Enumerating Shared Risk Link Groups of Circular Disk Failures Hitting k Nodes

Balázs Vass[a], Erika Bérczi-Kovács[b], and János Tapolcai[a]
[a]MTA-BME Future Internet Research Group, Budapest University of Technology, {vb,tapolcai}@tmit.bme.hu
[b]Department of Operations Research, Eötvös University, Budapest, Hungary, koverika@cs.elte.hu

Abstract

Current backbone networks are designed to protect a certain pre-defined list of failures, called Shared Risk Link Groups (SRLG). It has been observed that some type of failure events manifested at multiple locations of the network which are physically close to each other. Such failure events are called regional failures, and are often caused by a natural disaster. The aim of the paper is to bring the conventional SRLG based pre-planned protection and regional failures closer to each other by providing a systematic approach to generate the list of SRLGs for regional failures. In our study we overestimate the regional failures with failures having a shape of circular disk; however, instead of fixing the radius we classify the regional failures according to the network elements they hit. In particular we are interested in the number of nodes the failure can hit. Formally, we focus on circular disk failures that hit exactly k nodes, where k is part of the input. According to simulation results, this list is short with $O((k+1)|V|)$ SRLGs in total, and can be computed in $O(|V|^3)$, where V denotes the set of the nodes. Applying the obtained SRLG list network operators can design their networks to be protected against regional and random failures.

1 Introduction

Current backbone networks are built to protect a certain list of failures. Each of these failures (or termed failure states) forms a *Shared Risk Link Group* (SRLG), which is a set of links that is expected to fail simultaneously. The network is designed to be able to automatically reconfigure in case of a single SRLG failure, such that every connection further operates after a very short interruption. In practice it means the connections are reconfigured to by-pass the failed set of nodes and links. Thus the network can recover if an SRLG or a subset of links in an SRLG fails simultaneously; however, there is no performance guarantee when a network is hit by a failure that involves links not a subset of an SRLG. Nevertheless, the list of SRLGs must be defined very carefully, because not getting prepared for one likely simultaneous failure event means a significant degradation in the observed reliability of the network.

Operators have numerous answers what simultaneous failures mean. One extreme is to list every *single link or node failure* as an SRLG. Here the concept is that the failure first hits a single network element for the protection of which the network is already pre-configured. Often there is a known risk of a simultaneous multiple failure that can be added as an SRLG, for example if two links between different pair of nodes traverse the same bridge, etc. On the other hand, we have witnessed serious network outages [1, 2, 3, 4, 5, 6, 7] because of a failure event that takes down almost every equipment in a physical region as a result of a disaster, such as weapons of mass destruction attacks, earthquakes, hurricanes, tsunamis, tornadoes, etc. These type of failures are called *regional failures*, which are simultaneous failures of nodes/links located in specific geographic areas .It is still a challenging open problem how to prepare a network to protect such failure events, as their location and size is not known at planning stage. In the paper we propose a solution to this problem with a technique that can significantly reduce the number of possible failure states that should be added as an SRLG to cover all regional failures.

Regional failures can have any location, size and shape. The shape of a regional failure can be arbitrary; however, it is a common practice to overestimate the size of the regional failure by ignoring its shape and rather focus on its radius [10] [11]. Roughly speaking by overestimating the size of the regional failure, we may require to have larger distances between the working and backup resources than it is needed in reality. However, adopting a failure model that captures only the diameter of the failed region results in a manageable list of SRGLs [11]. Nevertheless it is not trivial to define a network wide parameter for the maximal radius the regional failure may have. Thus, to have a generic model instead of fixing the radius we classify the regional failures according to the network elements it hits. In particular we are interested in the number of nodes the failure covers. Note that the networks are more sensible to node failures than link failures. The intuition behind this model is that there are more network nodes in the crowded areas, where we would like to generate more SRLGs.

In our previous studies [12] and [13] regional failures hitting exactly $k = 0$ and $k = 1$ nodes have been studied. The lists proposed for protecting these failures have a length linear in the number of network nodes and can be calculated in $O(|V|\log|V|)$ and $O(|V^2|)$ for $k = 0$ and $k = 1$,

respectively, in practice.
In this study the number of SRLGs is significantly reduced applying computational geometric tools based on the following assumptions:

1. The network is a geometric graph $G(V,E)$ embedded in a 2D plane.
2. The shape of the regional failure is assumed to be a circle with arbitrary radius and center position.
3. We focus on **regional link k-node failures**, failures that hit k nodes.

We will show that with these assumptions the number of SRLGs is small, $O(k|V|)$ in a typical backbone network topology, and at most $O(|E||V|)$ in an artificial worst case scenario (shown in [12]), where $|V|$ denotes the number of nodes in the network, and $|E|$ the number of links. We propose a systematic approach based on computational geometric tools that can generate the list of SRLGs in $O(|V|^3)$ steps on typical networks.
Using the obtained SRLG list network operators can design their networks to be protected against regional and random failures. Backbone networks designed according to our new failure model should have higher reliability.
The paper is organized as follows. After Sec. 1 of introduction, in Sec. 2 our model will be presented. In Sections 3 and 4 there will be introduced a naive and an improved algorithm for computing the SRLG list, respectively. After this, we further refine our insight on proposed algorithms and auxiliary graphs in Sec. 5. Simulation results are presented in Sec. 6. Finally, conclusions are drawn in Sec 7.

2 Model and Assumptions

The network is modeled as an undirected connected geometric graph $G(V,E)$ with $n = |V|$ nodes and $m = |E|$ edges. E is given as an edge list, and $n \geq 3$ is assumed. The nodes of the graph are embedded as points in the Euclidean plane, and the edges are embedded as line segments between the end points. It will be assumed that basic arithmetic functions $(+,-,\times,/,\sqrt{\ })$ have constant computational complexity. For simplicity we assume nodes of V are situated in general positions of the plane, i.e. there are no three points on the same line, and no four on the same circle.
In this study failures are caused by a regional disaster having an area overestimated by a disk, i.e we overestimate the disaster such that it erases all network elements that intersect the interior of a circle c, and leaves all other network elements untouched. Note that we do not assume the failed region has a shape of disk, instead this modeling technique overestimates the size of the failed region in order to have tractable problem space.
Due to mathematical considerations the erasing effect of a circle is formally defined as follows.

Definition 1. *If an edge e (considered as a geometric object) intersects the interior or the boundary of a circle c then we say e is hit by c. A node u is hit by circle c if it is located in interior of c (and not on its boundary.)*

In this study we focus on disk failures hitting exactly k nodes, where $k \in \{0,\ldots,n-2\}$ is a given number.

Definition 2. *Let $\mathscr{C}$ denote the set of all circles in the plane, and let $\mathscr{C}_k \subseteq \mathscr{C}$ denote the set of those hitting exactly k nodes.*

From the viewpoint of connectivity listing failed nodes beside listing failed edges has no additional information, thus only the failed edge sets will be considered.

Definition 3. *Let E_c denote the set of edges hit by a circle c.*

Based on the above we can define the set of failure states.

Definition 4. *Let set $F(\mathscr{C})$ denote the set of link sets which can be hit by a circle $c \in \mathscr{C}$.*

If a network is prepared for the failure of a given edge set F, then it is prepared for the failure of every edge set $F' \subseteq F$ too, thus it is enough to list only the maximal elements of $F(\mathscr{C})$ as SRLGs.

Definition 5. *For a set $\mathscr{C}'$ of circles let $M(\mathscr{C}')$ denote the set of maximal link sets which can be hit by a circle $c \in \mathscr{C}'$.*

The aim of this study is to offer fast algorithms computing $M(\mathscr{C}_k)$ for various values of k, which is also denoted by $M_k = M(\mathscr{C}_k)$.

3 Naive Algorithm for Enumerating Maximal Failures

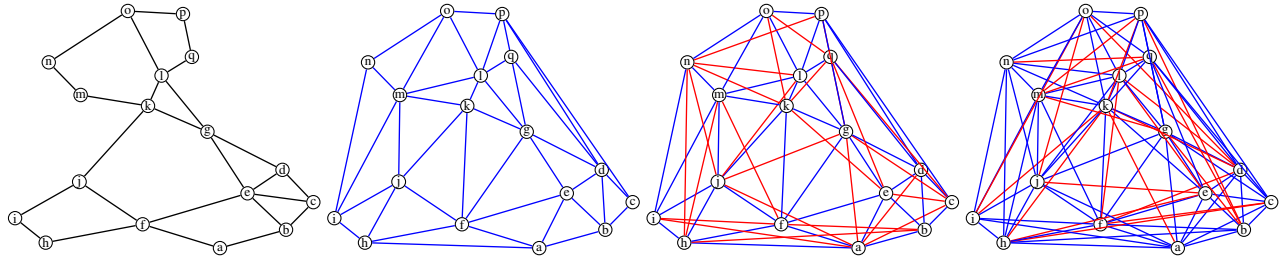
In this section a naive polynomial algorithm is presented for computing M_k. The basic idea is that determining M_k can be decomposed into several simpler tasks.

3.1 Basic Observations

Our key observation is the following.

Claim 6. *For every $f \in M_k$ $(k \leq n-2)$ there exists a circle $c \in \mathscr{C}_k$ such that f is hit by c and c has at least 2 nodes on its boundary.*

Proof. Since $f \in M_k$, there exists a $c_0 \in \mathscr{C}_k$ that hits f. If c_0 has at least 2 nodes on its boundary, it fulfills the requirements of the claim. On the other hand, if c_0 does not have any node on its boundary, it can be magnified until it reaches a node u while keeping its centre point. Remains to prove that if c_0 has exactly 1 node on its boundary then there exists a 'good' circle having 2 nodes on its boundary. Let us denote the node on the boundary of c_0 with u, and the centre point of c_0 with P. In most cases, c_0 can be magnified until it reaches a node v while keeping u on its boundary and its centre point on line uP. The resulting circle c still hits f, is element of $\mathscr{C}_k$, and has at least 2 nodes on its boundary. If the described v does not exist, then all the edges from f have at least one of theire endpoints



(a) Input graph, $n = 17$ **(b)** Graph $D_0(V,E_0)$, $|E_0| = 40$ **(c)** Graph $D_1(V,E_1)$, $|E_1| = 66$ **(d)** Graph $D_2(V,E_2)$, $|E_2| = 89$

Figure 1 Input topology and auxiliary graphs D_k for $k = 0,1,2$.

inside c_0; and c_0 can be magnified while keeping P as centre point until it reaches a node u_2. Let this circle be c_1, which can be shrunken until its boundary reaches a node v_2 while keeping its centre point on line u_2P and node u_2 on its boundary. If the resulting circle c_2 has a third node v_3 on its boundary, then c_2 can be magnified a bit while keeping u_2 and v_2 on its boundary such that v_3 becomes a node in interior of the resulting c_3, which still hits f, is element of $\mathscr{C}_k$, and has at least 2 nodes on its boundary, completing the proof.

□

Claim 6 suggests the following simple method to compute M_k. First, for every node pair $\{u,v\}$ we compute a set $M_k^{u,v}$ of failures which contain all the elements of M_k that can be hit by a circle $c \in \mathscr{C}_k$ having u and v on its boundary. M_k can be computed by merging these sets. Let us formalise the above method.

Definition 7. *For every node pair u and v let $\mathscr{C}_k^{u,v}$ be the set of disks from $\mathscr{C}_k$ having u and v on their boundary.*

Based on the above definition we can define an auxiliary graph as follows.

Definition 8. *Let $D_k(V,E_k)$ denote the graph with node set V and edge set E_k, where $\{u,v\} \in E_k$ if and only if $\mathscr{C}_k^{u,v} \neq \emptyset$.*

Fig. 1 shows an example of the input topology and the auxiliary graphs D_k for $k = 0,1,2$. It can be shown that D_0 is actually the Delaunay triangulation [14] of the graph; thus, we call D_k the Delaunay-k graph. Clearly, D_k can be computed in polynomial time. In Sec. 6 we will show that D_k is sparse for small k. In other words, $\mathscr{C}_k^{u,v} = \emptyset$ for most node pairs u,v for small k.

We aim to compute $M_k^{u,v} = M(\mathscr{C}_k^{u,v})$. Fortunately, according to Claim 6 we have

$$M_k \subseteq \bigcup_{u,v \in E_k} M_k^{u,v} \ . \tag{1}$$

3.2 The Apple Data Structure

Assume for a moment that for a given node pair $\{u,v\}$ there are at least k nodes in both half planes determined by line uv. Let us place a Cartesian coordinate system in the plane such that line uv be identical to the Oy axis. Assuming that $\mathscr{C}_k^{u,v}$ is not empty, this means that $\mathscr{C}_k^{u,v}$ has a rightmost c_+ and a leftmost c_- element (i.e. the centre point of c_+ and c_- have the biggest and smallest x coordinates among circles from $\mathscr{C}_k^{u,v}$, respectively), see also Fig. 2. It is easy to prove indirectly that circle c_+ has 3 nodes on its boundary, say u,v and w_+. Similarly, c_- has u,v and w_- on its boundary.

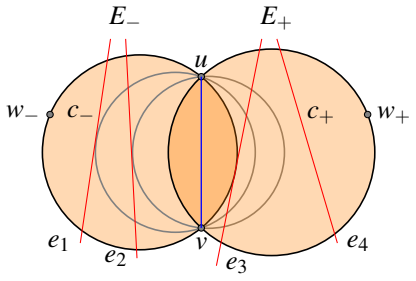


Figure 2 Illustration of an apple with $k = 0$. Apple $A_k^{u,v}$ consists of circles c_+ and c_-, nodes w_+ and w_- from $V \cup \{v_\emptyset\}$, and ordered lists of edges E_+ and E_-, where $E_+ = \{e_4,e_3\}$ and $E_- = \{e_2,e_1\}$.

If there are no k nodes on the left side of uv, then let c_+ be the right open half plane h_+ determined by line uv. In this case h_+ is considered to be a degenerate element of $\mathscr{C}_k^{u,v}$. Same applies for the left side of uv. Let $v_\emptyset$ be considered as an imaginary node. For easier discussion we will use $v_\emptyset$ to indicate that our node object does not exist. For example in the above case we have $w_+ = v_\emptyset$.

Let E_{c_+} and E_{c_-} denote the edge sets hit by c_+ and c_-, respectively. To compute $M_k^{u,v}$ we use the following observation.

Claim 9. *For all $f \in F(\mathscr{C}_k^{u,v})$, $f \subseteq E_{c_+} \cup E_{c_-}$.*

Proof. It is easy to see that for every circle $c \in \mathscr{C}_k^{u,v}$, $c \subseteq c_+ \cup c_-$. □

According to Claim 9, a first step towards computing $M_k^{u,v}$ is to determine the edge sets hit by c_+ and c_-. Trivially, this can be done in polynomial time. The remaining question is how to calculate systematically $M_k^{u,v}$ from $E_{c_+} \cup E_{c_-}$. Some additional notations and definitions precede the presentation of the solution.

For each edge $e \in E_{c_+} \cup E_{c_-}$ we will compute the maximal subtending angle of the line segment u,v. Here we consider the two half planes determined by line u,v separately. Let ◗ denote the right side of circle c_+ cut by the vertical line uv, and ◖ the left side of circle c_- cut by the vertical line

uv. Let E_+ denote the list of edges hit by D, and similarly, let E_- be the list of edges hit by C. Thus, we have $E_+ \subseteq E_{c_+}$ and $E_- \subseteq E_{c_-}$, and also $E_+ \cup E_- \equiv E_{c_+} \cup E_{c_-}$.

Let $\sphericalangle_+^{u,v}(e)$ denote the maximal subtending angle to line segment $[uv]$ from a point of edge e in D. Similarly, let $\sphericalangle_-^{u,v}(e)$ denote the maximal subtending angle to line segment $[uv]$ from a point of edge e in C. Let edges in E_+ be ordered ascending by their $\sphericalangle_+^{u,v}$ values, and let edges in E_- be ordered descending by their $\sphericalangle_-^{u,v}$ values.

Note that according to Claim 43 from the Appendix, both $\sphericalangle_+^{u,v}(e)$ and $\sphericalangle_-^{u,v}(e)$ can be computed in $O(1)$.

Now we can define the data structure *apple* for each edge of the Delaunay-k graph.

Definition 10. *For an edge $\{u,v\} \in E_k$, apple $A_k^{u,v}$ is an ordered system $A_k^{u,v} = (c_+, c_-, w_+, w_-, E_+, E_-)$, where circles c_+ and c_-, nodes w_+ and w_-, and ordered lists of edges E_+ and E_- are as described in the subsection before.*

3.3 Concept of Sweep Circle Algorithms

3.3.1 Concept

In this subsection we highlight the paradigm of sweep circle algorithms, which is similar to algorithmic paradigm of sweep line (sweep surface) algorithms in computational geometry.

In case of sweep line algorithms it is imagined that a line is moved across the plane, keeping its orientation and stopping at some points. Geometric operations are restricted to the immediate vicinity of the sweep line whenever it stops, and the complete solution is available once the line has passed over all objects. For example Fortune's sweep line algorithm for computing the Voronoi diagram of a point set is a sweep line algorithm [14].

Definition 11. *Let $\mathscr{C}^{u,v}$ be the set of circles having u and v on the boundary.*

Our sweep circle algorithms will scan through circle sets $\mathscr{C}^{u,v}$. In this sense in contrast of the sweep surface paradigm, our circles have different diameters, and instead of keeping "orientation" the invariant will be that all circles have u and v on the boundary. Thus our circle to sweep is "elastic", in the sense that it can change its diameter, but not its shape.

3.3.2 Example

Our first sweep circle algorithm is used for determining w_+ and w_- for a given $A_k^{u,v}$. The algorithm works as follows. Starting from a circle $c \in \mathscr{C}^{u,v}$ having centre point 'very far away' on the right side of line uv, c is swept throughout the elements of $\mathscr{C}^{u,v}$ until a position 'very far away' on the left. Meanwhile the number of nodes hit is followed. Nodes w_+ and w_- can be determined at the first and last moment when c hits exactly k nodes, respectively. (Non-existence of such moments would mean that $\{u,v\} \notin E_k$.)

Technically this can be done as follows. Let $V_+ \subseteq V$ be the list of nodes right from line uv ordered increasingly by their subtending angles $\sphericalangle(uwv)$. Similarly, let V_- be the list of nodes left from line uv ordered decreasingly by their subtaining angles. Applying the fact that a node pair $z_+ \in V_+$ and $z_- \in V_-$ can be hit by the same circle $c \in \mathscr{C}^{u,v}$ iff $\sphericalangle(uz_+v) + \sphericalangle(uz_-v) \geq \pi$, sweeping can be imitated as in Algorithm 1.

Algorithm 1: Sweeping through $\mathscr{C}^{u,v}$ checking the nodes

Input: V and $u, v \in V$ **begin**
1. Compute V_+ and V_-;
2. $i, j \leftarrow 1$;
3. **while** *not reached the end of V_+ or V_-* **do**
4. increase j until $V_+[i]$ and $V_-[j]$ cannot be hit by the same $c \in \mathscr{C}^{u,v}$;
5. increase i until $V_+[i]$ and $V_-[j]$ can be hit by the same $c \in \mathscr{C}^{u,v}$;

From the following Proposition 12, one can check that the number of hit nodes can be easily followed with the help of an additional variable.

Proposition 12. *Let $c \in \mathscr{C}^{u,v}$. If $V_+[i-1]$ is not hit by c, then all the preceding elements in V_+ are not hit by c. If $V_+[i]$ is hit by c, then all the following elements are hit by c.*

Similarly, if $V_-[i-1]$ is hit by c, then all the preceding elements are hit by c. If $V_-[i]$ is not hit by c, then all the following elements are not hit by c. □

Claim 13. *For a given $\{u,v\} \in E_k$, w_+ and w_- can be determined in $O(n \log n)$ time.*

Proof. According to those written in this subsection, both V_+ and V_- can be determined in $O(n \log n)$ the dominant step being a sorting algorithm. Sweeping can be trivially done in $O(n)$, meanwhile both w_+ and w_- can be determined. □

3.4 Determining Apples

Claim 14. *An apple can be determined in $O(m \log m)$.*

Proof. Checking if $\{u,v\} \in E_k$ can be done with the help of Algorithm 1 in $O(n \log n)$ time. If $\{u,v\} \in E_k$, then w_+ and w_- can be determined in $O(n \log n)$ according to Claim 13. From w_+ and w_- we also know c_+ and c_-, so it remains to determine E_+ and E_-. With this aim $O(m)$ edges have to be checked and sorted which gives a total complexity of $O(m \log m)$. □

Definition 15. *Let A_k be the set of apples $A_k^{u,v}$.*

Corollary 16. *For a given k, the set of apples A_k can be determined in $O(n^2 m \log m)$.*

Proof. $O(n^2)$ node pairs have to be examined. According to Claim 13, a node pair can be examined in $O(m \log m)$, which completes the proof. □

3.5 Computing the Set of SRLGs by Sweeping Through Each Apple

In order to compute $M_k^{u,v}$, the algorithm has to be able to decide if there exists a circle $c \in \mathscr{C}_k^{u,v}$ which hits a given

edge pair e and f from $E_+ \cup E_-$. If both e and f are part of E_+ or E_-, then they can be both hit by c_+ or c_-. If it is not the case, the following proposition will help in this decision.

Proposition 17. *Let $e \in E_+$, and $f \in E_-$, they can be hit by the same $c \in \mathscr{C}_k^{u,v}$ iff $\sphericalangle_+^{u,v}(e) + \sphericalangle_-^{u,v}(f) \geq \pi$.* □

Determining $M_k^{u,v}$ from apple $A_k^{u,v}$ can be done with the help of a sweep circle algorithm as a subroutine of Algorithm 2 similar to Algorithm 1, the only difference is that here we check the edges instead of nodes.
Implementation of Algorithm 2 has to be made carefully. On one hand, while sweeping through $\mathscr{C}^{u,v}$ from c_+ until c_-, not necessarily all the circles are from $\mathscr{C}_k^{u,v}$. On the other hand, edges intersecting segment $[u,v]$ should be stored exactly once in any element of $M_k^{u,v}$.

Algorithm 2: Processing an apple

Input: Apple $A_k^{u,v}$
Output: Set $M_k^{u,v}$ of locally maximal failures.
begin
1 **while** *Sweeping through $\mathscr{C}^{u,v}$ from c_+, until c_-; checking the edges* **do**
2 Gather in $(M_k^{u,v})'$ the failures hit by a $c \in \mathscr{C}_k^{u,v}$ with locally maximal cardinalities
3 $M_k^{u,v} \leftarrow$ maximal elements of $(M_k^{u,v})'$;
4 **return** $M_k^{u,v}$

Claim 18. *Algorithm 2 calculates $M_k^{u,v}$ in $O(m^3)$.*

Proof. Correctness of the algorithm can be easily checked. Since while sweeping an edge can get hit or unhit at most once on one side of line uv, there are at most $O(m)$ failures with locally maximal cardinalities, each of them having $O(m)$ edges, thus $(M_k^{u,v})'$ has $O(m)$ elements of $O(m)$ size. Trivially, every pair of sets from $(M_k^{u,v})'$ can compared in $O(m)$. This means that from $(M_k^{u,v})'$, $M_k^{u,v}$ can be determined in $O(m^3)$. It can be checked that all the other operations have complexity at most $O(m^3)$. □

Corollary 19. *All $M_k^{u,v}$ can be determined in $O(n^2m^3)$.* □

3.6 The Naive Algorithm

As presented before, a naive algorithm should determine and process the apples, and finally merge the obtained lists $M_k^{u,v}$ in M_k. This way the naive algorithm could be written as follows:

Algorithm 3: Algorithm for computing M_k

Input: $G(V,E)$, k
Output: M_k
begin
1 Determine E_k;
2 Determine set A_k of nonempty apples;
3 Process apples from A_k;
4 Merge lists $M_k^{u,v}$;
5 **return** M_k

Theorem 20 gives a loose time complexity for the algorithm:

Theorem 20. *Algorithm 3 calculates set M_k in $O(n^4m^3)$ time. M_k has $O(n^2m)$ elements, and a total length of $O(n^2m^2)$.*

Proof. Let the four phases of the algorithm be examined separately.
Determining E_k: As mentioned in the proof of Claim 14, the sweep circle Algorithm 1 can be used to decide whether a given pair of nodes $\{u,v\} \in E_k$. According to Claim 13, it gives the answer in $O(n \log n)$. Since there are $O(n^2)$ node pairs, the total complexity of this phase if $O(n^3 \log n)$.
Determining A_k: According to Corollary 16, A_k can be determined in $O(n^2 m \log m)$.
Processing apples: In Cor. 19 it was shown that all of the apples can be processed in $O(n^2m^3)$.
Merging lists: M_k can be obtained by deleting the non-maximal and redundant elements while comparing all the possible list pairs $\{M_k^{u_1,v_1}, M_k^{u_2,v_2}\}$, then concatenating the remaining sets. There are $O(n^2)$ apples, and therefore $O(n^2)$ $M_k^{u,v}$ lists with at most m elements containing at most m edges (thus M_k has $O(n^2m)$ elements, and a total length of $O(n^2m^2)$). This means $O(n^4m^2)$ comparisons and deletions. Since both comparision and deletion can be done in $O(m)$, the total complexity of this phase is $O(n^4m^3)$.
We conclude that Algorithm 3 computes M_k in $O(n^4m^3)$. □

In the following sections we will improve this algorithm. Note that in Table 1 the obtained time complexities are summarized.

4 Improved Algorithm for Enumerating Maximal Failures

In this section Algorithm 3 will be improved while keeping its four phases. These phases either will be improved, or their complexity will be better estimated.

4.1 Determining E_k Faster

Claim 21. *Decision whether a node pair $\{u,v\} \in E_k$ can be made in $O(n + k \log k)$.*

Proof. For a given node pair $u,v \in V$, let V_+ and V_- be as in Subsubsec. 3.3.2. A useful observation is that based on Prop. 12, there is no need to determine the whole lists V_+ and V_-, we only need to store and sort the k elements of both of these lists with the biggest subtending angles $\sphericalangle(u.v)$ in $V_{+,k}$ and $V_{-,k}$ (if there is no k elements right from uv, then let $V_{+,k}$ be V_+, same for left side). Sweep circle algorithm described in Subsubsec. 3.3.2 run with $V_{+,k}$ and $V_{-,k}$ can be used for deciding if $\{u,v\} \in E_k$, having a complexity of $O(k)$.
It remains to prove that $V_{+,k}$ and $V_{-,k}$ can be determined in $O(n + k \log k)$. Node $v_{+,k}$ with the k. biggest viewing angle from V_+ can be determined in $O(n)$ from V, same for $v_{-,k}$, V_-. After this, set containing nodes from V_+ with the k biggest viewing angle can be calculated in $O(n)$ too, same

for V_-. $V_{+,k}$ and $V_{-,k}$ can be obtained by sorting the two determined sets in $O(k \log k)$. □

Lemma 22. *E_k can be determined in $O(n^2(n+k\log k))$.*

Proof. The proof is straightforward from Claim 21 and the fact that there are $O(n^2)$ node pairs to examine. □

4.2 Better Complexity Bounds for Determining Apples

Up to this point the fact that $G(V,E)$ is a graph of a communication network, and thus it is 'almost planar' was not used. Intuitively, an almost planar graph has $O(n)$ edges. In the followings this will be formalised with the help of a graph density parameter.

Definition 23. *For all $i \in \{0, n-2\}$, let θ_i be the maximum number of edges hit by a circle from $\mathscr{C}_i$.*

Since parameters θ_i measure local properties of the networks, often it will be assumed that these parameters are not exceeding a constant. For example θ_0 is not going to be large, since where there are many links, it is likely to appear a node too.

Observation 24. *For any $0 \le i < j \le n-2$, $\theta_i \le \theta_j$.*

Claim 25. *In any apple $a \in A_k$ there are at most $2\theta_k$ edges.*

Proof. All edges in a are hit by either c_+ or c_-, which together hit at most $2\theta_k$ edges. □

Lemma 26. *The number of edges is $O(n\theta_0)$, more precisely $m \le (2n-5)\theta_0$.*

Proof. Consider set E_0, i.e. the set of node pairs $\{u,v\}$ for which $\mathscr{C}_0^{u,v}$ is not empty, in other words there exists a circle not hitting any node and having u and v on its boundary. It can be observed that no two edges from E_0 are intersecting each other in inner points, and that the edges of E_0 generate a triangulation $D_0(V,E_0)$ of V. In fact, this particular triangulation is the so-called Delaunay triangulation [14] of node set V. A triangle in D_0 is called a Delunay triangle.
The Delaunay triangulation D_0 is a planar graph, thus $|E_0| \le 3n-6$. Since every Delunay triangle has 3 Delaunay edges and a Delaunay edge is the edge of at most 2 Delaunay triangles, and there are at least 3 Delaunay edges on the convex hull of V, the number of Delaunay triangles is at most

$$\frac{2|E_0|-3}{3} \le \frac{2}{3}(3n-6)-1 = 2n-5.$$

Since every edge intersects at least one triangle, and every triangle can be covered by a circle $c \in \mathscr{C}_0$, which intersects at most θ_0 edges of the network, we get that the number m of edges cannot be larger than θ_0 times the number of the Delaunay triangles. We get $m \le (2n-5)\theta_0$. □

Lemma 27. *Set A_k of apples can be calculated in $O(|E_k|(n\theta_0 + \theta_k \log \theta_k)$.*

Proof. There are $|E_k|$ apples to determine. For each, $O(m)$, or alternatively in $O(n\theta_0)$ edges have to be checked if they are in the apple. After this, based on Claim 25, there are $O(\theta_k)$ edges to order, which gives the proposed complexity. □

Corollary 28. *If θ_0 is upper bounded by a constant, then A_k can be determined in $O(|E_k|n)$.* □

4.3 Storing $M_k^{u,v}$ more Economically and Better Bound for Processing Apples

One can observe that in case of an apple $A_k^{u,v}$, a list of edges can be constructed such that every $f \in M_k^{u,v}$ forms an interval in this list, and this way $M_k^{u,v}$ can be stored more economically.

Definition 29. *For an apple $A_k^{u,v}$ let L_k be the concatenation of list E_+ and list E_- minus those edges from E_- which intersect segment $[uv]$.*

Claim 30. *For a nonempty apple $A_k^{u,v}$, $M_k^{u,v}$ can be stored in $O(\theta_k)$ space.*

Proof. It can be observed that every $f \in M_k^{u,v}$ forms a list in L_k, this way after storing L_k, for identifying f it is only needed to store the index of 'beginning' and 'end'of f. Since $M_k^{u,v}$ has at most $|L_k|$ elements, for storing $M_k^{u,v}$ we only have to store at most $|L_k|$ index pairs beside storing $|L_k|$, which means $O(|L_k|)$ space.
Since all the edges in L_k can be hit with at least one of c_+ and c_-, and every edge e appears at most 2 times in the list (it can appear twice only if it is hit by both c_+ and c_-), $|L_k|$ is at most $2\theta_k$. This means that $M_k^{u,v}$ can be stored in $O(\theta_k)$. □

Cor. 19 gave a complexity bound on processing apples. Now it can be rephrased applying the new notions.

Lemma 31. *All the apples from A_k can be processed in $O(|E_k|\theta_k^2)$.*

Proof. There are $|E_k|$ apples to process. Each of them can be processed in θ_k^2 time by constructing $M_k^{u,v\prime}$ while scanning through $\mathscr{C}_k^{u,v}$, then eliminating its nonmaximal and redundant elements. Note that if the elements of $M_k^{u,v\prime}$ are stored by noting the indexes of their first and last edges, each pair of elements from $M_k^{u,v\prime}$ can be compared in $O(1)$. □

4.4 Better Complexity Bound on Merging Lists $M_k^{u,v}$

Lemma 32. *M_k can be computed in $O(|E_k|^2\theta_k^3)$ from lists $M_k^{u,v}$.*

Proof. There are $|E_k|$ lists containing $O(\theta_k)$ sets containing $O(\theta_k)$ edges. M_k can be computed by comparing all the set pairs (and eliminating the redundant or nonmaximal elements), which means $O(|E_k|^2\theta_k^2)$ comparisions. Since comparing two sets takes $O(\theta_k)$ time, the total complexity is $O(|E_k|^2\theta_k^3)$. □

4.5 Complexity of an Improved Algorithm for Computing M_k

Theorem 33 gives complexity bound on an improved algorithm for computing M_k.

Theorem 33. *M_k can be computed in $O(n^2(n+k\log k)+|E_k|n\theta_0+|E_k|^2\theta_k^3)$. M_k has $O(|E_k|\theta_k)$ elements with at most θ_k edges, and can be stored in $O(|E_k|\theta_k)$ space.*

Proof. As presented perviously in this section (in Lemmas 22, 27, 31 and 32), each of the four phases of Alg. 3 can be examinated in the proposed complexity. There are $|E_k|$ lists $M_k^{u,v}$ to merge, each of them has at most θ_k edges, and according to Claim 30, each $M_k^{u,v}$ can be stored in $O(\theta_k)$, completing the proof. □

Corollary 34. *If θ_k is upper bounded by a constant, then M_k can be computed in $O(n^2(n+k\log k)+|E_k|n+|E_k|^2)$. M_k has $O(|E_k|)$ elements, and can be stored in $O(|E_k|)$ space.*

	Naive	Improved
Determining E_k	$O(n^3\log n)$	$O(n^2(n+k\log k))$
Determining apples	$O(n^2m\log m)$	$O(\|E_k\|(n\theta_0+\theta_k\log\theta_k))$
Processing apples	$O(n^2m^3)$	$O(\|E_k\|\theta_k^2)$
Merging lists	$O(n^4m^3)$	$O(\|E_k\|^2\theta_k^3)$
Total complexity	$O(n^4m^3)$	$O(\dots)$
T. c. if $\theta_i=O(1)$		$O(n^2(n+k\log k)+\|E_k\|^2)$

Table 1 Time complexities of naive and improved algorithms for computing M_k

As it was seen, $|E_0|<3n$, this way M_0 can be computed in $O(n^3)$. It can be proven (like in [13]) that $|E_1|<6n$, this way M_1 also can be computed in $O(n^3)$. Furthermore, according to simulation results, $|E_k|<3(k+1)n$ (Obs. 40). Based on these it worths to formulate the next corollary.

Corollary 35. *If θ_k is upper bounded by a constant, $|E_k|$ is $O(n^{1.5})$ and k is $O(\frac{n}{\log n})$ then M_k can be computed in $O(n^3)$.*

There are possibilities to write even faster algorithms for smaller values of k. In [12] and [13] we presented algorithms which compute M_0 and M_1 in $O(n\log n)$ and $O(n^2)$, respectively, if some conditions hold in a very similar mathematical model. Proposing similar algorithms would exceed the limits of this paper.

5 On Graph Delaunay-k and Calculating $M_0, M_1, \dots, M_k$ Simultaneously

As mentioned before, graph $D_0(V,E_0)$ is the Delaunay triangulation of node set V, therefore it is a plane graph.

Claim 36. *For every $k\in\{1,\dots,\lceil n/2\rceil-1\}$, $E_{k-1}\subseteq E_k$.*

Proof. Let $\{u,v\}\in E_i$ for an $i<k$. While sweeping through $\mathscr{C}^{u,v}$ with a circle c, nodes are getting hit and uncovered, but maximum one at a time, this way the number of hit nodes $\#_c$ changes one-by-one while sweeping.
Since $\{u,v\}\in E_i$, $\#_c$ can drop to i. On the other hand, since either h_+ or h_- hits at least $\lceil n/2\rceil-1$ nodes, thus $\#_c$ reaches $\lceil n/2\rceil-1$. Since $i<k\le\lceil n/2\rceil-1$, this means that there exists a moment when $\#_c=k$, thus $\{u,v\}\in E_k$. □

Claim 37. *Graph $D_{\lceil n/2\rceil-1}$ is the complete graph on V.*

Proof. Let $\{u,v\}\subseteq V$. Let the number of nodes hit by h_+ and h_- be $\#_{h_+}$ and $\#_{h_-}$, respectively. Clearly, $\#_{h_+}+\#_{h_-}=n-2$, and a node from $V\setminus\{u,v\}$ is hit by h_+ iff it is not hit by h_-, this way it can be assumed w.l.o.g. that $\#_{h_+}\ge\lceil n/2\rceil-1$ and $\#_{h_-}\le\lceil n/2\rceil-1$. This way while sweeping from h_+ to h_- in $\mathscr{C}^{u,v}$, there exists a moment, when the number of hit nodes equals $\lceil n/2\rceil-1$, thus $\{u,v\}\in E_{\lceil n/2\rceil-1}$ □

Definition 38. *Let $M_{0,k}$ be the concatenation of $M_0,M_1,\dots,M_k$.*

As it turns out, calculating $M_{0,k}$ is not harder than calculating M_k:

Theorem 39. *If $k\le\lceil n/2\rceil-1$, then $M_{0,k}$ can be computed in $O(n^2(n+k\log k)+|E_k|n\theta_0+|E_k|^2\theta_k^3)$. $M_{0,k}$ has $O(|E_k|\theta_k)$ elements with at most θ_k edges, and can be stored in $O(|E_k|\theta_k)$ space.*

Proof. To calculate $M_{0,k}$, we only need a small modification of the improved algorithm from Section 4. First, E_k, then A_k should be determined as in phases 1 and 2 in the original algorithm. Processing an apple $A_k^{u,v}$ should be modified: while a single sweeping we construct all of $M_0^{u,v},M_1^{u,v},\dots,M_k^{u,v}$; it can be proved that this is still possible in the same complexity as proven for phase 3 in the original improved algorithm (Lemma 31). Construction of lists $M_0,\dots,M_k$ by merging, then concatenation can be also done in the same complexity proven for original phase 4 in Lemma 32. □

6 Simulation Results

In this section we focus on the number of edges of graph Delaunay-k. As seen in Fig. 1, graph D_0 is a planar graph, this way $|E_0|\le3n-6$. As proven in [13], $|E_1|\le6n-15$. We deduct that $|E_k|<3(k+1)n$ for $k=0$ and $k=1$. However we coud not prove it mathematically yet, simulation results show that that $|E_k|<3(k+1)n$ holds for greater values of k too, as shows Fig. 3. Note that if a graph has an average nodal degree d, than it has $m=nd/2$ edges.
While increasing k, the linear growth of $|E_k|$ slightly slows as in Fig. 4, and finally, at $k=\lceil n/2\rceil-1$, $D_{\lceil n/2\rceil-1}$ becomes the complete graph on V, as proven in Claim 37. If k is increased further from $\lceil n/2\rceil-1$, another process starts: edges are dropping out of D_k causing a linear decrease in $|E_k|$.
Let us formalise some of these observations and their corollaries.

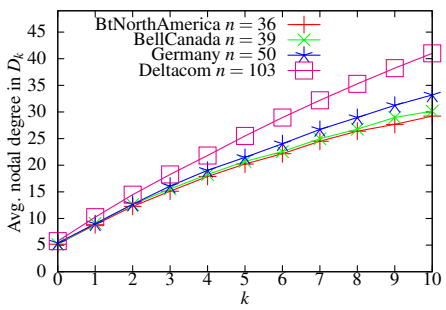


Figure 3 Average nodal degrees in D_k for small k values

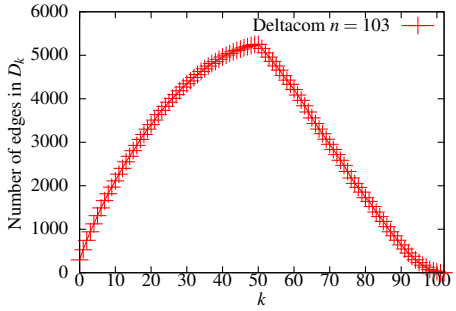


Figure 4 $|E_k|$ for all possible k values

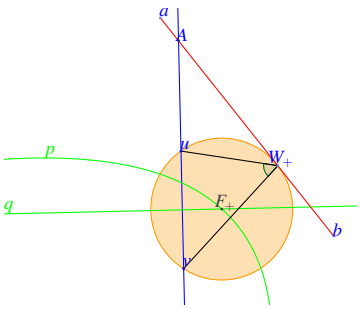


Figure 5 Illustration for Subsec. 8.1

Observation 40. *According to simulation results, $|E_k|$ is $O((k+1)n)$, more precisely $|E_k| < 3(k+1)n$.* □

Using Obs. 40, the complexity results could be rephrased. Instead of this we show most of these results in a tabular form in Table 2.

	Improved	Improved + "$|E_k| < 3(k+1)n$"
Determining E_k	$O(n^2(n+k\log k))$	$O(n^2(n+k\log k))$
Determining apples	$O(\|E_k\|(n\theta_0+\theta_k\log\theta_k))$	$O(n(k+1)(n\theta_0+\theta_k\log\theta_k))$
Processing apples	$O(\|E_k\|\theta_k^2)$	$O(n(k+1)\theta_k^2)$
Merging lists	$O(\|E_k\|^2\theta_k^3)$	$O(n^2(k+1)^2\theta_k^3)$
Total complexity	$O(\dots)$	$O(n^2(n+(k+1)^2\theta_k^3))$

Table 2 Time complexity of the improved algorithm for computing M_k with and without using $|E_k| < 3(k+1)n$

Using simulation resulsts Theorem 39 can be written as follows:

Claim 41. *If $k \le \lceil n/2 \rceil - 1$, then $M_{0,k}$ can be computed in $O(n^2(n+(k+1)^2\theta_k^3))$. $M_{0,k}$ has $O(n(k+1)\theta_k)$ elements with at most θ_k edges, and can be stored in $O(n(k+1)\theta_k)$ space.* □

Corollary 42. *If $k \le \lceil n/2 \rceil - 1$ and θ_k is upper bounded by a constant, then $M_{0,k}$ can be computed in $O(n^2(n+(k+1)^2))$. $M_{0,k}$ has $O(n(k+1))$ elements with $O(1)$ edges, and can be stored in $O(n(k+1))$ space.* □

7 Conclusions

In this paper we propose a fast and systematic approach to enumerate the list of possible circular disk failures having at most k nodes inside. Defining the size of the region as the number nodes inside is a very general failure model, which does not require the knowledge of absolute coordinates and the physical distances in the network topology. Note that, we do not assume the failed region has a disk shape, instead this modelling technique overestimates the size of the failed region in order to have tractable problem space. Roughly speaking, by setting a too large maximum radius, we require to have larger distances between the working and backup resources, than it is actually needed in reality. However, adopting a failure model that captures the number of nodes of the failed region results in model manageable list of SRGLs. This opens up a straightforward way of protecting regional failures by simply configuring them as a list of SRLGs for the current self-healing mechanisms.

Acknowledgement

This article is based upon work from COST Action CA15127 ("Resilient communication services protecting end-user applications from disaster-based failures - RECODIS") supported by COST (European Cooperation in Science and Technology). J.T. is partially supported by the Hungarian Scientific Research Fund (grant No. OTKA 108947) E.B.-K. is partially supported by OTKA grant K109240.

8 Literature

[1] S. Neumayer, G. Zussman, R. Cohen, and E. Modiano, "Assessing the vulnerability of the fiber infrastructure to disasters," *IEEE/ACM Transactions on Networking*, 2011.

[2] O. Gerstel, M. Jinno, A. Lord, and S. B. Yoo, "Elastic optical networking: A new dawn for the optical layer?" *IEEE Communications Magazine*, vol. 50, no. 2, 2012.

[3] J. Heidemann, L. Quan, and Y. Pradkin, *A preliminary analysis of network outages during hurricane sandy*. University of Southern California, 2012.

[4] I. B. B. Harter, D. Schupke, M. Hoffmann, G. Carle *et al.*, "Network virtualization for disaster resilience of cloud services," *IEEE Communications Magazine*, 2014.

[5] X. Long, D. Tipper, and T. Gomes, "Measuring the survivability of networks to geographic correlated failures," *Optical Switching and Networking*, 2014.

[6] B. Mukherjee, M. Habib, and F. Dikbiyik, "Network adaptability from disaster disruptions and cascading failures," *IEEE Communications Magazine*, 2014.

[7] R. Souza Couto, S. Secci, M. Mitre Campista, K. Costa, and L. Maciel, "Network design requirements for disaster resilience in iaas clouds," *IEEE Com. Magazine*, 2014.

[8] D. M. Masi, E. E. Smith, and M. J. Fischer, "Understanding and mitigating catastrophic disruption and attack," *Sigma Journal*, pp. 16–22, 2010.

[9] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and

survivability in communication networks: Strategies, principles, and survey of disciplines," *Comp. Networks*, 2010.

[10] S. Neumayer, A. Efrat, and E. Modiano, "Geographic max-flow and min-cut under a circular disk failure model," *Computer Networks*, vol. 77, pp. 117–127, 2015.

[11] J. Tapolcai, L. Rónyai, B. Vass, and L. Gyimóthi, "List of shared risk link groups representing regional failures with limited size," in *Proc. IEEE INFOCOM*, 2017.

[12] B. Vass, E. Bérczi-Kovács, and J. Tapolcai, "Shared risk link group enumeration of node excluding disaster failures," in *NaNA*, 2016.

[13] B. Vass, E. Bérczi-Kovács, and J. Tapolcai, "Enumerating circular disk failures covering a single node," in *RNDM*, 2016.

[14] F. Aurenhammer, "Voronoi diagrams: a survey of a fundamental geometric data structure," *ACM Computing Surveys (CSUR)*, vol. 23, no. 3, pp. 345–405, 1991.

Appendix

8.1 Determining $\sphericalangle_+^{u,v}(e)$ and $\sphericalangle_-^{u,v}(e)$ in Constant Time

Claim 43. *For any edge $e = \{a,b\}$ and node pair $\{u,v\} \subset V$, both $\sphericalangle_+^{u,v}(e)$ and $\sphericalangle_-^{u,v}(e)$ can be calucated in $O(1)$.*

Proof. Let us concentrate on calculation of $\sphericalangle_+^{u,v}(e)$, because $\sphericalangle_-^{u,v}(e)$ can be determined similarly.

Maximal subtended angle from line: First let us compute the maximal subtended angle by segment $[uv]$ from a point of line ab right from uv. Let $\{A\} = uv \cap ab$. Clearly, the subtending angle function $s_{u,v}(W) := \sphericalangle(uWv)$ is unimodal in both rays (R_+ and R_-) defined by line ab and point A (i.e. it has a strictly monotone increasing and a strictly monotone decreasing interval). Let the two points where local maximum is reached be W_+ on the right side and W_- on the left side of uv. Let the cetre point of circles $c(uvW_+)$ and $c(uvW_-)$ be F_+ and F_-, respectively. W_+ can be determined via determining F_+.

Since point F_+ is located equidistant from u and v, it is located on perpendicular bisector q of segment $[uv]$. On the other hand, F_+ is located equidistant from v and line ab, thus it is on parabola p defined by point v and line ab.

Since q can be described with a linear equasion and p with a quadratic one, determining their intersections can be made via solving a quadratic expression, which can be done in $O(1)$.

From this point it is easy to describe circles $c(uvW_+)$ and $c(uvW_-)$, and to determine and distinguish F_+ and F_- via solving quadratic expressions in constant time.

Maximal subtended angle from segment: Clearly, if $W_+ \in [AB]$, then $\sphericalangle_+^{u,v}(e) = \sphericalangle(uW_+v)$, since $s_{u,v}$ is unimodal in ray R_+. If it is not the case then $\sphericalangle_+^{u,v}(e) = \max(\sphericalangle(uAv), \sphericalangle(uBv))$, again due to unimodality. This way $\sphericalangle_+^{u,v}(e)$ can be determined in $O(1)$. □