A Survey of Component Order Connectivity Models of Graph Theoretic Networks

DANIEL GROSS
Seton Hall University
Dept of Math and CS
South Orange, NJ 07079
USA
daniel.gross@shu.edu

MONIKA HEINIG
Stevens Institute
Dept of Mathematical Sciences
Hoboken, NJ 07030
USA
mheinig@stevens.edu

LAKSHMI ISWARA
Stevens Institute
Dept of Mathematical Sciences
Hoboken, NJ 07030
USA
liswarac@stevens.edu

L. WILLIAM KAZMIERCZAK
Stevens Institute
Dept of Mathematical Sciences
Hoboken, NJ 07030
USA
lkazmier@stevens.edu

KRISTI LUTTRELL
Stevens Institute
Dept of Mathematical Sciences
Hoboken, NJ 07030
USA
kmartini@stevens.edu

JOHN T. SACCOMAN
Seton Hall University
Dept of Math and CS
South Orange, NJ 07079
USA
john.saccoman2@shu.edu

CHARLES SUFFEL
Stevens Institute
Dept of Mathematical Sciences
Hoboken, NJ 07030
USA
csuffel@stevens.edu

Abstract: The traditional vulnerability parameter connectivity is the minimum number of nodes needed to be removed to disconnect a network. Likewise, edge connectivity is the minimum number of edges needed to be removed to disconnect. A disconnected network may still be viable if it contains a sufficiently large component. Component order connectivity and component order edge connectivity are the minimum number of nodes, respectively edges needed to be removed so that all components of the resulting network have order less than some preassigned threshold value. In this paper we survey some results of the component order connectivity models.

Key–Words: Connectivity, edge connectivity, component order connectivity, component order edge connectivity, component order neighbor connectivity

1 Introduction

This paper is a chronicle of the ongoing study by the Stevens/Seton Hall Graph Theory Group of a vulnerability model of a graph-theoretic network. Specifically, a network is modeled by a (simple) graph with either its nodes or its edges subject to either random or purposeful failure. We consider the nodes and edges to be the elements of the associated system and the operating elements remaining after the failure of some of the elements at a snapshot in time is referred to as a **state of the system**. Of course an edge being operable presupposes that both end nodes are also operable. In the traditional models of connectivity, a failure state occurs when the surviving subgraph is either disconnected or trivial. Thus, a failure state may contain a large component, a subset of a failure state may be an operating state, and relatively small operating states are tolerated. In many cases the surviving network may still be able to perform the function of the network provided there is a sufficiently large component remaining, regardless of whether or not the graph is connected. An example of such a network is a distributed computer system, which may still function at a reasonable level as long as there are a sufficient number of computers still interconnected.

In our model a threshold value $k \in \mathbb{Z}^+$ is determined beforehand and a state is an **operating state** provided that the induced subgraph defined by the operating elements has at least one component of order at least k. Otherwise, if the operating elements induce a subgraph with all components of order at most $k-1$, the state is called a **failure state** and the associated set of failed elements is called a **failure set**.

For the most part we limit the study to scenarios where either nodes may fail but edges do not or edges may fail but nodes do not. Of course if a node fails the edges incident at it become inoperable and must be removed from the graph. We also introduce one other scenario currently being studied. In this case nodes fail, edges do not but nodes adjacent to failed nodes are subverted and are therefore considered inoperable. Thus the state of the associated system consists of the subgraph induced by those nodes that have neither failed nor have been subverted.

In the next section we formally introduce the models and essential parameters along with some elementary facts. In the third section we study the relationship between our new node vulnerability parameter and κ the usual (node) connectivity parameter. In the fourth section we investigate the relationship between our new edge vulnerability parameter and other graph parameters. In the fifth section we introduce some algorithms to compute the parameters for trees and cycles. In our sixth and final section we introduce the model where nodes adjacent to failed nodes are subverted and present some preliminary results.

2 Preliminaries

Consider a network modeled by a (simple) graph $G = (V, E)$ having $n = |V|$ nodes and $e = |E|$ edges. We consider two scenarios, either the nodes are subject to failure and the edges do not fail or the edges are subject to failure and the nodes do not. Note if a node fails, all edges incident at the node become inoperable. In the case that nodes fail we define an associated system on the set of nodes V. A state of the system is the collection of operating nodes at a given instance of time. For convenience, we will also refer to the subgraph of G spanned by the operating nodes as a state of the system. Observe that if W is the set of nodes that have failed, then this subgraph is precisely $G - W$. In the traditional node failure scenario, a state is an operating state provided the subgraph is connected and has at least 2 nodes; otherwise it is a failure state. The set of nodes W whose failure created the failure state is called a failure set and the (node) connectivity parameter κ is defined as the minimum cardinality of a failure set. Similarly, in the case that edges fail, we define an associated system on the set of edges E. A state of the system is the collection of operating edges at a given instance of time. For convenience, we will also refer to the subgraph of G spanned by the operating edges as a state of the system. Observe that if F is the set of edges that have failed, then this subgraph is precisely $G - F$. In the traditional edge failure scenario, a state is an operating-state provided the subgraph is connected; otherwise it is a failure state. The set of edges W whose failure created the failure state is called an edge failure set and the edge connectivity parameter λ is defined as the minimum cardinality of an edge failure set.

In our study a state is an operating-state provided the subgraph that remains after the failure of some nodes or some edges contains a component of order at least k, where $2 \leq k \leq n$ is a pre-assigned threshold value. Otherwise a failure state occurs, i.e. all components of the surviving subgraph have order at most $k-1$. We now more formally define these terms and the associated parameters.

Definition 1 *Let $G = (V, E)$ be a graph having $n = |V|$ nodes and $e = |E|$ edges. Assume $2 \leq k \leq n$.*

a) *If nodes fail but edges do not, a* **state** *is the set of operating nodes at a snapshot in time. A state is an* **operating-state** *if the operating nodes induce a subgraph with at least one component of order at least k. Otherwise, it is a* **failure state**, *i.e. all components of the induced subgraph have order at most $k-1$ and, the associated set of failed nodes is called a* **failure set**. *The* **k-component order connectivity,** *$\kappa_c^{(k)}(G)$ or simply $\kappa_c^{(k)}$, is the minimum cardinality of a failure set.*

b) *If edges fail but nodes do not, a state is the set of operating edges at a snapshot in time. A state is an* **edge operating state** *if the operating edges induce a subgraph with at least one component of order at least k. Otherwise, it is an* **edge failure state**, *i.e. all components of the induced subgraph have order at most $k-1$ and the associated set of failed edges is called an* **edge failure set**. *The* **k-component order edge connectivity,** *$\lambda_c^{(k)}(G)$ or simply $\lambda_c^{(k)}$, is the minimum cardinality of an edge failure set.*

Note that for convenience, we will also refer to the subgraph induced by the operating nodes or edges as a state. If W is the set of nodes which have failed then the state is precisely the subgraph $G - W$. If F is the set of edges that have failed, then the state is the subgraph $G - F$.

We give some examples for these new parameters. Note that we compute $\kappa_c^{(k)}$ by either finding a failure set of minimum cardinality or finding a failure state of maximum order, and we can compute $\lambda_c^{(k)}$ by either finding an edge failure set of minimum cardinality or an edge failure state of maximum size.

Example 2 *Assume* $2 \leq k \leq n$ *.*

a) *Let* $G = P_n$*, the path on* n *nodes. Starting from an end node, label the nodes from* 1 *to* n*. It is easily seen that the set of all nodes whose label is divisible by* k *forms a minimum cardinality failure set, thus* $\kappa_c^{(k)}(P_n) = \left\lfloor \frac{n}{k} \right\rfloor$ *. If we instead label the edges from* 1 *to* $n-1$*, then the set of all edges whose label is divisible by* $k-1$ *forms a minimum cardinality edge failure set, thus* $\lambda_c^{(k)}(P_n) = \left\lfloor \frac{n-1}{k-1} \right\rfloor$*.*

b) *Let* $G = C_n$ *, the cycle on* n *nodes. To find a minimum failure set, let* u *be an arbitrary node and label the nodes sequentially on the cycle from* 0 *to* $n-1$*, with* u *the node labeled* 0*. Then* u *along with the nodes found by applying the process in a) to* $C_n - u = P_{n-1}$ *, already labeled forms a minimum cardinality failure set. Thus* $\kappa_c^{(k)}(C_n) = \left\lceil \frac{n}{k} \right\rceil$ *. An analogous procedure shows* $\lambda_c^{(k)}(C_n) = \left\lceil \frac{n}{k-1} \right\rceil$*.*

c) *Let* $G = K_{1,n-1}$ *, the complete bipartite graph with one part of order* 1*. Removal of the center node isolates all the other nodes, thus* $\kappa_c^{(k)}(K_{1,n-1}) = 1$*. Removal of* $n-k+1$ *edges leaves one non-trivial component of order* $k-1$*, thus an edge failure state. Thus* $\lambda_c^{(k)}(K_{1,n-1}) = n-k+1$ *.*

d) *Let* $G = K_n$*, the complete graph on* n *nodes. Since the deletion of any set of nodes does not disconnect, a failure state occurs only when* $k-1$ *or fewer nodes operate. Thus* $\kappa_c^{(k)}(K_n) = n-k+1$ *. All components of a maximum size edge failure state of* K_n *will be complete. An algebraic computation shows that the maximum size edge failure state occurs when the number of components of order* $k-1$ *is maximized. Thus*

$$\lambda_c^{(k)}(K_n) = \binom{n}{2} - \left\lfloor \frac{n}{k-1} \right\rfloor \binom{k-1}{2} - \binom{r}{2},$$

where $n = \left\lfloor \frac{n}{k-1} \right\rfloor (k-1) + r$*,* $0 \leq r \leq k-2$*.*

e) *Let* $G = K_{p,q}$*, the complete bipartite graph on* $p+q = n$ *nodes with* $p \leq q$*. The computation of* $\kappa_c^{(k)}(K_{p,q})$ *is relatively easy and depends on the size of* k*. When* $k \leq q$*, then* $\kappa_c^{(k)}(K_{p,q}) = p$*, since removing the part of order* p *leaves a subgraph consisting of* q *isolates and thus a failure state while removing fewer than* p *nodes does not disconnect and leaves a component of order greater than* q *and therefore greater than* k*. When* $q < k$ *it is not necessary to disconnect to obtain a failure state, i.e. a failure state also occurs when* $k-1$ *or fewer nodes operate. Therefore if* $q < k$*, then*

$$\kappa_c^{(k)}(K_{p,q}) = p + q - k + 1 = n - k + 1.$$

On the other hand, the computation of $\lambda_c^{(k)}(K_{p,q})$ *is surprisingly difficult. All non-trivial components of a maximum size edge failure state of* $K_{p,q}$ *will be complete-bipartite. In the case that* $p \leq \left\lfloor \frac{p+q}{k-1} \right\rfloor$ *, a maximum size edge failure state consists of* p *copies of* $K_{1,k-2}$ *and* $q - p(k-2)$ *isolated nodes. Thus when* $p \leq \left\lfloor \frac{p+q}{k-1} \right\rfloor$*,* $\lambda_c^{(k)}(K_{p,q}) = p(q-k+2)$*[1]. When* $p > \left\lfloor \frac{p+q}{k-1} \right\rfloor$*, the analysis of the composition of a maximum size edge failure state is more involved and we refer the readers interested in this result to [1].*

Remark 3 *Based on the examples just discussed, one could surmise that, for example, a maximum order failure state can always be constructed by judiciously breaking off components of order* $k-1$*. But consider the graph in Figure* 1 *with* $k = 5$*. The deletion of nodes* u *and* v *leaves a component of order* 4*, thus a failure state, while the deletion of node* w *leaves two components, one of order* 3 *the other order* 2*, thus a failure state. So a maximum order failure state exists with no component of order* $k-1$*. We note also that in the example the bridge is the unique minimum edge failure set and its deletion leaves no component of order* $k-1$*.*

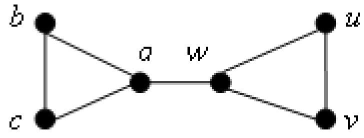


Figure 1

In general, the computation of $\kappa_c^{(k)}$ and $\lambda_c^{(k)}$ for an arbitrary graph G is not an easy problem. In Section 5 we give algorithms that compute these parameters for trees and unicycles. We do not have a general procedure that applies to any graph G.

We now give some results pertaining to these new parameters. The first gives values of $\kappa_c^{(k)}$ and $\lambda_c^{(k)}$

for some specific values of k. We denote by $\beta(G)$ and $\beta_1(G)$ the independence number, i.e. the maximum number of non-adjacent nodes, and the edge independence number, i.e. the maximum number of non-adjacent edges, respectively.

Proposition 4 *[2]: Let G be a graph on n nodes and e edges.*

1. $\kappa_c^{(2)}(G) = n - \beta(G)$ *and* $\kappa_c^{(n)}(G) = 1$.
2. $\lambda_c^{(2)}(G) = e$, $\lambda_c^{(3)}(G) = e - \beta_1(G)$, *and* $\lambda_c^{(n)}(G) = \lambda(G)$.

Our next easily established result shows that $\kappa_c^{(k)}$ and $\lambda_c^{(k)}$ satisfy the *subgraph property*.

Proposition 5 *Let H be a subgraph of G on n nodes, then for any* $2 \le k \le n$, $\kappa_c^{(k)}(H) \le \kappa_c^{(k)}(G)$ *and* $\lambda_c^{(k)}(H) \le \lambda_c^{(k)}(G)$

The next result, also easily established, shows the relationship of the parameters for different threshold values.

Proposition 6 *Let* $2 \le k \le l \le n$, *then*
$\kappa_c^{(l)}(G) \le \kappa_c^{(k)}(G)$ *and* $\lambda_c^{(l)}(G) \le \lambda_c^{(k)}(G)$.

Remark 7 *The last proposition provides another difficulty with finding a general procedure to compute the parameters. Consider once again the graph in Figure 1. It is easy to see by inspection that* $\lambda_c^{(3)} = 4$ *with the only minimum edge failure set* $F_3 = \{ab, ac, wu, wv\}$ *while* $\lambda_c^{(4)} = 1$ *with the only minimum edge failure set* $F_4 = \{aw\}$. *Thus* $\lambda_c^{(4)} = |F_4| \le |F_3| = \lambda_c^{(3)}$ *, which satisfies the proposition but* $F_4 \not\subset F_3$ *; in fact in this example* $F_4 \cap F_3 = \emptyset$.

We conclude this section with some results involving trees. In Example 2 a) and c) we computed the values of $\kappa_c^{(k)}$ and $\lambda_c^{(k)}$ for the trees P_n and $K_{1,n-1}$. These are the extremal trees for these parameters.

Proposition 8 *[3]: Let* T_n *be a tree on* n *nodes, then for any value k,* $2 \le k \le n$

1.
$$1 = \kappa_c^{(k)}(K_{1,n-1}) \le \kappa_c^{(k)}(T_n) \le \kappa_c^{(k)}(P_n) = \left\lfloor \frac{n}{k} \right\rfloor .$$
Moreover if α *is any value* $1 \le \alpha \le \lfloor \frac{n}{k} \rfloor$, *then there exists a tree T on* n *nodes with* $\kappa_c^{(k)}(T) = \alpha$.

2.
$$\left\lfloor \frac{n-1}{k-1} \right\rfloor = \lambda_c^{(k)}(P_n) \le \lambda_c^{(k)}(T_n) \le \lambda_c^{(k)}(K_{1,n-1}) = n-k+1.$$
Moreover if α *is any value* $\left\lfloor \frac{n-1}{k-1} \right\rfloor \le \alpha \le n-k+1$, *then there exists a tree T on* n *nodes with* $\lambda_c^{(k)}(T) = \alpha$.

Remark 9 *Thus, for any value of k, the worst tree for* $\kappa_c^{(k)}$ *is the best tree for* $\lambda_c^{(k)}$ *, while the best tree for* $\kappa_c^{(k)}$ *is the worst tree for* $\lambda_c^{(k)}$. *Therefore, for the class of graphs with* n *nodes and* $n-1$ *edges there is no value of k for which it is possible to simultaneously maximize both* $\kappa_c^{(k)}$ *and* $\lambda_c^{(k)}$.

3 κ versus $\kappa_c^{(k)}$

Although the parameter κ is not a pragmatic measure of system failure, it does have a pragmatic routing interpretation. Indeed, the Menger-Whitney Theorem [4] states that κ equals the maximum number of internally disjoint paths joining any pair of non-adjacent nodes of a network. Hence a study of the interrelationships between κ and $\kappa_c^{(k)}$ is called for. Issues regarding how far apart these parameters can be from each other as well as the determination of pairs (n, e) such that κ and $\kappa_c^{(k)}$ may be simultaneously maximized are significant network analysis and design concerns. We begin this discussion with a result that compares relative sizes of κ and $\kappa_c^{(k)}$ followed by a "realizability" result describing necessary and sufficient conditions on a 4-tuple (n, k, a, b) or 5-tuple (n, e, k, a, b) for which a graph on n nodes and e edges exists having $\kappa = a$ and $\kappa_c^{(k)} = b$.

Theorem 10 *[5]: If G is a graph on n nodes and* $2 \le k \le n$, *then*

1. $\kappa(G) \ge n-k+1$ *implies that*
$$\kappa_c^{(k)}(G) = n-k+1 \text{ and}$$
2. $\kappa(G) \le n-k$ *implies that*
$$\kappa(G) \le \kappa_c^{(k)}(G) \le n-k.$$

Remark 11 *It is easy to see that* $\kappa_c^{(k)}(G) \le n-k+1$ *since the removal of any set of* $n-k+1$ *nodes* *leaves only $k-1$ survivors. Hence, the first condition guarantees that G attains the maximum possible value of $\kappa_c^{(k)}$ while the second indicates that if κ is too small $n-k+1$ is unattainable, rather $n-k$ is the maximum possible.*

In our next result we establish necessary and sufficient conditions which specify precisely the combinations of κ and $\kappa_c^{(k)}$ that are realizable.

Theorem 12 *[6]:*

1. *The 5-tuple (n, e, k, a, b), where $2 \leq k \leq n$ and $a \geq n-k+1$, is realizable by a graph on n nodes and e edges having $\kappa = a$ and $\kappa_c^{(k)} = b$ if and only if either*

   (a) $n = k$, $a = b = 1$ and $k-1 \leq e \leq \frac{(n-1)(n-2)}{2}+1$, or

   (b) $2 \leq k \leq n-1, a \geq 2, b = n-k+1$, and $\lceil \frac{na}{2} \rceil \leq e \leq \frac{(n-1)(n-2)}{2}+a$

2. *The 4-tuple (n, k, a, b) is realizable by a graph on n nodes, where $2 \leq k \leq n$ and $a \leq n-k$, provided $0 \leq a \leq b \leq n-k$.*

Remark 13 *The result given in Theorem 12(1) follows immediately from Theorem 10(1) and the well-known result of Harary [7] that a graph G exists with $\kappa = a$ if and only if the bounds given on e are satisfied. As for Theorem 12(2) we indicate a construction which proves realizability. A modification of this construction yields ranges of e values which we do not indicate here but are given in [6].*

Construction 14 *Consider*

$$G = \left(K_{k-1+b-a} \cup \overline{K}_{n-k+1-b}\right) + K_a;$$

it has $\kappa = a$ and $\kappa_c^{(k)} = b$ where $0 \leq a \leq b \leq n-k$.

In the following discussion we consider the possibility of simultaneously maximizing κ and $\kappa_c^{(k)}$. We begin with two simple results that follow from a result of Harary [7] and Theorem 10. Specifically, Harary showed that, given n and e such that $\lfloor \frac{2e}{n} \rfloor \geq 2$, there exists a graph on n nodes and e edges, namely a power of the cycle C_n with additional edges if necessary, having $\kappa = \lambda = \delta = \lfloor \frac{2e}{n} \rfloor$. As $\delta \leq \lfloor \frac{2e}{n} \rfloor$ such a graph has maximum value of κ over all graphs with n nodes and e edges and is referred to as max-κ.

Theorem 15 1. *If $\lfloor \frac{2e}{n} \rfloor \geq n-k+1$ then every max-κ graph is max-$\kappa_c^{(k)}$, i.e. $\kappa_c^{(k)} = n-k+1$.*

2. *If $\lfloor \frac{2e}{n} \rfloor = n-k \geq 2$ then every max-κ graph is max-$\kappa_c^{(k)}$, i.e. $\kappa_c^{(k)} = n-k$.*

The next theorem concerns a forbidden subgraph condition on $\overline{G}$, the complement of G, that guarantees max-κ and max-$\kappa_c^{(k)}$ for $k \geq 3$ and minimum diameter. This result requires two preliminary results that are interesting in and of themselves. Although we do not indicate the proof of the main result, we present the two requisite results anyway.

Proposition 16 *[8]: Suppose $n \geq 4$ and G is a graph on n nodes. Then*

1. *$\kappa(G) = \delta(G)$ if and only if $K_{m,n-\delta+1-m} \not\subset \overline{G}$ for every $2 \leq m \leq n-\delta+1-m$, and*

2. *for $k \geq 3$, $\kappa_c^{(k)}(G) \geq n-k$ if and only if $K_{q,k+1-q} \not\subset \overline{G}$ for every $2 \leq q \leq k+1-q$.*

As indicated above, the next general result guarantees a strong network design for relatively dense situations.

Theorem 17 *[8]: Suppose $\delta(G) = \lfloor \frac{2e}{n} \rfloor \geq \lfloor \frac{n}{2} \rfloor$ and $K_{2,2} = C_4 \not\subset \overline{G}$. Then*

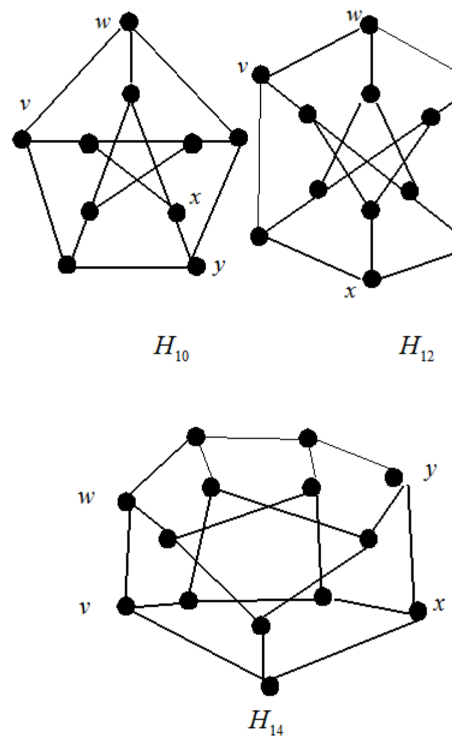
1. *G is max-κ,*

2. *G is max-$\kappa_c^{(k)}$ for every $k \geq 3$, i.e. $\kappa_c^{(k)} = n-k$ if $\lfloor \frac{2e}{n} \rfloor \leq n-k$ or $\kappa_c^{(k)} = n-k+1$ if $\lfloor \frac{2e}{n} \rfloor \geq n-k+1$, and*

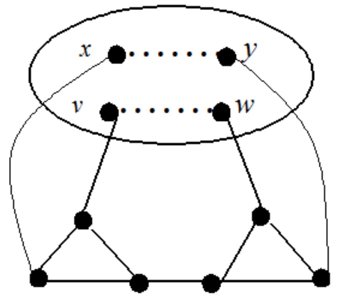
3. *d(G) ≤ 2, where d(G) is the diameter of G.*

We have obtained constructions for $k \leq n-4$ [9] but only describe the one for $k = n-4$.

Construction 18 *Observe that the three graphs H_{10} (the Peterson graph), H_{12} and H_{14} shown in Figure 2(a) are cubic and C_4-free. Then beginning with these graphs, inserted for H in the graph of Figure 2(b) and continuing recursively, one obtains cubic, C_4-free graphs for all even orders $n \geq 10$. Next we remove at most $\frac{n}{2}-1$ edges from each of them so that the maximum degree is exactly three. By taking complements of the resulting graphs we obtain G's with n nodes and e edges such that $\delta(G) = \lfloor \frac{2e}{n} \rfloor = n-4$ and $C_4 \not\subset \overline{G}$.*



a) The base graphs H_{10}, H_{12} and H_{14}



b) The Construction

Figure 2

We have obtained a partial generalization of Theorem 17.

Theorem 19 *[9]: Suppose that $\delta(G) = \lfloor \frac{2e}{n} \rfloor = m - l$, where $l \geq 3$, G is almost regular (i.e. degrees differ by at most one) and $m \geq l$ is the smallest value such that $K_{q,m+1-q} \not\subset \overline{G}$ for every $2 \leq q \leq m+1-q$. Then*

1. *G is max-κ,*
2. *for $k > m$, G is max-$\kappa_c^{(k)}$ with $\kappa_c^{(k)} = n - k + 1$,*
3. *for $l \leq k \leq m$, G is max-κ with $\kappa_c^{(k)} = n - k$, and*
4. *if $l > 3$ and $3 \leq k \leq l$, then $\kappa_c^{(k)} \geq n - l$.*

The final discussion in this section concerns $\kappa_c^{(2)}$ versus κ. As $\kappa_c^{(2)}(G) = n - \beta(G)$, where $\beta(G)$ is the independence number of G, and calculations of $\beta(G)$ is an NP-hard problem, it is not surprising that this discussion turns out to be relatively involved. However, first we present an easily established result.

Theorem 20 *[8]: If $\delta(G) = \lfloor \frac{2e}{n} \rfloor = n - 1$ or $n - 2$, then G is max-κ , max-$\kappa_c^{(2)}$ and $d(G) \leq 2$.*

We next present a result that gives max-$\kappa_c^{(2)}$ dependent on the value of e; but first some preliminaries must be stated, the first of which requires no proof.

Proposition 21 *For $n \geq m \geq 2$, $\kappa_c^{(2)}(G) = n - m$ if and only if $\overline{G}$ is K_{m+1}-free but $\overline{G}$ contains a K_m.*

Our next requisite fact is the famous theorem of Turan [10].

Proposition 22 *If H is K_{m+1}-free, then $e(H) \leq t_n(m)$, where $t_n(m)$ is the number of edges in the almost complete multipartite graph K(m,n) on n nodes with m parts. Furthermore, K(m,n) is the unique extremal graph.*

Now the theorem:

Theorem 23 *[10]: Given n, e and $\overline{e} = \binom{n}{2} - e$ the maximum $\kappa_c^{(k)}$ value is $n - m$ if and only if*

$$t_n(m-1) < \overline{e} \leq t_n(m).$$

This result follows readily by removing the appropriate number of edges from $K(m,n)$ and applying the two previous propositions.

Ballista and Bollabas [11] proved the following generalization of a theorem of Erdos [12] which we shall use to obtain a range of values for which κ and $\kappa_c^{(2)}$ cannot be simultaneously maximized.

Proposition 24 *If H has n nodes and*

$$t_n(m) - \left\lfloor \frac{n}{m} \right\rfloor + 2 \leq e(H) \leq t_n(m),$$

then H is K_{m+1}-free if and only if H is m-partite. Furthermore, the lower bound is tight.

The following theorem yields a range of e values for which no max-κ , max-$\kappa_c^{(k)}$ graph exists.

Theorem 25 *[14]: If G has n nodes and e edges and $\overline{e} = \left(\begin{array}{c} n \\ 2 \end{array}\right) - e$, then $t_n(m) - \left\lfloor \frac{n}{m} \right\rfloor + 2 \leq \overline{e} \leq t_n(m)$ implies that G cannot be both max-κ and max-$\kappa_c^{(2)}$.*

This result follows from the observation that G being max-$\kappa_c^{(2)}$ forces $\cup_{i=1}^{m} K_{n_i} \subset G$ where $\sum_{i=1}^{m} n_i = n$, subsequently followed by a nontrivial argument that $\kappa(G) < \left\lfloor \frac{2e}{n} \right\rfloor$ [14].

We complete this discussion with a positive result. It is a partial answer to the question as to whether simultaneous maximization is possible when $\overline{e} \leq t_n(m) - \left\lfloor \frac{n}{m} \right\rfloor + 1$.

Theorem 26 *[14]: If n = mr, m is even and $m \geq 4$ or m = 3, $r = \frac{n}{m} \geq 2$ and $t_n(m-1) \leq \overline{e} \leq t_n(m) - \frac{n}{2}$, then there exists a graph G with n nodes and e edges which is max-κ and max-$\kappa_c^{(2)}$.*

Indeed, a graph obtained by judiciously adding a matching to m disjoint copies of K_r handles the case of $\overline{e} = t_n(m) - \frac{n}{2}$. The construction for the remaining values of $\overline{e}$ can be found in [14].

Remark 27 *Unfortunately we cannot shed light on the missing range $t_n(m) - \frac{n}{2} \leq \overline{e} \leq t_n(m) - \left\lfloor \frac{n}{m} \right\rfloor + 1$.*

4 The Relationship of $\lambda_c^{(k)}$ with the Other Parameters

We first note that the relationship between $\lambda_c^{(k)}$ and λ is straightforward; in fact since a $\lambda_c^{(k)}$-failure state must be disconnected, it follows that $\lambda_c^{(k)} \leq \lambda$ for all k, $2 \leq k \leq n$. The following result, which follows immediately from proposition 4(2) and Proposition 6, extends this inequality.

Proposition 28 *For any connected graph G on n nodes and e edges,*

$$\lambda(G) = \lambda_c^{n}(G) \leq \lambda_c^{(n-1)}(G) \leq \cdots \leq \lambda_c^{(2)}(G) = e.$$

It is well known that for any graph G, $\kappa \leq \lambda \leq \delta$. An analogous inequality to the first holds when we consider component order connectivity and component order edge connectivity.

Theorem 29 *[3]: For any connected graph G on n nodes and e edges and for any k, $2 \leq k \leq n$, $\kappa_c^{(k)}(G) \leq \lambda_c^{(k)}(G)$.*

When we apply this to trees along with Proposition 2.48 we come up with the surprising result: if T_1 and T_2 are two trees on n nodes, then $\kappa_c^{(k)}(T_1) \leq \lambda_c^{(k)}(T_2)$, for all $k, 2 \leq k \leq n$.

Given a graph G on n nodes and e edges $\lambda(G) \leq \delta(G) \leq e$, and therefore $\lambda_c^{(n)}(G) \leq \delta(G) \leq \lambda_c^{(2)}(G)$. Taking into consideration the string of inequalities from Proposition 28, it is natural to ask where $\delta(G)$ appears, i.e. is there a value of k such that $\lambda_c^{(k+1)}(G) < \delta(G) \leq \lambda_c^{(k)}(G)$. We answer this question when $\delta(G)$ is sufficiently large by generalizing a result of Chartrand.

In 1966 Chartrand proved the following:

Theorem 30 *[15]: Let G be a connected graph of order n. If $\delta(G) \geq \left\lfloor \frac{n}{2} \right\rfloor$ then $\lambda(G) = \delta(G)$ Moreover, this bound is best possible in the sense that there exists a connected graph G' of order $n \geq 6$ with $\delta(G') = \left\lfloor \frac{n}{2} \right\rfloor - 1$ and $\lambda(G') < \delta(G')$.*

The next theorem generalizes Chartrand's result.

Theorem 31 *[16]: Let G be a connected graph of order n. If $\delta(G) \geq \left\lfloor \frac{n}{l+1} \right\rfloor$, $1 \leq l \leq n-1$, then $\lambda_c^{(\lceil \frac{n}{l} \rfloor)}(G) \geq \delta(G)$. Moreover, if $n \geq l(l+1)$ this is best possible in the sense that for all δ such that $\left\lfloor \frac{n}{l+1} \right\rfloor \leq \delta \leq \left\lfloor \frac{n}{l} \right\rfloor - 1$, there exists a connected graph G' of order n with $\delta(G') = \delta$ and $\lambda_c^{(\lceil \frac{n}{l} \rfloor + 1)}(G) < \delta(G)$.*

Observe that when $l = 1$, we obtain Theorem 30.

The previous theorem does not give the best possible result when $n < l(l+1)$. Our next theorem shows that in fact in this case $\left\lceil \frac{n}{l} \right\rceil$ is not best possible.

Theorem 32 *Let G be any connected graph of order n with minimum degree $\delta = \left\lfloor \frac{n}{l+1} \right\rfloor$, where $n < l(l+1)$ and $l > 1$. Write $n = \left\lfloor \frac{n}{l+1} \right\rfloor (l+1) + r$, where $0 \leq r \leq l$ and set $m = \left\lceil \frac{r}{\left\lfloor \frac{n}{l+1} \right\rfloor} \right\rceil$. Then $\lambda_c^{(l+m+1)}(G) \geq \delta(G) = \left\lfloor \frac{n}{l+1} \right\rfloor$. Moreover, this is best possible in the sense that there exists a connected graph G' of order n with $\delta(G') = \left\lfloor \frac{n}{l+1} \right\rfloor$ and $\lambda_c^{(l+m+2)}(G') < \delta(G')$.*

5 Algorithmic Considerations

We begin by observing that the problem of computing $\kappa_c^{(k)}(G)$ for arbitrary k and arbitrary G is NP-hard since $\kappa_c^{(2)}(G) = n - \beta(G)$ where $\beta(G)$ is the independence number of G.

On the other hand, suppose a graph is equipped with nonnegative weights on the vertices and $2 \leq k \leq w_G$, where w_G denotes the sum of all weights of G. Let $\kappa_{wc}^{(k)}(G)$ denote the minimum order of a set of vertices whose failure leaves a subgraph with each component having weight at most $k-1$. We present a polynomial algorithm for computing $\kappa_{wc}^{(k)}(T)$ for an arbitrary tree T. We also present an algorithm for a weighted unicycle. Regarding the edge model, we define $\lambda_{wc}^{(k)}$ analogously for graphs with weighted nodes, and present polynomial algorithms for computing $\lambda_{wc}^{(k)}$ of an arbitrary tree and an arbitrary unicycle. We also present an efficient algorithm for computing $\kappa_{wc}^{(n-\alpha)}(G)$ for constant α and arbitrary G.

Definition 33 *Suppose G is equipped with non-negative weights on the vertices and let $2 \leq k \leq w_G$, where w_G denotes the sum of all weights of G. Then $\kappa_{wc}^{(k)}(G)$ is the minimum number of vertices whose removal leaves a subgraph with each component having weight at most $k-1$.*

Remark 34 *If all nodes have weight equal to 1, then $\kappa_{wc}^{(k)}(G) = \kappa_c^{(k)}(G)$.*

Algorithm 1 Algorithm for $\kappa_{wc}^{(k)}(T)$ for a Weighted Tree T on n Nodes.

- **Input:** *Tree T with weighting function $w(u) \geq 0$ for $u \in V(T)$; Integer k such that*

$$2 \leq k \leq w(T) = \sum_{u \in V(T)} w(u).$$

- **Output:** *$\kappa_{wc}^{(k)}(T)$ and an associated minimum failure set W.*

- **Initialization:** *Root T at any vertex $r \in V(T)$ and for each $v \in V(T)$ let $l(v)$, the level of v, denote the distance of v from r. Set $L = max\{l(v) | v \in V(T)\}$, $W = \emptyset$, CompSize(u)=0 for each $u \in V(T)$, and $T' = T$.*

- *While $L \geq 0$ do*

  1. *For each u such that $l(u) = L$, let CompSize(u) = w(u) + $\sum_{v_i \text{ a child of } u \text{ in } T'}$ CompSize(v_i).*
  2. *For each u at level L if $CompSize(u) \geq k$ add u to W and set $T' = T' - T(u)$, where T(u) is the subtree of T' rooted at u.*
  3. *Set $L = L - 1$.*

  end while.

- **Return:** *$\kappa_{wc}^{(k)}(T) = |W|$ and minimum failure set W.*

We defer proving correctness and refer the reader to [17]. Instead, we provide an example

Example 35 *Consider weighted tree T as shown in Figure 3 with weights $\overrightarrow{w} = (1,3,2,1,0,3,2,1,1,1,2,3)$ where $\overrightarrow{w}_i = w(u_i)$ and $k = 4$. In the parentheses we indicate the weight and the level of the node.*

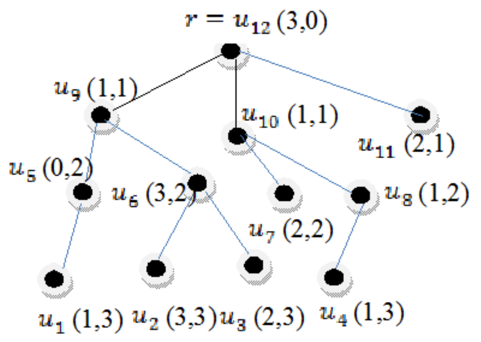


Figure 3: Tree T

Initialization: $L = 3$, $W = \emptyset$, $T' = T$, *CompSize(u) = 0 for all $u \in V(T)$.*

L = 3:

1. *CompSize$(u_1) = 1$, CompSize$(u_2) = 3$, CompSize$(u_3) = 2$, CompSize$(u_4) = 1$.*
2. *Since CompSize$(u_i) < 4$ for all u_1 with $l = 3$, T' remains the same.*
3. *Set $L = 2$.*

L = 2:

1. *CompSize$(u_5) = 0 + 1 = 1$, CompSize$(u_6) = 3 + 3 + 2 = 8$, CompSize$(u_7) = 2$, CompSize$(u_8) = 1 + 1 = 2$.*

2. *Since CompSize*$(u_6) = 8 \geq 4$, *set* $W = \{u_6\}$ *and* $T' = T' - T(u_6)$, *see Figure 4.*

3. *Set* $L = 1$.

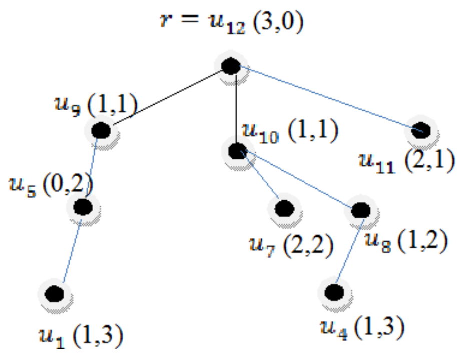


Figure 4: T' *at the next stage*

L = 1:

1. *CompSize*$(u_9) = 1 + 1 = 2$, *CompSize*$(u_{10}) = 1 + 2 + 2 = 5$, *CompSize*(u_{11})*=2.*

2. *Since CompSize*$(u_{10}) = 5 \geq 4$, *set* $W = \{u_6, u_{10}\}$ *and* $T' = T' - T(u_{10})$, *see Figure 5.*

3. *Set* $L = 0$.

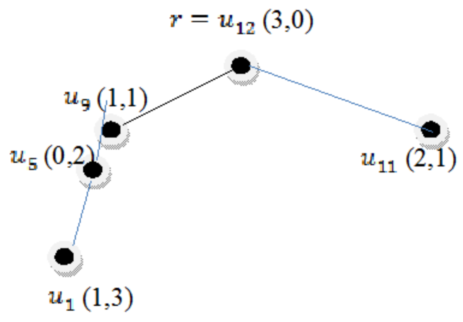


Figure 5: T' *at the next stage*

L = 0:

1. *CompSize*$(r = u_{12}) = 3 + 2 + 2 = 7$.

2. *Since CompSize*$(u_{12}) = 7 > 4$, *set* $W = \{u_6, u_{10}, u_{12}\}$ *and* $T' = T'(u_{12})$ *(which results in an empty tree).*

3. *Set* $L = -1$.

The while loop is done: $\kappa_{wc}^{(4)}(T) = |W| = 3$ *and* $W = \{u_6, u_{10}, u_{12}\}$ *is a minimum failure set.*

We now consider $\kappa_{wc}^{(k)}$ of a weighted unicycle. We first observe that either $\kappa_{wc}^{(k)}(C_n) = 0$ when $w(C_n) \leq k - 1$ or $\kappa_{wc}^{(k)}(C_n) = min\left\{1 + \kappa_{wc}^{(k)}(C_n - u) | u \in V(C_n)\right\}$, so that n applications of the tree algorithm serves to determine $\kappa_{wc}^{(k)}(C_n)$. Next consider a weighted unicycle U on n nodes with unique cycle C_m with vertices $u_1, u_2, \ldots, u_m$. The remaining vertices, if any, form trees rooted at the vertices $u_1, u_2, \ldots, u_m$. Label each vertex with its distance from its associated root and refer to theses labels as levels (so that $u_1, u_2, \ldots, u_m$ are all at level 0).

Algorithm 2 Algorithm for $\kappa_{wc}^{(k)}(U)$ of a Weighted Unicycle U on n Nodes.

- **Input:** *Unicycle U on n nodes with unique cycle* C_m *on the vertices* $\{u_1, u_2, \ldots, u_m\}$ *and weighting function* $w(u) \geq 0$ *for* $u \in V(U)$; *Integer k such that* $2 \leq k \leq w(U) = \sum_{u \in V(U)} w(u)$.

- **Output:** $\kappa_{wc}^{(k)}(U)$ *and an associated minimum failure set W.*

- **Initialization:** *For each* $v \in V(U)$ *let l(v) denote the distance of v from* C_m. *Set* $L = max\{l(v)|v \in V(U)$, $W = \emptyset$, *CompSize(u)* = 0 *for each* $u \in V(U)$, *and* $U' = U$.

- *While* $L \geq 1$ *do*

```
1. For each u such that $l(u) = L$, let Compsize(u) = $w(u) + \sum_{v_i \text{ a child of } u \in U'} CompSize(v_i)$.
2. For each u at level L if $CompSize(u) \geq k$, add u to S and set $U' = U' - T(u)$, where T(u) is the subtree of $U'$ rooted at u.
3. Set $L = L - 1$.
end while.
```

- *For j* = 1 *to m, let* $CompSize(u_j) = w(u_j) + \sum_{v_i \text{ a child of } u_j \in U'} CompSize(v_i)$.

- *Let* $w \in V(C_m)$ *have maximum CompSize.*

- *If* $CompSize(w) \geq k$ *, then*
  add w to W, apply the tree algorithm to
  $U' - T(w)$, *rooted at a node from*

```
        the cycle to obtain a minimum failure
        set $W_w$ of $U' - T(w)$.
        Set $W = W \cup W_w$.
    else if $w(C_m) \geq k$, then
        for $j = 1$ to $m$, apply the tree algorithm to
        $W' - T(u_j)$, rooted at a node of the cycle
        and let $W_j$ be the failure set returned.
        Let $j$ be such that $|W_j|$ is minimum
        and set $W = W \cup \{u_j\} \cup W_j$.
```

- **Return:** $\kappa_{wc}^{(k)}(T) = |W|$ *and minimum failure set W.*

Again in lieu of providing a proof, which can be found in [17], we present examples.

Example 36 *Consider the weighted unicycle U as shown in Figure 6 with weights* $\overrightarrow{w} = (3, 1, 2, 3, 1, 2, 0, 1, 1, 3, 2, 1)$ *where* $\overrightarrow{w}_i = w(u_i)$ *and* $k = 4$*. In the parentheses we indicate the weight and the level (i.e. distance from the cycle* $C_5 = u_1, u_2, u_3, u_4, u_5, u_1$*) of the node.*

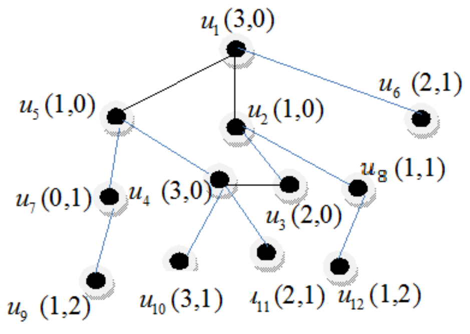


Figure 6: Weighted unicycle U

Initialization: $L = 2$, $W = \emptyset$, *CompSize*$(u) = 0$ *for all* $u \in V(U)$.
L = 2:

1. *CompSize*$(u_9) = 1$ *and CompSize*$(u_{12} = 1$.
2. *Since CompSize*$(u_i) < 4$ *for all* u_i *with* $l = 2$, U' *remains the same.*
3. *Set* $L = 1$.

L = 1:

1. *CompSize*$(u_6) = 2$, *CompSize*$(u_7) = 0 + 1 = 1$, *CompSize*$(u_8) = 1 + 1 = 2$, *CompSize*$(u_{10}) = 3$, *CompSize*$(u_{11}) = 2$.
2. *Since CompSize*$(u_i) < 4$ *for all* u_i *with* $l = 1$, U' *remains the same.*
3. *Set* $L = 0$.

L = 0: *The while loop is done.*
CompSize$(u_1) = 3+2 = 5$, *CompSize*$(u_2) = 1+2 = 3$, *CompSize*$(u_3) = 2$, *CompSize*$(u_4) = 3+3+2 = 8$, *CompSize*$(u_5) = 1 + 1 = 2$.
Set $w = u_4$.
Since CompSize$(u_4) = 8 > 4$, *set* $W = \{u_4\}$,
Apply the tree algorithm to $U' - w$ *rooted at* u_1*, see Figure 7.* **Note the level is now distance from the root** u_1.

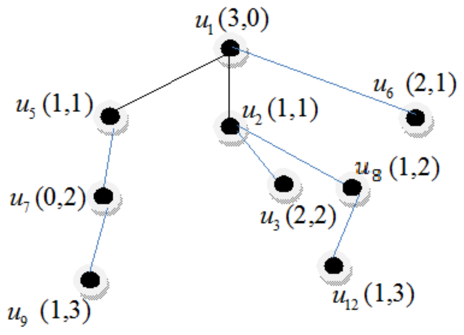


Figure 7: The tree $U' - w$ *rooted at* u_1

The minimum failure set obtained by the tree algorithm is $W_w = \{u_1, u_2\}$*. Set* $W = W \cup W_w = \{u_1, u_2, u_4\}$*. Then* $\kappa_{wc}^{(k)}(U) = |W|$ *and* $W = \{u_1, u_2, u_4\}$ *is a minimum failure set.*

Example 37 *Consider weighted unicycle U as shown in Figure 8 with weights* $\overrightarrow{w} = (3, 1, 2, 1, 0, 3, 2, 1, 1, 3, 2, 1)$ *where* $\overrightarrow{w}_i = w(u_i)$ *and* $k = 4$*. In the parentheses we indicate the weight and the level (i.e. distance from the cycle* $C_3 = u_1, u_2, u_3, u_1$*) of the node.*

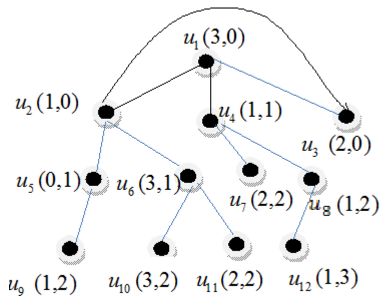


Figure 8: Unicycle U

In this discussion we will summarize the steps in the algorithm, rather than list them all. The first nodes encountered with CompSize $\geq k$ *occur when* $l = 1$*, namely CompSize*$(u_4) = 5$ *and CompSize*$(u_6) = 8$.

Then set $W = \{u_4, u_6\}$ and $U' = (U' - T(u_4)) - T(u_6)$, see Figure 9.

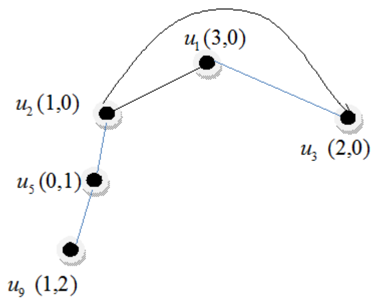


Figure 9: Unicycle

The while loop is done. CompSize$(u_1) = 3$, CompSize$(u_2) = 1 + 0 + 1 = 2$, CompSize$(u_3) = 2$. Thus $w(C_3) = 7 \geq 4$. Applying the tree algorithm to $U' - T(u_1)$ rooted at u_2 we obtain $W_1 = \{u_2\}$, applying the tree algorithm to $U' - T(u_2)$ rooted at u_3 we obtain $W_2 = \{u_3\}$, and applying the tree algorithm to $U' - T(u_3)$ rooted at u_1 we obtain $W_3 = \{u_1\}$. Since $|W_i| = 1$ for each i, we choose any of them, say W_1, and obtain $W = W \cup \{u_1\} \cup W_1 = \{u_1, u_2, u_4, u_6\}$. Therefore $W = \{u_1, u_2, u_4, u_6\}$ is a minimum failure set and $\kappa_{wc}^{(4)}(U) = |W| = 4$.

We now consider the case where edges fail.

Definition 38 *Suppose G equipped with non-negative weights on the edges and let $2 \leq k \leq w_G$, where w_G denotes the sum of all weights of G. Then $\lambda_{wc}^{(k)}(G)$ is the minimum number of edges whose removal leaves a subgraph with each component having weight at most $k - 1$.*

Remark 39 *If all nodes have weight equal to 1, then $\lambda_{wc}^{(k)}(G) = \lambda_c^{(k)}(G)$.*

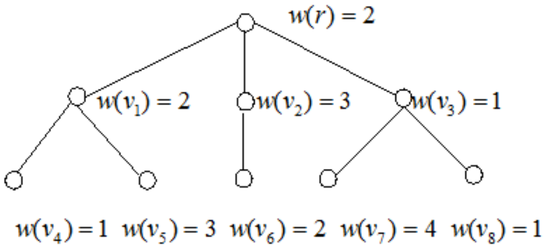
We now present efficient algorithms for computing $\lambda_{wc}^{(k)}$ of an arbitrary tree and of an arbitrary unicycle, using the presentation we gave in [18]. We shall omit proofs of validity given in [18] but we shall include the examples given there.

Algorithm 3 Algorithm for $\lambda_{wc}^{(k)}(T)$ of a Weighted Tree T on n Nodes.

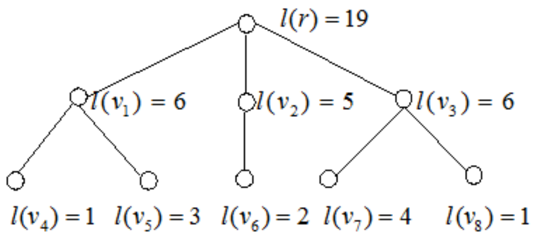
- **Input:** *Tree T with weighting function $w(u) \geq 0$ for $u \in V(T)$; Integer k such that*
$$2 \leq k \leq w(T) = \sum_{u \in V(T)} w(u).$$
- **Output:** *$\lambda_{wc}^{(k)}(T)$ and an associated minimum failure set F_a.*
- **Initialization:** *Root T at any vertex $r \in V(T)$ and for each $v \in V(T)$ set $l(v) = \sum_{u \in V(T(v))} w(u)$, where $T(v)$ denotes the subtree rooted at v. We refer to $l(v)$ as the label of v. Set $F_a = \emptyset$.*
- *While there exists $v \in V$ such that $l(v) \geq k$ but $l(u) \leq k - 1$ for all descendants of v do*
  1. *Let $u_1, u_2, \ldots, u_d$ denote the children of v ordered such that $l(u_1) \geq l(u_2) \geq \cdots \geq (u_d)$.*
  2. *Determine the minimum i such that $l(v) - \sum_{j=1}^{i} l(u_j) \leq k - 1$ and add $\{\{v, u_1\}.\{v, u_2\}, \ldots, \{v, u_i\}\}$ to F_a.*
  3. *Relabel the nodes in the component of $T - F_a$ containing the root r, i.e.*
  $$l(v) := l(v) - \sum_{j=1}^{i} l(u_j)$$
  and then relabel all ancestors of v.

  end while.
- **Return:** *$\lambda_{wc}^{(k)}(T) = |F_a|$ and minimum edge failure set F_a.*

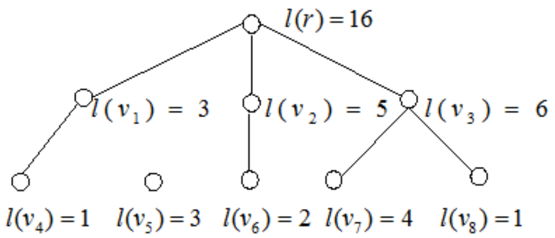
Example 40 *Consider the tree on $n = 9$ nodes with weights on the nodes as shown. We assume that $k = 5$.*



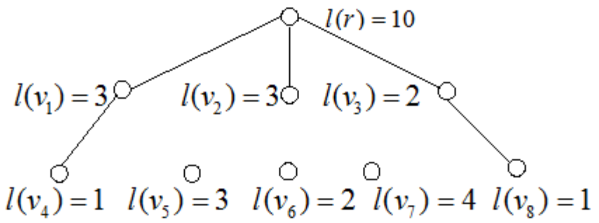
Next we provide the labels.



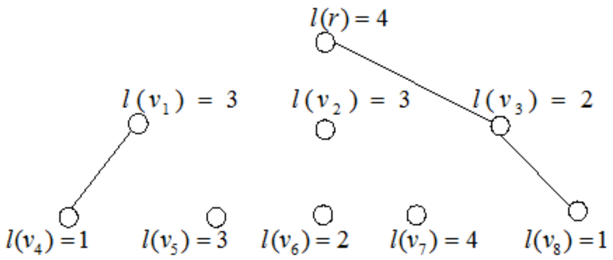
Since $l(v_1) = 6 \geq 5$ and $l(v_5) = 3$ add edge $\{v_1, v_5\}$ to F_a and set $l(v_1) = 3, l(r) = 16$. We indicate the resulting graph with edge $\{v_1, v_5\}$ deleted.



Similarly $l(v_2), l(v_3) \geq 5$, we add edges $\{v_2, v_6\}$ and $\{v_3, v_7\}$ to F_a and relabel $l(v_2) = 3, l(v_3) = 2, l(r) = 10$.



Finally $l(r) = 10 \geq 5$, add edges $\{r, v_1\}$ and $\{r, v_2\}$ to F_a and relabel $l(r) = 4$.



Since $l(v) < 5$ for all $v \in V$, the algorithm is complete. Therefore $\lambda_{wc}^{(5)}(T) = |F_a|$, where $F_a = \{\{v_1, v_5\}, \{v_2, v_6\}, \{v_3, v_6\}, \{r, v_1\}, \{r, v_2\}\}$ is a minimum edge failure set.

As regards an algorithm for a unicycle U, realize that if the total weight of the unique cycle C_m is at least k, then a minimum edge set must contain an edge from C_m so that $\lambda_{wc}^{(k)}$ is just $\min_{x \in E(C_m)} (1 + \lambda_{wc}^{(k)}(U - x))$ and may be computed by m repetitions of the tree algorithm. On the other hand, if the total weight of the cycle C_m is at most $k - 1$, a minimum edge failure set may or may not contain an edge of C_m. If it doesn't then $\lambda_{wc}^{(k)}$ can be computed by applying the tree algorithm to the weighted tree T_C obtained from U by coalescing C_m to a node and assigning it a weight equal to the total weight of C_m. Hence we have the following algorithm.

Algorithm 4 Algorithm for $\lambda_{wc}^{(k)}$ of a Weighted Unicycle U on n Nodes.

- **Input:** *Unicycle U on n nodes with unique cycle C_m on the vertices $u_1, u_2, \ldots, u_m$ and weighting function $w(u) \geq 0$ for $u \in V(U)$; Integer k such that $2 \leq k \leq w(U) = \sum_{u \in V(U)} w(u)$.*

- **Output:** *$\lambda_{wc}^{(k)}(U)$ and an associated minimum failure set F_a.*

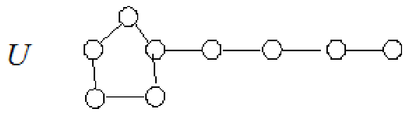
- **Initialization:** *Set $\overline{w} = \sum_{u \in V(C_m)} w(u)$ and $F_a = \emptyset$. Let T_C be the weighted tree obtained from U by coalescing C_m to a node u_C and setting $w(u_C) = \overline{w}$.*

  1. *Using the $\lambda_{wc}^{(k)}(T)$-algorithm on each $U - x$ where $x \in E(C_m)$ determine $\min_{x \in E(C_m)} (\lambda_{wc}^{(k)}(U - x))$ and the edge failure set F'_a for the minimum. If x is the edge deleted from C_m to obtain F'_a, set $F''_a = F'_a \cup \{x\}$.*
  2. *If $\overline{w} \geq k$ set $F_a = F''_a$.*
  3. *If $\overline{w} \leq k - 1$ determine $\lambda_{wc}^{(k)}(T_C)$ using the $\lambda_{wc}^{(k)}(T)$-algorithm and the corresponding minimum weighted failure set F'''_a. If $|F'''_a| \leq |F''_a|$ then set $F_a = F'''_a$ else set $F_a = F''_a$.*

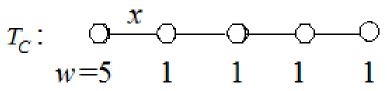
- **Return:** $\lambda_{wc}^{(k)}(U) = |F_a|$ *and minimum edge failure set F_a.*

We present two examples of the algorithm.

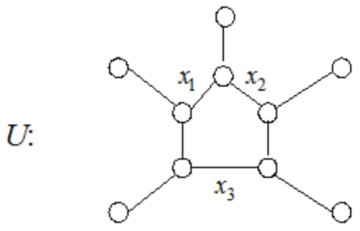
Example 41 *Consider the unicycle U shown below and assign the weight 1 to each node.*



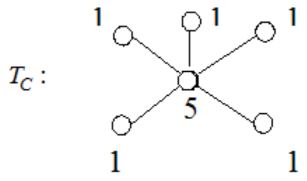
If $k = 6$ *then* $\underset{x \in E(C)}{min} (\lambda_{wc}^{(6)}(U - x) + 1) = 2$, *but* T_C *as shown below has* $\lambda_{wc}^{(6)} = 1$. *So* $\lambda_{wc}^{(6)}(U) = 1$ *and* $F_a = \{x\}$.



Example 42 *Consider the unicycle U as shown and assign weight 1 to each node.*



If $k = 6$ *then* $\underset{x \in E(C)}{min} (\lambda_{wc}^{(6)}(U - x) + 1) = 3$, *but* T_C *as shown below has* $\lambda_{wc}^{(6)} = 5$. *So* $\lambda_{wc}^{(6)}(U) = 3$ *and* $F_a = \{x_1, x_2, x_3\}$.



We conclude this section by describing an efficient procedure to determine $\kappa_c^{(n-\alpha)}(G)$ for an arbitrary graph G having n nodes and e edges and constant α. Recall by Theorem 10, if G is a graph on n nodes and $2 \le k \le n$, then

(i) $\kappa(G) \ge n - k + 1$ implies that $\kappa_c^{(k)} = n - k + 1$ and

(ii) $\kappa(G) \le n - k$ implies that $\kappa \le \kappa_c^{(k)} \le n - k$

Now applying this result to the case where $k = n - \alpha$ we see that

(i′) $\kappa(G) \ge \alpha + 1$ implies that $\kappa_v^{(k)} = \alpha + 1$ and

(ii′) $\kappa(G) \le \alpha$ implies that $\kappa \le \kappa_c^{(k)} \le \alpha$

Because of these facts we determine $\kappa_c^{(n-\alpha)}(G)$ in a brute force manner. Indeed, if $\kappa(G)$ is at least $\alpha +1$ we may conclude that $\kappa_c^{(n-\alpha)}(G) = \alpha + 1$. On the other hand if $\kappa(G)$ is at most α we proceed as follows: For each subset S of $\alpha - 1$ nodes we determine the order of the components of $G - S$. If for all S there exists a component of $G - S$ of order at least $n - \alpha$ we declare $\kappa_c^{(n-\alpha)}(G)$ to be α. Otherwise, we consider each subset S of order $\alpha - 2$. If for all S there exists a component of $G - S$ of order at least $n - \alpha$ we declare $\kappa_c^{(n-\alpha)}(G)$ to be $\alpha - 1$. Continuing on in this fashion it is clear that if γ is the largest integer, $k - 1 \le \gamma \le \alpha - 1$ such that a subset of nodes S exists such that $|S| = \gamma$ and $G - S$ has a component of order at least $n - \alpha$ then $\kappa_c^{(n-\alpha)}(G) = \gamma + 1$.

6 Neighbor Component Order Connectivity

In this final section we introduce a new scenario currently being studied along with some preliminary results. Consider a network modeled by a (simple) graph $G = (V, E)$ having $n = |V|$ nodes and $e = |E|$ edges. With this model, nodes fail and the nodes adjacent to failed nodes are subverted and therefore considered inoperable. All edges connected to failed and subverted nodes are also subverted. We note that the subversion of a node does not preclude it from failing. Thus the state of the associated system consists of those nodes which are operating, i.e. those that have neither failed nor are subverted. For convenience, we will also refer to the subgraph of G induced by the operating nodes as a state of the system. Observe that if W is the set of nodes that have failed, then this subgraph is precisely $G-[W]$, where $G[W]$ is the closed neighborhood of W. Following the terminology of Gunther and Hartnell [19] a state is a neighbor-failure state if it is either empty or nonempty and disconnected. The minimum number of nodes whose failure along with the subversion of neighbors results in either a neighbor-failure state or a complete subgraph is called the neighbor-connectivity of the graph and is denoted $\kappa_{nc}(G)$ or simply κ_{nc}.

Gunther, Hartnell and Nowkowski proved the following result.

Theorem 43 *[20]: For a graph G,* $\kappa_{nc}(G) \le \kappa_{nc}(G)$.

Now combining the notions of neighbor-connectivity and component order connectivity, the vulnerability parameter $\kappa_{nc}^{(k)}(G)$ neighbor-component-order connectivity is defined as follows.

Definition 44 *[21]: Let G =*(E, V) *be a graph having* $n = |V|$ *nodes and* $e = |E|$ *edges and assume* $2 \le k \le n$ *. Nodes fail but edges do not, when a node fails all adjacent nodes are subverted and are considered inoperable. A* **state** *is the set of operating nodes* *at a snapshot in time. A state is a* **neighbor operating state** *if the operating nodes induce a subgraph with at least one component of order at least k. Otherwise, it is a* **neighbor failure state***, i.e. all components of the induced subgraph have order at most $k-1$ and, the associated set of failed nodes is called a* **neighbor failure set***. The k-neighbor component connectivity or neighbor component order connectivity of G, denoted by $\kappa_{nc}(G)$ or simply κ_{nc}, is the minimum cardinality of a neighbor failure state.*

Note that for convenience, we will also refer to the subgraph induced by the operating nodes as a state. If W is the set of nodes which have failed then the state is precisely the subgraph $G-[W]$.

We give some examples for this new parameter.

Example 45 *Assume $2 \le k \le n$.*

a) *Let $G = P_n$, the path on n nodes. Starting from an end node, number the nodes from 1 to $k+2$. Then repeat numbering until all of the nodes are labeled. Next, delete the closed neighborhoods of nodes labeled $k+1$. (Note, this will also remove nodes labeled k and $k+2$ as they are neighbors of node $k+1$). Continue to remove closed neighborhoods from the path in this manner. The surviving subgraph consists of a disconnected graph with components of order $k-1$ and one remainder component of order $\le k$. If the remainder component has order k, remove one more closed neighborhood from that component. Given n and k, it follows by the division algorithm, n = $\left\lfloor \frac{n}{k+2} \right\rfloor$(k+2)+r where $0 \le r \le k+1$ and*

$$\kappa_{nc}^{(k)}(P_n) = \begin{cases} \left\lfloor \frac{n}{k+2} \right\rfloor + 1, r = k, k+1 \\ \left\lfloor \frac{n}{k+2} \right\rfloor, 0 \le r \le k-1 \end{cases}.$$

b) *Let $G = C_n$, the cycle on n nodes. To find a minimum neighbor-failure set, let u be an arbitrary node. The deletion of u and the subversion of its two neighbors leaves a surviving subgraph which is a path of length $n-3$. Applying the procedure in a) to P_{n-3} and the division algorithm n = $\left\lfloor \frac{n-3}{k+2} \right\rfloor$(k+2)+r where $0 \le r \le k+1$, we get*

$$\kappa_{nc}^{(k)}(C_n) = \begin{cases} \left\lfloor \frac{n-3}{k+2} \right\rfloor + 2, r = k, k+1 \\ \left\lfloor \frac{n-3}{k+2} \right\rfloor + 1, 0 \le r \le k-1 \end{cases}.$$

c) *Let $G = K_{1,n-1}$, the complete bipartite graph with one part of order 1. Removal of the center node subverts all the other nodes, thus $\kappa_{nc}^{(k)}(K_{1,n-1}) = 1$.*

d) *Let $G = K_n$, the complete graph on n nodes. The deletion of any node subverts all the other nodes, thus $\kappa_{nc}^{(k)}(K_n) = 1$.*

e) *Let $G = K_{p,q}$, the complete bipartite graph on $p + q = n$ nodes with $p \le q$. The deletion of any node from one part subverts all the nodes from the other part and isolates the remaining nodes, thus $\kappa_{nc}^{(k)}(K_{p,q}) = 1$.*

We now state some properties of $\kappa_{nc}^{(k)}$. If W is a minimum cardinality failure set for $\kappa_c^{(k)}$, i.e. all components of $G-W$ have order at most $k-1$, then clearly W is a neighbor failure state as well. Thus we have the following theorem.

Theorem 46 *[21]: Let G be a graph on n nodes and e edges, then $\kappa_{nc}^{(k)} \le \kappa_c^{(k)}$.*

By Proposition 5 It is known that if $H \subset G$ then $\kappa_c^{(k)}(H) \le \kappa_c^{(k)}(G)$. However, for neighbor-connectivity and neighbor-component order connectivity, this inequality is not necessarily true. We demonstrate this in the following example.

Example 47 *Consider the wheel on 9 nodes, denoted W_9 and its induced subgraph the cycle C_8, see Figure 10. From Example 45 b) we know that $\kappa_{nc}^{(k)}(C_8) = 2$ for $2 \le k \le 5$. But $\kappa_{nc}^{(k)}(W_9) = 1$ for all k since the central node is a failure set.*

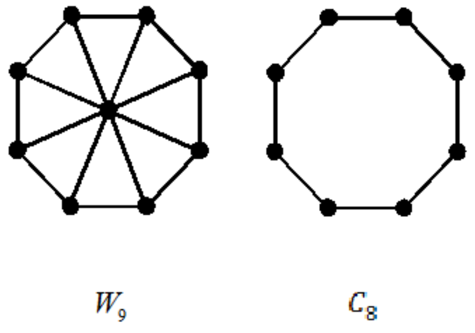


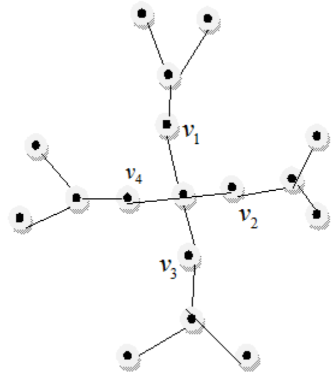
Figure 10

Remark 48 *$\kappa_{nc}^{(k)}(H)$ may be larger than $\kappa_{nc}^{(k)}(G)$ even if H is a spanning subgraph of G.*

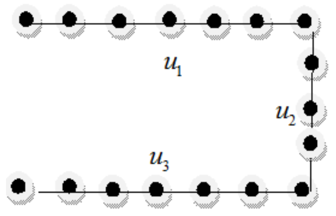
Our next results focus on obtaining a realizability result for trees. Recall by Proposition 2.48 (1), for any tree T on n nodes $\kappa_c^{(k)}(T) \le \kappa_c^{(k)}(P_n)$. As the next example shows, $\kappa_{nc}^{(k)}$ of an arbitrary tree can be larger than that of the path.

Example 49 *: Consider the tree T_{17} on 17 nodes with indicated specific nodes as depicted in Figure 11(a) and P_{17} with indicated nodes as depicted in Figure 11(b). Let $k = 3$. By inspection we see that $W_a = \{v_1, v_2, v_3, v_4\}$ and $W_b = \{u_1, u_2, u_3\}$ are minimum neighbor-failure sets of T_{17} and P_{17}, respectively. Therefore*

$$\kappa_{nc}^{(3)}(T_{17}) = |W_b| = 3 < 4 = |W_a| = \kappa_{nc}^{(3)}(T_{17}).$$



(a)



(b)

Figure 11

Thus if T is an arbitrary tree on n nodes $\kappa_{nc}^{k}(T)$ may not be less than $\kappa_{nc}^{k}(P_n)$. Now by Theorem 43, $\kappa_{nc}^{(k)}(T) \le \kappa_c(k)(T)$ and by Proposition 8 (1), $\kappa_c(k)(T) \le \kappa_c(k)(P_n)$. We conclude this section with the following result.

Theorem 50 *[21]: Let T be a tree on n nodes. Then for any value k, $2 \le k \le n$,*

$$1 = \kappa_c^{(k)}(K_{1,n-1}) \le \kappa_c^{(k)}(T) \le \kappa_c^{(k)}(P_n) = \left\lfloor \frac{n}{k} \right\rfloor.$$

An algorithm to compute $\kappa_{nc}^{(k)}$ for an arbitrary tree exists and will be found in the thesis of K. Luttrell which is in preparation.

References:

[1] D. Gross, L. Kazmierczak, J. T. Saccoman, C. Suffel, A. Suhartomo, On Component Order Edge Connectivity of a Complete Bipartite Graph, *ARS Combinatoria* accepted to appear.

[2] F. Boesch, D. Gross, C. Suffel, Component order connectivity graph invariant related to operating component reliability, *Combinatorics, graph theory, and algorithms* Vol. I, II, pp. 109–116, New Issues Press, Kalamazoo, MI, 1999.

[3] F. Boesch, D. Gross, L.W. Kazmierczak, C. Suffel, A. Suhartomo, Component order edge connectivity an introduction, *Congr. Numer.* 178 (2006), pp. 7–14.

[4] G.Chartrand, L.Lesniak and Ping Zhang, *Graphs and Digraphs*,Chapman/Hall/CRC, Boca Raton, 2011.

[5] F. Boesch, D. Gross, C. Suffel, Component order connectivity, *Congr. Numer.* 131 (1998), pp. 145–155.

[6] L.W. Kazmierczak, F. Boesch, D. Gross, C. Suffel, Realizability results involving two connectivity parameters, *Ars Combin.* 82 (2007), pp. 181–191.

[7] F. Harary, The Maximum Connectivity of a Graph, *Proceedings of the National Academy of Sciences USA* 48 (1962), pp. 1142–1146.

[8] L.W. Kazmierczak, F. Boesch, D. Gross, C. Suffel, Forbidden subgraph conditions on the complement of a graph that insure a strong network design, *Congr. Numer.* 161 (2003), pp. 65–74

[9] L.W. Kazmierczak, On the relationship between connectivity and component order connectivity.Thesis (Ph.D.) Stevens Institute of Technology. 2003.

[10] F, Boesch; D. Gross; L. Kazmierczak; J. Stiles; C. Suffel, On Extensions of Turans Theorem, *Graph Theory Notes XL - Proceedings of the New York Academy of sciences* Graph Theory Day (2001), pp. 42–45.

[11] P. Ballister, B. Bollobas, private communication.

[12] P. Erdos, On a Theorem of Rademacher-Turan, *Illinois J. Math.*, 6 (1962), pp. 122–127.

[13] F. Boesch, D. Gross, C. Suffel, Component order connectivity: a graph invariant related to multiprocessor network reliability. *Graph Theory Notes - Proceedings of the New York Academy of sciences* (1997),pp. 45–50

[14] On component order connectivity: κ_c^2, Stevens Institute of Technology Technical Report, to appear.

[15] G. Chartrand, A graph-theoretic approach to a communication problem, *SIAM J. Appl Math* 5 (1966), pp. 778–781.

[16] Boesch, Frank; Gross, Daniel; Saccoman, John T.; Kazmierczak, L. William; Suffel, Charles; Suhartomo, Antonius A generalization of an edge-connectivity theorem of Chartrand, *Networks* 54 (2009), no. 2, pp. 82–89.

[17] Algorithms for Determining Component Order Node and Edge Connectivities, Stevens Institue of Technology Technical report, to appear.

[18] L. Isawara, D. Gross, L. Kazmierczak, J. T. Saccoman, A. Suhartomo, C. Suffel, On Weighted ComponentEdge Connectivity of Trees and Unicycles, *Congr. Numer.* 206 (2010), pp. 85–97.

[19] G. Gunther. Neighbour-connectivity in regular graphs, *Discrete Appl. Math*, 11 (1985), pp. 233–243.

[20] G. Gunther, B. Hartnell, R. Nowakowski. Neighbour-connected graphs and projective planes, *Networks*, 17 (1987), pp. 241–247.

[21] K. Luttrell, L. Iswara, L.W. Kazmierczak, C. Suffel, D. Gross, J.T. Saccoman. The relationship between neighbor-connectivity, component order connectivity and neighbor-component order connectivity. *Congressus Numeratium*, accepted - to appear.