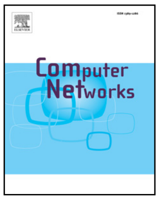
Computer Networks 128 (2017) 63–77

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Optimized repair of a partitioned network topology

Wassila Lalouani[a], Mohamed Younis[b,*], Nadjib Badache[c]

[a] High National School of Computing Science, Algiers, Algeria
[b] Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, MD, USA
[c] Department of Theories and Computer Engineering, CERIST, Algiers, Algeria

ARTICLE INFO

Article history:
Received 26 July 2016
Revised 18 January 2017
Accepted 6 February 2017
Available online 7 February 2017

Keywords:
Topology repair
Fault tolerance
Relay node placement
Connectivity restoration, ESTPO

ABSTRACT

We consider the problem of deploying the least count of relay nodes to restore connectivity in a network that got partitioned into multiple disjoint segments. Such a problem has been generally formalized as a Steiner Minimum Tree (SMT) by assuming that each segment is a terminal, e.g., by picking a single node in a segment to serve as an interface point. We argue that such formulation is ineffective since the size and the shape of the segment are not factored in. To overcome this drawback, we propose a novel approach for Boundary-aware optimized Interconnection of Disjoint segments (BIND). BIND opts to restore network connectivity by forming the shortest length topology in the Euclidean plane that interconnects a subset of nodes on the segment boundaries through extra Steiner points so that there is a path between every pair of segments. Since constructing the SMT connecting boundary nodes (terminals) subject to obstacle avoidance is NP-hard, BIND pursues a heuristic based on the generation and concatenation of full Steiner trees (FSTs). As the number of distinct FSTs grows exponentially with the number of terminals, BIND further promotes a new geosteiner technique based on the straight skeleton of the segment boundaries within the deployment area in order to reduce the complexity. The simulation results confirm the effectiveness of BIND and its advantage compared to competing schemes.

1. Introduction

In wireless networks that operate in harsh environments nodes may be susceptible to electronic breakdown and even damage. In these cases, the failure of nodes not only could degrade the coverage quality of the monitored area, but also break some communication paths among nodes. Example application scenarios include the deployment of a network to serve security surveillance of a vast border, search-and rescue in disaster area, reconnaissance of a combat field, etc. In addition, the node failure may be due to depleted energy supply or the small form-factor of the node design, e.g., the case of wireless sensor network (WSNs). When only a single node is lost, the effect on the network connectivity is minimal unless the failed node serves as a cut-vertex, i.e., gateway, in the topology causing the network to be split to disjoint segments. A single node failure often can be detected by neighboring nodes through the exchange of periodic heartbeat messages and failure tolerance can be autonomously achieved by the coordinated relocation of these neighboring nodes to restore the connectivity. However, such a recovery process is ineffective in case the scope of failure involves multiple collocated nodes. This failure scenario may be experienced in applications such as battlefield surveillance, forest fire detection, volcano monitoring, etc. where parts of the area get affected by an explosion, fire, lava, etc. Basically, the scope of the damage in this case is too broad to be determined by the healthy nodes in the vicinity and thus coordinated repositioning of some nodes will not be feasible as a recovery strategy due to the loss of many communication links [1].

Published schemes for overcoming the simultaneous failure of multiple collocated nodes mostly pursue centralized recovery strategies [2–18]. Basically, additional mobile nodes are deployed to serve as relay nodes among the isolated segments. The main optimization objective of the recovery process in such a case is to engage the least number of relay nodes in order to cope with resource availability and/or deployment constraints. In many studies, the problem of relay nodes placement has been formalized as a Steiner Minimum Tree (SMT) problem by representing each segment as a terminal, e.g., by picking a single node in a segment to serve as an interface point [2,11–18]. Since such problem formulation is NP-hard [20], heuristic solutions have been pursued. The most commonly used heuristics seek to establish the SMT by forming sets of connected components (each typically consists of three terminals) and then incrementally interconnecting them. We argue that this solution strategy is ineffective since the size and the

* Corresponding author.
E-mail addresses: w_lalouani@esi.dz (W. Lalouani), younis@cs.umbc.edu (M. Younis), badache@mail.cerist.dz (N. Badache).

http://dx.doi.org/10.1016/j.comnet.2017.02.003
1389-1286/© 2017 Elsevier B.V. All rights reserved.

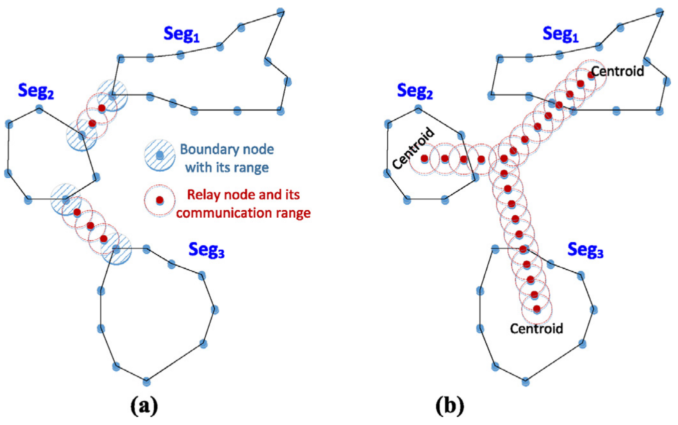

Seg₁
Seg₂
Seg₃
Boundary node with its range
Relay node and its communication range
(a)
Seg₁
Seg₂
Seg₃
Centroid
Centroid
Centroid
(b)

Fig. 1. Illustrating the dramatic effect of segment size and shape on the relay count required for establishing connectivity. A circle denotes the relay communication range; (a) the links between segments reflect the solution while factoring in the boundary nodes, where a total of 5 relays will be needed. Note that Seg_2 has two interface points. Seg_1 and Seg_3 are connected through a path that uses all new relay nodes and leverage the inter-segment links of Seg_2 (recall that segments are strongly connected; (b) an inter-segment topology linking the centroids of segments, i.e., SMT for the centroids, employs 23 relays to federate the network in this case. Obviously using a single interface point in this case is a costly solution.

shape of the segment are not factored in and the number of relays usually exceeds what is needed. Instead of having just a single segment interface, pairs of segments could be interconnected through different boundary nodes and fewer relays could be used, as illustrated by the example in Fig. 1. Furthermore, the pursued SMT heuristics do not enable optimized interconnection of the formed connected components. Basically, incremental SMT solution strategies found in the literature do not revisit the intra-component connectivity and thus do not allow updating how certain segments are connected even if a better way exists. Two components could conflict if they share at least one segment. Existing incremental SMT formation strategies do not provide an appropriate mechanism for the management of conflicts between connected components because they do not allow discarding a connected component. In other words, there is no guarantee that the connected components, i.e., subsets of segments, are complementary in the sense that the overall solution converges to an optimal Steiner tree. Although, the authors of [21,22] consider all nodes located on the boundary of the individual segments, they did not deal with this drawback.

To overcome the aforementioned shortcomings, this paper proposes BIND, a novel approach for Boundary-aware optimized Interconnection of Disjoint segments. In BIND the segments are represented by a simple polygon delimited by its boundary nodes. BIND opts to restore network connectivity by forming the shortest length topology in the Euclidean plane that interconnects a subset of nodes on the segment boundaries through extra Steiner points so that there is a path between every pair of segments. The problem could be seen as a variant of Euclidean Steiner Tree Problem with Obstacles (ESTPO). ESTPO is a Steiner tree constrained by the fact that the edges should not penetrate any given segment in the polygon P and may thus end at the boundary. In addition to the fact that the SMT is an NP-hard problem for a known set of terminals, ESTPO is further complicated by the fact that we should determine the best boundary nodes to serve as inter-segment gateways. A brute-force search that runs the ESTPO for each distinct set of boundary nodes is fundamentally a set partitioning problem and incurs excessive computational complexity that is exponential in the number of boundary nodes. Therefore, BIND first determines the SMT that spans all boundary nodes using the least number of relays and then retrieves the minimum cost tree that spans segments from the SMT. Hence, the solution consists of two main phases: (1) connecting boundary nodes, and (2) federating segments.

To connect boundary nodes in the first phase, BIND adopts an approach based on full Steiner trees (FSTs), which has been deemed the best approximate solution for Euclidean Steiner Minimal Tree problems [23]. The idea is to form all FSTs for small subsets of boundary nodes and then concatenate some of the generated FSTs such that an SMT is established for all nodes. The most prominent advantage of this approach is that federating the segments could be formalized as a hypergraph that includes all possible combination of terminals (boundary nodes) and thus conflicts between the FST of various terminals (connected components) could be considered. In other words, the hypergraph formalization will overcome the shortcoming of the existing recovery solutions. Since the number of distinct FSTs grows exponentially with the number of terminals, geosteiner techniques, e.g., [23], are often employed to reduce the complexity without changing the optimal solution. BIND promotes a new geosteiner technique based on the straight skeleton of the segment boundaries within the deployment area. Basically, the straight skeleton enables BIND to factor in the position of segments relative to one another, i.e., the visibility proprieties among obstacles in an ESTPO, and prevents intra-segment edges from being part of an FST. Finally, in the second phase of BIND, the SMT connecting boundary nodes is mapped to a hypergraph where boundary nodes serve as vertices with the objective of determining a subset of the SMT edges connecting the boundary nodes that interconnect the segments with the least cost in terms of the number of required relays. The effectiveness of BIND is validated through extensive simulation experiments. The validation results demonstrate that BIND significantly outperforms competing recovery schemes in the literature.

In summary, the paper makes the following contributions: (1) a novel formulation for the inter-segment connectivity problem that factors in the segment boundary; (2) a novel algorithm for federating disjoint segments using the fewest relays. The rest of the paper is organized as follows. The next section sets BIND apart from published schemes. Section 3 formally defines the problem and provides fundamental definitions. BIND is described in detail in Sections 4–6. Section 7 reports the validation results. Finally, the paper is concluded in Section 8.

2. Related work

Tolerating node failure in the context of wireless networks has received increased attention in recent years. The most notable efforts have been dedicated to wireless sensor networks given the harsh environment that they operate in. Although loss of area coverage has been considered, given the focus of BIND we shall limit the discussion to work that focuses on restoring severed connectivity due to node failure. According to Younis et al. [1], published failure recovery techniques can be classified into three categories. The first reestablishes communication links by repositioning a subset of the healthy nodes. In the second category additional resources, i.e. relays, are employed to form new paths. Approaches in the third category provision even more resources to establish k-connectivity so that the network operation does not get affected when a node fails. Another way to categorize existing work is based on the scope of the failure depending on whether a single or multiple nodes are lost. Tolerating single node failure has received the most attention. The loss of a node will cause the network to partition only if it serves as a cut-vertex in the topology. Tolerance techniques for single node failure have mainly focused on either provisioning resources or adapting the topology structure to mitigate the loss of critical nodes [24,25], or how the network self-heals without external coordination [26–31]. Basically the neighbors of a failed node detect and spearhead the recovery process.

For example Basu and Redi [26] and Tamura et al. [27] move the nodes in blocks while RIM [28], PADRA [29], LeDiR [30], and ACR [31] relocate nodes individually by exploiting cascading motion. The main objective considered by these techniques is to reduce the travel overhead and the number of involved nodes. Nevertheless, those solutions cannot be applied for simultaneous failure of multiple nodes since the healthy nodes cannot assess the scope of the damage that the network topology has suffered.

Approaches that handle simultaneous node failures can be grouped based on whether the lost nodes are collocated or dispersed. The latter resembles the case of multiple independent single-node failures when solved in a centralized manner [32–36]; however distributed solutions are complicated by the fact that resource conflicts may be experienced during autonomous recovery. Basically a node may find itself required to simultaneously move to two different locations to serve in the recovery from two distinct single-node failures. The use of synchronization tokens have been used to mitigate such a challenge [29,37]. On the other hand dealing with simultaneous failure of multiple collocated nodes proved to be the most challenging connectivity-loss scenario since the scope is hard to determine in a distributed manner. Recovery solutions falls into one of three categories, (1) node spreading, (2) touring segments to transfer data (data ferrying); and (3) relay node placement. The former relies on the fact that nodes can move and orchestrate network-wide motion in a known direction [38]. Meanwhile the use of mobile data collectors to transport data among segments is deemed a temporary recovery solution since it involves overhead and suffers increased data latency [12,39]. Repairing the damage through the deployment of relays has been more popular, especially when the inter-segment topology is expected to serve the application for an extended duration. Given the contribution of the paper, the focus of the rest of this section is on comparing BIND with prior work on restoring connectivity using relay node placement.

Multiple variants of the relay placement problem have been studied in the literature. The most popular problem model is to determine the least count and position of the relays to establish a connected topology of a set of terminals. For example, Cheng et al. consider a three-step heuristic [3], where nodes with distance less than *R* are connected first. Then, 3-stars are formed for each subset of three nodes that can have SMT using only one relay, i.e., Steiner point. Finally, they populate relays along the minimum spanning tree (*mst*) edges that are not connected by 3-stars. Lloyd and Xue [4] form an SMT by employing a Geometric Disk Cover algorithm to deploy the fewest relays such that each node is connected to at least one relay, and a connected inter-relay topology is established. Efrat et al. [5] model the network as unit disk graph where relays are positioned in the overlapping unit disk regions to connect the network. ORC [14] deploys relays inwards from the periphery of the area identified by the convex hull. FeSTA [15] computes the cost for all possible 3-tuples of terminals (triangles) and selects triangles that connects the segments by applying Discrete Fermat Point optimization. However, the FeSTA algorithm is very complex. A bio-inspired [16] heuristic named SpiderWeb targets network federation while achieving a better node coverage and enabling balanced distribution of traffic load on the employed relays. CORP [17] focuses on forming a highly-connected inter-segment topology where the node degree is more than two. The approach of Senel and Younis [19] consider Delaney triangulation and 3-stars for edges on the *mst* to optimize the selection of Steiner points. The approach matches the performance of FeSTA, which considers all combination of three terminals, in terms of the relay count, with a major drop in complexity. Nonetheless, all these solutions assume that a segment is represented by one terminal and do not factor the size and the shape of the segment. BIND formalizes the problem as an ESTPO where the position of segments relative to each other is the main criterion to restore the connectivity. Unlike [3,14–19], BIND does not restrict to only components of three.

Variants of the relay node placement optimization have also been considered. In [11–13], the connectivity objective is to be achieved under constrained relay count. Turjman et al. [11] propose a grid-based placement approach in which the grid vertices are considered as placeholders for relays to connect the disjointed segments; a subset of these vertices is then populated with relays such that the inter-segment connectivity is maximized using the available set of relays. In [13] the same authors considered a variant of the optimization where inter-segment data delivery delay is to be minimized instead. Meanwhile, the approach of [12] assumes insufficient relay count and uses a mix of stationary relay nodes and mobile data carriers to interconnect the segment. Optimized placement of relay nodes has also been used to support high degree of connectivity in order to tolerate occasional single node failures in the future. For example, Tang et al. [6], Han et al. [8], Li et al. [9], Chen et al. [10], and Lee and Younis [40,41] strive to form a bi-connected inter-segment topology using the fewest relays. Moreover, some work considered quality of service requirements for the inter-segment data paths; for example EQAR models the area as a grid and duplicates relays as needed in order to meet bandwidth requirements [42].

Some work pursues self-spreading of relay nodes to fill the void area that lost collocated nodes and then applies optimization procedure to limit the relay count. For example, DORMS [18] moves nodes from each segment toward the center of the deployment area to assess the damage. As soon as those relays come in range of each other, the partitioned segments resume operation. DORMS further models such initial inter-segment topology as a Steiner tree in order to minimize the count of required relays. Similar methodology with an enhanced optimization procedure is used in [38]. Meanwhile, AuR [43] expands segment boundaries and orchestrates block node movement towards the center in order for the segments to re-connect. The presence of obstacles in the area is factored in [44] when determining which nodes move and what path they should travel along. Senturk et al. [21] propose a game theoretic approach assuming the existence of cameras on mobile nodes. Boundary nodes move outward and use cameras to detect nodes in other segments. Another game-theoretic approach is proposed by the same authors in [22], where relays are deployed at the center of the area and a virtual force-based model is employed to guide the relays spreading to reach to the boundary of segments. The performance of BIND will be compared with that of this approach in Section 7.

Compared to the aforementioned approaches, BIND factors in the topological boundary of segments; we thus argue that BIND addresses the practical aspects of the recovery problem. We formulate the problem as ESTPO and use a FST-based methodology, which has been shown to be efficient for solving SMT problems. We further propose a geosteiner technique to expedite the convergence and lower the complexity of the recovery process.

3. System model and problem formualtion

3.1. System model

We consider a network of nodes that communicate over a single shared wireless channel. A wireless link can be established between a pair of nodes if they are within the radio range of each other. During operation, the network may suffer simultaneous failure of multiple collocated nodes due to external events such as a forest fire, an explosion in a combat field, or a sand storm, and consequently becomes partitioned into multiple disjoint segments. Fig. 2 shows an articulation. In such a failure scenario, the network may also experience degraded sensor coverage. However, the

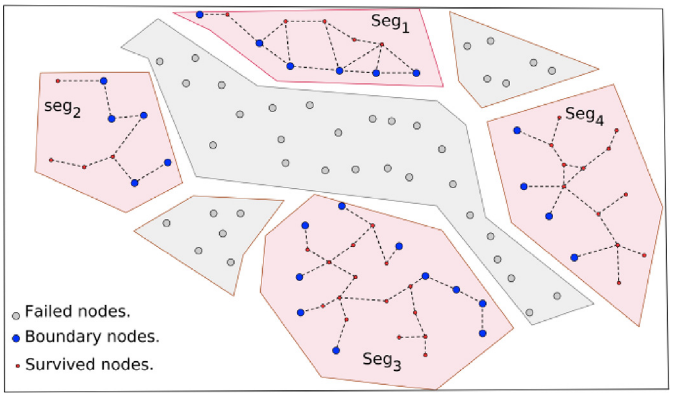


Fig. 2. An articulation of a sample damaged network topology.

focus of the paper is on how to restore connectivity using limited resources, in terms of additional relays that can be deployed or existing nodes that may be spared from the individual segments. Given the resource constraints, coverage may be viewed as secondary and establishing a connected inter-segment topology to enable data sharing among the surviving nodes is what matters the most.

Detecting the failure and assessing its scope can be done by surveying the area using a flying drone or in-situ robots that are equipped with cameras [21], or by sending a representative node from each segment to rendezvous at the center of the area if the network has mobile nodes [18]. In other words, the failure assessment can be conducted in a centralized or distributed manner depending on the application setup. To determine the boundary of segments, contour determination techniques like Voronoi, Alpha shapes, or Unit ball could be used; quite a few algorithms can be applied to do so in a distributed manner [45–47]. It is worth noting that BIND is applicable to other application scenarios in which multiple autonomous, i.e., independently-operating, networks that need to combine their capabilities or to support large-scale Internet-of-Things applications such as smart cities [48]. For example, multiple WSNs may need to collaborate in order to perform a critical task, or serve multiple users with diverse quality-of-information requirements [49].

To restore connectivity a set of relay nodes are to be employed and carefully placed in order to form a connected inter-segment topology. For stationary relays, we assume that they can be deterministically positioned in the area, either manually or using remotely-controlled robots. On the other hand, a relay could be a mobile node with significantly more energy reserve and could some navigation capabilities. In that case, relays may conduct damage assessment in a distributed manner and collectively execute BIND. Once the positions are determined the relays stay still and form a stable inter-segment data paths. Regardless of whether the relays are mobile or not, the number of engaged relays is to be minimized in order to limit the recovery cost and simplify the logistics. It is assumed that all deployed relays have the same communication range "R" that is equal to that of the network nodes. All communication links are assumed to be bidirectional. Moreover, the focus of the paper is on the algorithmic aspect of the recovery while assuming robust link-layer communication protocol. It is important to note that BIND does not consider the energy of relays and focuses only on the restoring connectivity in the fragmented network. Although selecting which node serves as a relay is beyond the scope of the paper, it is conceivable for these relays to be selected based on their energy reserve.

3.2. Problem definition and solution strategy

We consider a set of nodes S deployed in a harsh environment. Due to a major event in the area some nodes fail causing the network to split into multiple disjoint segments. While nodes within the same segment can reach each other, the large scale damage inflicted to the network prevents nodes in one segment from reaching others in different segments and thus become unaware of the extent of the failure. Like all failure recovery solutions, BIND assumes that the network topology will not change during the recovery process, e.g., additional nodes will fail until the recovery is complete. We assume that the deployment area is represented as a polygon P. A segment is a simple polygon inside P delimited by the convex hull of its nodes. A polygon is simple if no nonadjacent edges intersect. The objective of BIND is to form the shortest total length tree in the Euclidean plane that interconnects a subset of boundary nodes (referred to as terminals) with extra Steiner points such that no edge goes across a segment. As discussed in Section 1, the problem could be seen as a variant of Euclidean Steiner Tree problem with Obstacles (ESTPO). Nonetheless, the set of terminals is unknown, i.e., to be picked among the all boundary nodes, in our case. In other words, we should further determine the best boundary nodes to serve as inter-segment gateways. A brute-force search that solves the ESTPO for each distinct subset of boundary nodes would incur excessive computational complexity that is exponential in the number of boundary nodes. Therefore, our solution first determines the SMT for all boundary nodes using the least number of relays and then identifies the minimum cost tree that spans segments based on edges of such SMT. Basically, we formulate and solve the problem as ESTPO and then exploit the solution to determine the least cost inter-segment topology. Therefore, the recovery process consists of two main phases: (1) connecting boundary nodes, and (2) federating segments.

4. Connecting boundary nodes

The first phase of BIND opts to connect boundary nodes (terminals) subject to obstacle avoidance, which is in principle solving an ESTPO problem. Since the ESTPO problem is NP-hard [52], we pursue heuristics. Particularly, we follow a solution strategy based on the generation and concatenation of full Steiner trees (FSTs) which has been shown to be very successful for obstacle free SMT problems [51] and been recommended for solving ESTPO problems [52].

4.1. FST-based SMT solution

An FST is a topology of a non-degenerate SMT in which every terminal has a degree of one, i.e., a leaf in the tree. Generally, a geometric network is called non-degenerate if all edges in the embedding have non-zero length [23]. An FST-based SMT approach contains two phases, namely, the FST generation phase and the FST concatenation phase under the condition that the FSTs concatenation should yield an SMT. The FST generation in essence forms connected components, each of which is SMT for the largest subsets of terminals while meeting the condition that every terminal is a leaf. In other words, each FST is the SMT of a subset of terminals. Since none of these connected components will include all terminals, the FST concatenation step opts to select some of the formed connected components to concatenate in order to form the SMT for all terminals. Thus, the FST-based approach reduces the formation of SMT to a combinatorial problem.

Warme [53] has observed that the FST concatenation can be reduced to finding a minimum spanning tree in a hypergraph whose vertices are the terminals and whose hyper-edges correspond to the generated FSTs. Fig. 3 illustrates the idea behind the FST-based

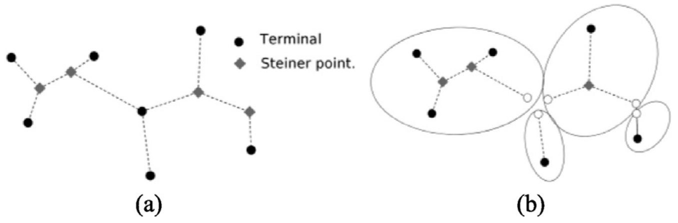


Fig. 3. (a) SMT tree of a set of terminals (dots); (b) The connected components (FSTs) of the SMT in part (a).

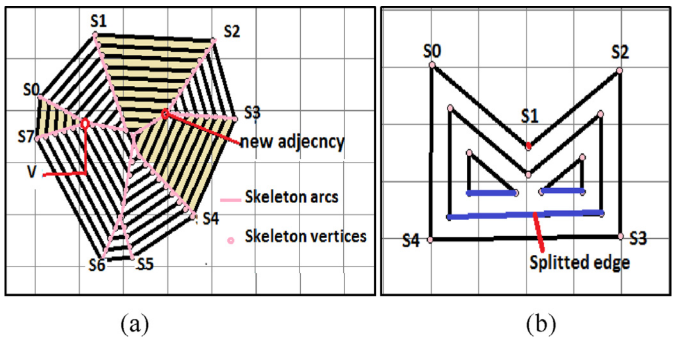


Fig. 4. Illustrating the wavefront propagation process to form straight skeleton; (a) Edge Event; (b) Split Event.

approach. Fig. 3(a) shows an SMT for an example set of terminals represented by circles. Fig. 3(b) reflects a decomposition of the SMT of Fig. 3(a) into FSTs, i.e., sub-Steiner trees. Those FSTs could be seen as edges of a hypergraph whose concatenation forms the SMT of Fig. 3(a). Although, the FST-based approach reduces the complexity of the ESTPO problem, the challenge imposed by such an approach is that the number of full components increases rapidly as the number of terminals increases.

Lemma 1 [22]**.** *The number of distinct full Steiner topologies (full components) for n terminals is:* $\frac{(2n-4)!}{(n-2)!*2^{n-2}}$, *where* $n > 3$.

Lemma 1 shows that the number of distinct FSTs grows super-exponentially with the number of terminals, which makes it impractical for large problems to consider all possible FSTs in the generation and concatenation steps. Geosteiner techniques [51] are usually applied to reduce the number of FSTs through factoring in geometric properties. The idea is to apply geometric techniques to reduce the size of the problem with a guarantee that they preserve the optimal solution within the search space. Basically, according to the geometric disposition of the boundary nodes, it may be clear that certain pairs of FSTs cannot appear together in the same SMT. Most approaches in the literature, for example Brazil and Zachariasen [23], deal with such a problem by discarding the full Steiner trees that go across an obstacle. BIND promotes a new geosteiner approach that generates only useful FSTs to concatenate them directly rather than determining all possible FSTs and then discarding those that traverse an obstacle (segment) or those that can be better connected through other FSTs.

Fundamentally, BIND deals with the following challenges: (1) the selection of set of terminals that should be within the same FST; picking these terminals, which are referred to as "neighboring terminals", should factor in the presence of obstacles and the disposition of terminals relative to one another. In contrary to most of existing geosteiner techniques that identify inutile FSTs based on distance between terminals, we use a unified structure, namely the straight skeleton, that avoid such a complex combinatorial problem; (2) placing Steiner points should be only outside segments. Obviously, we do not want to add redundant nodes inside an already-connected segment; (3) the edges connecting terminals to the Steiner points should be inside the polygon P (i.e., the network area) and should not traverse the interior of any segment in order to connect terminals. This avoids the consideration of long path (edges connecting terminals) that cannot be part of the optimal solutions.

These three challenges are imposed by both the terminals (boundary nodes) dispositions over the deployment area and the position of segments relative to one another, i.e., the visibility proprieties among obstacles in an ESTPO, which should prevent intra-segment edges and Steiner points from being part of an FST. In geometrical terms, we would like to form a structure that does not penetrate inside segments. Furthermore, such a structure should respect the visibility propriety without incurring additional computation complexity, e.g., computing the visibility graph of the set of terminals [50]. To address the abovementioned challenges, BIND promotes the use of straight skeleton as an FST generation technique. In the next subsection, we will explain how the straight skeleton could be used to determine the set of neighboring terminals, which should form FSTs. Section 6 shows how the skeleton preserves the visibility propriety.

4.2. FST generation using the straight skeleton

4.2.1. Straight skeleton

The definition of the straight skeleton is based on the so-called wavefront propagation process [54], where the boundaries of obstacles (segments) expand outward at a constant speed. The wavefront vertices determine the straight-line segments known also as arcs and the position of the topological changes of the wavefront. The topological changes are due to two types of events, namely, edge events and split events. An edge event occurs when an edge shrinks to zero making its neighboring edges adjacent (i.e., share a vertex). As shown in Fig. 4(a), the edge event for $\overline{S_0S_7}$ creates new adjacencies between $\overline{S_0S_1}$ and $\overline{S_6S_7}$ at the skeleton vertex "v". A reflex vertex within the polygon P generates split event when meeting an edge during the wavefront propagation process. The split event partitions the whole polygon. In the Fig. 4(b), the edge $\overline{S_3S_4}$ is split by S_1. New adjacencies occur between the split edge vertices and the reflex vertex (blue thick lines on Fig. 4(b)).

Lemma 2. [54]. *Let P be a simple polygon with n vertices and SS(P) be its straight skeleton, the following hold: (1) The straight skeleton divides the interior of P into n polygonal faces, one for each edge of P, which is known as its base; (2) The polygonal face having an edge "e" as its base, tracks all events that split the edge and the two resulting split edges.*

In fact, the face $F(e)$ of edge "e" starts at each edge and completes when "e" (and its split edges) has shrunk to zero. Three of the formed faces are highlighted in Fig. 4(a). Therefore, the face captures all changes of "e" until it disappears. In other words, the changes of each edge could be tracked by analyzing its faces. From the aforementioned, we can conclude that the topological changes are defined based on meeting edges of internal segments within P (hole edges), as the speed of the wavefront is constant and is the same of all edges. Thus, the straight skeleton preserves the topology of the original polygon due to the fact that it tracks those topological changes during the wavefront movement. As explained next, BIND exploits these changes to determine the neighboring terminals, i.e., the set of terminals that could form a connected component.

4.2.2. Determining unessential FST

BIND opts to reduce the complexity of forming FSTs by avoiding those FSTs that may not be part of the SMT solution of the ESTPO

```
1. For each boundary node $S_i$:
2. Neighbor[Si]=[Si-1,Si+1]
3. While ($\overline{S_\iota S_{\iota-1}}$ and $\overline{S_\iota S_{\iota+1}}$ do not shrink to zero):
4.     Expand $\overline{S_\iota S_{\iota-1}}$ and $\overline{S_\iota S_{\iota+1}}$.
5.     If a vertex v split $\overline{S_\iota S_{\iota-1}}$ (resp. $\overline{S_\iota S_{\iota+1}}$)
6.         Neighbor[Si ]= Neighbor[Si ]+{v}
7.         create skeleton point v.
8.     if $\overline{S_j S_k}$ shrink to zero and $\overline{S_j S_\iota}$ exists
9.         Neighbor[Si ]= Neighbor[Si ]+{Sk}
10.    If Si splits $\overline{S_j S_k}$:
11.        Neighbor[Si ]= Neighbor[Si ]+{Sj, Sk}
12.    If Si belongs to a triangle that shrinks to one vertex
         //multiple edge shrinking to zero in the same wavefront
13.        Add all nodes in the edge in consideration to the list.
14  End
```

Fig. 5. Pseudo code for neighboring set determination process to determine appropriate terminals for forming FST.

problem. To do so, BIND factors in the visibility property, i.e., the open space between segments and exploits the fact that the presence of an edge implies that the end terminals are part of the same segment and consequently connected through intra-segment links. Thus, an edge $\overline{S_iS_j}$ between S_i and S_j that shrinks to zero-length by an edge event before meeting another edge implies that S_i and S_j are not visible to another segment and will not be together in the SMT, i.e., not part of the optimal solution of the ESTPO problem. In other words, the edge $\overline{S_iS_j}$ is between two boundary nodes of the same segment; thus S_i and S_j are already connected, and are not part of a polygonal face that involves edges from other segments, i.e., the edge $\overline{S_iS_j}$ is not essential in connecting their segment to another segment. BIND employs this fact in selecting what terminals should be together in an FST.

First, BIND defines the notion of neighboring terminals. Two terminals S_i and S_j are neighbors if, (1) they are part of the same face, and (2) $\overline{S_iS_j}$ does not shrink on a wavefront until vanishing without meeting the wavefront of another edge, or an edge of S_i meets the wavefront of an edge of S_j. Therefore, BIND constructs a neighboring list for each terminal S_i by tracking the different events that an edge of S_i encounters during the wavefront movement. To elaborate, a node is ware of its neighbors on the segment boundary and augments its neighbor list when it gets in contact with other nodes during the wavefront motion. The use of the neighbor lists enables BIND to decompose the set of terminals into a maximum of n small adjacency lists, each corresponds to the neighbors of a terminal S_i. Each adjacency list will be considered for a FST. Thus, rather than considering the construction of all FSTs (combination of each subset of terminals), BIND automatically excludes the set of terminals that could not be into the same FST. Consequently, the complexity of FST generation step is reduced to only FSTs for the n small adjacency lists, as will be also shown in Section 6. The pseudo code, in Fig. 5, illustrates how to determine the adjacency sets of individual terminals.

4.2.3. Determining the FSTs

The objective of this step is to form the FST for each adjacency list, i.e., the SMT of a node and its neighbors. To do so, BIND considers the polygon delimited by the terminals of the adjacency list; we refer to such polygon as the Skeleton hull of these terminals. In fact, the skeleton hull is delimited by the base of neighbor faces and does not contain any terminals. The following definitions opts to relate the Skeleton hull to known concepts in FST formation.

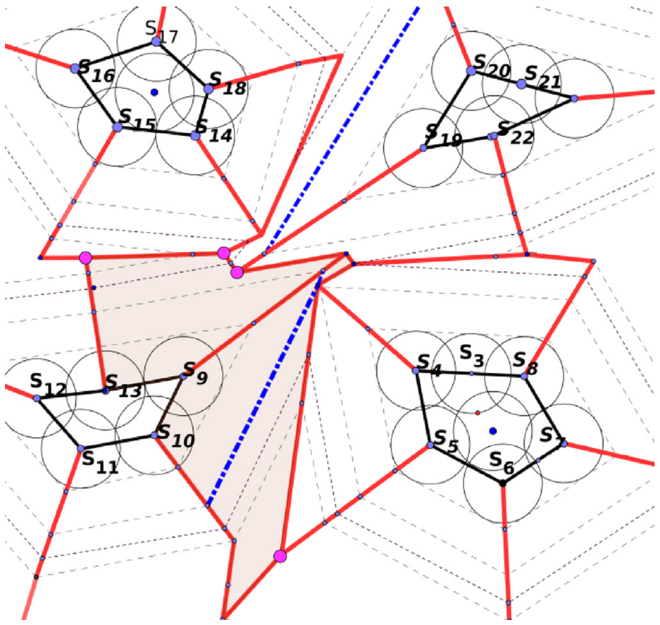


Fig. 6. An illustrative example for how the straight skeleton is used in BIND to define faces and Skeleton hulls. The red lines reflect the boundary of faces (straight skeleton arcs). The highlighted area is the union of faces of node S_9.

Definition. [Steiner hull] [23]. Given a set of n terminals, a Steiner hull is a bounded region of the plane containing every SMT of all n terminals.

Theorem 1. [55]. *The convex hull is a Steiner hull.*

Definition. [Steiner polygon] [55]. It is the Steiner hull, for which all terminals lie on the hull boundary.

The Steiner polygon concept is the basis of widely-used FST formation algorithms. As shown in [56], SMT decomposition based on a Steiner polygon criterion achieves major reduction in computation time. Because the super-polynomial nature of the Euclidean Steiner tree problem, a divide-and-conquer strategy is very effective in limiting the search space by solving smaller problems. Since the skeleton hull does not contain any terminal and all terminals lie on its boundary, it is considered a Steiner polygon and consequently Steiner polygon based FST formation algorithms can be applied to the Skeleton hull. BIND exploits such a property to determine the FSTs. Basically, BIND applies the approach of Warme et al. [51], which assumes convex Steiner polygons. When the Skeleton hull is convex, it reflects the convex hull of terminals (of an adjacency list) and the FSTs can be formed using [51]. Meanwhile, when the polygon is concave, BIND cuts the polygon at each concave vertex along a segment to divide the interior angle into two convex (less than a straight angle) angles [57]. Once the polygons are convex, we apply the FST formation algorithm to each of them individually. In the extreme case, the concave polygon will be dissected to finite number of triangles. The FST concatenation step enables the establishment of an ESTPO of the segments as will be discussed in the next section.

4.2.4. Detailed illustrative example

To illustrate the idea we refer to the example in Fig. 6, which includes four segments represented by simple polygons. According to the implementation of the straight skeleton [54], the segment edges move at constant speed outward according to the angular bisectors. We highlight how BIND operates using terminal S_9. To determine the set of neighbors for S_9, we follow the straight skele-

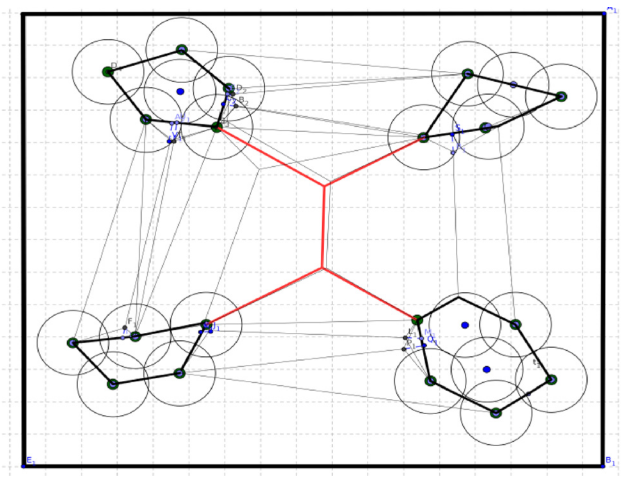


Fig. 7. Illustrating the segment federation phase, where the red lines are for the *mst* of the formed hypergraph and in essence reflect the final SMT solution of the ESTPO problem.

ton within the highlighted area, which reflects the two faces corresponding to edges $\overline{S_9S_{10}}$ and $\overline{S_9S_{13}}$.

Considering $\overline{S_9S_{10}}$, this edge is first split by node S_4 creating the edge $\overline{S_4S_9}$. S_9 in turns splits the edge $\overline{S_{19}S_{22}}$ into two edges, namely, $\overline{S_9S_{19}}$ and $\overline{S_9S_{22}}$, where the edge $\overline{S_9S_{19}}$ is also connected to S_{14}. The unused edges are shown with dashed blue lines in the figure. During the propagation of those edges, the edge $\overline{S_{19}S_{14}}$ shrink to zero-length making S_9 and S_{14} adjacent. Likewise, we form the adjacency list of S_9 in order to determine its neighbor set. In Fig. 6, the nodes S_4, S_{19}, S_{14} and S_{22}} constitute the neighbors of S_9. The polygon {S_4, S_{19}, S_{14}, S_9, S_{22}} is concave. When decomposing the polygon, we get two convex polygons: {S_4, S_{19}, S_{14}, S_9} and {S_4, S_{19}, S_{22}}. When applying the FST formation algorithm for each convex polygon, we get two FSTs, namely, {S_4, S_{19}, S_{14}, S_9} and {S_{19}, S_{22}}. Similarly, we can determine the FSTs of all terminals, as shown in Fig. 6. The effectiveness of the straight skeleton as geosteiner techniques will be shown in the analysis part where we demonstrate that the skeleton does not miss any utile FST.

5. Optimized segment federation

The objective of this phase is to concatenate the FSTs in order to form an SMT that spans all segments. As pointed by Warme [53], the FST concatenation step can be reduced to finding a minimum spanning tree (*mst*) in a hypergraph whose vertices are the terminals and whose hyper-edges correspond to the generated FSTs. The FSTs are represented using a hypergraph rather than a simple graph as the cardinality of the edge (FST) could be more than 2 as shown in the Fig. 7. The idea is based on the fact that *mst* in a hypergraph has the same key property of a spanning tree in an ordinary graph—namely that there is a unique path in the tree between every pair of vertices spanned by the tree. Therefore, an *mst* of hyper-edges concatenates the appropriate set of FST in such way that it spans all terminals and that any two distinct hyper-edges in a tree contain at most one common vertex (terminal).

However, BIND aims to find the SMT that spans segments with appropriate set of boundaries nodes (terminals). The main difference from contemporary FST concatenation problems is that in BIND the set of terminals is not known in advance. In fact, we aim not only to connect segments but also to identify the most appropriate set of interface nodes. However, as pointed out in Section 1, if we consider an incremental concatenation of FSTs (i.e., linking connected components) the effectiveness of the solution depends on the order of insertion of edges and the way the component are connected. Therefore, we adopt the hypergraph-based approach that generates an *mst* to interconnect the boundary nodes (terminals) and introduce some optimization in order to generate a segment-based SMT solution. It is worth noting that one of the key advantage of the *mst* of the hypergraph is that it allows the management of all conflicts between FSTs at once [53]. We would also like to note that the determination of the minimum cost tree of all terminals is not equivalent to finding an *mst* of a graph G that represents the union of all formed FST vertices since simply not all Steiner points are needed. To this end, contemporary FST concatenation algorithms instead opt to determine the subset of FSTs that have links in the minimum cost tree of all terminals. In other words, forming a minimum cost tree of all terminals is FST selection rather than Steiner points and edges selection optimization.

As mentioned above, we form the minimum cost tree that spans all boundary nodes in such way that we get the SMT of the segments. Given the set of FSTs denoted as $F = \{T_1, \ldots, T_n\}$ where each T_i is an FST connecting some of the boundaries nodes, we associate a cost for each T_i that represents the length (hop count) of the path connecting terminals of T_i. Note that the cost of T_i reflects the number of relays needed when Steinerizing the edges of the tree, which is consistent with our quest for minimizing the required relay count for forming a connected inter-segment topology. In fact, an FST represents the optimal way to connect the correspondent set of terminals. While connecting two terminals needs the Steinerization of the direct path (straight line) between them, the optimal way to connect three terminals could be through one Steiner point (3-star [3]). Therefore, the number of relays varies according to the formed FST. Although the Steiner edges of T_i may not be the optimal path connecting any pair of terminals within T_i, it represents the optimal way to connect all terminals in T_i's adjacency list. This justifies further the use of a hypergraph in order to select a subset of the FSTs. More importantly, the cost of the hyper-edge plays a primordial role in enforcing the selection of inter segment link.

To optimize the selection of inter-segment links, we apply the following improvement: (1) the links that connect the boundaries nodes of the same segment should have zero cost as no relays are needed to ensure intra-segment connectivity. This forces the FST concatenation algorithm to select only the required hyper-edges connecting different segments to reduce the cost of SMT; (2) As we aim to connect distinct segments, a FST that includes more than one terminal of the same segment requires special consideration. For example, the tree connecting S_i^1, S_j^1, S_k^2 have two nodes that belong to *Seg*$_1$, namely, S_i^1, S_j^1 and one node from *Seg*$_2$, i.e., S_k^2. Since a link between S_i^1 and S_j^1 exists, it is preferable to connect *Seg*$_1$ and *Seg*$_2$ using only one terminal (S_i^1 or S_j^1). In such a case, we consider the cost of $\overline{vS_i^1}$ and $\overline{vS_j^1}$, where "v" is the Steiner point connecting S_i^1, S_j^1, S_k^2. The technique used for finding the *mst* in hypergraphs is based on an integer programming algorithm [51]. The *mst* connects all terminals and consequently federate the network segments. Fig. 7 illustrates all FSTs that BIND has determined from the topology considered in Fig. 6. The optimal concatenation leads to the red lines that federate the segments. The implementation and performance of BIND are analyzed next.

6. Approach implementation and analysis

6.1. BIND implementation

BIND opts to federate the WSN segments using the least number of relays. The presentation of BIND in the previous two sec-

```
1. For each boundary node $S_i$:
2    determine Neighbor[$S_i$]
3.    form the convex polygons relating $S_i$ and Neighbor[$S_i$]
4.    compute the FST for each polygon.
5.    estimate the cost of FSTs.
6. End
7. construct the hypergraph $H$ of the FSTs.
8. find the mst of $H$.
9. steinerized the edges of the mst
```

Fig. 8. Pseudo code summary of BIND.

tions has focused on justifying the recovery methodology and outlining the relay placement optimization process. Briefly, BIND consists of the following steps: (1) determine the set of neighbors for each boundary sensor according to the straight skeleton of the damaged area; (2) form the FSTs connecting boundary nodes and calculate their cost; (3) define a hypergraph based on the identified FSTs; (4) find the *mst* of the hypergraph; (5) steinerize the edges of the found *mst* based on the communication range of relay nodes and place relays on the identified Steiner points. Steps (3) and (4) constitute the FST concatenation phase. The pseudo code summary can be found in Fig. 8.

It is important to note, nonetheless, that BIND requires knowledge of the scope of network damage. According to the failure detection approach, BIND may be implemented in both centralized and distributed manners. A centralized implementation will obviously be pursued if the entire network state can be determined by external means, e.g., using a camera on an airborne unit or using the approach of Lee and Younis [18] and Senturk et al. [22]. In that case, the entire network state is known and BIND can be executed on the airborne unit, or at a remote commend center to determine the least number of relays and where to place them in order to restore connectivity.

On the other hand, distributed implementation requires the use of mobile relays to detect and recover from failure. The mobile relays will explore the area to determine the scope of the damage. Optimized area exploration is a well-known problem in robotics [58]. Efficient path planning solutions pursue topological skeletonization in a distributed manner [59]. BIND can thus leverage the exploration process in order to determine the adjacency lists. Furthermore, the multi-robot exploration process guarantees that at the end of the exploration all relays will be aware of the overall the Straight skeleton, and consequently the neighbor lists. Thus, the relays can form FSTs and individually apply the FST concatenation phase to determine where they will be positioned to serve in the inter-segment topology.

6.2. Sustaining the visibility property

As shown in Section 4, the events generated during the wavefront propagation process are of significance and useful for determining the set of FSTs. In the following, we show how the straight skeleton preserves the visibility property. The importance of the visibility propriety is because we deal with an ESTPO problem that should not allow the penetration a segment by a Steiner edge, i.e. no Steiner points inside segments. The straight skeleton enables the determination of the visibility from one point to all other points in one sweep without the need for checking every pair of terminals, as will be shown below. First, let us define the visibility graph.

Definition (Visibility graph [23]). Given a finite set of points N and a finite set of polygonal obstacles Ω in a plane, the *visibility graph* for N and Ω is a graph $G_{N,\Omega} = (V(G_{N,\Omega}), E(G_{N,\Omega}))$, where $V(G|N, \Omega) = N \cup^{V_\Omega}$ and $E(G|N, \Omega)$ is the set edges between elements of $V(G|N, \Omega)$ that do not intersect, i.e., cut across, any obstacle in Ω.

In other words, the visibility graph of a set of nonintersecting polygonal obstacles in a plane is a graph whose set of vertices consists of boundary points on the obstacles and whose edges are pairs of vertices u and v such that the edge $\overline{uv}$ between u and v does not intersect any of the obstacles.

Lemma 3. *The straight skeleton of a polygon with holes ensures the visibility propriety, where a hole refers to a segment in the context of BIND*

Proof. We prove this lemma by contradiction. Assume that u and v are adjacent on the skeleton (i.e., there exists a shared event between their edges during the wavefront propagation process) and that $\overline{uv}$ cut across with a segment (hole in the polygon). According to the skeleton formation procedure, during the expansion (wavefront propagation) process the lines of the segment (hole) boundary move parallel guided by the angle bisectors of the end vertices. According to Felkel and Obdrzalek [62], for each segment edge the wavefronts move outward until meeting the boundary of another segment or shrink to zero. In all cases, such an edge either gets split or disappears. Therefore, the wavefront could not penetrate inside any segments and the terminals constitute the leaves of the skeleton structure. Thus, if u and v are adjacent, it means that their respective segment boundaries meet first with the skeleton lines and do not meet another segment boundary in their wavefront movement. Therefore, the edge $\overline{uv}$ could not pass through segments, which contradicts the assumption. □

6.3. The correctness of the BIND

BIND reduces the complexity of SMT formation by applying an FST-based approach. However, it is shown in Lemma 1 that the number of FSTs grows rapidly with increasing problem size, i.e., the number of terminals. Therefore, BIND promotes a new geosteiner technique for ESPO problem to generate only useful topologies that may be part of the optimal solution. To avoid excluding important topologies, BIND decomposes the area into Steiner polygons (delimited by neighboring nodes). In the following, we show the correctness of BIND as a geosteiner technique. First, we show that the skeleton hull is a Steiner hull. Then, we prove that BIND does not discard any useful FSTs and successfully federates the segments.

Lemma 4. *The skeleton hull is the smallest hull containing terminals of an adjacency list.*

Proof. We prove this theorem by contradiction. Assume that the skeleton hull of an adjacency list η contains an internal terminal S. Thus, S does not belong to η since it does not lie on the polygon boundary. If such a terminal exists, it should belong to a segment edge E as we deal with ESTPO. According to Lemma 2, we have a face in the skeleton for each segment edge. For an edge E to be inside the polygon, E should be the base for some face F. Using Lemma 2, the face F tracks the changes of E. Since E is inside the polygon, its face F is adjacent to at least one face of terminals in η. Therefore, those faces share a common event (split or edge event) and consequently S should be considered within the set of neighbors η, i.e., on the boundary of the skeleton hull, which contradicts the assumption that S lies inside the skeleton hull. □

Lemma 5. *BIND does not discard useful FSTs.*

Proof. We prove this lemma by showing that the FSTs discarded by BIND are those that do not meet the ESTPO requirements mentioned in Section 4.1. In other words, the ignored FSTs will include at least one Steiner point inside a segment or Steiner edge that traverses a segment.

For each terminal S, BIND considers the set of nodes that have a shared face with an edge involving S. Therefore, for a given node S_i that belongs to faces F^1_{Si} and F^2_{Si}, BIND excludes the nodes that do not share a skeleton arc with F^1_{Si} or F^2_{Si}. Assume that F^1_{Sj} and F^2_{Sj} do not share any skeleton arc F^1_{Si} and F^2_{Si}; therefore, S_j does not belong to the adjacency list of S_i. As the set of FSTs are connected, there exists at least one face of node S_k separating the faces of S_i and S_j, where S_i and S_k are within the same skeleton hull. Given that S_i and S_k are neighbors, we will show that connecting S_i to S_j should traverse at least one segment. The following outlines all possibilities:

(1) S_i and S_k belong to the same polygon: In this case either S_k is the nearest to S_j because their wavefronts meet first, or the three terminals belong to the same segment and there is no need to connect them.
(2) S_j and S_k belong to same polygon: S_j is thus closer to S_i than S_k as their wavefronts meet first.
(3) S_j and S_k are not in the same polygon: The following cases are distinguished:
   - S_k is connected to S_j and S_i, and S_i is not connected to S_j; consequently S_j is between S_i and S_k. Therefore, connecting the three segments should be though $S_{j.}$
   - S_k and S_j are not connected directly; this mean that the path connecting S_i to S_j should imperatively traverse at least the polygon of S_j since it separates S_i and S_k.

Therefore, all the ignored FSTs cut across segments and thus BIND generates the correct set of FSTs. □

Lemma 6. *BIND considers all segment interfaces.*

Proof. This is referred to as the completeness property. To prove this property, we refer to the properties of the straight skeleton. Per Lemma 2, The straight skeleton *SS(P)* partitions a polygon *P* into *n* faces that are defined by the skeleton vertices (resulting from the intersection of wavefronts). Each face *F(e)* is associated with just one edge “*e*”. Each boundary node (terminal) in a segment belongs to exactly two edges and thus becomes a vertex in two faces. Considering all *n* terminals implies considering all edges and consequently pursuing all *n* faces. Since BIND processes all polygonal faces of the straight skeleton, it considers all segment interfaces. □

6.4. The complexity of BIND

We analytically show the complexity improvement that BIND achieves using the straight skeleton.

Theorem 2. (Melzak–Hwang Theorem) [60]. *Given a full Steiner topology T with n terminals, there is an O(n) time algorithm to either compute a full Steiner tree for T or decide that no such a tree exists.*

Fundamentally, the Melzak–Hwang Theorem implies that the complexity of FST generation depends on the generation of full Steiner topologies. According the correctness analysis, BIND generates only useful full Steiner topologies. In order to lower the complexity, BIND exploits the proprieties of the Steiner polygon to reduce on the number of generated full Steiner topologies, which in turns significantly reduce the computation time. As shown in Theorem 1, the convex hull of a set of terminals Ψ is a Steiner hull of Ψ. Therefore, BIND can benefit from the following Steiner polygon proprieties.

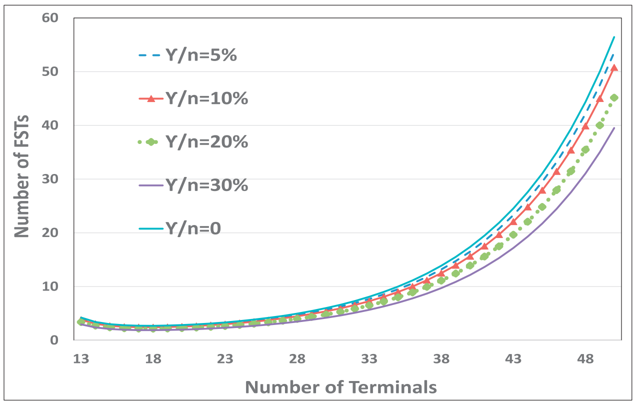


Fig. 9. The number of full Steiner topologies for n terminals as a function of the number of faces that BIND ignores during the generation process.

Table 1
Comparison between BIND and the analytical estimate of the FST complexity while varying γ and fixing β at 5%.

n	γ/n = 5%	γ/n = 10%	γ/n = 20%	γ/n = 30%	γ/n = 0	Γ(n)
25	4	4	4	3	4	2.54E+28
30	6	6	5	5	6	8.69E+36
35	10	9	8	7	10	7.3E+45
40	17	16	14	13	18	1.31E+55
45	30	28	25	22	32	4.56E+64
50	54	51	46	40	57	2.84E+74

Definition. [cherry]. A pair of terminals that are adjacent to a common Steiner point in a Steiner topology is called a cherry for the topology.

Corollary 1. [23]. *If a set of terminals Ψ has the property that all elements of Ψ lie on the boundary P_Ψ of the convex hull of Ψ, then every cherry of a SMT consists of an adjacent pair of vertices of P_Ψ.*

The following theorem is directly deduced from Corollary 1.

Theorem 3. [61]. *If all n terminals lie on a Steiner hull, then the number of full Steiner topologies is:* $\frac{4^n}{16n(\sqrt{\pi n})}$

Based on the aforementioned proprieties, we show an analytical estimate of the number of full Steiner topologies. Based on Lemma 1, the straight skeleton forms n topological faces; we assume that each face involves an average number of β terminals. Thus, the largest number of neighbors for each face is equal to $(2\beta-1)$ reflecting the case of two neighboring faces that share only one terminal. However, BIND considers only faces that can connect distinct segments. If we assume that γ edges shrink to zero without meeting any other edges, the corresponding γ faces (for which these γ edges are bases) are ignored by BIND. Thus, we have *(n-γ)* faces for which we form FSTs. In other words, BIND forms *(n-γ)* skeleton hulls. According to Theorem 3, BIND will generate the following number of FSTs**:**

$$(n-\gamma)*\frac{4^{(2\beta-1)}}{16(2\beta-1)(\sqrt{\pi(2\beta-1)})}$$

Recall that the number full Steiner topologies for n terminals from Lemma 1 is $\Gamma(n)=\frac{(2n-4)!}{(n-2)!*2^{n-2}}$.

In Figs. 9 and 10, we show the number of FST as functions of β, γ and n. Tables 1 and 2 compare the FST count of BIND with the baseline case of Lemma 1. As evident from the aforementioned formula, BIND achieves major optimization by punning inutile set of neighboring. Tables 1 and 2 demonstrate the massive reduction in the number of generated FSTs compared to the baseline (Lemma 1), as γ and β vary, respectively. Both β and γ are defined

Table 2
The effect of β on complexity in terms of the number of generated FSTs as size of the neighbor list varies, assuming no face has been ignores (i.e., $\gamma = 0$).

N	β/n = 5%	β/n = 10%	β/n = 20%	β/n = 30%	β/n = 40%	Γ(n)
16	3	4	81	3370	177,020	2.13E+14
20	3	9	625	81,100	13,037,916	2.22E+20
25	4	29	8562	4,518,591	2.93E+09	2.54E+28
30	6	97	121,649	2.59E+08	6.75E+11	8.69E+36
40	18	1249	26,075,832	9E+11	3.77E+16	1.31E+55
50	57	17,123	5.85E+09	3.25E+15	2.19E+21	2.84E+74

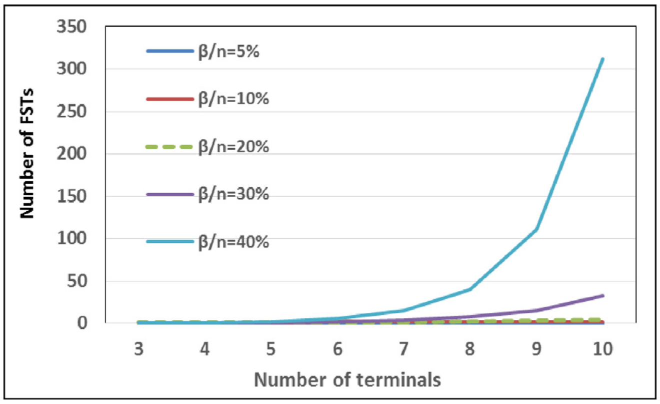


Fig. 10. The effect of the cardinality of the adjacency set on the number of full Steiner topologies for n terminals.

relative to n. Figs. 9 and 10 capture the effect of γ and β on BIND in more details. Due to the major difference in scale, the baseline count (Lemma 1) is not shown in the plots. Fig. 9 highlights the effect of ignored edges on the complexity. Obviously more reduction in the FST count is possible as γ increases since more faces are skipped. Meanwhile the effect of β is more dramatic (as seen in Fig. 10) since fundamentally β is the exponent in the complexity estimate. A small β implies fewer neighbors and consequently fewer FSTs. Although a large number of terminals per face diminishes the effect of the optimization, the reduction is still major compared to the baseline bound.

7. Pefromance validation

The effectiveness of BIND is validated through simulation. This section discusses the simulation setup, performance metrics and results.

7.1. Simualtion environment and performance metrics

The performance of BIND has been validated through extensive simulation experiments. The simulation environment is developed in Python. In the simulation, nodes are deployed using a uniform random distribution in a $900 \times 900\,\mathrm{m}^2$ area. The transmission range is set to 50 m for both the nodes and the relays. Overall, the parameters settings are consistent with the related work discussed in Section 2. We gauge the effectiveness of BIND using the following metrics:

- *The number of deployed relays*: This metric reflects the number of relay nodes required to restore inter-segment connectivity. The main objective of BIND is to minimize the recovery cost by employing the fewest relays.
- *Average path length (APL)*: This metric uses the Euclidean distance between a pair for segments to indicate the efficiency of the formed paths and appropriateness of the picked interface nodes. Shorter distances imply more efficient paths in terms of delay and communication. The average over all pair-wise paths is used to capture performance for all inter-segment topology.
- *Maximum inter-segment path length (MPL)*: Like the previous metric, the Euclidean distance again is used to gauge the efficiency of the formed inter-segment paths. This metric reports the maximum path length to show the worst-case.
- *Average inter-segment hop count (AHC)*: Unlike the previous two metrics, this metric measures the length of the data route between a pair of segments. The length here is in terms of the number of hops (relays) that a packet from the interface node of segment Seg_1 has to pass through until reaching the interface node of segment Seg_2. The average number of hops for all segment pairs is reported. Reducing the average hop count indicates decreased data delivery delay.
- *Maximum hop count (MHC)*: The metric measures the longest data routes between two segments. It is indicative of the worst case delay.

To baseline the performance of recovery strategy of BIND, we compare it with the game-theory based approach (GTA) of [22]. Although GTA considers the overall number of survived sensors within the segment during the restoration process, it does not factor in the shape and the size of the segments. Nonetheless, it exploits the visibility propriety to federate segments incrementally. Two parameters are varied in the experiments, namely, the scope of the topology damage and node density in the network. For the former, the number of nodes is fixed to 200 and the number of segments is varied between 2 and 16, which mainly reflects how many partitions have to be considered in the recovery. Basically, nodes are randomly deployed and then clustered into the required number of partitions based on proximity. Then some nodes are removed to ensure that the communication ranges of two nodes in separate clusters do not overlap. Then, substitutes of the removed nodes are deterministically placed within the formed partitions so that total count matches the planned node population. To test the impact of node density, 100–200 nodes are deployed in a similar way so that 6 disjoint partitions are formed.

7.2. Experiment setup and results

The performance results are depicted in Figs. 11–20. The plotted results reflect the average of 30 different random topologies. Error bars are also shown to indicate the statistical stability if the simulation results.

7.2.1. Number of involved relays

As pointed out earlier, minimizing the required number of relays to restore connectivity is the prime objective of BIND. Figs. 11 and 12 show the effect of node density and number of segments on the performance of both BIND and the baseline approach. As evident from the figures, BIND significantly outperforms GTA. The performance advantage for BIND is mainly attributed to the use of FSTs and the straight skeleton. When fixing the number of segments, Fig. 11 shows that BIND only needs only 30–60% of the relays used by GTA to restore connectivity. The performance gap

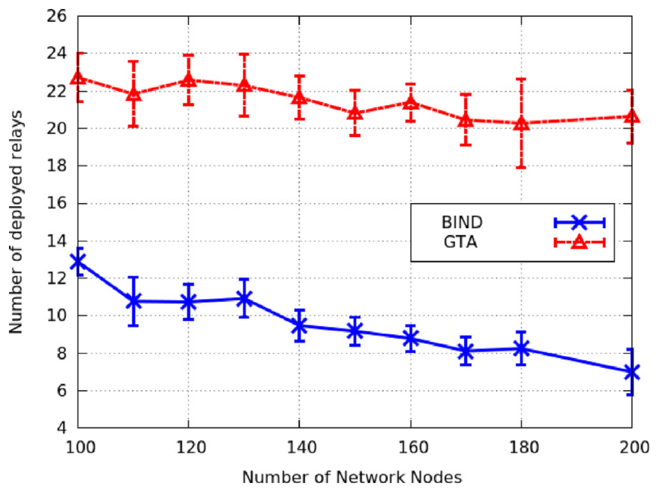


Fig. 11. The number of deployed relays with varying number of nodes.

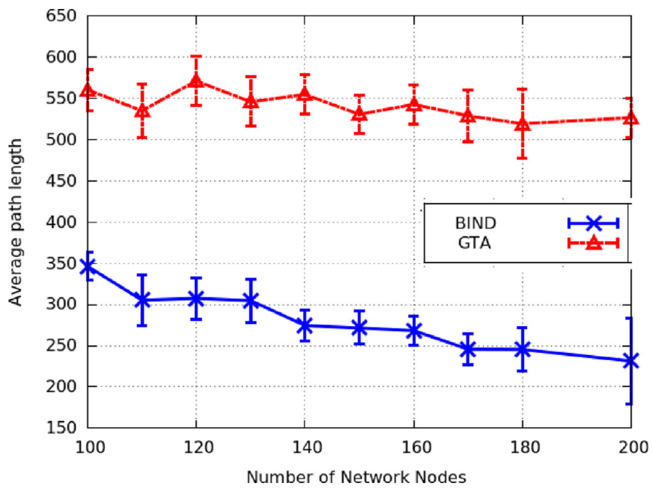


Fig. 13. APL with varying number of nodes.

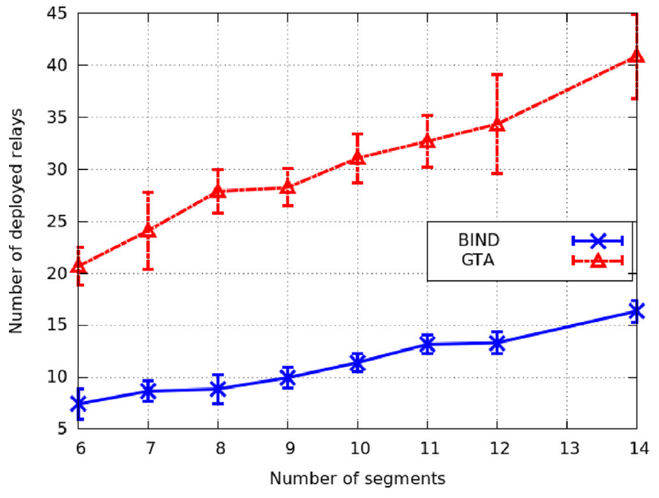


Fig. 12. The number of deployed relays with varying number of segments.

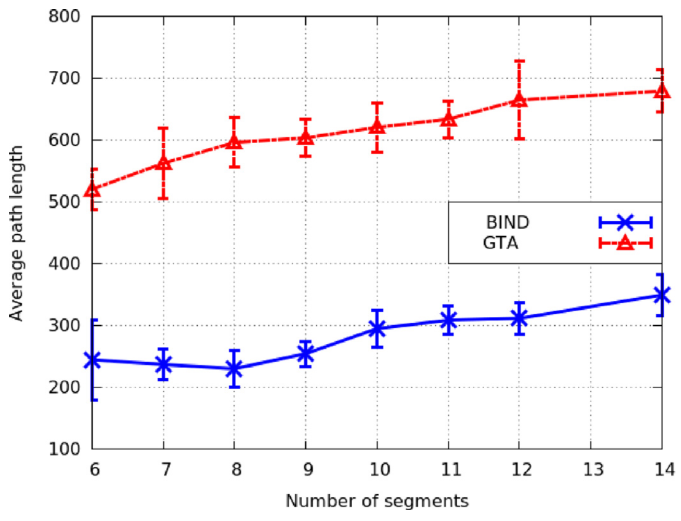


Fig. 14. APL with varying number of segments.

widens when the network is denser due to the fact that BIND determines the nearest segments (neighbor segments) in order to connect them while GTA forms connected components incrementally and thus the results depend on the order of segment federation.

Fig. 12 shows that BIND consistently connects the segments using about 1/3 of what GTA needs. This is quite distinct performance and is more evident for larger segment counts. Generally, growing the number of segments increases the connectivity requirement and consequently more relays are needed for forming a connected inter-segment topology. Nonetheless, BIND scales nicely in that regard, and widens the performance gap with the baseline approach. Basically, GTA forms components of segments incrementally using pair-wise segment proximity without considering the relay count savings that could be achieved by forming a partial Steiner tree for these components. BIND employs FSTs and strives to select the best subset of FSTs using a hyper-graph model. On the other hand, GTA is based only on visibility among the segments and does not factor the shape of segment.

7.2.2. Path length

Figs. 13 and 14 depict the average Euclidian distance between pairs of segments under varying node and segment counts, respectively. As pointed out, this metric gauges the optimized selection of nodes that interface the individual segments. Fig. 13 shows how successful BIND in determining the interface nodes that ought to be part of the inter-segment topology, where the proximity of the end nodes of a path between a pair of segments is about 38% shorter than that of GTA. The results can be attributed to the fact that the restoration pattern in GTA is not imperatively a tree that connects all segments. Basically, GTA connects segment incrementally. In each step, it connects a segment with the highly populated segment. Thus, GTA may have some redundant nodes given the greedy nature, yet the path length is not necessarily shorter than BIND since it depends on the segment handling order. As the node density decreases the distance between the corresponding interface-nodes decrease on the average; yet BIND takes advantage of the increased node population much better than GTA and the performance gap widens. Fig. 14 shows the effect of segment count on *APL*, where BIND yields 100% improvement compared to GTA and such distinct performance is sustained for highly fragmented networks, i.e., larger segment count. As expected, *APL* grows when growing the number of segments, mainly due to the increased connectivity requirements where more segments are to be interconnected. The results of *MPL* are shown in Figs. 15 and 16 are consistent with those of *APL*. The most notable observation is how the increased node count positively affects the *MPL* for BIND, yet GTA does not seem to be significantly impacted.

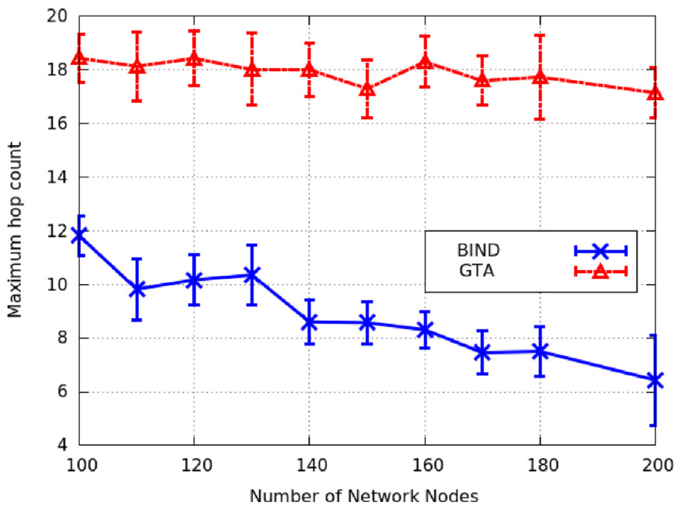


Fig. 15. MPL with varying number of nodes.

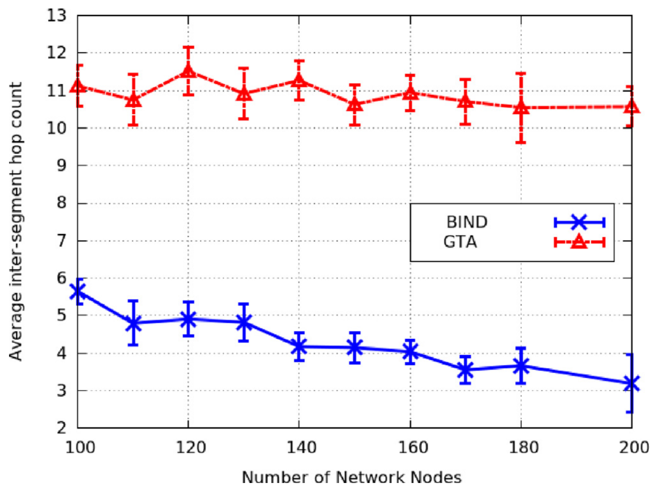


Fig. 17. AHC with varying number of nodes.

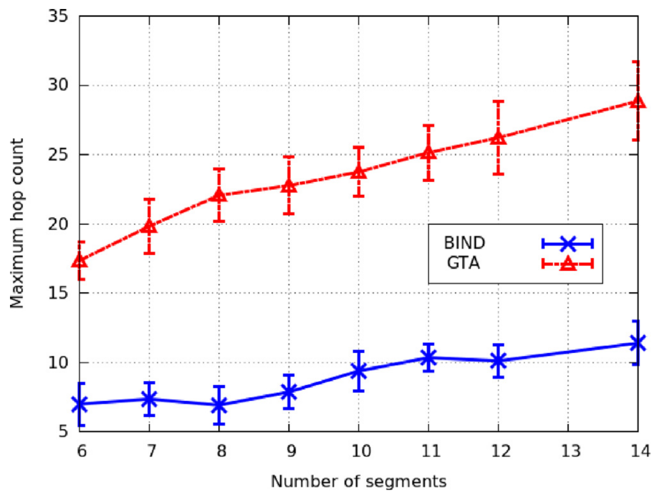


Fig. 16. MPL with varying number of segments.

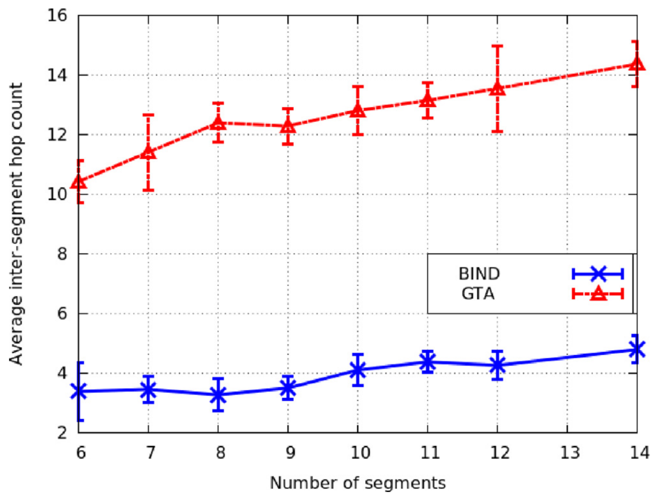


Fig. 18. AHC with varying number of segments.

7.2.3. Inter-segment hop count

Having many relays on a path between two segments will boost the end-to-end delay, and also the communication energy, both of which are detrimental to the network operation. Figs. 17–20 show the observed *AHC* and *MHC* results, where BIND is shown to yield excellent performance. Although BIND strives to reduce the overall number of relays, at the same time it optimizes the routes between individual segment-pairs. In fact, BIND determines the *mst* of the hypergraph during the concatenation phase while setting the hyper-edge weight to the hop count. Furthermore, BIND favors FSTs that interconnect multiple terminals at the same time. This shows how much it is beneficial to consider our FST-based approach compared to incremental federation of connected components (the strategy that GTA applies). Both *AHC* and *MHC* improve in dense networks since more boundary nodes are available for forming paths with fewer relays. GTA does not take advantage of the increase node density and consequently the performance gap with BIND widens as the node count increases. This is also due to the fact that GTA connects the nodes incrementally creating a backbone for the network that favors long tree branches. Increasing the number of segments extends both the average and largest relay count on an inter-segment path due to the increased connectivity requirements; however, the impact is significantly less in BIND compared to GTA.

8. Conclusion

In this paper, we have presented a novel approach for solving the problem of connectivity restoration in highly partitioned networks. Contrarily to the bulk of existing solutions in the literature, we factor in the size and the shape of the segment in order to form the shortest length topology in the Euclidean plane that interconnects a subset of nodes on the segment boundaries through extra Steiner points so that there is a path between every pair of segments. We first formulate the problem as variant of the ESPO problem and propose a novel approach for Boundary-aware (BIND). BIND pursues a heuristic based on the generation and concatenation of full Steiner trees (FSTs). Since the number of distinct FSTs grows exponentially with the terminal count, BIND employs a new geosteiner technique based on the straight skeleton of the segment boundaries within the deployment area. The simulation results confirm the effectiveness of BIND and its advantage over competing schemes. It is worth to mentioning that the straight skeleton has been widely used in multi-robots for area exploration. Therefore, BIND can leverage such work in order to realize optimized failure recovery in distributed manner. BIND deals with a static segment topology. Dealing of dynamic changes in the segment boundary is worth future investigation. In addition, we would like to study mitigation of coverage and connectivity loss due multi-node failure.

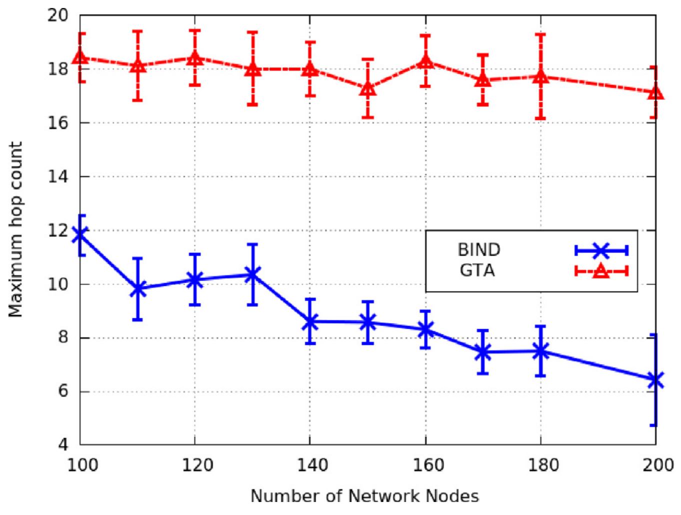


Fig. 19. MHC with varying number of nodes.

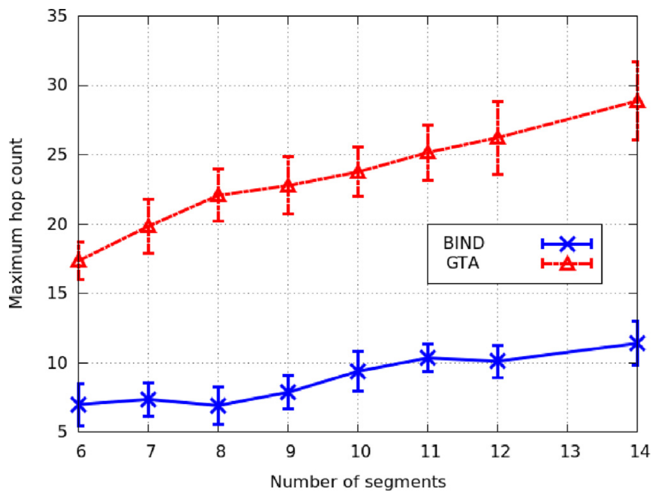


Fig. 20. MHC with varying number of segments.

References

[1] M. Younis, I. Senturk, K. Akkaya, S. Lee, F. Senel, Topology management techniques for tolerating node failures in wireless sensor networks: a survey, Comput. Netw. 58 (January) (2014) 254–283.
[2] F. Senel, M. Younis, Novel relay node placement algorithms for establishing connected topologies, J. Netw. Comput. Appl. 70 (July) (2016) 114–130.
[3] X. Cheng, D.-z. Du, L. Wang, B. Xu, Relay sensor placement in wireless sensor networks, Wireless Netw. 14 (3) (2008) 347–355.
[4] E.L. Lloyd, G. Xue, Relay node placement in wireless sensor networks, IEEE Trans. Comput. 56 (January (1)) (2007) 134–138.
[5] A. Efrat, et al., Improved approximation algorithms for relay placement, in: Proceedings of the 16th European Symposium on Algorithms, September, Karlsruhe, Germany, 2008.
[6] J. Tang, B. Hao, A. Sen, Relay node placement in large scale wireless sensor networks, Comput. Commun. 29 (2006) 490–501 special issue on wireless sensor networks.
[7] C. Ma, W. Liang, M. Zheng, H. Sharif, A connectivity-aware approximation algorithm for relay node placement in wireless sensor networks, IEEE Sens. J. 16 (2) (2016) 515–528.
[8] X. Han, X. Cao, E.L. Lloyd, C.-C. Shen, Fault-tolerant relay nodes placement in heterogeneous wireless sensor networks, in: Proceedings of the 26th IEEE/ACM Joint Conference on Computers and Communication (INFOCOM'07) May, Anchorage AK, 2007.
[9] N. Li, J.C. Hou, Flss: a fault-tolerant topology control algorithm for wireless networks, in: Proceedings 10th ACM Annual International Conference on Mobile Computing and Networking (MobiCom 2004) September, Philadelphia, PA, 2004.
[10] D. Chen, D.-Z. Du, X.-D. Hu, G.-H. Lin, L. Wang, G. Xue, Approximations for steiner trees with minimum number of steiner points, Theor. Comput. Sci. 262 (1) (2001) 83–99.
[11] F. Al-Turjman, H. Hassanein, W. Alsalih, M. Ibnkahla, Optimized relay placement for wireless sensor networks federation in environmental applications, Wireless Commun. Mobile Comput. 11 (December (12)) (2011) 1677–1688.
[12] A. Abbas, M. Younis, Establishing connectivity among disjoint terminals using a mix of stationary and mobile relays computer communications, Comput. Comm. 36 (2013) 1411–1421.
[13] F. Al-Turjman, H. Hassanein, Towards augmented connectivity with delay constraints in WSN federation, Int. J. Ad Hoc Ubiquitous Comput. 11 (2/3) (2012) 97–108.
[14] S. Lee, M. Younis, Optimized relay node placement for connecting disjoint wireless sensor networks, Comput. Netw. 56 (12) (2012) 2788–2804.
[15] F. Senel, M. Younis, Relay node placement in structurally damaged wireless sensor networks via triangular steiner tree approximation, Comput. Commun. 34 (16) (2011) 1932–1941.
[16] F. Senel, M. Younis, K. Akkaya, Bio-inspired relay node placement heuristics for repairing damaged wireless sensor networks, IEEE Trans. Veh. Technol. 60 (April (4)) (2011) 1835–1848.
[17] S. Lee, M. Younis, Optimized relay placement to federate segments in wireless sensor networks, IEEE J. Sp. Topics Commun. 28 (June (5)) (2010) 742–752 Special Issue on Mission Critical Networking.
[18] S. Lee, M. Younis, Recovery from multiple simultaneous failures in wireless sensor networks using minimum steiner tree, J. Parallel Distrib. Syst. 70 (2010) 525–536.
[19] F. Senel, M. Younis, Novel relay node placement algorithms for establishing connected topologies, J. Netw. Comput. Appl. 70 (July) (2016) 114–130.
[20] G. Lin, G. Xue, Steiner tree problem with minimum number of steiner points and bounded edge-length, Inf. Process. Lett. 69 (1999) 53–57.
[21] I.F. Senturk, S. Yilmaz, K. Akkaya, Connectivity restoration in delay tolerant sensor networks using game theory, Int. J. Ad Hoc Ubiquitous Comput. 11 (November (2/3)) (2012) 109–124.
[22] I.F. Senturk, K. Akkaya, S. Yilmaz, Relay placement for restoring connectivity in partitioned wireless sensor networks under limited information, Ad Hoc Netw., Part B 13 (February) (2014) 487–503.
[23] M. Brazil, M. Zachariasen, Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications, Springer, 2015.
[24] S.A. Khan, L. Bölöni, D. Turgut, Bridge protection algorithms – a technique for fault-tolerance in sensor networks, Ad Hoc Netw. Part A 24 (January) (2015) 186–199.
[25] K. Saleem, A. Derhab, J. Al-Muhtadi, M.A. Orgun, Analyzing ant colony optimization based routing protocol against the hole problem for enhancing user's connectivity experience, Comput. Human Behav. Part B 51 (October) (2015) 1340–1350.
[26] P. Basu, J. Redi, Movement control algorithms for realization of fault-tolerant ad hoc robot networks, IEEE Netw. 18 (4) (2004) 36–44.
[27] Y. Tamura, et al., Concurrent Moving-based connection restoration scheme between actors to ensure the continuous connectivity in WSANs, in: the Proceedings of the High Performance Computing and Communications, Paris, France, August, 2014.
[28] M. Younis, S. Lee, A. Abbasi, A localized algorithm for restoring inter-node connectivity in networks of moveable sensors, IEEE Trans. Comput. 59 (December (12)) (2010) 1669–1682.
[29] K. Akkaya, F. Senel, A. Thimmapuram, S. Uludag, Distributed recovery from network partitioning in movable sensor/actor networks via controlled mobility, IEEE Trans. Comput. 59 (2) (2010) 258–271.
[30] A. Abbasi, M. Younis, U. Baroudi, Restoring connectivity in wireless sensor-actor networks with minimal topology changes, IEEE Trans. Veh. Technol. 62 (January (1)) (2013) 256–271.
[31] M. Imran, A.M. Said, M. Younis, H. Hasbullah, Application-centric recovery algorithm for wireless sensor and actor networks, Int. J. Commun. Netw. Distrib. Syst. (IJCNDS) 10 (4) (2013).
[32] M. Sir, I. Senturk, E. Sisikoglu, K. Akkaya, An optimization-based approach for connecting partitioned mobile sensor/actuator networks, in: the Proceedings of International Workshop on Wireless Sensor, Actuator and Robot Networks (WiSARN), Shanghai, China, April, 2011.
[33] S.J. Habib, P.N. Marimuthu, An integrated restoration framework for coverage and communication within wireless sensor networks, Int. J. Ad Hoc Ubiquitous Comput. 15 (1-3) (2014) 95–105.
[34] T.T. Truong, K.N. Brown, C.J. Sreenan, Multi-objective hierarchical algorithms for restoring wireless sensor network connectivity in known environments, Ad Hoc Netw. 33 (October) (2015) 190–208.
[35] H. Wang, X. Ding, C. Huang, X. Wu, Adaptive connectivity restoration from node failure (s) in wireless sensor networks, Sensors 16 (10) (2016) 1487–1513.
[36] H. Essam, M. Younis, E. Sha'aban, Minimum cost flow solution for tolerating multiple node failures in wireless sensor networks, in: the Proceedings of the IEEE International Conference on Communications (ICC 2015), London, UK, June, 2015.
[37] M. Imran, M. Younis, A.M. Said, H. Hasbullah, Localized motion-based connectivity restoration algorithms for wireless sensor actor networks, J. Netw. Comput. Appl. 35 (March (2)) (2012) 844–856.
[38] Y. Joshi, M. Younis, Restoring connectivity in a resource constrained WSN, Elsevier J. Netw. Comput. Appl. 66 (May) (2016) 151–165.
[39] H. Almasaeid, A.E. Kamal, Data delivery in fragmented wireless sensor networks using mobile agents, in: the Proceedings of the 10th ACM/IEEE Int'l Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), Chania, Greece, October, 2007.

[40] S. Lee, M. Younis, M. Lee, Connectivity restoration in a partitioned wireless sensor network with assured fault tolerance, J. Ad Hoc Netw. Part A 24 (January) (2015) 1–19.
[41] S. Lee, M. Younis, M. Lee, Optimized bi-connected federation of multiple sensor network segments, J. Ad-hoc Netw. 38 (March) (2016) 1–18.
[42] S. Lee, M. Younis, EQAR: effective QoS-aware relay node placement algorithm for connecting disjoint wireless sensor sub-networks, IEEE Trans. Comput. 60 (December (12)) (2011) 1772–1778.
[43] Y. Joshi, M. Younis, Autonomous recovery from multi-node failure in wireless sensor network, in: the Proceedings of the IEEE Global Communications Conference (GLOBECOM 2012), Anaheim, CA, December, 2012.
[44] I.F. Senturk, K. Akkaya, Energy and terrain aware connectivity restoration in disjoint mobile sensor networks, in: the Proceedings of the 12th IEEE International Workshop on Wireless Local Networks (WLN'12), Clearwater, FL, October, 2012.
[45] Y. Wang, J. Gao, J.S. Mitchell, Boundary recognition in sensor networks by topological methods, in: Proceedings of 12th Annual International Conference on Mobile Computing and Networking (MobiCom), Los Angeles, CA, September, 2006.
[46] C. Zhang, Y. Zhang, Y. Fang, Localized algorithms for coverage boundary detection in wireless sensor networks, Wireless Netw. 15 (1) (2009) 3–20.
[47] M. Fayed, H. Mouftah, Localised alpha-shape computations for boundary recognition in sensor networks, Ad Hoc Netw. 7 (6) (2009) 1259–1269.
[48] G. Singh, F. Al-Turjman, Learning data delivery paths in QoI-aware information-centric sensor networks, IEEE Internet Things J. 3 (August (4)) (2016) 572–580.
[49] G. Singh, F. Al-Turjman, A data delivery framework for cognitive information–centric sensor networks in smart outdoor monitoring, Computer Commu 74 (1) (2016) 38–51.
[50] Kapoor, S.N. Maheshwari, Efficiently constructing the visibility graph of a simple polygon with obstacles, SIAM J. Comput. 30 (2000) 847–871.
[51] D.M. Warme, P. Winter, M. Zachariasen, Exact algorithms for plane Steiner tree problems: a computational study, in: D.-Z. Du, J.M. Smith, J.H. Rubinstein (Eds.), Advances in Steiner Trees, Kluwer Academic Publishers, Boston, 2000, pp. 81–116.
[52] M. Zachariasen, P. Winter, Obstacle-avoiding euclidean steiner trees in the plane: an exact algorithm, in: the Proceedings of the Workshop on Algorithm Engineering and Experimentation (ALENEX), Lecture Notes in Computer Science 1619, Springer, 1999, pp. 282–295.
[53] D.M. Warme, Spanning Trees in Hypergraphs with Applications to Steiner Trees PhD thesis, Computer Science Department, The University of Virginia, 1998.
[54] O. Aichholzer, D. Alberts, F. Aurenhammer, Bernd Gärtner, A novel type of skeleton for polygons, J. Univ. Comput. Sci. 1 (12) (1995) 752–761.
[55] E.N. Gilbert, H.O. Pollak, Steiner minimal trees, SIAM J. Appl. Math. 16 (1) (1968) 1–29.
[56] Grimwood, The Euclidean Steiner Tree Problem: Simulated Annealing and Other Heuristics MS Thesis, Institute of Statistics and Operations Research, The Victoria University of Wellington, 1994.
[57] Chung-Wu Ho, Decomposition of a polygon into triangles, Math. Gazette 60 (1976) 132–134.
[58] I.M. Rekleitis, G. Dudek, E.E. Milios, Multi-robot collaboration for robust exploration, in: the Proceedings of IEEE International Conference on Robotics and Automation (ICRA'00), San Francisco, CA, April, 2000.
[59] J. de Hoog, Role-Based Multi-Robot Exploration, Department of Computer Science, University of Oxford, 2011.
[60] F.K. Hwang, A linear time algorithm for full Steiner trees, Operat. Res. Lett. 4 (5) (1986) 235–237.
[61] E.J. Cockayne, On the Steiner problem, Canadian Math. Bulletin 10 (1967) 431–450.
[62] P. Felkel, S. Obdrzalek, Straight skeleton implementation, in: Proceedings of the 14th Spring Conference on Computer Graphics, 1998, pp. 210–218.

Wassila Lalouani received her BS and MS degrees from the Department of Computer Science at Computer Science at the university of science and technologies Houari Boumediene (USTHB), Algiers, Algeria, in 2006 and 2009, respectively. She is currently a graduate student in the department of computer science at USTHB Algiers, Algeria. She is also a lecturer at the High National School of Computing Science, Algiers, Algeria. Her research interest include fault tolerance and network management strategies for wireless sensor networks.

Dr. Younis is currently an associate professor in the department of computer science and electrical engineering at the university of Maryland Baltimore County (UMBC). He received his Ph.D. degree in computer science from New Jersey Institute of Technology, USA. Before joining UMBC, he was with the Advanced Systems Technology Group, an Aerospace Electronic Systems R&D organization of Honeywell International Inc. While at Honeywell he led multiple projects for building integrated fault tolerant avionics and dependable computing infrastructure. He also participated in the development of the Redundancy Management System, which is a key component of the Vehicle and Mission Computer for NASA's X-33 space launch vehicle. Dr. Younis' technical interest includes network architectures and protocols, wireless sensor networks, embedded systems, fault tolerant computing, secure communication and distributed real-time systems. He has published over 200 technical papers in refereed conferences and journals. Dr. Younis has six granted and two pending patents. In addition, he serves/served on the editorial board of multiple journals and the organizing and technical program committees of numerous conferences. Dr. Younis is a senior member of the IEEE and the IEEE communications society.