A Note on Network Vulnerability

My T. Thai

Abstract Assessing network vulnerability is a central research topic to understand networks structures, thus providing an efficient way to protect them from attacks and other disruptive events. Existing vulnerability assessments mainly focus on investigating the inhomogeneous properties of graph elements, node degree for example, however, these measures and the corresponding heuristic solutions can provide neither an accurate evaluation over general network topologies, nor performance guarantees to large scale networks. To this end, we discuss two new optimization models to quantify the network vulnerability, which aim to discover the set of key node/edge disruptors, whose removal results in the maximum decline of the global pairwise connectivity as well as degrade the network’s quality of service (QoS). This note may contain some errors; therefore, it is not for any references.

Key words: Network Vulnerability, Pairwise Connectivity, Approximation Algorithms, Critical Node Detection

1 Introduction

Complex network systems such as the Internet, WWW, MANETs, and the power grids, are often greatly affected by several uncertain factors, including external natural or man-made interferences (e.g., severe weather, enemy attacks, malicious attacks.) Additionally, they are extremely vulnerable to attacks, that is, the failures of a few critical nodes (links) which play a vital role in maintaining the network’s functionality can break down its operation [1].

Dept. of Comp. & Info. Sci. & Eng., University of Florida, Gainesville, FL 32611, e-mail: mythai@cise.ufl.edu
In order to assess the network vulnerability, we need to address several fundamental questions such as: How do we quantitatively measure the vulnerability degree of a network? What are the key nodes (links) that play a vital role in maintaining the network’s functionality? What is the minimum number of nodes (links) do we need to take down in order to break down a network, i.e. an adversarial network?

In this note, we study the following two problems:

**Problem 1. β−Vertex (Edge) Disruptor [β−VD (β−ED)]:** Given a directed graph $G = (V, E)$ (representing a network’s topology) and $0 \leq \beta < 1$, find a subset of vertices (edges) $S \subseteq V$ ($S \subseteq E$) with the minimum cardinality so that the total pairwise connectivity of $G[V \setminus S]$ ($G[E \setminus S]$) is at most $\beta \binom{n}{2}$, where the total pairwise connectivity in $G$ is defined as the total number of connected pairs of vertices in $G$. A pair of nodes is connected if there is a path between them in graph $G$.

Note that $E$ is a set of links where links can be physical links or logical links between two endpoints. For example, if there is a communication between nodes $u$ and $v$ in the network, there will be an edge $(u, v)$ in the representing graph $G$. If we use $G$ to represent the functional dependency between each node in a network, then there is an edge $(u, v)$ in $G$ iff there is a functional relationship between $u$ and $v$ in the network. Therefore, the connectivity discussed in this chapter is not simply a physical connectivity in a given network.

Clearly, this model reveals the vulnerability degree of the networks. That is, the more key nodes (edges) there are (i.e., the more nodes (edges) whose deletion is required to meet the requirement of pairwise connectivity), the less vulnerability the network is. Conversely, the fewer the key nodes (edges), the more vulnerability the network will be to the attacks. This model can be used in the study of breaking down an adversarial social network to a certain degree with the minimum cost. Moreover, the model presents a deeper meaning and greater potentials on the study of multiple disruption levels (different values of $\beta$). Several recent studies in the context of wireless networks have aimed to discover the nodes/edges whose removal disconnects the network, regardless of how disconnected it is [2, 3, 4]. However, it is not reasonable to limit the possible disruption to only disconnecting the graph, ignoring how fragmented it is since the giant connected component still exists and the network may function well. For example, a scale-free network can tolerate high random failure rates [1], since the destructions to boundary vertices may not significantly decline the network connectivity even though the whole graph becomes disconnected. In addition, different disruption levels may require different sets of disruptors on which these two models can differentiate whereas existing methods cannot. For example, the node centrality method always returns a set of nodes with non-increasing degrees regardless of the disruption level.
Another aspect is QoS-aware topology vulnerability, which is critical for the Internet. As the Internet serves as the main carrier of more and more real-time applications, it has to satisfy several QoS measures with predefined thresholds, which include jitter, delay, bandwidth, packet loss and etc. Plenty of QoS routing protocols, e.g., Q-OSPF and PNNI have been developed to meet these requirements [5]. In practical networks, malfunctions often take place at intermediate network nodes/links for routing, consequently, even the optimal routing path from source to destination can satisfy few of the QoS constraints. In this cases, only improvements over routing protocols cannot enhance the robustness in unreliable network environments. Therefore, we are interested in study how many node/link failures are required to break down the network to such an extent. Notice that even the subproblem of this study, detecting an optimal routing path satisfying a set of QoS constraints, is nontrivial.

In practical applications, the constraints that are satisfied by the QoS optimal routing path can be categorized into additive and non-additive ones. Specifically, jitter, delay and packet loss of a routing path are the sum of each metric over all the links belonging to this path. However, constraints like bandwidth are not additive from edge to edge, but min/max or multiplicative functions. In principle, multiplicative measures can be converted into additive ones in a logarithmic manner and min/max non-additive measures can be satisfied by ruling out all the unsatisfied single links. Therefore, classic theoretical studies over the QoS routing are normally formulated as a multi-additive-constraint path (MCP) problem [6][7][8]: Consider a network $G(V, E, s, t)$ with designated source node $s$, destination node $t$, and $m$ additive constraint $(c_1, \ldots, c_m)$, where each edge $(u, v) \in E$ has $m$ additive weights $w_i(u, v) \geq 0, i \in [1, m]$. Find a path $P$ from $s$ to $t$ with

$$w_i(P) \triangleq \sum_{(u, v) \in P} w_i(u, v) \leq C_i$$

for all $i \in [1, m]$, if it exists.

The purpose of vulnerability assessment is to discover the weakness of the object network topology, whose result can be applied to optimizing network topology design, enhancing network robustness or destroying terrorist networks. We refer to these weak nodes/links as critical nodes/links, specifically, the minimum set of nodes/links whose failure can bring down the network QoS to a certain low level are called QoS critical node/link set. Therefore, given two networks of the same size, the one which has a smaller QoS critical node/link set is of course more vulnerable. In this paper, we measure the network QoS by the optimal QoS source-destination routing path, which satisfy the most QoS constraints over all routing paths. By requiring how much such an optimal path satisfies the multiple QoS constraints, we put a threshold on the network QoS and discover the QoS critical node/link set correspondingly.
Problem 2. QoS-Critical Vertices (QoSCV) / QoS-Critical Edges (QoSCE):
Given a directed graph $G(V, E, s, t)$ with $m$-dim edge weight vector $(u, v) \in E: (w_1(u, v), w_2(u, v), \ldots, w_m(u, v))$. The weight vector for each $s-t$ path $P$ is defined as $(w_1(P), w_2(P), \ldots, w_m(P))$ where $w_i(P) = \sum_{(u, v) \in P} w_i(u, v)$ for all $i \in [1, \ldots, m]$. Given a constraint threshold vector $(c_1, c_2, \ldots, c_m)$ with corresponding credit vector $(\lambda_1, \lambda_2, \ldots, \lambda_m)$, an $s-t$ path $P$ satisfies the $i$th constraint (denoted as $p \propto i$) if $w_i(P) \leq c_i$, and an SAT score $\phi(P)$ is defined as $\phi(P) = \sum_{j: P \propto i} \lambda_j$. The SAT score for the graph $G$ is $\phi(G) = \max_{P \in G} \phi(P)$, i.e. the maximum score among all $s-t$ paths.

The QoSCV/QoSCE problem is to find a minimum set $S$ of edges/vertices such that $\phi(G \setminus S) \leq \rho$ for a given score threshold $\rho$. The solution edges/vertices are referred as QoS critical edges/vertices respectively. Notice that QoSCV can be readily converted into QoSCE through a classic technique \cite{??}, we only present solutions for QoSCE.

In the following sections, we present the hardness complexity and solutions for the above 2 main problems.

2 Hardness Results

2.1 NP-completeness of $\beta$-Edge Disruptor

We use a reduction from the balanced cut problem.

Definition 1. A cut $\langle S, V \setminus S \rangle$ corresponding to a subset $S \in V$ in $G$ is the set of edges with exactly one endpoint in $S$. The cost of a cut is the sum of its edges’ costs (or simply its cardinality in the case all edges have unit costs). We often denote $V \setminus S$ by $\bar{S}$.

Finding a min cut in the graph is polynomial solvable \cite{9}. However, if one asks for a somewhat “balanced” cut of minimum size, the problem becomes intractable. A balanced cut is defined as following:

Definition 2. (Balanced cut) Let $f$ be a function from the positive integers to the positive reals. An $f$-balanced cut of a graph $G = (V, E)$ asks us to find a cut $\langle S, \bar{S} \rangle$ with the minimum size such that $\min\{|S|, |\bar{S}|\} \geq f(|V|)$.

Abusing notations, for $0 < c \leq \frac{1}{2}$, we also use $c$-balanced cut to find the cut $\langle S, \bar{S} \rangle$ with the minimum size such that $\min\{|S|, |\bar{S}|\} \geq c|V|$. We will use in our proofs the following results on balanced cut shown in \cite{10}:

Corollary 1. (Monotony) Let $g$ be a function with

$$0 \leq g(n) - g(n - 1) \leq 1$$

Then $f(n) \leq g(n)$ for all $n$, implies $f$-balanced cut is polynomially reducible to $g$-balanced cut.
Corollary 2. (Upper bound) $\alpha \epsilon$-balanced cut is NP-complete for $\alpha, \epsilon > 0$.

It follows from Corollaries 1 and 2 that for every $f = O(\alpha n)$ $f$-balanced cut is NP-complete. We are ready to prove the NP-completeness of $\beta$-edge disruptor:

Theorem 1. $\beta$-Edge Disruptor is NP-complete.

Proof. We prove Theorem 1 for a special case when $\beta = \frac{1}{2}$. For other values of $\beta$ the proof can go through with a slight modification of the reduction. We consider $n$ to be a large enough number in our proof, say $n > 10^3$.

Consider the decision version of the problem that asks whether an undirected graph $G = (V, E)$ contains a $\frac{1}{2}$-edge disruptor of a specified size:

$$\frac{1}{2}\text{-ED} = \{ (G, K) \mid G \text{ has a } \frac{1}{2} \text{-edge disruptor of size } K \}$$

To show that $\frac{1}{2}\text{-ED}$ is in NP-complete we must show that it is in NP and that all NP-problems are polynomial time reducible to it. The first part is easy; given a candidate subset of edges, we can easily check in polynomial time if it is a $\frac{1}{2}$-edge disruptor of size $K$. To prove the second part, we show that $f$-balanced cut is polynomial time reducible to $\frac{1}{2}\text{-ED}$ where $f = \lceil \frac{n-\sqrt{2|V|/n}+n}{2} \rceil$.

Let $G = (V, E)$ be a graph in which one seeks to find an $f$-balanced cut of size $k$. Now construct the graph $H(V_H, E_H)$ as follows: $V_H = V' \cup C_1 \cup C_2$ where $V' = \{ v_i \mid i \in V \}$ and $C_1, C_2$ are two cliques of size $\lceil \frac{n^2}{4} \rceil$. The total number of nodes in $H$ is, hence, $N = 2\lceil \frac{n^2}{4} \rceil + n$. Beside edges inside two cliques, we add an edge $(v_i, v_j)$ for each edge $(i, j) \in E$. We next connect each vertex $v_i$ to $\lceil \frac{n^2}{4} \rceil + 1$ vertices in $C_1$ and $\lceil \frac{n^2}{4} \rceil + 1$ vertices in $C_2$ so that the degree difference of nodes in the cliques are at most one. We illustrate the construction of $H(V_H, E_H)$ in Figure 1. We show that there is a $f$-balanced cut of size $k$ in $G$ if $H$ has an $\frac{1}{2}$-edge disruptor of size $K = n\lceil \frac{n^2}{4} \rceil + 1 \rceil + k$ where $0 \leq k \leq \lceil \frac{n^2}{4} \rceil$. Note that the cost of any cut $(S, V \setminus S)$ in $G$ is at most $|S||V \setminus S| \leq \lceil \frac{|S|+|V\setminus S|}{4} \rceil = \lceil \frac{n^2}{4} \rceil$. 

![Figure 1](attachment:image1.png)
On the one hand, an \( f \)-balanced cut \( \langle S, S' \rangle \) of size \( k \) in \( G \) induces a cut \( \langle C_1 \cup S, C_2 \cup S' \rangle \) with size exactly \( n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) + k \). If we select the cut as the disruptor, the pairwise connectivity will be at most \( \frac{1}{2} \binom{n}{2} \).

On the other hand, assume that \( H \) has a \( \frac{1}{2} \)-edge disruptor of size \( K = n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) + k \). Because separating \( n \) nodes in a clique requires cutting at least \( n \left( \left\lfloor \frac{n^2}{4} \right\rfloor - n \right) > n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) + k \) edges, there are at most \( n \) nodes separated from both the cliques. A direct consequence is that at least \( \left( \left\lfloor \frac{n^2}{4} \right\rfloor - n \right) \) nodes remain connected in each clique after removing edges in the disruptor. Denote the sets of those nodes by \( C_1' \) and \( C_2' \) respectively. \( C_1' \) and \( C_2' \) cannot be connected otherwise the pairwise connectivity will exceed \( \frac{1}{2} \binom{n}{2} \). Denote by \( X_1, X_2 \) the set of nodes in \( V' \) that are connected to \( C_1' \) and \( C_2' \) respectively. Since, \( C_1' \) and \( C_2' \) are disconnected we must have \( X_1 \cap X_2 = \emptyset \).

We will modify the disruptor without increasing its size and the pairwise connectivity such that no nodes in the the cliques are split. For each \( u \in C_1 \setminus C_1' \), we remove from the disruptor all edges connecting \( u \) to \( C_1' \) and add to the disruptor all edges connecting \( u \) to \( X_2 \). This will move \( u \) back to the connected component that contains \( C_1' \) while reducing the size of the disruptor at least \( \left( \left\lfloor \frac{n^2}{4} \right\rfloor - n \right) - n - \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) + n > 0 \). In addition, the pairwise connectivity will not increase when we connect \( u \) to \( C_1' \) and disconnect \( v \) from \( C_1' \). If no nodes are left in \( X_1 \), we can select \( v \in X_2 \) (as in that case \( |C_2' \cup X_2| > |C_1' \cup X_1| \)) that makes sure the pairwise connectivity will not be increased. We repeat the same process for every node in \( C_2 \setminus C_2' \) and at the end of the process, \( C_1' = C_1 \) and \( C_2' = C_2 \).

We will prove that \( X_1 \cup X_2 = V' \) i.e. \( \langle X_1, X_2 \rangle \) induces a cut in \( V(G) \). Assume not, the cost to separate \( C_1 \cup X_1 \) from \( C_2 \cup X_2 \) will be at least \( n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right)(|V' - X_1| + |V' - X_2|) = n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right)(2n - |X_1| - |X_2|) \geq n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) + k \) that is a contradiction.

Since \( X_1 \cup X_2 = V' \), the disruptor induces a cut in \( G \). To have the pairwise connectivity at most \( \frac{1}{2} \binom{n}{2} \), both \( (C_1 \cup X_1) \) and \( (C_2 \cup X_2) \) must have size at least \( \frac{N - \sqrt{N}}{4} \). If follows that \( X_1 \) and \( X_2 \) must have size at least \( f(n) = \left\lfloor \frac{n \sqrt{2n} + n}{2} \right\rfloor \). The cost of the cut induced by \( \langle X_1, X_2 \rangle \) in \( G \) will be \( n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) + k - n \left( \left\lfloor \frac{n^2}{4} \right\rfloor + 1 \right) = k \). \( \square \)


2.2 Hardness of $\beta$-Vertex Disruptor

**Theorem 2.** $\beta$-Vertex Disruptor is NP-complete.

**Proof.** We ignore the details and present here the sketch of the proof. We show that vertex cover is polynomial time reducible to $\beta$-vertex disruptor. Let $G = (V, E)$ be a graph in which one seeks to find a vertex cover of size $k$. Note that if we remove nodes in a vertex cover from the graph, the pairwise connectivity in the graph will be zero. Hence, by setting $\beta = 0$, $G$ has a vertex cover of size $k$ iff $G$ has an $\beta$-vertex disruptor of size $k$.

One can also avoid using $\beta = 0$ by replacing each vertex in $G$ by a clique of large enough sizes, say $O(n)$, that ensures no vertices in cliques will be selected in the $\beta$-vertex disruptor.

Given the NP-hardness of the $\beta$-vertex disruptor, the best possible result is a polynomial time approximation scheme (PTAS) that given a parameter $\epsilon > 0$, produces a $(1 + \epsilon)$-approximation solution in polynomial time. Unfortunately, this is impossible unless $P = NP$.

**Theorem 3.** Unless $P = NP$, $\beta$-vertex disruptor has no polynomial time approximation scheme.

**Proof.** In the case $\beta = 0$, the problem is equivalent to finding the minimum vertex cover in the graph. In [11], Dinur and Safra showed that approximating vertex cover within constant factor less than 1.36 is NP-hard. Hence, a PTAS scheme for $\beta$-vertex disruptor does not exist unless $P = NP$. $\square$

2.3 Hardness of QoSCE

QoSCE problem is quite challenging and its complexity class is still an open issue. We find that it does not belong to NP class through the following discussions.

Given an edge subset $S$ as a certificate to QoSCE problem on graph $G$, the process of verifying this certificate equals to another decision problem QoS-SP on the remaining graph $G \setminus S$: does there exist a path $P \in G \setminus S$ satisfying $\phi(P) \leq \rho$? This problem is NP-Complete.

**Lemma 1.** QoS-SP is NP-Complete.

**Proof.** This proof is straightforwardly by reduction from MCP[7] problem by letting $\rho = \sum_{i=1}^{m} \lambda_i$. Then if there exists a solution for the QoS-SP problem, then the path is a feasible path to MCP, otherwise, MCP has no solution. Since MCP is NP-hard, the proof completes. $\square$

Therefore, the certificate of QoSCE is not verifiable in polynomial time, thus not in the class of NP. So it is quite challenging to be tackled.
3 Approximation Algorithm for $\beta$-edge Disruptor

In this section, we present an $O(\log^2 n)$ pseudo-approximation algorithm for the $\beta$-edge disruptor problem in directed graphs. (Note that in directed graphs, a pair $(u,v)$ is said connected if there exist a directed path from $u$ to $v$ and vice versa, from $v$ to $u$.) Formally, our algorithm finds in a directed graph $G$ a $\beta'$-edge disruptor whose the cost is at most $O(\log^2 n)\text{OPT}_{\beta-ED}$, where $\frac{\beta'}{2} < \beta < \beta'$ and $\text{OPT}_{\beta-ED}$ is the cost of an optimal $\beta$-edge disruptor.

As shown in Algorithm 1, the approximation solution consists of two main steps. First, we construct a decomposition tree of $G$ by recursively partitioning the graph into two halves with a directed $c$-balanced cut. Second, we solve the problem on the obtained tree using a dynamic programming algorithm and transform this solution to the original graph. These two main steps are explained in the next two sections.

3.1 Balanced Tree-Decomposition

Let us first introduce some definitions before describing the balanced tree-decomposition algorithm.

Definition 3. Given a directed graph $G = (V,E)$ and a subset of vertices $S \subset V$. We denote the set of edges outgoing from $S$ by $\delta^+(S)$; the set of edges incoming to $S$ by $\delta^-(S)$. A cut $(S,V \setminus S)$ in $G$ is defined as $\delta^+(S)$. A $c$-balanced cut is a cut $(S,V \setminus S)$ s.t. $\min\{|S|,|V \setminus S|\} \geq c|V|$. The directed $c$-balanced cut problem is to find the minimum $c$-balanced cut. And finally, we denote $P(G)$ as the total pairwise connectivity in $G$.

Note that a cut $(S,V \setminus S)$ separate pairs $(u,v) \in S \times (V \setminus S)$ so that there is no path $u$ to $v$, that is, there is no strongly connected component (SCC) containing vertices both in $S$ and $V \setminus S$.

Our tree decomposition procedure is as follows. As shown in Algorithm 1 (lines 3 to 13), we start with a tree $T = (V_T,E_T)$ containing only one root node $t_0$. We associate $t_0$ with the vertex set $V$ of $G$, that is, $V(t_0) = V(G)$. For each node $t_i \in V_T$ whose $V(t_i)$ contains more than one vertex and $V(t_i)$ has not been partitioned, we partition the subgraph $G[V(t_i)]$ induced by $V(t_i)$ in $G$ using a $c$-balanced cut algorithm [12] where constant $c = 1 - \sqrt{\frac{3}{\beta'}}$. For each $c$-balanced cut on $G[V(t_i)]$, we create two children nodes $t_{i1}$ and $t_{i2}$ of $t_i$ in $V_T$ corresponding to two sets of vertices returning by the cut. Finally, we assign each node $t_i$ a cost $\text{cost}(t_i)$ which is equal to the cost of the cut performed on $G[V(t_i)]$. This procedure continues until $V(t_i)$ contains only a single vertex. That is, the leaves of $T$ represent nodes in $G$. Therefore, $T$ has $n$ leaves and $n - 1$ internal nodes where each internal node in $T$ represents a subset of nodes in $V$. 
We define the root node $t_0$ to be on level 1. If a node is on level $l$, all its children are defined to be on level $l+1$. Note that all collections of vertices corresponding to nodes in a same level forms a partition in $V$.

**Lemma 2.** The height of $T$ obtained in the above balanced tree decomposition procedure is at most $O(-\log_{1-c'} n)$

**Proof.** Note that the directed $c$-balanced cut algorithm [12] finds in polynomial time a $c'$-balanced cut within a factor of $O(\sqrt{\log n})$ from the the optimal $c$-balanced cut for $c' = \alpha c$ and fixed constant $\alpha$. Thus when separating $G[V(t_i)]$ using $c$-balanced cut, the size of the larger part is at most $(1 - c')|V(t_i)|$. By induction method, we can show that if a node $t_i$ is on level $l$, the size of the corresponding collection $V(t_i)$ is at most $|V| \times (1 - c')^{l-1}$. It follows that the tree’s height is at most $O(-\log_{1-c'} n)$  

### 3.2 Pseudo-Approximation Algorithm and Analysis

**Algorithm 1** $\beta$-edge Disruptor

1. **Input:** A directed graph $G = (V, E)$ and $0 \leq \beta < \beta' < 1$
2. **Output:** A $\beta'$-edge disruptor of $G$.
   
   {Construct the decomposition tree}

3. $c \leftarrow 1 - \sqrt{\frac{\beta}{n}}$
4. $T(V_T, E_T) \leftarrow (\{t_0\}, \phi), V(t_0) \leftarrow V(G), l(t_0) = 1$
5. **while** $\exists$ unvisited $t_i$ with $|V(t_i)| \geq 2$ **do**
6. Mark $t_i$ visited, create new child nodes $t_{i1}, t_{i2}$ of $t_i$
7. $l(t_{i1}), l(t_{i2}) \leftarrow l(t_i) + 1$
8. $V_T \leftarrow V_T \cup \{t_{i1}, t_{i2}\}$
9. $E_T \leftarrow E_T \cup \{(t_i, t_{i1}), (t_i, t_{i2})\}$
10. Separate $G[V(t_i)]$ into two using directed $c$-balanced cut
11. Assign two obtained partitions to $V(t_{i1}), V(t_{i2})$
12. $cost(t_i) \leftarrow$ The cost of the balanced cut
13. **end while**
   
   {Find the minimum cost $G$-partition}

14. **for** $t_i \in T$ in reversed BFS order from root node $t_0$ **do**
15. **for** $p \leftarrow 0$ to $\beta'(\frac{n}{2})$ **do**
16. if $P(G[V(t_i)]) \leq p$ then
17. $cost(t_i, p) \leftarrow 0$
18. else
19. $cost(t_i, p) \leftarrow \min\{cost(t_{i1}, \pi) + cost(t_{i2}, p - \pi) + cost(t_i) \mid \pi \leq p\}$
20. **end if**
21. **end for**
22. **end for**
23. Find $F$ with $P(F) = \min\{cost(t_0, p) \mid p \leq \beta'(\frac{n}{2})\}$
24. Return union of cuts used at $A(F)$ during tree construction
In this section, we present the second main step which uses the dynamic programming to search for the right set of nodes in $T$ such that the cuts to partition those corresponding sets of vertices in $G$ have the minimum cost and the obtained pairwise connectivity is at most $\beta'(\frac{n}{2})$ where $\beta < \beta' < 1$. The details of this step are shown in Algorithm 1 (lines 14 to 23).

Denote a set $F = \{t_1, t_2, \ldots, t_k\} \subset V_T$ such that $V(t_1), V(t_2), \ldots, V(t_k)$ is a partition of $V(G)$ i.e. $V(G) = \bigcup_{i=1}^{k} V(t_i)$ and for any pair $t_i$ and $t_j \in F$, $V(t_i) \cap V(t_j) = \emptyset$. We say such a subset $F$ is $G$-partition. Denote by $\mathcal{A}(t_i)$ the set of nodes corresponding to ancestor of $t_i$ in $T$ and $\mathcal{A}(F) = \bigcup_{t_i \in F} \mathcal{A}(t_i)$. It is clear that a $F$ is $G$-partition iff $F$ satisfies:

1. $\forall t_i, t_j \in F: t_i \notin \mathcal{A}(t_j)$ and $t_j \notin \mathcal{A}(t_i)$
2. $\forall t_i \in V_T, t_i$ is a leaf: $\mathcal{A}(t_i) \cap F \neq \emptyset$

\[\text{Fig. 2} \ \text{A part of a decomposition tree.} \ \ F = \{t_2, t_3, t_5, t_6\} \ \text{is a} \ G\text{-partition. The corresponding partition} \ \{V(t_2), V(t_3), V(t_5), V(t_6)\} \ \text{in} \ G \ \text{can be obtained by using cuts at ancestors of nodes in} \ F \ \text{i.e.} \ t_0, t_1, t_4.\]

In case $F = \{t_1, t_2, \ldots, t_k\}$ is $G$-partition, we can separate $V(t_1), V(t_2), \ldots, V(t_k)$ in $G$ by performing the cuts corresponding to ancestors of nodes in $F$ during the tree construction. For example in Figure 2, we show a decomposition tree with a $G$-partition set $\{t_2, t_3, t_5, t_6\}$. The corresponding partition $\{V(t_2), V(t_3), V(t_5), V(t_6)\}$ in $G$ can be obtained by cutting $G[V(t_0)], G[V(t_1)], G[V(t_4)]$ successively using $c$-balanced cuts in the tree construction. The cut cost, hence, will be $\text{cost}(t_0) + \text{cost}(t_1) + \text{cost}(t_4)$. In general, the total cost of all the cuts to separate $V(t_1), V(t_2), \ldots, V(t_k)$ is equal to:
\[ \text{cost}(F) = \sum_{t_i \in A(F)} \text{cost}(t_i) \]

And the pairwise connectivity in \( G \) is equal to:
\[ \mathcal{P}(F) = \sum_{t_i \in F} \mathcal{P}(G[V(t_i)]) \]

Our goal now is to find a \( G \)-partition \( F \in V_T \) so that \( \mathcal{P}(F) \leq \beta'(\frac{n}{2}) \) with a minimum \( \text{cost}(F) \) since the cut associated to \( F \) on \( T \) is the \( \beta' \)-edge disruptor of \( G \). Clearly finding such a set \( F \) has an optimal substructure, thus it can be found in \( O(n^3) \) using dynamic programming as described next.

Let \( \text{cost}(t_i, p) \) denote the minimum cut cost to make the pairwise connectivity in \( G[V(t_i)] \) be less than or equal to \( p \) using only \( c \)-balanced cuts corresponding to nodes in the subtree rooted at \( t_i \). The minimum cost for a \( G \)-partition subset \( F \) that induces a \( \beta' \)-edge disruptor of \( G \) is then
\[ \min \{ \text{cost}(t_0, p) \mid p \leq \beta'(\frac{n}{2}) \} \]
where \( t_0 \) is the root node in \( T \).

The value of \( \text{cost}(t_i, p) \) can be calculated using following recursive formula:
\[
\text{cost}(t_i, p) = \begin{cases} 
0 & \text{if } \mathcal{P}(G[V(t_i)]) \leq p \\
\min_{\pi \leq p} \{ \text{cost}(t_{i1}, \pi) + \text{cost}(t_{i2}, p - \pi) + \text{cost}(t_i) \} & \text{otherwise}
\end{cases}
\]

where \( t_{i1}, t_{i2} \) are children of \( t_i \).

In the first case, when \( \mathcal{P}(G[V(t_i)]) \leq p \), no cut is required, thus \( \text{cost}(t_i, p) = 0 \). Otherwise, we try all possible combinations of pairwise connectivity \( \pi \) in \( V(t_{i1}) \) and \( p - \pi \) in \( V(t_{i2}) \) for all \( \pi \leq p \). The combination with the smallest cut cost is then selected.

Based on the above recursive formula, a dynamic programming (shown in lines 14 to 23 of Algorithm 1) can find a minimum cost \( G \)-partition set \( F \) that induces a \( \beta' \)-edge disruptor in \( G \).

We now prove that such a set \( F \) exists in \( T \) and the cost of the \( \beta' \)-edge disruptor induced from \( F \) found in the dynamic programming algorithm is no more than \( O(\log \frac{3}{2} n) \) \text{Opt}_{\beta', \text{ED}} \) as follows:

**Lemma 3.** There exists a \( G \)-partition subset \( F \) in \( T \) that induces a \( \beta' \)-edge disruptor whose cost is no more than \( O\left( \log \frac{3}{2} n \right) \) \text{Opt}_{\beta', \text{ED}} \).

**Proof.** We first show that such a set \( F \) exists. That is, we are able to find a set \( G \)-partition set \( F \) such that \( \mathcal{P}(F) \leq \beta'(\frac{n}{2}) \). Denote \( D_\beta \) an optimal \( \beta \)-edge disruptor in \( G \). Removing \( D_\beta \) from \( G \) we obtain a set of strongly connected components (SCCs), denote as \( C_\beta = \{C_1, C_2, \ldots, C_k \} \).

We construct a \( G \)-partition subset \( F \) based on \( C_\beta \) as shown in the Algorithm 2. We visit nodes in \( T \) in a top-down manner i.e. every parent must be visited before its children. This can be done by visiting nodes in Breath First Search (BFS) order from the root node \( t_0 \). For each node \( t_i \), if there exists
some component $C_j \in C_\beta$ such that $V(t_i)$ contains more than $(1-c)|V(t_i)|$ nodes in $C_j$ (all leaves in $T$ satisfies this condition) and no ancestors of $t_i$ has ever been selected into $F$, then we select $t_i$ as a member of $F$. It is clear to see that $F$ is a $G$-partition as it satisfies two conditions mentioned earlier.

The total pairwise connectivity of $G$ induced by $F$ is bounded as:

$$\mathcal{P}(F) \leq \sum_{t_i \in F} \left( \frac{|V(t_i)|}{2} \right)$$

$$= \frac{1}{2} \sum_{C_j \in C_\beta} \sum_{|V(t_i) \cap C_j| \geq (1-c)|V(t_i)|} |V(t_i)|^2 - \frac{n}{2}$$

$$\leq \frac{1}{2} \sum_{C_j \in C_\beta} \left( \sum_{|V(t_i) \cap C_j| \geq \sqrt{\frac{\beta}{\beta_c}}|V(t_i)|} |V(t_i)| \right)^2 - \frac{n}{2}$$

$$\leq \frac{1}{2} \sum_{C_j \in C_\beta} \left( \sqrt{\frac{\beta}{\beta_c}} |C_j| \right)^2 - \frac{n}{2}$$

$$< \frac{\beta'}{2\beta} \left( \sum_{C_j \in C_\beta} |C_j|^2 - n \right) \leq \beta' \left( \frac{n}{2} \right)$$

Thus, such a set $F$ exists.

Next, we show that $\text{cost}(F) \leq O(\log^3 n)\text{Opt}_{\beta-ED}$. Let $h(T)$ and $L_u$ denote the height of $T$ and the set of nodes at level $u$ of $T$, respectively. We have:

$$\text{cost}(F) = \sum_{u=1}^{h(T)} \sum_{t_i \in (L_u \cap A(F))} \text{cost}(t_i)$$

If $t_i \in A(F)$ then $t_i$ is not selected to $F$. Hence, there exists $C_j \in C_\beta$ so that $|V(t_i) \cap C_j| < (1-c)|V(t_i)|$ (otherwise $t_i$ was selected into $F$ as it satisfies the conditions in the line 3, Algorithm 2). To guarantee $c < 1 - \epsilon$ we constrain $c < 1/2$ i.e. $\beta > \beta'$.

The edges in the optimal $\beta$-edge disruptor $D_\beta$ separate $C_j$ from the other SCCs. Hence, $D_\beta$ also separates $C_j \cap V(t_i)$ from the $V(t_i) \backslash C_j$ in $G[V(t_i)]$. Denote $\text{sep}(t_i, D_\beta)$ the set of edges in $D_\beta$ separating $C_j \cap V(t_i)$ from the rest in $G[V(t_i)]$. Clearly, $\text{sep}(t_i, D_\beta)$ is a directed $c$-balanced cut of $G[V(t_i)]$. Since, the cut algorithm we used in the tree construction has a pseudo-approximation ratio of only $O(\sqrt{\log n})$, we have $\text{cost}(t_u) \leq O(\sqrt{\log n})|\text{sep}(t_i, D_\beta)|$.

Recall that if two nodes $t_i$ and $t_j$ are on the same level then $V(t_i)$ and $V(t_j)$ are two disjoint subsets. It follows that $\text{sep}(t_i, D_\beta)$ and $\text{sep}(t_j, D_\beta)$ are also disjoint sets. Therefore, we have:
A Note on Network Vulnerability

\[
\sum_{t_i \in (L^* \cap A(F))} \text{cost}(t_i)
\leq O(\sqrt{\log n}) \sum_{t_i \in (L^* \cap A(F))} |\text{sep}(t_i, D\beta)|
\leq O(\sqrt{\log n}) \bigcup_{t_i \in (L^* \cap A(F))} |\text{sep}(t_i, D\beta)|
= O(\sqrt{\log n})\text{Opt}_{\beta-\text{ED}}
\]

Since \(h(T)\) is at most \(O(\log n)\) (Lemma 2), it follows from Equation 1 that \(\text{cost}(F) \leq O(\log^{\frac{3}{2}} n)\text{Opt}_{\beta-\text{ED}}\). It completes the proof. \(\square\)

**Algorithm 2** Find a good \(G\)-partition set \(F\) of \(T\) that induces a \(\beta'\)-edge disruptor in \(G\)

\[
F \leftarrow \phi
\text{for } t_i \in T \text{ in BFS order from } t_0 \text{ do}
\quad \text{if } (\exists C_j \in C_\beta : |V(t_i) \cap C_j| \geq (1 - c)|V(t_i)|) \text{ and } (A(t_i) \cap F = \emptyset) \text{ then}
\qquad F \leftarrow F \cup \{t_i\}
\quad \text{end if}
\text{end for}
\]

Since there exists a \(G\)-partition subset of \(T\) that induces a \(\beta'\)-edge disruptor whose cost is no more than \(O(\log^{\frac{3}{2}} n)\text{Opt}_{\beta-\text{ED}}\) as shown in Lemma 3 and the dynamic programming is always able to find such a set \(F\), the following theorem follows immediately.

**Theorem 4.** Algorithm 1 achieves a pseudo-approximation ratio of \(O(\log^{\frac{3}{2}} n)\) for the \(\beta\)-edges disruptor problem.

4 Approximation Algorithm for \(\beta\)-vertex Disruptor

In this section, we present a polynomial time algorithm (shown in Algorithm 3) that finds in a directed graph \(G = (V, E)\) a \(\beta'\)-vertex disruptor whose the size is at most \(O(\log n \log \log n)\) times the optimal \(\beta\)-vertex disruptor where \(0 < \beta < \beta'^2\).

At the high level, Algorithm 3 consists of two main phases: (1) Vertex Conversion and (2) Size Constraint Cut. In the first phase, we convert a given graph \(G\) into \(G'\) in a way that removing \(v \in G\) has the same effects as removing edge in \(G'\). In the second phase, we try to cut \(G'\) into SCCs capping the sizes of the largest component while minimizing the number of removed edges. The constraint on the size of each component is kept relaxing until the set of cut edges induces a \(\beta'\)-vertex disruptor of \(G\).
4.1 Algorithm Description

Phase 1: Vertex Conversion. Given a directed graph $G = (V, E)$ for which we want to find a small $\beta'$-vertex disruptor, we construct a new directed graph $G' = (V', E')$ as follows:

1. $V'$ construction: For each vertex $v \in V$, create two vertices $v^-$ and $v^+$ in $V'$. Thus $V' = \{ v^-, v^+ | v \in V \}$.

2. $E'$ construction: For each vertex $v \in V$, add a directed edge $(v^- \rightarrow v^+)$ to $E'$. And for each directed edge $(u \rightarrow v) \in E$, add a directed edge $(u^+ \rightarrow v^-) \in E'$. That is, $E' = \{ (v^- \rightarrow v^+) | v \in V \} \cup \{ (u^+ \rightarrow v^-) | (u \rightarrow v) \in E \}$.

3. Edge cost assignment: Assign 1 for all edges $(v^- \rightarrow v^+)$ and $+\infty$ to other edges in $E'$.

An example of this construction is shown in Figure 3.

![Fig. 3 Vertex Conversion](image)

Based on this conversion, a careful edge cut on $G'$ will return a vertex disruptor on $G$. Indeed, let $E'_V = \{ (v^- \rightarrow v^+) | v \in V \}$ and consider a cut set $D'_c \subset E'$ that contains only edge in $E'_V$, we have a one-to-one correspondence between $D'_c$ and $D_v = \{ v | (v^- \rightarrow v^+) \in D'_c \}$ which is a vertex disruptor set of $G$. However, $G$ and $G'$ have different maximum pairwise connectivity, $\frac{(n-1)n}{2}$ for $G$ and $\frac{(2n-1)2n}{2}$ for $G'$, the fractions of pairwise connectivity remaining in $G$ and $G'$ after removing $D_v$ and $D'_c$ are, therefore, not simply related to each other. Thus we next describe what a "careful edge cut" is and how good a vertex disruptor of $G$ can be obtained.

Phase 2: Size Constraint Cut. Observe that if we remove edges and separate a graph into SCCs then there is a relation between the pairwise connectivity in the remaining graph and the maximum size of SCC. The smaller the maximum size of SCC, the smaller pairwise connectivity in the graph. However, the smaller the maximum size of each SCC, the more edges are needed to be cut. Therefore, we need to find a good range of the maximum size of SCC in $G'$ in order to return a good vertex disruptor of $G$ with the minimum cut. Along this direction, we perform a binary search to find a right upper bound $\beta |V'|$ and lower bound $\beta |V'|$ of the size of each SCC in $G'$. As shown in Algorithm 3, at each step, we find in $G' = (V', E')$ a minimum cost edge set whose removal partitions the graph into strongly connected...
Algorithm 3 $\beta'$-vertex disruptor

**Input:** Directed graph $G = (V, E)$ and fixed $0 < \beta' < 1$.
**Output:** A $\beta'$-vertex disruptor of $G$

\[ G'(V', E') \leftarrow (\phi, \phi) \]
\[ \forall v \in V : V' \leftarrow V' \cup \{v^+, v^-\} \]
\[ \forall v \in V : E' \leftarrow E' \cup \{(v^- \rightarrow v^+)\}, c(v^-, v^+) \leftarrow 1 \]
\[ \forall (u \rightarrow v) \in E : E' \leftarrow E' \cup \{u^+ \rightarrow v^-, c(u^+, v^-) \leftarrow \infty \} \]
\[ \beta \leftarrow 0, \beta' \leftarrow 1 \]
\[ D_V \leftarrow V(G) \]

while $\left(\beta - \beta' > \epsilon\right)$ do

\[ \bar{\beta} \leftarrow \left\lfloor \frac{\beta + \beta'}{2} \right\rfloor \times \epsilon \]

Find $D_e \subset E'$ to separate $G'$ into strongly connected components of sizes at most $\bar{\beta}|V'|$ using algorithm in [13]
\[ D_e \leftarrow \{v \in V(G) \mid (v^+ \rightarrow v^-) \in D_e\} \]

if $\mathcal{P}(G[V \setminus D_e]) \leq \beta\binom{n}{2}$ then

\[ \beta = \bar{\beta} \]

Remove nodes from $D_e$ as long as $\mathcal{P}(G[V \setminus D_e]) \leq \beta\binom{n}{2}$

if $|D_V| > |D_e|$ then

\[ D_V = D_e \]

end if

derseelse

\[ \bar{\beta} = \bar{\beta} \]

end if

end while

Return $D_V$

components, each has size at most $\bar{\beta}|V'|$, where $\bar{\beta} = \left\lfloor \frac{\beta + \beta'}{2} \right\rfloor \times \epsilon$. We round the value of $\bar{\beta}$ to the nearest multiple of $\epsilon$ so that the number of steps for the binary search is bounded by $\log \frac{1}{\epsilon}$. The problem of finding a minimum cost edge set to decompose a graph of size $n$ into strongly connected components of size at most $\rho n$ is known as $\rho$-separator problem. We use here the algorithm presented in [13] which finds a $\rho$-separator in a directed graph $G$ with a pseudo-approximation ratio of $O\left(\frac{1}{\epsilon^2} \log n \log \log n\right)$ for a fixed $\epsilon > 0$. In our context, $\rho = \bar{\beta}|V'|$. By the cost assignment step in the vertex construction phase, the edge cut obtained in this step must be a subset of $E'_V$ as other edges $e'$ not in $E'_V$ have a cost of $+\infty$, hence, $e'$ never got selected. Finally, we convert the cut edges in $G'$ to vertices in $G$ to obtain the $\beta'$-vertex disruptor. Simply, for each edge $(v^- \rightarrow v^+)$ in a cut set, we have a corresponding vertex $v \in V$.

### 4.2 Theoretical Analysis

**Lemma 4.** Algorithm 3 always terminates with a $\beta'$-vertex disruptor.
Proof. We show that whenever \( \tilde{\beta} \leq \beta' \) then the corresponding \( D_v \) found in Algorithm 3 is a \( \beta' \)-vertex disruptor of \( G \). Consider the edge separator \( D'_v \) of \( G' \) induced by \( D_v \). We first show the mapping between SCCs in \( G[V \setminus D_v] \) and SCCs in \( G'[E \setminus D'_v] \), the graph obtained by removing \( D'_v \) from \( G' \). Partition the vertex set \( V \) of \( G \) into: (1) \( D_v \): the set of removed nodes (2) \( V_{\text{singleton}} \): the set of SCCs whose sizes are one, that is, each SCC has only one node, and (3) \( V_{\text{connected}} \): the set of remaining SCCs whose sizes are at least two, denote \( V_{\text{connected}} = \bigcup_{i=1}^{j} C_i, |C_i| \geq 2 \). Vertices in \( V_{\text{connected}} \) belong to at least one cycle in \( G \) whereas vertices in \( V_{\text{singleton}} \) are all singleton.

We have the following corresponding SCCs in \( G'[E \setminus D'_v] \):

1. \( v \in D_v \leftrightarrow \) SCCs \( \{v^+\} \) and \( \{v^-\} \) in \( G'[E \setminus D'_v] \). Because after removing \( (v^- \rightarrow v^+) \), \( v^+ \) does not have incoming edges and \( v^- \) does not have outgoing edges.

2. \( v \in V_{\text{singleton}} \leftrightarrow \) SCCs \( \{v^+\} \) and \( \{v^-\} \). Assume \( v^+ \) belong to some SCC of size at least 2 i.e. \( v^+ \) lies on some cycle in \( G' \). Because the only incoming edge to \( v^+ \) is from \( v^- \), it follows that \( v^- \) is preceding \( v^+ \) on that cycle. Let \( u^-, u^+ \) be the successive vertices of \( v^+ \) on that cycle. We have \( u \) and \( v \) belong to the same SCC in \( G \) which yields a contradiction as \( v \in V_{\text{singleton}} \).

Similarly, \( v^- \) cannot lie on any cycle in \( G' \).

3. SCC \( C_i \subset V_{\text{connected}} \leftrightarrow \) SCC \( C'_i = \{v^-, v^+ | v \in C_i\} \). This can be shown using a similar argument as above.

Since \( D'_v \) is a \( \tilde{\beta} \)-separator, the sizes of SCCs in \( G'[E \setminus D'_v] \) are at most \( \tilde{\beta} 2n \). It follows that the sizes of SCCs in \( G[V \setminus D_v] \) are bounded by \( \tilde{\beta} n \).

Denote the set of SCCs in \( G[V \setminus D_v] \) by \( \mathcal{C} \) with the convention that vertices in \( D_v \) become singleton SCC in \( G[V \setminus D_v] \). Therefore, we have:

\[
\mathcal{P}(G[V \setminus D_v]) = \sum_{C_i \in \mathcal{C}} \left( \frac{|C_i|}{2} \right) = \frac{1}{2} \left( \sum_{C_i \in \mathcal{C}} |C_i|^2 - |V| \right) \\
\leq \frac{1}{2} \left( \sum_{C_i \in \mathcal{C}} \tilde{\beta} |V||C_i| - |V| \right) \\
= \frac{1}{2} \left( \tilde{\beta} |V|^2 - |V| \right) \leq \tilde{\beta} \left( \frac{|V|}{2} \right) < \beta' \left( \frac{|V|}{2} \right)
\]

This guarantees that the binary search always finds a \( \beta' \)-vertex disruptor and completes the proof.

\[\square\]

**Theorem 5.** Algorithm 3 achieves a pseudo-approximation ratio of \( O(\log n \log \log n) \) for the \( \beta \)-vertex disruptor problem.

Proof. We will show that Algorithm 3 always finds a \( \beta' \)-vertex disruptor whose the size is at most \( O(\log n \log \log n) \) times the optimal \( \beta \)-vertex disruptor for \( \beta'^2 > \beta > 0 \). First, it follows from the Lemma 4 that Algorithm 3 returns a \( \beta' \)-vertex disruptor \( D_v \). At some step, the size of \( D_v \) equals to
the cost of $\hat{\beta}$-separator $D'_v$ in $G'$ where $\hat{\beta}$ is at least $\beta' - \epsilon$ according to Lemma 4 and the binary search scheme. The cost of the separator is at most $O(\log n \log \log n)$ times the $Opt(\hat{\beta} - \epsilon)$-separator using the algorithm in [13].

Consider an optimal $(\beta'^2 - 9\epsilon)$-vertex disruptor $D'_v$ of $G$ and its corresponding edge disruptor $D'_e$ in $G'$. Denote the cost of that optimal vertex disruptor by $Opt(\beta'^2 - 9\epsilon)$-VD. If there exists in $G[V \setminus D_v]$ a SCC $C_i$ so that $|C_i| > (\beta' - 2\epsilon)n$ then

$$\mathcal{P}(G[V \setminus D_v]) > \frac{1}{2}((\beta' - 2\epsilon)n - 2)((\beta' - 2\epsilon)n - 1) > (\beta'^2 - 9\epsilon)(\frac{n}{2})$$

when $n > \frac{20(\beta' + 1)}{\epsilon}$. Hence, every SCC in $G'[V \setminus D'_v]$ has size at most $(\beta' - 2\epsilon)(2n)$ i.e. $D'_e$ is an $(\beta' - 2\epsilon)$-separator in $G'$. It follows that $Opt(\beta'^2 - 9\epsilon)$-VD $\geq Opt(\beta' - 2\epsilon)$-separator in $G'$.

Because $\beta - \epsilon \geq \beta' - 2\epsilon$, we have:

$$Opt(\hat{\beta} - \epsilon) - \text{separator} \leq Opt(\beta' - 2\epsilon) - \text{separator} \leq Opt(\beta'^2 - 9\epsilon) - \text{VD}$$

The size of the vertex disrupter $|D_v| = |D'_v|$ is at most $O(\log n \log \log n)$ times $Opt(\hat{\beta} - \epsilon)$-separator. Thus, the size of found $\beta'$-vertex disruptor $D_v$ is at most $O(\log n \log \log n)$ times the optimal $(\beta'^2 - 9\epsilon)$-vertex disruptor. As we can choose arbitrary small $\epsilon$, setting $\beta = \beta'^2 - 9\epsilon$ completes the proof. □

5 Exaction Solutions to QoSCE

In real application scenarios, the set of constraints that are taken into account is very limited, which may only consist jitter, packet loss, delay and some others. Therefore, it is quite practical to consider $m$ as a not quite large value, which gives rise to an exact solution called MFMCS.

The basic idea is to first enumerate all the possible satisfiable combinations of constraints, for example $(c_1, c_3, c_5)$ if $\lambda_1 + \lambda_3 + \lambda_5 \geq \rho$. Therefore $s - t$ paths that satisfy all the constraints within such a combination is a satisfiable path. We only consider the set of minimal constraint combinations, since the set of paths satisfying a set of constraints is surely the super set of those satisfying a superset of these constraints. (set of paths satisfying $(c_1, c_3)$ of course contains those satisfying $(c_1, c_3, c_5)$.) With this set of minimal combinations, we revise the classic Edmonds – Karp algorithm [14] to find the minimum size of edge cut to cut all the augmenting $s - t$ paths which at the same time satisfy any of these combinations. The pesudo-code of this algorithm is included in Algorithm 4 where we employ the A* MCSP algorithm in [15] to discover the shortest path that satisfies a specific set of constraints, i.e., a combination of
Fig. 4 Correctness of MFMCSP

corrections enumerated above. The correctness of this exact solution is shown in Theorem 6.

Algorithm 4 MFMCSP

1: **Input:** directed graph \( G = (V, E) \), constraint set \( M = \{c_1, \ldots, c_m\} \), credit vector \((\lambda_1, \lambda_2, \ldots, \lambda_m)\), satisfactory score threshold \( \rho \);
2: **Output:** solution set of edge of QoSCE.
3: \( S \leftarrow \) all the minimal combinations \( ss \) of \( M \) with \( \sum_{c_i \in ss} \lambda_i > \rho \);
4: for each edge \( (i, j) \in E \) do
5: \( f(i, j) = f(j, i) = 0 \);
6: \( c_f(i, j) = 1 \) and \( c_f(j, i) = 0 \).
7: end for
8: while \( S \neq \emptyset \) do
9: \( ss \leftarrow \) extracted from \( S \);
10: while \( \exists q \leftarrow \) the shortest path satisfying all the constraints in \( ss \) do
11: \( f(u, v) = f(v, u) = -f(u, v) \);
12: \( c_f(u, v) = c(u, v) - f(u, v) ; c_f(v, u) = c(v, u) - f(v, u) \);
13: end for
14: end while
15: all the vertices reachable from \( s \) on the residual network induces a cut \( \mathcal{T} \).
16: **Return** \( \mathcal{T} \).

**Theorem 6.** The edge cut returned by MFMCSP is an optimal solution of QoSCE.

**Proof.** As shown by Fig. 4, the satisfiable paths are within the dotted circle, while others are outside the circle. As stated in MFMCSP, all the satisfiable combinations of constraints are enumerated and based on the argument about the minimal combination, so each of the satisfiable paths should satisfy at least one minimal combination in \( S \).

Consider the subgraph induced by all these satisfiable paths as \( G' \). It is evident that no satisfiable path will be augmenting after the loop 8-17, otherwise, it is supposed to be discovered in Line 10. Since only the satisfiable paths are augmented within this loop, it can be regarded as a discovery of max-flow on the induced graph \( G' \). Therefore, the cut returned \( \mathcal{T} \) is a min cut of \( G' \), denoted as \( m(G') \). The equivalence of \( m(G') \) to \( \text{opt}(\text{QoSCE}) \) is proved by contradiction.
Assume there exists an edge $e \in \text{opt}(\text{QoSCE})$, while $e$ does not belong to any satisfiable paths, as shown in Fig. 4. Then adding $e$ back to $G$ will not bring back any satisfiable paths, while decreases the optimal solution and draws the contradiction. Therefore, $\text{opt}(\text{QoSCE})$ is a subset of $G'$. Moreover since removing all edges in $\text{opt}(\text{QoSCE})$ will disconnect all satisfiable paths, i.e. all $s-t$ path in $G'$, $\text{opt}(\text{QoSCE})$ is an edge cut of $G'$.

On the other hand, any cut of $G'$ is a feasible solution to QoSCE. Suppose that the min cut of $G'$ is smaller than $\text{opt}(\text{QoSCE})$, then this min cut becomes a smaller optimal solution to QoSCE, which draws a contradiction. Otherwise, if the min cut is larger than $\text{opt}(\text{QoSCE})$, then $\text{opt}(\text{QoSCE})$ becomes a new min cut.

Therefore, the min cut $T$ is an optimal solution to QoSCE.

6 Literature

Several existing works on network vulnerability and survivability have been investigated, roughly categorized into two periods. Researches during the early period mainly focused on investigating the node’s centrality and prestige, using functions of node degree [16, 17, 18, 19] as the only measure. The main measures of centrality can be classified as: degree centrality, betweenness, closeness, and eigenvector centrality. Readers are referred to [20] to find more details about this centrality measurement.

As wireless ad hoc networks came forth, several recent studies have aimed to discover the critical nodes whose removal causes the network disconnected, regardless of how fragmented it is! Several centralized, distributed, and localized heuristics have been proposed. Some representatives are: centralized DFS-based search [2, 21, 22, 23], distributed disjoint path [24, 3] and localized $k$-critical [4]. Most of these algorithms suffer from high communication overhead to discover the network partition and could not efficiently locate the critical nodes. In addition, none of these methods has been proven theoretically.

Unfortunately, none of the above work reveals the global damage done on the network in the case of multiple nodes/links failing simultaneously, thus inaccurately assessing the network vulnerability. None of these methods proves the performance bound of their solutions either.

In this chapter, we present a novel paradigm via two new optimization models for quantitatively characterizing the vulnerability of networks. The proposed approaches aim to identify the key nodes who play a vital role in the network overall performance, such as whose removal maximizes the network fragmented. This study defines the objective function differently in terms of the connected components rather than a traditional direct measure of algebraic connectivity, thereby providing an excellent way to assess the network vulnerability.
References