SAPIENZA
UNIVERSITÀ DI ROMA

Network analysis and algorithm solutions in critical emergency scenarios

PhD School of the Department of Computer Science at Sapienza University of Rome

Dottorato di Ricerca in Informatica – XXIX Ciclo

Candidate

Stefano Ciavarella
ID number 1274110

Thesis Advisor

Prof. Novella Bartolini

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science

January 2017

Thesis defended on February 2017
in front of a Board of Examiners composed by:

Network analysis and algorithm solutions in critical emergency scenarios
Ph.D. thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: January 30, 2017

Author's email: ciavarella@di.uniroma1.it

to those who love me and believed in me...

Contents

Introduction

The purpose of this thesis is to analyze and study the critical issues that influence the operation of computer networks. In fact, nowadays each computer network field should deal with the increasing problem of designing networks that are able to work under unpredictable conditions. In particular, such conditions include natural disasters (e.g., earthquake, tsunami, flooding, tornado, hurricane, etc.), human hostile environments (e.g., wildfire, loss of gas, presence of radiations, etc.) and even intentional attacks performed by an external attacker that aims to drastically reduce the operation of the network. Due to the general and wide application of the communication networks in different areas of computer science, these vulnerabilities can seriously compromise and influence the safety of modern society.
For instance, a widespread collapse of the communication networks, occurred after Hurricane Katrina hit the Gulf Coast of the United States in 2005. The damage extended for an area of approximately 93,000 square miles. More than 2,000 cell towers were knocked out. The backbone conduit for landline service was flooded as well as many central switching centers [59, 69]. Another example, in 2011 the "great east Japan earthquake" hit a large part of the north-east of Japan. The earthquake was just the start of a widespread disaster, which also included a huge tsunami and the nuclear failure at Fukushima. The tsunami destroyed most terrestrial communication infrastructures including many of the wired communication networks and emergency municipal radio communication systems [77, 71]. In both cases, the communication outage consequent to the disaster hampered the assessment of residents' safety. It also precluded efficient rescue operations by government and public organizations, such as distribution of medical aid and emergency supplies. The restoration of the communication infrastructure and its related services took months, a time window that is far from meeting the requirements of critical services or normal local communications of people living in the affected areas. For these reasons, a major challenge in disaster management scenarios is to *sufficiently recover* the communication network infrastructures so that it may support mission critical applications. In this scenario, our goal is to optimize the restoration actions in the shortest time and with minimum interventions, under the constraints posed by the critical services requirements.
In the field of Mobile Wireless Sensors Network (MWSN), mobile wireless sensors are used for monitoring inaccessible or hostile environments, where manual positioning of static sensors is not feasible [80] due to natural disasters as for example wildfires, chemical plumes or nuclear failures. These devices can autonomously deploy over an Area of Interest (AoI) using algorithms that determine the device movement and positioning rules. As shown in [9], an adversary may capture several sensors and reprogram them to perform

several attacks to damage the network, exploiting the specific vulnerabilities of the *deployment algorithm* in use. An attacker can influence the deployment of MWSN to reduce the area in which sensors are deployed, making the network unable to monitor the AoI with sufficient coverage and raising security issues [9]. In such a non monitored zone, the network is subject to several vulnerabilities, as for example intrusion actions by an attacker. In this thesis, we address the vulnerabilities of deployment algorithms for Mobile Wireless Sensors Network (MWSN) based on Voronoi diagrams to coordinate mobile sensors and guide their movements.
Since communication networks are widely applied in several fields, another example of critical and emergency scenarios comes from the power grids. Natural disaster phenomena or intentional attacks could damage the power systems operation resulting in large-scale blackouts. A report by the U.S. Executive Office of the President estimates that between 2003 and 2012, 679 large scale power outages occurred in the U.S., each affecting at least 50,000 customers [36, 18, 19]. The economical impact of this outage was significant, as the costs range from 18 to 33 billion dollars per year [36]. We focused on the power grids contingency analysis in Smart Grids to study the impact of potential component failures. In particular we addressed the problem of how to balance the end-users energy's loads, after a system component failure in the smart grid, to prevent a cascading failure phenomena that could lead to a large-scale blackout.

Considering the detrimental effects of such phenomenons on several different kind of networks, this work analyzes and proposes at the same time, protocols and algorithmic solutions to restore the full operation of the network under emergency scenario. The rest of the thesis is organized as follows. In part I, we focus on the problem of efficiently restoring sufficient resources in a communications network to support the demands of mission critical services (e.g., government offices, police stations, fire stations, power plants, hospitals, etc.) after a large scale disruption. Because our society heavily depends on communication networks to support these mission critical services, especially in times of emergency, it is important that such infrastructures be repaired quickly, at least to the point where mission critical services are restored. We consider scenarios in which a major disruption of the communication network makes it unable to meet the bandwidth requirements (hereby *demand flows*) of the mission critical services and therefore recovery actions are needed. We model the *recovery problem (MinR)* as a mixed integer linear programming (MILP) that aims to recover the damaged infrastructures in order to minimize the cost of the recovery actions. The restoration of the mission critical services are performed through the recovery of damaged network

components or the deployment of new network elements. We show that the problem is NP-hard and propose a polynomial time heuristic called *Iterative Split and Prune* (ISP) to recover the network efficiently with a solution close to the optimal, i.e., that minimizes the cost for reparations. ISP iteratively selects the node with the highest centrality value, repairs it if damaged, and *splits* some demand flows to force them to pass through the selected node. Furthermore, ISP minimizes the repairs by concentrating flows towards the areas of the network already repaired and it *prunes* the demand flows which can be routed on the currently repaired network. ISP terminates in a finite number of steps by returning both a recovery strategy and a routing solution for the demand flows. Experimental results, show that ISP performs very close to the optimal in terms of number of elements repaired.
To work properly, ISP assumes to have a perfect knowledge of the disrupted area, i.e., ISP needs to know the exact sets of failed nodes and links. However, this information is not always available or it is only partially available. In this context, a complete and detailed damage assessment required by ISP could take long time for extensive monitoring and local inspections. It is therefore fundamental that recovery interventions start as soon as possible even if knowledge of the damage extension is incomplete. For this purpose, we study the problem of *progressive restoration* of the mission critical services in condition of partial knowledge. By progressive restoration we mean a progressive process of monitor placement, network probing and repair interventions, to gain information and restore the networks' components over time. We formulate the problem of *Progressive Damage Assessment and network Recovery (PDAR)* which aims at progressively restoring critical services in the shortest possible time, under constraints on the availability of recovery resources and partial knowledge on the disruption. We propose a polynomial time algorithm called *Centrality based Damage Assessment and Restoration (CeDAR)* to dynamically schedule repair interventions, local inspections and remote probing of network components, with the objective to restore critical services in the shortest possible time with efficient use of recovery resources. CeDAR restores critical demand flows iteratively by planning repair schedules that are based on the current global view of the network. It schedules the repairs of components that can be utilized immediately and with the highest advantage for the largest number of critical services first, maximizing the accumulative service flow during the recovery process. Through extensive simulations, we show that CeDAR recovers the network with the lower cost of repairs, lower number of local inspections to discover the network's status and with the higher flow restored over time, compared to the other approaches in all the experimental scenarios.
In part II, we move the attention on the vulnerabilities of the deployment

algorithms for Mobile Wireless Sensors Network (MWSN) based on Voronoi diagrams to coordinate mobile sensors and guide their movements. Since, previous deployment algorithms do not address potential security issues, an attacker can easily prevent the network from achieving its coverage goals by taking control of few nodes. We give a geometric characterization of possible attack configurations, proving that a simple attack consisting of a barrier of few compromised sensors can severely reduce network coverage, creating a non monitored area subject to malicious actions by an attacker. Based on the above characterization, we propose a new secure deployment algorithm, named *SSD (Secure Swap Deployment)*. This algorithm allows a sensor to detect compromised nodes by analyzing their movements. We show that the proposed algorithm is effective in defeating a barrier attack, achieving the total coverage of the AoI and it has guaranteed termination. We perform extensive simulations showing that SSD has better robustness and flexibility, excellent coverage capabilities and deployment time, even in the presence of an attack.
In part III, we study the detrimental effect of system failures in the Smart Grid. Since these failures could lead to cascading failures phenomena and could induce a large-scale blackout, it is extremely important to perform preventive actions. We propose a novel framework to alleviate this type of risks by adjusting a large number of end-user loads through the concept of the Internet of Things (IoT) [5, 91]. The assumption is that, for such emergency cases, curtailing some non-critical loads to prevent cascading failures yields greater aggregate utility than leaving the lights on and having cascading failures later. The proposed framework comprehensively involves the System Operator (SO), the Load Serving Entities (LSEs), and the end-users' smart systems. The system operator prevents cascading failures by calculating the load energy curtailment. When an LSE is notified of a load curtailment amount, it computes the individual load curtailments of each users, in order to maximize the aggregate utility, i.e., end-users' satisfaction. Finally, the Smart Home Management System calculates an *emergency schedule*, which defines the best set of appliances that the user is allowed to use. This schedule minimizes the impact of the curtailment on the user's habits, while satisfying the power allowance requested by the LSE. The results show that the proposed framework is effective in keeping the system stable during contingencies, preventing cascading failures while maximizing the aggregate user utility.

Part I

Network Recovery after Massive Failures

Introduction

Natural disasters or intentional attacks can severely disrupt critical infrastructures such as communication, power, and emergency control networks [68] at a large scale. A complete recovery of these infrastructures may require months, during which there would be no sufficient support to the most critical services, not to mention normal communications in the devastated areas. Disaster management requires the restoration of at least the minimum necessary infrastructures to perform safety critical services, with the utmost urgency. Because our society heavily depends on communication networks to support mission critical services, especially in times of emergency, it is important that such infrastructures be repaired quickly, at least to the point where mission critical services are restored.
As example, a widespread collapse of critical infrastructures occurred after Hurricane Katrina that hit the Gulf Coast of the United States in 2005. The damage extended for an area of approximately 93,000 square miles. More than 2,000 cell towers were knocked out. The backbone conduit for landline service was flooded as well as many central switching centers [59, 69].
In 2011, the "great east Japan earthquake" hit a large part of the north-east of Japan. The earthquake was just the start of a widespread disaster, which also included a huge tsunami and the nuclear failure at Fukushima. The tsunami destroyed most terrestrial communication infrastructures including many of the wired communication networks and emergency municipal radio communication systems [77, 71].
In both cases, the communication outage consequent to the disaster hampered the assessment of residents' safety. It also precluded efficient rescue operations by government and public organizations, such as distribution of medical aid and emergency supplies. The restoration of the communication infrastructure and its related services took months, a time window that is far from meeting the requirements of critical services or normal local communications of people living in the affected areas.
For this reason, a major challenge in disaster management scenarios is to *sufficiently recover* the communication network infrastructures so that it may support mission critical applications in the shortest time and with minimum interventions.
In this first part, we focus on the *communication network* and the mission critical applications it supports. These applications represents critical services such as communication between government offices, police stations, fire stations, power plants, gas-duct control centers and hospitals, that rely on the communication network for control and operation. These services are

critical for first responders and typically show increased rate of requests as a consequence of the occurred incidents [7]. We address the problem of restoring critical services provided by the communication network through the recovery of damaged network components or the deployment of new network elements. Our goal is to optimize the restoration cost under the constraints posed by the critical service requirements.

In Chapter 1, we initially model the communication network as a graph of nodes (routers and access points) and links (physical lines) considering them as the elements that may fail after a massive failure. We model the recovery problem as a mixed integer linear programming (MILP) (Section 1.1). The problem looks for the best strategy that recovers the damaged infrastructure and deploys new links and nodes in order to minimize the cost of the recovery actions under the constraints on network capacity and while satisfy the demand flows. We show that the problem is NP-hard and propose a heuristic called *Iterative Split and Prune* (ISP) to recover the network efficiently in polynomial time with a solution close to the optimal (Section 1.2). ISP is based on a new metric called *demand based centrality*, specifically designed to measure the importance of a node with respect to multiple demand flows of interest. ISP makes use of this metric to determine the most important nodes to be repaired. In particular, ISP iteratively selects the node with the highest centrality, repairs it if damaged, and *splits* some demand flows to force them to pass through the selected node. This way, ISP minimizes the repairs by concentrating flows towards the areas of the network already repaired. Additionally, it *prunes* the demand flows which can be satisfied by the currently repaired network.

To work properly, ISP assumes a perfect knowledge of the disrupted area, i.e. ISP needs to know the exact sets of nodes and links failed. However, this information is not always available or it is only available a partial knowledge on the disrupted area. In this context, a complete and detailed damage assessment required by ISP could take long time for extensive monitoring and local inspections. It is therefore fundamental that recovery interventions start as soon as possible even if knowledge of the damage extension is incomplete. For this purpose, in Chapter 2 we focus on the operative phases of damage assessment and network recovery of the communication infrastructures. In particular, we study the problem of *progressive restoration* of the mission critical services in condition of partial knowledge. By progressive restoration we mean a progressive process of monitor placement, network probing and repair interventions, to gain information and restore the networks' components over time. The reason why repair actions need

to be performed progressively is twofold: 1) knowledge of the status of the network elements can be increased incrementally by placing new monitors on repaired network components; 2) the schedule of recovery interventions can be adapted to the incremental knowledge availability and result in a more efficient utilization of recovery resources (e.g. human workers). Performing repairs in a progressive manner is also necessary for coping with unpredicted variability and surges of the demand [7]. We formulate the problem of *Progressive Damage Assessment and network Recovery (PDAR)* which aims at progressively restoring critical services in the shortest possible time, under constraints on the availability of recovery resources and with partial knowledge on the disrupted area (Section 2.1). We show that the PDAR problem is NP-hard and may require an unsustainable computation time for large networks. We propose a polynomial time algorithm called Centrality based Damage Assessment and Restoration (CeDAR) which dynamically schedules repair interventions, local inspections and remote probing of network components, with the objective to restore critical services in the shortest possible time with efficient use of recovery resources (Section 2.2). CeDAR restores critical demand flows iteratively by planning repair schedules which are based on the current global view of the network. It schedules the repair of components that can be utilized immediately and with the highest advantage for the largest number of critical services first, maximizing the accumulative service flow during the recovery process. CeDAR chooses the best node to place a monitor on the basis of a demand-based centrality metric that takes into account the demand flows. Based on the information obtained by placing monitors in each stage, CeDAR adapts its repair schedule by planning more efficient interventions.

In both scenarios, we prove some properties of the proposed algorithms, proving that ISP and CeDAR terminate in a finite number of steps and in polynomial time, by returning both a recovery strategy and a routing solution for the demand flows (Sections 1.3 and 2.3). We also propose other heuristics for comparison, based on the standard multi-commodity approach, greedy approaches as well as other solutions proposed in literature and adapted to correctly work in the recovery scenario. We compare the performances of ISP and CeDAR with other solutions in a variety of scenarios. Such scenarios include both real and synthetic network topologies, geographically correlated failures, as well as different demand requirements (Sections 1.4 and 2.4). Results show that ISP and CeDAR always outperform other approaches and always perform close to the optimal solution that is NP-hard (Sections 1.5 and 2.5).

In summary the original contribution of our work is the following:

- We model, for the first time, the recovery problem (MinR) and progressive recovery problem (PDAR) under complete and incomplete knowledge, respectively.
- We show that the MinR and PDAR problems are NP-hard.
- We introduce a new metric of demand based centrality, specifically meant to measure the importance of a node in a network of a multi-commodity problem instance.
- We propose two polynomial time heuristic called Iterative Split and Prune (ISP) and Centrality based Damage Assessment and Recovery (CeDAR), to solve the MinR and PDAR problems, respectively.
- We analyze the properties of ISP and CeDAR and prove its correctness, termination, and polynomial time complexity.
- We propose several heuristics based on the standard multi-commodity approach, greedy heuristics and shortest paths repair approaches, as well as we modified previous approaches in literature as baseline solutions to MinR and CeDAR.
- We evaluate the proposed solutions through simulations under a wide variety of scenarios. Results show that ISP and CeDAR perform close to the optimal NP-hard solution, while other heuristics incur a much higher cost to accommodate the demand flows in all the considered scenarios.
- The ISP algorithm for network recovery is published on *Dependable Systems and Networks (DSN), 2016 46th Annual IEEE/IFIP International Conference*(http://ieeexplore.ieee.org/document/7579733/) [13]
- The CeDAR algorithm for the progressive network recovery is published on *IEEE Proceedings of the International Conference on Computer Communications (IEEE INFOCOM 2017)* [29]

Related Works

Numerous works address the case of sparse failures through the provision of alternative paths, provided either proactively, as in the work of Todimala et. al [82], or reactively, as in the work of Zheng et al. [92], whereas Suchara et. al [81] jointly address recovery and traffic engineering, to minimize congestion after a failure. Since our works address the problem of network recovery from large scale failures, in this section we do not consider the previous work on the first problem, and describe only works that are related or are applicable to the case of massive failures.

The multi-commodity flow problem, addressed in a large amount of research work, aims at finding the routing of several multi-commodity flows in a supply network, so as to optimize the totally routed flow. This problem seems the most reasonable reduction of our problem to a classic problem. Nevertheless this approach has considerable limitations when applied to the problem of recovery. We discuss these aspects in detail in Chapter 1, section 1.4.1.

Many heuristics have been proposed to solve several variants of the multi-commodity flow problem. Most of these works [42, 6] rely on the idea that a higher total flow can be obtained by balancing the load distributing the flow over many paths. This idea is opposite to what is needed in the recovery problem MinR, where we want to maximize the flow traversing repaired paths, and concentrate the flow towards shared paths.

Some works focus on the *rent or buy multi-commodity problem*, which aims at installing possibly unlimited capacities on the edges of a network so that a prescribed amount of flow can be routed between several pairs of terminals. The rent or buy problem assumes that each edge can obtain unlimited capacity at a given cost. The works by Kumar et al. [56] and Fleischer et al. [38] address this problem and propose polynomial time heuristics with a given approximation of the optimal solution.

Other works address the problem of service restoration in the case of heterogeneous non-telecommunication networks. Among these, in their work [60], Lee et al. address the problem of restoring service in an interconnected network by creating new links. They propose a formulation of the problem in terms of a high complexity optimization model.

Other works [3, 51] address the problem of recovery beyond the field of telecommunications with solutions tailored to the specific type of network being considered.

Finally, the work of Magnanti et al. [64] addresses the problem of network design under connectivity only requirements. It shows that the simplified version of MinR in which every demand pair requires only to be connected

regardless of the capacity of the interconnecting paths, is a specific instance of the Steiner Forest problem. The work by Wan, Qiao and Yu [86] introduced a problem related to ours. They study the impact of recovery actions in terms of improved throughput over time. Their work aims at formulating a schedule of repair interventions under limited daily budget, so as to optimize the achieved throughput. The authors modeled the problem as an MILP and showed that is it NP-hard. They proposed a greedy heuristic for solving the problem in multiple stages by analyzing the shadow prices of the related optimization problem and using an iterative evaluation of these values to repair the edges with highest potential for contributing to the objective function. Ferdousi et al. [37] tackle the problem of progressive datacenter recovery after a large-scale failure. The work aims at giving directions on how to select which datacenters should be recovered at each recovery stage to maximize cumulative content reachability from any source considering limited available network resources.
All the mentioned works have different targets and are not directly applicable to solve the MinR and PDAR problems. Furthermore, these works assume perfect knowledge of the status (working or damaged) of network elements, while PDAR specifically address the problem of recover a network in absence of knowledge. In Section 2.4.1, we extended the work in [86] and we adapt it to properly work in the scenario of partial knowledge to solve the PDAR problem. Moreover, our algorithms also produce a routing solution that guarantees that the demand flows are actually accommodated.

Nomenclature and Notation

In this section we define the nomenclature that will be used in the entire Part I to model the network recovery problem scenario. We model the communication network as an undirected graph $G = (V, E)$, called the *supply graph*, where V and E represent nodes and links of the network, respectively. Each edge $(i, j) \in E$ has capacity c_{ij}. We also consider a *demand* graph $H = (V_{\mathtt{H}}, E_{\mathtt{H}})$, where $V_{\mathtt{H}} \subseteq V$, and $E_{\mathtt{H}} \subseteq V_{\mathtt{H}} \times V_{\mathtt{H}}$ is the set of pairs of nodes in $V_{\mathtt{H}}$ having a positive flow demand. Each pair $(s_h, t_h) \in E_{\mathtt{H}}$ has a source s_h, a destination t_h and an associated demand flow d_{s_h,t_h}. For the sake of simplicity, we write $h \in E_{\mathtt{H}}$, when $(s_h, t_h) \in E_{\mathtt{H}}$, and we shortly use the notation d_h for d_{s_h,t_h} when the context allows. Notice that the demand flows modeled by the sets $V_{\mathtt{H}}$ and $E_{\mathtt{H}}$ can take emergency related priorities into account. These sets define the endpoints of critical communication services and an estimate of the related demand flow, which may account for the increased needs due to the disaster [7].

Notations	Descriptions
$G = (V, E)$	supply graph
$G^{(n)} = (V^{(n)}, E^{(n)})$	supply graph at iteration n
$H = (V_{\mathtt{H}}, E_{\mathtt{H}})$	demand graph
$H^{(n)} = (V_{\mathtt{H}}^{(n)}, E_{\mathtt{H}}^{(n)})$	demand graph at iteration n
c_{ij}	capacity of edge $(i, j) \in V$
$d_h = d_{s_h,t_h}$	demand flow of edge $(s_h, t_h) \in E_H$
$c_{ij}^{(n)}$, $d_{s_h t_h}^{(n)}$	capacity of (i, j), demand of (s_h, t_h) at the n=th iteration
$V_{\mathtt{B}} \subseteq V$ and $E_{\mathtt{B}} \subseteq E$	broken vertices and edges
$V_{\mathtt{B}}^{(n)}$ and $E_{\mathtt{B}}^{(n)}$	$V_{\mathtt{B}}$ and $E_{\mathtt{B}}$ at iteration n
$h \in E_{\mathtt{H}}$,	demand pair $(s_h, t_h) \in E_{\mathtt{H}}$
k_i^v, k_{ij}^e	cost of vertex i and edge (i, j)
f_{ij}^h	quantity of flow h from i to j
δ_{ij}	decision to use edge $(i, j) \in E$,
δ_i	decision to use vertex $i \in V$
$\eta_{\mathtt{max}}$	maximum degree of the network
b_i^h	flow h generated at node i
$\ell(p)$, $l(e_i)$	length of path p, length of edge e_i
$n(p)$	number of edges of p
$c(p)$	capacity of path p: $\min_{e \in p} c_e$
$\mathscr{P}(i, j)$	paths in G between i and j
$\mathscr{P}^*(i, j)$	shortest paths necessary to route demand d_{ij}
$\mathscr{P}_{ij}^*\|_v$	set of paths in $\mathscr{P}_{ij}^*$ that include v
$c_d(v)$	demand based centrality, see equation (1.3)
$v_{\mathtt{BC}}^{(n)}$	node with highest centrality at iteration n
$\mathcal{C}^{(n)}(v_{\mathtt{BC}}^{(n)}) \subseteq E_{\mathtt{H}}^{(n)}$	demand pairs that contributed to the centrality of $v_{\mathtt{BC}}^{(n)}$, updated at iteration n
$\mathcal{L}(n)$	list of repairs, updated at iteration n

Table 1: Nomenclature and notation.

In order to model the network failure, we define the sets $V_{\mathtt{B}} \subseteq V$ and $E_{\mathtt{B}} \subseteq E$ of damaged vertices and edges, respectively. We denote with k_i^v the cost of repairing vertex $i \in E_{\mathtt{B}}$ and with k_{ij}^e the cost of repairing the edge $(i, j) \in E_{\mathtt{B}}$[1]. The recovery costs are heterogeneous and dependent on the location and on the technology in use. This notations will be extended in the Chapter 2 to model the incompleteness of the information on the disruption. Table 1 summarizes the notation used throughout this part.

[1] Notice that this model can also be *adopted as is* to support decisions to replace broken links with new links of higher capacity, or to deploy and connect new nodes, by formulating a related decision space. These additional choices may be considered in the model as parts of the sets $E_{\mathtt{B}}$ and $V_{\mathtt{B}}$ and included in the correspondent supply graph G. The model can also be *extended* to the case of multiple choices for link technology and related capacity. For simplicity of presentation, in this work we refer to the only case of recovery decisions.

Chapter 1

ISP: network recovery under complete knowledge of the disruption

1.1 The Network Recovery Problem

In this section we formulate the MINIMUM RECOVERY (MinR) problem as a mixed integer linear optimization problem. MinR aims at minimizing the cost to repair broken nodes and links so as to restore the necessary network capacity to meet a given demand.

We introduce the decision variables $f_{ij}^h \in \mathbb{R}$, with $f_{ij}^h \geq 0$, to represent the fraction of the demand flow h that will be routed through the link $(i,j) \in E$, going from vertex i to vertex j. Notice that other flows may traverse the same edge in the opposite direction.

We also define the binary variables δ_{ij} and δ_i. The variable δ_{ij} represents the decision to use link $(i,j) \in E$, therefore $\delta_{ij} = 1$ if link (i,j) is used, and $\delta_{ij} = 0$ otherwise. If the link $(i,j) \in E_{\mathtt{B}}$, the decision to use this link implies that it must be recovered. Similarly, δ_i represents the binary decision to use the node $i \in V$, which has to be recovered if it is broken, that is if $i \in V_{\mathtt{B}}$.

The objective function of the MinR problem can be expressed as in Equation 1.1(a), where we optimize the cost of repairing the only vertices and edges that are both used (the corresponding binary decision variable is 1) and that were initially broken (the related vertices and edges belong to $V_{\mathtt{B}}$ and to $E_{\mathtt{B}}$, respectively).

The capacity constraint of our problem is expressed by Equation 1.1(b). According to this constraint the total amount of flow traversing the edge (i,j) in both directions cannot exceed the maximum capacity of the link.

Notice that if an edge (i,j) is used, the corresponding endpoints i and j are also used, which implies that $\delta_i \geq \delta_{ij}, \forall i,j \in V$. To express this constraint in a compact form, with fewer equations, we consider that the degree of each vertex is lower than or equal to the maximum degree $\eta_{\mathtt{max}}$ of the network. Therefore the relationship between δ_i and δ_{ij} can be expressed by the constraint given by Equation 1.1(c).

We consider a flow balance constraint, in the form expressed by Equation 1.1(d). In this equation $b_i^h = d_h$ if $i = s_h$, $b_i^h = -d_h$ if $i = t_h$, and $b_i^h = 0$ otherwise. This equation states that if vertex i is not the source nor the destination of flow h, then the total portion of flow h entering the vertex i should be the same as in the exit direction. By contrast if vertex i is the source of flow h, then the amount of flow h in the exit direction from vertex i is d_h. Similarly if vertex i is the destination of flow h, the amount of flow h entering vertex i is d_h. Finally, Equation 1.1(e) shows that we are considering non negative, continuous decision variables for the flow assignment to edges, while Equation 1.1(f) expresses the binary constraint for the decision

variables which determines whether some vertices and edges are used in the solution of the problem.

The MinR problem can therefore be formulated in linear terms in the variables δ_{ij}, δ_i and f_{ij}^h as follows:

$$\begin{array}{lll}
\min \sum_{(i,j)\in E_{\mathtt{B}}} k_{ij}^e \delta_{ij} + \sum_{i\in V_{\mathtt{B}}} k_i^v \delta_i & & (a)\\
c_{ij}\cdot\delta_{ij} \geq \sum_{h=1}^{|E_{\mathtt{H}}|} (f_{ij}^h + f_{ji}^h) & \forall (i,j)\in E & (b)\\
\delta_i \cdot \eta_{\mathtt{max}} \geq \sum_{j:(i,j)\in E} \delta_{ij} & \forall i \in V & (c)\\
\sum_{j\in V} f_{ij}^h = \sum_{k\in V} f_{ki}^h + b_i^h & \forall (i,h)\in V\times E_{\mathtt{H}} & (d)\\
f_{ij}^h \geq 0 & \forall (i,j)\in E,\ h\in E_{\mathtt{H}} & (e)\\
\delta_i, \delta_{ij} \in \{0,1\} & \forall i\in V,\ (i,j)\in E & (f)
\end{array} \tag{1.1}$$

Theorem 1.1.1. *The problem MinR is NP-Hard.*

Proof. Let us consider a generic instance of the Steiner Forest problem [49, 64]. Given a graph $G_{\mathtt{sf}} = (V_{\mathtt{sf}}, E_{\mathtt{sf}})$, a set of node pairs $S_{\mathtt{sf}} = \{(s_1, t_1), \ldots, (s_n, t_n)\}$ and a cost function $c_{\mathtt{sf}} : E \to \mathbb{R}^+$, the goal of the Steiner Forest problem is to find a forest $F_{\mathtt{sf}} \subseteq E$ with minimum cost, such that for each pair (s_i, t_i), s_i and t_i belong to the same connected component in $F_{\mathtt{sf}}$.

We reduce this problem to an instance of MinR as follows. We consider a supply graph $G = (V, E)$ with $V = V_{\mathtt{sf}}$ and $E = E_{\mathtt{sf}}$. We consider $E_{\mathtt{B}} = E$ and $V_{\mathtt{B}} = \emptyset$. We create a unitary demand flow for each pair in $S_{\mathtt{sf}}$. For each edge in E we set the cost of repair equal to the cost of the corresponding edge in $G_{\mathtt{sf}}$, and its capacity equal to a value L that is sufficiently large that any link of E can accommodate the sum of all demand flows. Therefore, considering a requirement of one unit of flow for each demand pair, it is $L \gg |S_{\mathtt{sf}}|$.

Given such instance, MinR returns the set of nodes $V^* \subseteq V$ and edges $E^* \subseteq E$ to be repaired to accommodate all the demand flows. However, $V^* = \emptyset$, since no node is damaged. Additionally, since the capacity of each edge in E is large enough to accommodate an amount of flow exceeding the sum of all demand flows, for each demand pair (s_i, t_i) a single path from s_i to t_i is sufficient to accommodate the demand flow between s_i and t_i. As a result, the union of the links in E^* generates a Steiner forest, since any cycle would imply unnecessary repairs. This is also the forest with minimum cost, since MinR minimizes the costs of repairs.

We can therefore conclude the reducibility of the Steiner Forest problem to MinR, and consequently that the problem MinR is NP-Hard. □

1.2 Iterative Split and Prune

To solve the MinR problem presented in the Section 1.1, we developed a novel heuristic called ISP. The algorithm ISP (Iterative Split and Prune)

works by iteratively selecting the best candidate nodes and links for repair, then simplifying the demand by either removing (pruning) or reducing it in smaller segments (split), so as to consider simpler instances of the problem at every iteration. The termination condition is the complete removal of the demand or the achievement of an instance whose demand is routable through the currently working links. Notice that at the end of its execution the algorithm ISP will output both the set of repairing interventions and the corresponding routing of demands.

The pseudo-code of the algorithm is shown in Algorithm 1. More details on the single activities can be found in the following sections.

ALGORITHM : Iterative Split and Prune (ISP)

Input: Supply graph G, demand graph H, broken nodes $V_{\mathtt{B}}$ and broken edges $E_{\mathtt{B}}$

```
1  while routability test fails do
2      while pruning condition do
3          Prune demands satisfying pruning condition;
4          Update G and H;
5      if there are repairable links then
6          Repair broken repairable links;
7          Update G and E_B;
8      else
9          Find best candidate v_BC for split;
10         Repair v_BC if broken;
11         Find best demand d to split on v_BC;
12         Calculate the maximum splittable amount d_x;
13         Split amount d_x of demand d on v_BC;
14         Update G, H, V_B;
```

1.2.1 Routability test

At the basis of the algorithm is the use of flow balance equations and capacity constraints to determine the feasibility of an action or the termination condition. The algorithm should terminate whenever there is no demand left, or the current demand can be routed without additional repairs.

For some specific topologies of both supply and demand graphs, as discussed by Schrijver in [78], the question whether a demand can be routed through the links of the supply graph can be answered by verifying the so called *cut condition*, namely whether for every cut the total capacity crossing the cut is no less than the total demand crossing it. While the cut condition is always necessary to ensure the routability of a set of demand flows through a supply graph, it is not always sufficient, for example when the graphs G and

H admit an *odd p-spindle* as a minor as motivated by Chakrabarty, Fleischer and Weible in [26], or a *bad-k4-pair* as discussed in the already mentioned work by Schijver [78].

The specific instances of graph pairs G and H of a multi-commodity flow problem for which the verification of the cut condition is a necessary and sufficient condition for the routability are called *cut-sufficient* instances. For this part on the network recovery, we are *not* assuming cut-sufficiency for our works, since we address general graph instances.

Without assuming any structural property of the supply and demand graph, the routability of the demand over the supply graph can be determined by solving the following set of inequalities, to which we will refer under the name of *routability conditions*:

$$\begin{cases} \sum_{h \in E_{\mathtt{H}}} (f_{ij}^h + f_{ji}^h) \leq c_{ij} & \forall (i,j) \in E \\ \sum_{j \in V} f_{ij}^h = \sum_{k \in V} f_{ki}^h + b_i^h & \forall (i,h) \in V \times E_{\mathtt{H}} \\ f_{ij}^h \geq 0 & \forall (i,j) \in E, h \in E_{\mathtt{H}} \end{cases} \tag{1.2}$$

If the constraint system given by the routability conditions of Equation (1.2) determines a non empty region, then we can assert that the supply graph G has enough capacity to ensure the routability of the considered demand H. Any feasible solution of the above system is a routing policy that can be adopted to satisfy the demand H with routes in G.

Notice that at any iteration, the demand graph H and the residual capacities of the edges of graph G are updated as a consequence of either prune, or split actions. The sets $V_{\mathtt{B}}$ and $E_{\mathtt{B}}$ are also updated after any repair decision.

For this reason we define the *supply graph at iteration* n as $G^{(n)} = (V^{(n)}, E^{(n)})$, with link capacities $c_{ij}^{(n)}$, and where $V^{(n)} = V \setminus V_{\mathtt{B}}^{(n)}$, and $E^{(n)} = (E \setminus E_{\mathtt{B}}^{(n)}) \setminus \{(i,j) \text{ s.t. } |\{i,j\} \cap V_{\mathtt{B}}^{(n)}| \geq 1\}$. Analogously, we consider the demand graph $H^{(n)}$, updated at iteration n. When necessary, the routability test is performed on the problem instance defined at iteration n, with supply graph $G^{(n)}$ and demand graph $H^{(n)}$.

1.2.2 Centrality based ranking

According to the classic definition, the betweenness centrality of a node v is proportional to the number of shortest paths in the supply graph $G = (V, E)$ between any two vertices $i, j \in V \setminus v$, divided by the total number of node pairs. When there are multiple shortest paths of equal length, their nodes share centrality credits in equal proportions. Unfortunately, when demands and capacities are taken into account, such a metric is no longer able to quantitatively evaluate the centrality of a node, as it considers all the node

pairs as equally important, and does not take account of the different amount of flow potentially traversing each node.

Unlike previous definitions of node centrality [40, 22, 20, 48, 74], we introduce a new measure of centrality (hereby *demand based centrality*), that takes account of the ability of each node to route the demand flows throughout the network. Our metric is a generalization of the classic notion of *betweenness centrality* [40, 22].

We generalize the notion of *betweenness centrality* as follows. A path p in the graph G is hereby defined as a list of edges $p =< e_1, e_2, \ldots, e_n >$. For shortness of notation, we will also say that a vertex $v \in p$ when v is an endpoint of an edge belonging to p. We denote with $\ell(p)$ the length of the path p, therefore $\ell(p) = \sum_{e_i \in p} l(e_i)$, where $l(e_i)$ is the length of the edge e_i.

The capacity of a path is denoted by $c(p)$ and is equal to the minimum capacity of the links in p, therefore $c(p) = \min_{(i,j)\in p} c_{ij}$.

We denote with $\mathscr{P}(i,j)$ the set of acyclic paths in G connecting nodes $i, j \in V$. We also denote with $\mathscr{P}^*(i,j) \subseteq \mathscr{P}(i,j)$ the set of the first shortest paths necessary to ensure the routability of the demand (i,j), when considered independently of the other demands.

The demand pair $(i,j) \in E_{\mathtt{H}}$ contributes to the centrality of a node v with all the paths $p \in \mathscr{P}^*_{ij}|_v$, where $\mathscr{P}^*_{ij}|_v \triangleq \{p | v \in p \wedge p \in \mathscr{P}^*_{ij}\}$. In particular, for each path $p \in \mathscr{P}^*_{ij}|_v$, the pair (i,j) contributes to the centrality of v with a fraction of the demand d_{ij} equal to the ratio between the capacity of p, $c(p)$, and the sum of the capacities of all the paths in $\mathscr{P}^*_{ij}$. Given the supply graph G (including broken elements) and the demand graph H, the *demand based centrality* $c_d(v)$ of node v is defined as:

$$c_d(v) \triangleq \sum_{(ij)\in E_{\mathtt{H}}} \left(\frac{\sum_{p\in\mathscr{P}^*_{ij}|_v} c(p)}{\sum_{p\in\mathscr{P}^*_{ij}} c(p)} \cdot d_{ij} \right). \tag{1.3}$$

Notice that, if a static distance metric is adopted to calculate the path length, $\mathscr{P}^*(i,j)$ can be calculated offline for any demand pair (i,j), and therefore it does not affect the complexity of ISP. Nevertheless, as we discuss in Section 1.2.4, a dynamic notion of path length which takes account of whether the considered network elements are working or not, may be used to attract more flow to repaired elements.

If the adopted distance metric is dynamic, the centrality of a node may vary significantly during the unfolding of the algorithm, according to the actions provided by ISP. In this case node centrality cannot be calculated offline and needs to be updated at every iteration. In order to have a low

complexity, we calculate an *estimated set of paths* $\hat{\mathscr{P}}^*_{ij}$ as follows. We use Dijkstra's algorithm to find the shortest path p between nodes i and j. Let $c(p)$ be the capacity of such path. If $c(p) \geq d_{ij}$ this path is sufficient, otherwise we consider the residual graph in which we reduce the capacity of p by $c(p)$, and we calculate the next shortest path to satisfy a demand $d_{ij} - c(p)$, if available. For each demand d_{ij}, with endpoints $(i, j) \in E_{\mathtt{H}}$, we calculate iteratively the estimated sets of shortest paths $\hat{\mathscr{P}}^*_{ij}$. For each shortest path in $\hat{\mathscr{P}}^*_{ij}$, we can update the centrality of its nodes in linear time with respect to the path length. As a result of this procedure, we obtain an estimate $\hat{c}_d(v)$ of the centrality of each node v, using the Equation 1.3, where we replace $\mathscr{P}^*_{ij}$ with $\hat{\mathscr{P}}^*_{ij}$.

Notice that, the calculation of the centrality based ranking is performed at each iteration considering the supply graph $G^{(n)}$ (including broken elements), the current demand graph $H^{(n)}$ and the current values of link capacities which may vary iteration by iteration as a consequence of pruning actions.

1.2.3 Split of the demand

At the n-th iteration, ISP selects the node $v^{(n)}_{\mathtt{BC}} \in V$ with highest demand based centrality. The centrality ranking does not take account of disruptions, but of the potentiality of a node to contribute to an efficient routing. Hence, the centrality calculation considers the original complete supply graph G (including the broken elements), with updated residual capacities, and the current demand graph $H^{(n)}$. If $v^{(n)}_{\mathtt{BC}} \in V^{(n)}_{\mathtt{B}}$, then $v^{(n)}_{\mathtt{BC}}$ is virtually repaired at the current iteration, therefore it is removed from the set $V^{(n)}_{\mathtt{B}}$ and it is added to the *list of items to be repaired*, referred to with $\mathcal{L}(n)$. Notice that once an element is inserted in the list $\mathcal{L}(n)$ it is thereafter considered by the algorithm as if it were already repaired (more details on this list can be found in Section 1.2.5).

The next step of the algorithm ISP is the split of a demand flow over the node $v^{(n)}_{\mathtt{BC}}$. Let us consider a split action occurring at the n-th iteration. Let us consider also a demand pair $(s_h, t_h) \in E^{(n)}_{\mathtt{H}}$ of value $d^{(n)}_h$. *Splitting d_x units of the demand $d^{(n)}_h$*, with $d_x \leq d^{(n)}_h$ is the action of removing d_x units from the demand associated to the couple (s_h, t_h) and creation of two new demand edges of d_x units of flow on the node couples $(s_h, v^{(n)}_{\mathtt{BC}})$ and $(v^{(n)}_{\mathtt{BC}}, t_h)$.

Figure 1.1 illustrates the described split action.

The set of demand couples $E^{(n)}_{\mathtt{H}}$ will be updated as follows

$$E^{(n+1)}_{\mathtt{H}} = \{(s_h, v^{(n)}_{\mathtt{BC}}), (v^{(n)}_{\mathtt{BC}}, t_h)\} \cup E^{(n)}_{\mathtt{H}}. \tag{1.4}$$

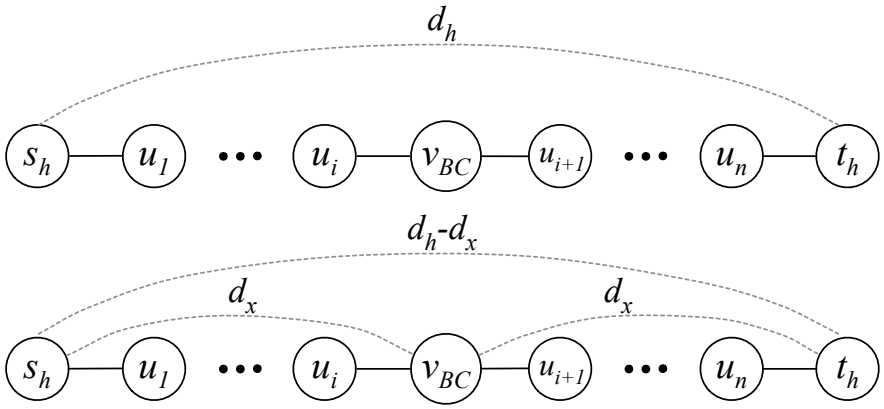


Figure 1.1: Split of d_x units of demand

The demand flows associated to the edges of $E_{\mathtt{H}}^{(n+1)}$ will be the same as in the previous iteration, with the exception of the split pair and the two new derived pairs. Therefore,

$$d_{zw}^{(n+1)} = d_{zw}^{(n)}, \forall (z,w) \neq (s_h, t_h), \tag{1.5}$$

while

$$d_{zw}^{(n+1)} = d_{zw}^{(n)} - d_x, \text{ if } (z,w) = (s_h, t_h) \tag{1.6}$$

and the new demand pairs have the following flows:

$$d_{zw}^{(n+1)} \quad = \quad d_x \text{ if } (z,w) = (s_h, v_{\mathtt{BC}}^{(n)}) | (v_{\mathtt{BC}}^{(n)}, t_h). \tag{1.7}$$

Notice also, that whenever a split action creates a new demand over an already existing demand pair, a unique demand link is created by summing the new demand to the previous.

The split action implies a routing decision, by imposing that d_x units of the split demand between s_h and t_h be routed across the intermediate node $v_{\mathtt{BC}}^{(n)}$ through which the demand has been split. Although this action requires the existence of a set of paths that can be used to route the demand, the only routing decision implied by the split action is the traversal of the node $v_{\mathtt{BC}}^{(n)}$ with d_x units of the original demand $d_h^{(n)}$. The algorithm ISP can be tuned to perform this action according to several criteria, to address two different aspects following the selection of the vertex $v_{\mathtt{BC}}^{(n)}$: (1) which demand should be split, and (2) the amount of flow to split.

Let $\mathcal{C}^{(n)}(v_{\mathtt{BC}}^{(n)}) \in E_{\mathtt{H}}^{(n)}$ be the set of demand pairs that positively contributed to the centrality value of the node $v_{\mathtt{BC}}^{(n)}$ at the current iteration, that is:

$$\mathcal{C}^{(n)}(v_{\mathtt{BC}}^{(n)}) = \bigcup_{(i,j) \in E_{\mathtt{H}}^{(n)}} \{(i,j) \text{ s.t. } \mathscr{P}^*(i,j)|_{v_{\mathtt{BC}}^{(n)}} \neq \emptyset\}.$$

Decision (1): The algorithm ISP selects the demand pair $h^{(n)} \in \mathcal{C}^{(n)}(v_{\mathtt{BC}}^{(n)})$ to be split as the one that can less likely be routed elsewhere, which can be roughly estimated by taking the demand which, if split onto $v_{\mathtt{BC}}$, would more likely use the major portion of the maximum flow between its endpoints. Therefore

$$h^{(n)} = \arg \max_{(i,j) \in E_{\mathtt{H}}^{(n)}} \frac{\min\{d_{ij}^{(n)}, \sum_{p \in \mathscr{P}^*(i,j) | v_{\mathtt{BC}}^{(n)}} c^{(n)}(p)\}}{f^*(i,j)} \tag{1.8}$$

where $f^*(i,j)$ is the maximum flow between nodes i and j on the complete supply graph G (including broken components) with currently updated capacities $c^{(n)}(\cdot)$, while $\min\{d_{ij}^{(n)}, \sum_{p \in \mathscr{P}^*(i,j) | v_{\mathtt{BC}}^{(n)}} c^{(n)}(p)\}$ is the part of demand $d_{ij}^{(n)}$ that can be routed across node $v_{\mathtt{BC}}^{(n)}$ in case of no conflicts with other demand pairs.

Decision (2): ISP decides the actual amount of demand that can be routed across $v_{\mathtt{BC}}^{(n)}$ by taking account of all potential conflicts with the other demands at the current iteration. Let d_x be such an amount, that is the part of $d_h^{(n)}$ that can be split on $v_{\mathtt{BC}}^{(n)}$ without affecting the routability of the current iteration instance of the problem on the supply graph $G^{(n)}$. The amount d_x can be calculated by solving the linear programming problem to maximize d_x under the constraints of $d_x \leq d_h^{(n)}$ and to the flow conservation and capacity constraints defined by equations (1.2), where the set $E_{\mathtt{H}}^{(n)}$ is defined according to Equation (1.4), and the demand flows are defined according to Equations (1.5), (1.6) and (1.7).

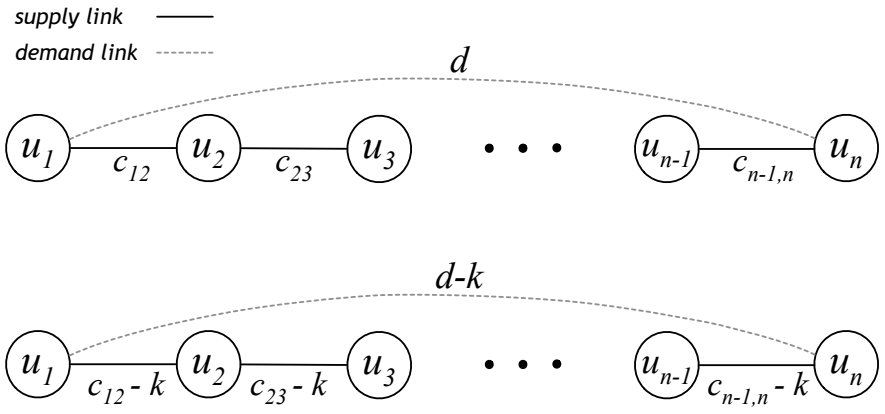


Figure 1.2: Pruning of k units of demand

1.2.4 On the use of a dynamic path metric

We use a measure of link length proportional to the cost of repairing the link or its endpoints, if any of these is broken, and inversely proportional to the link capacity. Such metric is updated every time a broken component is repaired or the residual capacity of a link is reduced due to a pruning action (see Section 1.2.6).

Formally, we define the length of the edge $e_{ij} = (i, j) \in E$ at iteration n as $l^{(n)}(e_{ij}) = [const + k^e_{ij}(n) + (k^v_i(n) + k^v_j(n))/2]/c_{ij}$, where the terms *const*, $k^v_i(n)$ and $k^e_{ij}(n)$ are as follows. The term *const* is a constant needed to account for the length of a working link. The terms $k^v_i(n)$ and $k^e_{ij}(n)$ are non null only if the corresponding elements are broken and not listed for repair in any previous iteration: therefore $k^v_i(n) = k^v_i$ if $i \in V_{\mathtt{B}}^{(n)}$, and null otherwise. Similarly, $k^e_{ij}(n) = k^e_{ij}$ if $(i, j) \in E_{\mathtt{B}}^{(n)}$ and null otherwise.

This path metric gives an extraordinary strength to the algorithm ISP because, if a decision to repair an element has been made, all successive actions will be performed accordingly. For instance, the nodes belonging to paths containing a repaired component will see an increased centrality measure after the repair, because they more likely belong to shortest paths. Henceforth, paths containing repaired components will more likely be selected for subsequent split and pruning actions.

1.2.5 Recovery of nodes and edges

The algorithm ISP works by virtually recovering network components during its execution until a sufficient number of edges and links are recovered to route the entire demand. These progressive recovery decisions alter the problem instance at any iteration. Therefore ISP considers a *list of items to be repaired* $\mathcal{L}(n)$, which is updated at any new repairing decision.

At any iteration n of the algorithm, if the best candidate $v_{\mathtt{BC}}$ is broken, that is $v_{\mathtt{BC}} \in V_{\mathtt{B}}^{(n)}$, it is added to the current list of repairs, so $\mathcal{L}(n+1) = \mathcal{L}(n) \cup \{v_{\mathtt{BC}}\}$, and the set of broken vertices is updated as follows: $V_{\mathtt{B}}^{(n+1)} = V_{\mathtt{B}}^{(n)} \setminus \{v_{\mathtt{BC}}\}$. Moreover, we repair a broken link in the supply graph if such link directly connects two endpoints of a demand, and such demand cannot be satisfied by the current repairs. Formally, if at any iteration n there is a demand $(s_h, t_h) \in E_{\mathtt{H}}^{(n)}$ that cannot be satisfied by any working path (including the links in $\mathcal{L}(n)$), and there is also a supply broken edge $(s_h, t_h) \in E \cap E_{\mathtt{B}}^{(n)}$ with the same endpoints, then the supply edge (s_h, t_h) is added to the list of repairs, that is $\mathcal{L}(n+1) = \mathcal{L}(n) \cup \{(s_h, t_h)\}$. The set of broken edges at the current iteration is also updated accordingly $E_{\mathtt{B}}^{(n+1)} = E_{\mathtt{B}}^{(n)} \setminus \{(s_h, t_h)\}$.

1.2.6 Pruning

The algorithm ISP executes the *pruning activity* to simplify the problem instance, when some units of demand can be routed over working paths. This may occur at the beginning of the algorithm execution or during its unfolding, after some split or repair actions.

According to ISP, k units of the demand flow d between a pair $(u_1, u_n) \in E_{\mathtt{H}}^{(n)}$, with $k \leq d$, can be pruned at iteration n only if there is a working path p between u_1 and u_n in the supply graph with capacity at least k. This is only a necessary condition for a demand to be *prunable*, and it does not imply that it will certainly be pruned. Figure 1.2 illustrates the pruning action. More formally, given the demand pair $(u_1, u_n) \in E_{\mathtt{H}}^{(n)}$ with a demand flow d, k units of this demand ($k \leq d$) can be pruned on path p if (1) $p \subseteq E^{(n)}$, and (2) $c(p) \geq k$. The pruning action consists in the removal of k units from the demand edge $(u_1, u_n) \in E_{\mathtt{H}}^{(n)}$ and routing these k units on a selected path p, thus subtracting the related capacity from any of the composing edges. Therefore, after the pruning action of k units, $d_{u_1,u_n}^{(n+1)} \leftarrow d_{u_1,u_n}^{(n)} - k$, and for any edge of the selected path $(i,j) \in p$, $c_{ij}^{(n+1)} \leftarrow c_{ij}^{(n)} - k$. If a demand is completely pruned, the demand pair is removed from $E_{\mathtt{H}}^{(n)}$. Moreover, if one or both of its endpoints do not belong to any other demand pair, then such endpoints are removed from $V_{\mathtt{H}}^{(n)}$.

It must be noted that, like the splitting action, the pruning action implies a routing decision which may possibly lead to an unfeasible solution of the problem. In the following, we give a sufficient condition for pruning to be feasible.

Given a demand h between the pair (s_h, t_h), the set $S_h \subset V$ is a *bubble for h* if it contains only vertices that cannot be reached by any demand node in $V_{\mathtt{H}}$ without traversing either s_h or t_h. More formally, we give the following definition.

Definition 1.2.1 (Bubble)**.** *Given a supply graph $G = (V, E)$ and a demand graph $H = (V_H, E_H)$, a set $S_h \subseteq V$, is a* bubble *for demand $h \in E_H$ if $S_h \cap V_H = \{s_h, t_h\}$, and $\forall (i,j) \in \delta_G(S_h)$, it holds that $|\{i,j\} \cap \{s_h, t_h\}| = 1$, where $\delta_G(S_h) = \{(i,j) \in E,\ s.t.\ |\{i,j\} \cap S_h| = 1\}$ is the supply cut of S_h.*

Theorem 1.2.2 (Prune conditions)**.** *Consider a supply graph G and a demand graph H, which satisfy the routability conditions given by equations (1.2). Let us consider a demand $h \in E_H$ between the pair (s_h, t_h) and flow d_h. If there is a set of working paths $\mathscr{P}(s_h, t_h)$ with maximum flow $f^*(\mathscr{P}(s_h, t_h))$ that can satisfy the demand, such that the set of vertices S_h forming the paths of $\mathscr{P}(s_h, t_h)$ is a bubble for the demand h, then the demand between s_h and*

t_h can be pruned on the paths of $\mathscr{P}(s_h, t_h)$ for an amount equal to $k_h \triangleq \min \{f^*(\mathscr{P}(s_h, t_h)), d_h\}$ without compromising the routability of the demand and without worsening the final solution in terms of recovered components.

Proof. As the paths of $\mathscr{P}(s_h, t_h)$ form a bubble, any potentially conflicting demand which requires capacity from the links of the paths of $\mathscr{P}(s_h, t_h)$ should traverse the endpoints s_h and t_h. Let us consider a potentially conflicting demand (s_q, t_q) requesting at least $f^*(s_h, t_h) - k_h + \epsilon$ units of flow, so that it is conflicting with demand (s_h, t_h) for an amount of capacity exactly equal to ϵ. Due to the hypothesis of routability of the overall demand, if the conflicting demand of ϵ of the couple (s_q, t_q) is routed in $\mathscr{P}(s_h, t_h)$, there is an alternative set of paths of capacity at least ϵ which goes from s_h to t_h traversing the nodes of $V \setminus S_h$. Therefore such an alternative path can equivalently be assigned to (s_q, t_q) without harming the routability of the demand. In terms of routability the two solutions, routing either one or the other of the two conflicting demands, are alike. Nevertheless in terms of resource consumption, the bandwidth consumed to route the demand d_h over its bubble is lower than the one potentially consumed by routing the conflicting demand d_q over the bubble of d_h. In fact, if d_q is routed over the bubble of d_h, this last demand will require the traversal of more edges than d_q to reach the alternative path. Hence routing d_h will result in the same or in a lower number of repairs than with the corresponding alternative solution. □

Notice that, in order to find demand bubbles, ISP adopts a modified breadth first search visit starting from one of the demand endpoints, and discarding all paths that lead to any endpoint of another demand. As the purpose of ISP is to minimize the number of repairs and not to find an efficient routing of the demand, any of the feasible assignments of a demand to one or several paths of one of its bubbles can be used for pruning. Moreover the pruning action must be performed by routing on the selected path the maximum amount of demand that is prunable, that is k_h which is the minimum between the maximum flow $f^*(\mathscr{P}(s_h, t_h))$ of the set of paths from s_h to t_h and the demand d_h.

1.3 Properties of ISP

Theorem 1.3.1. *The algorithm ISP terminates in a finite number of steps, which is polynomial in the input size.*

Proof. At each iteration, ISP performs either a repair, a split or a prune action. The number of repairs is limited by the number of broken network elements in the supply graph, that is $|V_B| + |E_B|$.

Let us consider the case of split actions. When a demand d_h between the pair (s_h, t_h), is split on the node v, ISP produces two new demand pairs for a flow d_x, namely (s_h, v) and (v, t_h), and updates the original pair to a demand $d - d_x$.

Let us consider the case of a partial split, where d_x is strictly lower than d. In such a case, d_x is the maximum value of splittable demand under the constraints given by Equations 1.2, with the updated demands. Due to the linearity of the problem, at least one capacity constraint acts as *binding constraint* of the linear programming problem, and is met with an equality in correspondence to the optimal. New partial splits will have new binding capacity constraints. As there is a capacity constraint for every edge, it follows that the number of partial splits is limited to the number of edges of the supply graph, that is $|E|$. This also shows that split actions can never produce infinitesimal demand values. This property is necessary to prove that also complete splits (which do not create binding capacity constraints) and pruning actions are executed a finite and limited amount of times.

We recall that the *surplus* [73] of a set of vertices $U \subset V$ is defined as: $\sigma(U) = \sum_{(i,j)\in\delta_{\mathtt{G}}(U)} c_{ij} - \sum_{(i,j)\in\delta_{\mathtt{H}}(U)} d_{ij}$, where $\delta_{\mathtt{G}}(U) = \{(i,j) \in E, \text{ s.t. } |\{i,j\} \cap U| = 1\}$ is a cut determined by U on the supply graph; similarly the cut on the demand is $\delta_{\mathtt{H}}(U) = \{(i,j) \in E_{\mathtt{H}}, \text{ s.t. } |\{i,j\} \cap U| = 1\}$. We denote with $\sigma^{(n)}(v)$ the surplus, at iteration n, of the set formed by the single vertex $v \in V$. By using the properties of cuts given in [26] we can prove that the algorithm actions affect the value of the surplus of single vertices as follows: a split action of d demand units over the intermediate vertex v decreases the surplus of v for a value of $2d$, while it leaves the other individual vertex cuts unaltered; a prune action of a demand amount of d along a path p causes a decrease of $2d$ in the surplus of the nodes belonging to p that are not endpoints of the pruned demand and leaves all other individual vertex cuts unaltered. As routability is a requirement for any action of ISP the action preserves the cut condition and all surplus will be non negative (cut condition). Therefore the number of split of any demand d on a node v is bounded by $\lfloor \sigma(v)/2d \rfloor$ which is finite and limited. Finally, let us consider the effect of pruning actions. A prune action of a demand d to a path p reduces the capacity of the edges of p of an amount equal to $\min\{d, c(p)\}$. Therefore, as the capacity of each edge is limited, the number of prune actions is also limited, as d is always finite.

□

Theorem 1.3.2. *The computational complexity of ISP is polynomial.*

Proof. Theorem 1.3.1 shows that ISP terminates in a polynomial number of iterations. We now show that each algorithm iteration is also polynomial. Let us consider the individual activities.

Complexity of routability test. Notice that the execution of the routability test requires to decide the feasibility of the set of inequalities on continuous variables (1.2), which has polynomial complexity, as detailed in [66, 45].

Notice that this complexity can be further reduced by considering the only incremental modifications of the routability problem at each iteration of the algorithm.

Complexity to calculate the demand based centrality ranking. If a static distance metric is adopted to calculate the path length, the set $\mathscr{P}^*(i,j)$ of any demand pair (i,j) can be calculated offline according to Equation 1.3, and therefore does not affect the complexity of ISP. If the distance metric is dynamic, as described in Section 1.2.4, this pre-calculation is not available and the demand based centrality is determined using the *estimate set* $\hat{\mathscr{P}}^*_{ij}$ described in Section 1.2.2. The resulting complexity is $O(|E^{(n)}| \times (|E^{(n)}| + |V^{(n)}| \log(|V^{(n)}|))$, since at each iteration we compute the shortest path between nodes i and j (complexity of the Dijkstra algorithm), which is either sufficient to route the entire demand $d(i,j)$ or at least one edge will be removed from the residual graph and a new shortest path will be considered. For each selected shortest path, we can update the centrality of its nodes in linear time with respect to the path length. Thanks to this procedure, we can refer to Equation (1.3) to obtain an estimate of the centrality of each node.

Complexity of the split action. Finding the best candidate requires $O(|V|)$ steps. In order to select the demand to be split (Decision 1), we rank all demands that contributed to the centrality of the best candidate on the basis of Equation (1.8). Calculating the demand rank costs $O(|E_{\mathtt{H}}^{(n)}|$ times the calculation of the max flow between any demand pair, which is also polynomial. We can then select the demand with highest rank in $O(|E_{\mathtt{H}}|^{(n)})$. Solving the linear programming problem to calculate d_x (Decision 2), has also polynomial complexity, using the interior point method [45] and, depending on the iteration, it is performed on problem instances of decreasing size.

Complexity of the recovery action. For each demand pair $(u,v) \in E_{\mathtt{H}}$, ISP checks if there exists a destroyed edge $(u,v) \in E$. The overall complexity is then $O(|E_{\mathtt{H}}|)$, using an adjacency matrix for E.

Complexity of the prune action. We can find the set of paths that form a bubble for each demand pair (s_h, t_h) by using a modified BFS visit starting

from s_h. Such visit discards all paths that lead to a demand endpoint which is not s_h or t_h. Since the pruning action of a demand on a path can be performed in linear time with respect to the path length, the complexity of the pruning activity is the complexity of the visits, i.e. $O(|E_{\mathtt{H}}| \times (|V| + |E|))$. □

1.4 Heuristics

In this section we describe some heuristics. Since the novelty of the problem, there are no previous solutions. We developed several heuristics, inspired by standard approaches in literature, that we propose as baseline comparison with ISP.

1.4.1 A multi-commodity based solution

One way to address the problem of finding the subset of broken components to be recovered is to minimize the amount of flow that makes use of broken links.

$$\begin{array}{lll} \min \sum_{(i,j)\in E_{\mathtt{B}}} k^e_{ij} \cdot \sum_{h\in E_{\mathtt{H}}} f^h_{ij} & & (a) \\ \sum_{h\in E_{\mathtt{H}}} (f^h_{ij} + f^h_{ji}) \leq c_{ij} & \forall (i,j) \in E & (b) \\ \sum_{j\in V} f^h_{ij} = \sum_{k\in V} f^h_{ki} + b^h_i & \forall (i,h) \in V \times E_{\mathtt{H}} & (c) \\ f^h_{ij} \geq 0 & \forall (i,j) \in E, h \in E_{\mathtt{H}} & (d) \end{array} \tag{1.9}$$

In terms of recovery decisions, this approach repairs only those broken links and vertices that are actually used by the optimal solution.

Notice that this new problem is a particular instance of the Multi-Commodity Flow problem.

Under this formulation, which is a relaxation of the problem of Equation (1.1), the problem is no longer NP-hard, but has polynomial time complexity, being it solvable efficiently with LP methods such as the interior point method [45].

Nevertheless, the multi-commodity flow formulation has a wide range of equally optimal solutions which vary significantly in the number of repaired edges and vertices. We denote with MCB and MCW the best and the worst of these solutions, respectively, in terms of number of repaired elements. Figure 1.3 illustrates the performance of MCB and MCW, versus the optimal solution of MinR and the trivial solution of repairing all broken elements (OPT and ALL in the figure, respectively). The results are obtained with the Bell-Canada topology [2, 55] by increasing the demand flow per pair under the experimental setting explained in Section 1.5. The results show

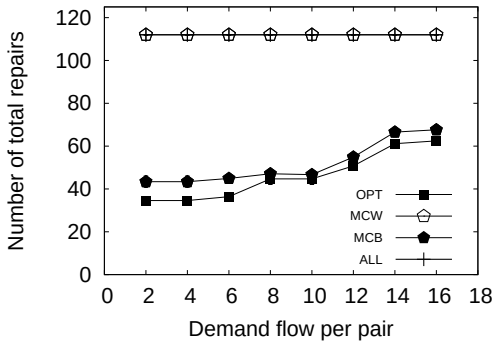


Figure 1.3: Total number of repairs of multi-commodity solution

that the multi-commodity approach has a wide solution space, which includes solutions close to the optimum as well as solutions equivalent to repairing all broken elements. Notice that the optimal solution of MinR repairs fewer network elements than MCB because it takes account of both vertex and edge repairs.

Note that finding MCB among the wide set of solutions is NP-hard, being it an instance of MinR.

For this reason we do not include the multi-commodity approach in our results.

1.4.2 Shortest Path Heuristic (SRT)

This heuristic is based on a very intuitive approach to the MinR problem, that is to consider all the demand pairs (s_i, t_i, d_i) in decreasing order of demand d_i, and repair all the shortest paths that are necessary to meet the demand requirements. Let S_i be the set including the first shortest paths for the $i-th$ demand, such that the maximum flow traversing the sub-graph formed by the only paths in S_i is at least d_i. According to SRT, for each demand d_i, all broken nodes and edges in S_i are repaired. The pseudo-code of SRT is shown in Algorithm SRT.

ALGORITHM SRT

Input: G, H, V_{B} and E_{B}

```
1 Sort demand pairs in E_H in decreasing order of d_i;
2 for i = 1, ..., |E_H| do
3     Calculate the set S_i for the demand pair (s_i, t_i, d_i);
4     Repair nodes and links of all paths in S_i;
```

This heuristic has polynomial time complexity, as it considers the demand pairs one at a time without considering potential conflicts with other demand pairs. For each demand pair, it requires to calculate the shortest paths to be repaired iteratively on a residual graph. Paths are selected until they are sufficient to meet the demand in a single flow scenario.

Notice that the sets of shortest paths of different demands may overlap, therefore the repaired links may be insufficient to route all flows and there can be some demand loss.

1.4.3 Greedy Heuristics

We developed two other heuristics based on a mapping between paths of the MinR problem and objects of an instance of a Knapsack problem. According to this mapping, we create a knapsack object for each path between a demand pair in H. The cost of repairing such path is the weight of the corresponding knapsack object, while the path capacity is the object value. Both heuristics make use of the set $P(H, G)$ of all simple paths between the demand pairs in H.

Notice that the number of paths in $P(H, G)$ is potentially exponential in the graph size, hence these heuristics can only be adopted if paths are pre-computed offline.

Thanks to the described Knapsack analogy, we can formulate two different heuristics based on the greedy approach to Knapsack [65].

ALGORITHM GRD-COM

Input: G, H, $V_{\mathtt{B}}$, and $E_{\mathtt{B}}$

```
1  Calculate (offline) P(H, G);
2  for p ∈ P(H, G) do w(p) = cost(p) / capacity(p);
3  Sort paths according to their weight;
4  while ∃ unsatisfied demands and available paths do
5      Let p be the next path, (s_i, t_i, d_i) its demand pair;
6      Repair p;
7      Assign a quantity of demand min{d_i, capacity(p)} to p;
8      Update G and H;
9      for each routable demand flow (s_k, t_k, d_k), k ≠ i do
10         Assign the maximum quantity of demand;
11         Update G and H;
```

Greedy Commitment (GRD-COM)

The first heuristic, called *Greedy Commitment* (GRD-COM), assigns to each path $p \in P(H, G)$ a weight $w(p) = \frac{cost(p)}{capacity(p)}$, where $cost(p)$ is the sum of the costs of repairing the edges composing p, while $capacity(p)$ is the residual capacity of p.

GRD-COM sorts the paths in $P(H, G)$ in ascending order of weight, and iteratively repairs paths following this order. Let p be the path repaired at the current iteration, and (s_i, t_i, d_i) the demand pair for which p was included in $P(H, G)$. GRD-COM assigns the maximum possible quantity of such demand to p, and updates the residual capacities of edges and the residual demand accordingly. It then verifies if also some other demands may be routed through the current graph, considering all the paths already repaired including p. The algorithm proceeds to the next iteration, selecting the next path in the order. GRD-COM terminates as soon as all demands are satisfied, or there are no more paths to repair. The pseudo code is shown in Algorithm GRD-COM.

Note that considering the residual graph capacities allows a lower amount of repairs with respect to the following greedy heuristics GRD-NC, but as in the case of SRT, there is no guarantee that all the demands can be satisfied due to the possibility to have wrong routing decisions, which may create inhibiting flow allocations, even if the capacity of the repaired edges is enough to route the demand.

ALGORITHM GRD-NC

Input: G, H, $V_{\mathtt{B}}$, and $E_{\mathtt{B}}$

1 Calculate (offline) $P(H, G)$;
2 **for** $p \in P(H, G)$ **do** $w(p) = \frac{cost(p)}{capacity(p)}$;
3 Sort paths according to their weight;
4 **while** *routability test fails* **do**
5 Repair the next path p;

Greedy Commitment (GRD-NC)

The second heuristic is called *Greedy No-Commitment* (GRD-NC). It is also inspired by the Knapsack heuristics, and similarly to GRD-COM, it makes use of the set of all paths $P(H, G)$ and path weights $w(\cdot)$.

GRD-NC repairs paths one by one following the ascending order of weights, but it does not provide a routing assignment of flows to paths unlike GRD-COM. On the contrary, it evaluates the routability of the overall demand,

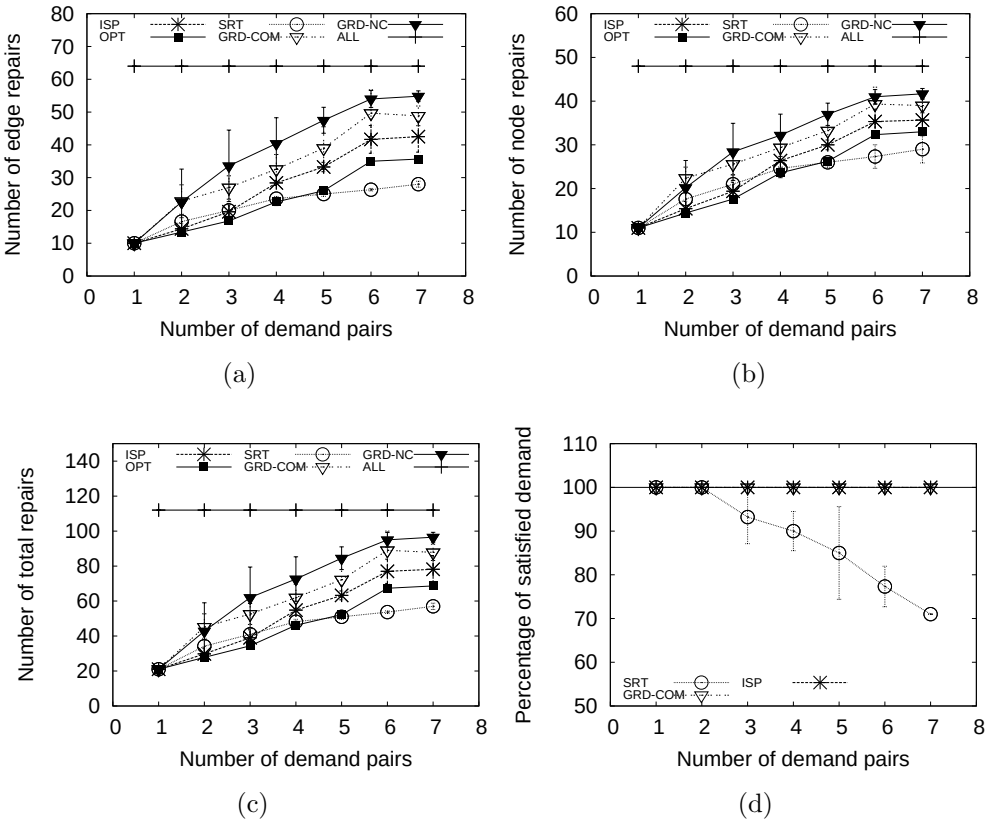


Figure 1.4: Bell-Canada topology. Varying number of demand pairs (10 flow units/pair). Repaired edges (a), repaired nodes (b), total repairs (c) and demand loss (d).

given the current repaired paths, using the routability test described in Section 1.2.1. GRD-NC terminates as soon as all demands are routable with the current repairs. The pseudo code is shown in Algorithm GRD-NC.

Note that unlike GRD-COM, GRD-NC does not provide an update of the path capacity at each step, since there is no routing assignment after the repairs. As a consequence, this heuristic can repair more edges and vertices than GRD-COM, but it has the advantage that if the demand is routable in the original graph before the disruption, the heuristic finds a solution with no demand loss.

1.5 Experiments

In the experiments we consider both real and synthetic topologies of various size to highlight different aspects of the performance of the algorithms.

We start the analysis with a real network topology of small size so that optimal solutions may be obtained in a reasonable time, and to provide a thorough experimental comparison of all the algorithms discussed in this work.

In the second scenario we instead show the results on a real large size topology, to evidence the good approximation of ISP to the optimal solution even with a large problem size.

The last experimental scenario is based on synthetic topologies of varying complexity, to study the computational time of the proposed heuristics and of the optimal solution. We will evidence the poor scalability of the optimal approach, motivating the need to resort to heuristic solutions.

In all the following experiments, where not otherwise stated, we average the results over 20 runs.

1.5.1 First scenario: small size topology

In this set of experiments we consider the Bell-Canada topology, taken from the Internet Topology Zoo [2, 55] collection. This network has 48 nodes and 64 edges. The data set provides uniform edge capacities, which we manually altered to consider non homogeneous capacities. In particular we consider two backbones with capacity 30 and 50, respectively, while all remaining edges have capacity 20. We use a homogeneous unitary repairing cost for damaged nodes and edges.

We build the demand graph $H = (V_{\mathtt{H}}, E_{\mathtt{H}})$ as follows. We select the demand pairs to be far apart in the supply graph. In particular, we randomly select the demand pairs among those which have a hop distance greater than or equal to half the diameter of the network.

We perform four sets of experiments. In the first set (Section 1.5.1) we fix the flow per pair, and increase the number of pairs in the demand graph. In the second set (Section 1.5.1), we fix the number of demand pairs and increase the demand flow per pair. In both these experiments, we considered a complete destruction of the supply graph, in order to have the maximum range of potential solutions. On the contrary, in the third set of experiments (Section 1.5.1) we consider different failure scenarios according to a geographically correlated failure model. Finally, the fourth scenario considers the case of heterogeneous costs.

Variation of the number of demand pairs

In these experiments we increase the number of demand pairs from 1 to 7, and each demand pair has a requirement of 10 flow units. Figures 1.4(a) and (b) show the number of edges and nodes repaired by the considered approaches, respectively. Figure 1.4(c) shows the cumulative number of repairs. In the figures, the line ALL refers to the total number of destroyed nodes and links.

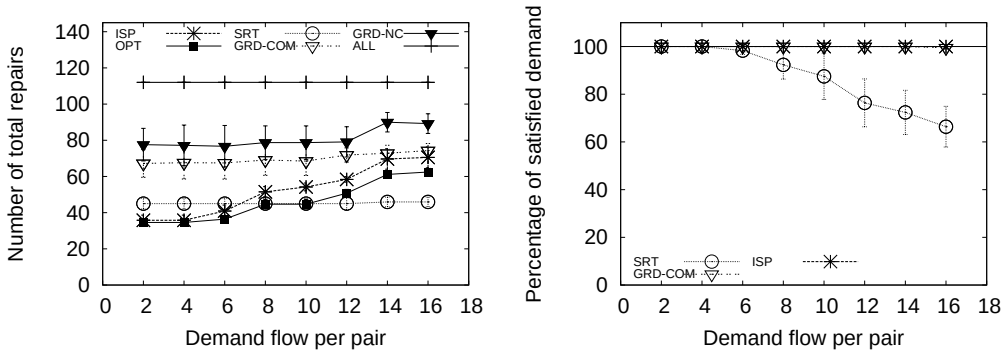
The experiments shown in Figures 1.4(a)-(c) highlight that by linearly increasing the number of demand pairs, the number of repaired edges and vertices also grows.

ISP is the closest to the optimal among the considered heuristics. In the most critical setting, with 7 demand pairs, OPT repairs 37 edges, ISP repairs 42 edges, while GRD-COM repairs 49 edges and GRD-NC repairs 55 edges. The number of repaired vertices are consequently proportional, as in this experimental scenario the entire network is damaged by the destruction. We highlight that the greedy solutions are much more computationally expensive than ISP, due to the necessity to find all paths between any demand pairs. It is also worth noting that ISP better approximates the optimal solution when the demand requirements are low with respect to the available bandwidth in the network. This result is visible in Figures 1.4 (a)-(c), when the number of demand pairs is less than 4.

As the figures show, SRT results in the lowest number of repairs, however SRT, and similarly GRD-COM, does not ensure that all demand flows can be routed. In particular, SRT repairs the number of shortest paths up to the minimum necessary to satisfy each demand, treating demands independently. As the number of demand pairs increases, the paths selected by SRT are more likely to be shared. Therefore when these shared paths are saturated, the policy SRT is not able to satisfy all demands, as Figure 1.4(d) shows. In these experiments, this occurs when the number of pairs grows from 2 to 3. This behavior reflects the fact that two pairs can be commonly routed on a path of 20 capacity units, but when 3 demand pairs require 30 capacity units, the shortest paths may have some edge in common and a portion of demand is lost. These arguments explain the initial constant behavior of the demand loss shown in Figure 1.4(d) and in the analogous figures of the following sets of experiments. Due to the similarity of the behavior of the policy SRT in all the following experiments, we will not comment on this policy any longer

Variation of the demand intensity

In this section we introduce a dual experiment in which we fix the number of demand pairs to 4, and we vary the intensity of demand per pair.



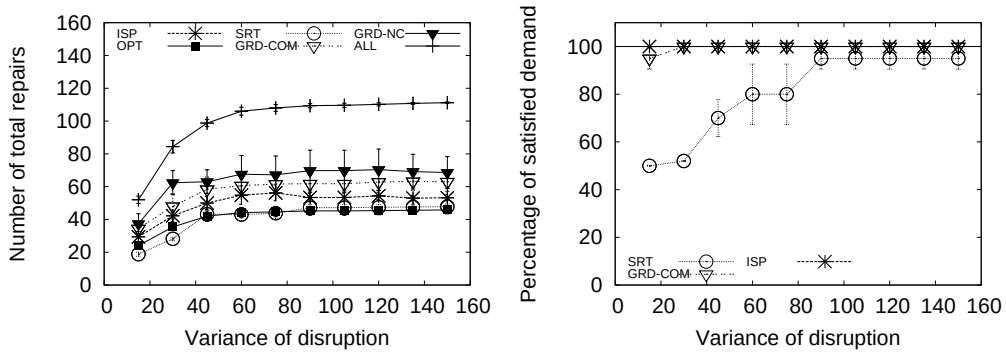
(a) (b)

Figure 1.5: Bell-Canada topology. Varying the intensity of demand flow (4 demand pairs). Total repairs (a), demand loss (b).

Figures 1.5(a) and (b) show the total number of repaired elements and the demand loss. We observe a similar behavior to what we discussed for the previous set of experiments. Nevertheless there are some aspects worthy of note.

Even if the global demand increase of this experiment is the same of the previous experiment, all policies tend to reveal a smoother increase in the number of repairs when the number of demand pairs is fixed. This is due to the need to repair damaged elements to at least connect the demand pairs, even when the demand intensity is low with respect to the link capacity. Such repairs are sufficient until the demand reaches an intensity for which more repairs are needed. This justifies the step-wise behavior of OPT and ISP.

The above reasoning helps understanding the trend of the greedy heuristics with respect to the intensity of the demand. These approaches blindly repair paths with high rank until all demands are satisfied. When the demand intensity is low, and basically only connectivity between demand pairs is needed, these heuristics still repair all paths in the list which have a higher rank than those required for connectivity. As the demand increases, this high number of paths is still sufficient to serve the demand, and hence further repairs are not needed. However, when the demand increases further, a bunch of additional paths are repaired, as shown in Figure 1.5(a) in correspondence of the increase in demand intensity from 12 to 14 for the heuristic GRD-NC.



(a) (b)

Figure 1.6: Bell-Canada topology. Varying the extent of destruction (4 demand pairs, 10 flow units/pair). Total repairs (a) and demand loss (b).

Variation of the extent of destruction

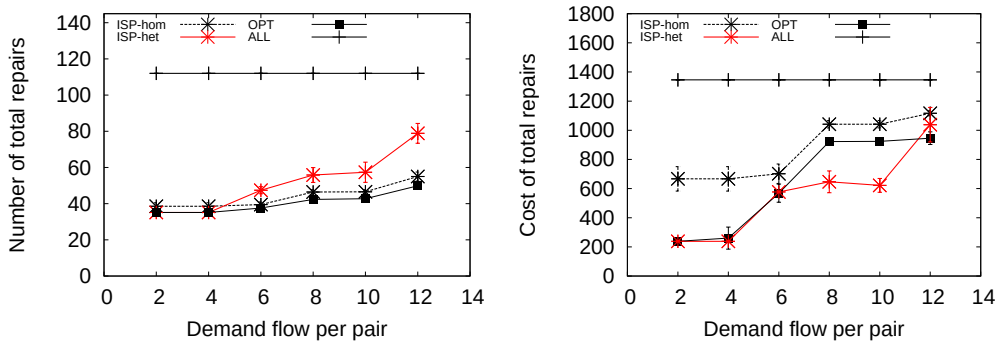
In this experiment we consider the impact of the extent of destruction. We consider a geographical failure model, to represent natural disasters and intentional attacks.

We generated the disruption according to a bi-variate Gaussian distribution of the disruption probability of network components. We varied the variance of such a distribution and scaled the probability accordingly to obtain larger failures with larger variance.

In these experiments we consider 4 demand pairs, each with a demand intensity of 10. We consider an increase in the amount of disrupted components obtained by varying the variance of the disruption. We consider the epicenter at the barycenter of the nodes in the network, and same variance in both dimensions of the bi-variate distribution of failures.

Figures 1.6(a) and (b) show the total number of repaired elements and the percentage of demand loss, respectively. The line labeled ALL shows how many edges or vertices are disrupted in the considered instance of the problem.

Even in this setting we observe similar behavior of the considered policies, which highlights the superiority of ISP. In particular, ISP performs close to the optimal, and when the network is almost completely destroyed (i.e. a variance equal to 150) ISP repairs only 53 elements, with respect to the 46 elements repaired by the optimal solution, whereas GRD-COM requires 63 repairs and GRD-NC requires 68 repairs.



(a) (b)

Figure 1.7: Bell Canada topology. Heterogeneous recovery costs. Total repairs (a), demand loss (b).

Heterogeneous costs

To complete our analysis on the Bell Canada topology, we considered a final set of experiments, in which we modeled heterogeneous repair costs to highlight the ability of ISP to adapt its choices to reduce the total cost of repairs and not simply the number of repairs. To make this scenario more realistic, we set the cost of the links according to their capacity, and considered a complete disruption of the network. We considered two backbones of different capacity: the first with 50 capacity units, with 80 cost units per link, and the second with 30 units of capacity and a corresponding cost per link of 30 cost units. The rest of the links have 20 units of capacity and a related cost of 4 units. We also considered a unitary uniform cost for node recovery. We considered 4 demand pairs and we increased the corresponding flow requirements.

The results are shown in Figure 1.7. In the figure, the optimal policy is calculated with objective function equal to the number of repairs. We consider two variants of ISP: *ISP-hom* which considers uniform repair costs and therefore acts in a cost-blind manner, and *ISP-het* which takes account of the cost when making its decisions, by considering a cost proportional distance metric.

In terms of number of repairs, shown in Figure 1.7(a), ISP-hom performs better than ISP-het, since at the same cost ISP-hom chooses the links with highest capacity. By contrast in terms of total cost of repairs, as shown by Figure 1.7(b), ISP-het performs better than ISP-hom by preferring decisions that minimize the total cost of repairs.

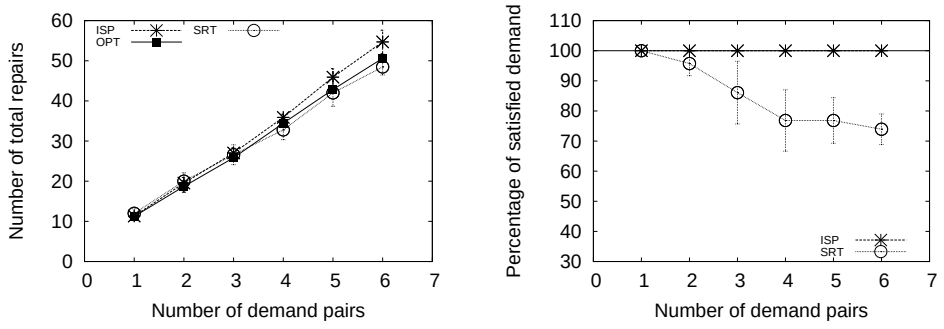
Figure 1.8: CAIDA topology AS28717, with 825 nodes and 1018 edges.

We did not run the greedy heuristics in these experiments, as despite their simplicity, they do not scale to large topologies due to the necessity to calculate all the paths between demand pairs, which is of exponential complexity in the size of V.

1.5.2 Second scenario: big size topology

For this second scenario, we consider the real topology AS28717 of Figure 1.8, taken from the CAIDA (Center for Applied Internet Data Analysis) resource collection [23]. This topology represents IP-level connections between backbone/gateway routers of several ASs from major Internet Service Providers (ISPs) around the globe. Since CAIDA topologies are often disconnected, we selected the giant connected component, which has 825 nodes and 1018 edges.

In this set of experiments, we consider random capacity links between 20 and 50 units, 22 units of flow per demand and we focus on vary the number



(a) (b)

Figure 1.9: CAIDA topology AS28717. Varying the number of demand pairs (22 flow units per pair). Total repairs (a), demand loss (b).

of demand pairs. Figure 1.9(a) shows the total number of repairs, while Figure 1.9(b) shows the demand loss. Even in this scenario, ISP performs close to the optimal, and does not show any demand loss. This behavior is similar to the one shown in the previous scenario for small size topology. This confirm that ISP is able to find a good approximation to MinR independently from the size or the structure of the network. Notice that even for big size topology scenario, the number of repairs under heuristic SRT is also comparable to the optimal, but the demand loss in this case is considerably high confirming that this heuristic is not able to accommodate all the demand flows.

1.5.3 Third scenario: simulation time comparison

In this last scenario, we analyze the scalability of ISP and OPT. We consider synthetic network topologies of increasing complexity and we evaluate the performance and the computation time of the two algorithms.

We considered an Erdos-Renyi topology [35] with 100 nodes. We recall that in an Erdos-Renyi graph, any two nodes are connected through an edge with probability p (*edge probability*). In the experiments of Figure 1.10 we varied the parameter p.

As the purpose of this set of experiments is to evaluate the algorithm scalability, we consider a relatively simple problem instance in which we have only a connectivity requirement, with a construction similar to the one of the proof of Theorem 1.1.1 (an instance of the Steiner Forest problem).

We modeled the link capacity and flow demand as follows: we considered 5 demand pairs, of one unit each, and we analyzed the case of a completely destroyed network, where each link has a capacity of 1,000 units of flow.

Despite the relative simplicity of the problem formulation (only connectivity requirements), by growing p we increase the difficulty of the problem.

In Figure 1.10(a) we focus on the execution time of ISP and OPT. For the calculation of the optimal solution we implemented problem 1.1 using Python and the Gurobi [1] library, which is known for its efficiency. For these experiments we used a 20 core/40 thread architecture composed of 2 Intel(R) Xeon(R) CPU ES-2680 v2 (2.80GHz) and 64GB RAM, running Ubuntu 14.04. The experiments show that the optimal solution has a prohibitive execution time, which as expected grows significantly with the parameter p. For instance, we observe that when p=0.9 OPT requires 10^5 secs (about 27 hours), on average.

The execution time of ISP is negligible and not affected by this parameter setting. When p=1 the problem becomes trivial, as the supply network is a clique, and the optimal solution consists in repairing the endpoints of each demand pair and the edges connecting them.

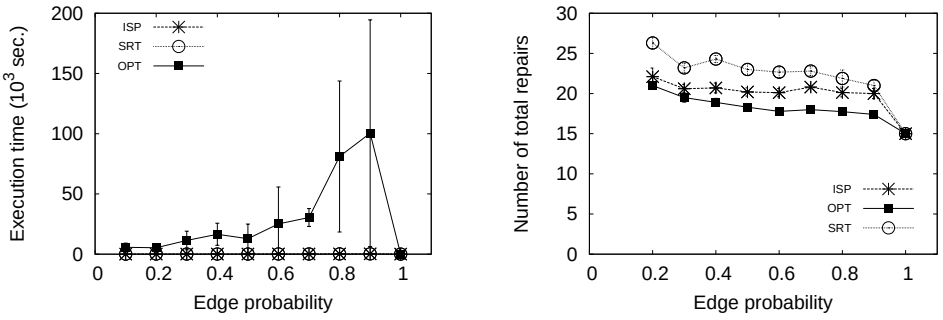
Notice that, when p grows, the graph becomes non planar and in the case of non-planar graphs, the Steiner Forest problem is known to be APX-hard [49], hence we do not expect a good approximation of the optimal solution.

In fact, Figure 1.10(b) shows that the gap between ISP and OPT is much higher than in the other experiments which used real topologies. This is because, as observed in [21], real topologies are typically planar or mostly planar. Nevertheless, ISP is still repairing a number of elements close to the optimal, and lower than the number of repairs under SRT. Notice also that in the case of p=1 the number of repairs is 15 for all the three plotted algorithms, as the supply network is a clique and all the algorithms are able to find the trivial solution of repairing the endpoints of each demand pair and the links between them, for a total of 5 pairs.

For these experiments, the link capacity is so high that none of the heuristics has any demand loss.

Notice also that we do not plot the greedy heuristics that are based on the pre-computation of the list of all paths, because with high values of p they would require $O(N!)$ steps.

In the next Chapter 2, we remove any assumptions on the knowledge of the disruption and we study how to progressively restore demand flows among the critical infrastructures by alternating interventions for repair and for monitoring the network status.



(a) (b)

Figure 1.10: Erdos-Renyi topology. Varying edge probability p. Execution time (a), number of total repairs (b).

Chapter 2

CEDAR: Progressive Network recovery Under Incomplete Knowledge of the Disruption

In the previous Chapter 1, we addressed the problem on how to efficiently restore critical services flow after a massive failure event. After modeling the problem of MinR, we developed a polynomial algorithm ISP that approximate the solution of the NP-hard problem. As discussed, ISP assumes, as input, the complete knowledge of the sets of nodes and links destroyed. However, this information is not always available or it is only available as a partial information. In this chapter we remove any assumption on the knowledge on the disrupted area, and we study how to alternate efficiently interventions for repair and interventions for monitoring the network in a progressive manner. Due to the progressive nature of the problem, the target is not anymore the number of elements repaired, but how fast the flow among the critical services is restored and made available.

2.1 Problem definition and assumptions

In this section, we introduce the problem of progressive damage assessment and network recovery (PDAR) which aims at finding a schedule of *repair interventions* to restore a set of critical demand flows as fast as possible, under constrained recovery resources. Furthermore, PDAR works with partial and progressively available knowledge of the status of the network which is the result of network probing. As long as the repair interventions provided by PDAR are executed, monitoring probes can find new working paths to explore new areas of the network. Moreover, new working nodes can be used as monitors.

The progressive damage assessment and network recovery considers subsequent stages of execution as illustrated in Figure 2.1. Whenever new information is available, the current stage ends and a new stage begins with the *information update* action. The new information is then used to determine a *decision* on the next schedule of repairs. A *recovery phase* follows, with repair interventions and monitor placement, until the next information update becomes available. Notice that, as the information available to PDAR is only partial, a repair intervention may be scheduled also on network elements for which no information is available on its status (*unknown status*). Due to the uncertainty on the status of such elements, at a local inspection, them may result to be properly working. In order to keep the problem formulation simple, we do not incorporate monitor placement actions in the decision problem, but we assume the following:

- software monitors are placed on all the nodes that are selected for a repair intervention (both in the case of broken nodes that have been

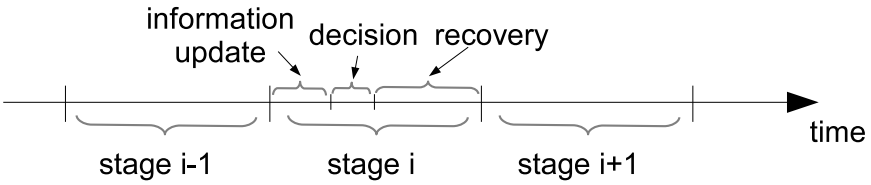


Figure 2.1: Stages of PDAR

repaired and in the case of nodes with unknown status that had been scheduled for repair but were found working after a local inspection);

- each new monitor node probes the surrounding network until it is able to determine its connected working component. In addition to probing, cable diagnostic devices, such as reflectometers, are used, when available, to determine the status of the adjacent lines of a monitor, if the next hop neighbors are unreachable;

- the demand endpoints are the first nodes to be repaired, and to host network monitors.

Although for practical purposes it is often desirable to limit the monitoring activity to a given number of hops from the monitor nodes, we assume that a monitor obtains knowledge of its entire connected component in the working graph $G_{\mathtt{W}}$. Notice that the monitoring activity and the consequent information update, trigger the transition to a new stage of PDAR, because it may find a more efficient repair schedule. Nevertheless, if the monitoring activity does not provide any update, or the only unknown elements that are discovered are actually broken or are isolated elements, the current schedule is kept unchanged as PDAR would provide the same solution.

2.1.1 The PDAR optimization problem

Before formalizing the PDAR problem, we recall and extend the nomenclature introduced in Chapter I to consider the partial knowledge of the network failures. Therefore the set V is partitioned into the three sets $V_{\mathtt{W}}$, $V_{\mathtt{B}}$, and V_u of working, broken and unknown-status nodes, respectively. The set E is likewise partitioned into the sets $E_{\mathtt{W}}$, $E_{\mathtt{B}}$, and E_u. We define the *working graph* $G_{\mathtt{W}} = (V_{\mathtt{W}}, E'_{\mathtt{W}})$, where $E'_{\mathtt{W}} = E_{\mathtt{W}} \setminus \{(i,j) \in E_{\mathtt{W}} | \{i,j\} \cap (V_{\mathtt{B}} \cup V_u) \neq \emptyset\}$. Namely, the working graph is formed by the working nodes of the supply graph, and by the working edges that are not incident to broken nodes or nodes with unknown status. Hereafter, we refer to a *recovery action* as the

local intervention on a node $v \in V_{\mathtt{B}} \cup V_u$ or a link $(i,j) \in E_{\mathtt{B}} \cup E_u$ to determine its status if unknown, and to restore its functionality if broken[1]. The recovery of a node includes the installation of a software monitor.

Since the PDAR problem works under the constrain of limited recovery resources, we consider a time based budget of repair resources, denoted with $B_{\mathtt{repairs}}$, for instance human personnel or vehicles, which determines the amount of repair interventions that can be performed in a same time period. Due to the constrained repair resource budget, only a limited set of repairs can be executed in parallel. For this reason, PDAR schedules repairs according to the time availability of repair resources. It considers the *stage* as a sequence of successive *steps* in which the maximum number of parallel repairs is bounded due to the budget constraint $B_{\mathtt{repair}}$.

Therefore, the PDAR optimization problem works in a sequence of at most N repair steps to be performed at each stage, where N is the maximum number of steps that are necessary to repair all the broken elements. Nevertheless, the sequence of repairs, which will be executed in the recovery phase of the stage, will be terminated if a useful information update is determined. In such a case PDAR will move to the next stage, before reaching the N-th step.

At each step n, there is an update of the composition of the sets of working, broken and unknown network elements. We will add the argument (n) to the notation of these sets when we want to refer to the specific composition they have at stage n. Each stage s starts with $n = 0$, with input consisting of the supply graph G, the demand graph H, and the current stage estimate of the damages, represented by the sets of certainly broken elements $V^s_{\mathtt{B}}(0)$ and $E^s_{\mathtt{B}}(0)$ and by the sets $V^s_u(0)$ and $E^s_u(0)$ of elements whose state is unknown. For example, the discovery of the status of a node $v \in V^s_u(n-1)$ after a local inspection implies that $V^s_u(n) \leftarrow V^s_u(n-1) \setminus \{v\}$ and the node v is added to either $V^s_{\mathtt{W}}(n)$ if v is functional, or to $V^s_{\mathtt{B}}(n)$ if v is broken. A similar update occurs when the monitoring activity brings new information on the status of a link which was previously unknown.

The purpose of PDAR is to find a step-based schedule of repairs, which determines the sequence of repair interventions within $V^* = V^s_{\mathtt{B}}(0) \cup V^s_u(0)$ and $E^* = E^s_{\mathtt{B}}(0) \cup E^s_u(0)$ that optimizes the *accumulative demand flow over N steps* $F^*(N)$. This value is defined as follows: $F^*(N) = \sum_{n=1}^{N} f(n)$, where $f(n) = \sum_{h \in E_{\mathtt{H}}} d_h \cdot \alpha_h(n)$, and $\alpha_h(n) \in [0,1]$ is a variable representing the percentage of the demand flow d_h that is routed at the n-th step.

[1]For simplicity of presentation we only consider repair actions, although a more general formulation is possible, considering that each broken network component can be repaired or replaced with devices of analogous or different characteristics.

Let the variables $f_{ij}^h(n) \in \mathbb{R}$, with $f_{ij}^h(n) \geq 0$, represent the fraction of the demand flow h that is routed through the link $(i,j) \in E$, going from vertex i to vertex j, at the completion of the n-th step. Notice that other flows may traverse the same edge in the opposite direction.

Also consider the binary variables $x_{ij}(n)$ and $y_i(n)$. The variable $x_{ij}(n) = 1$ if there is a recovery intervention on edge $(i,j) \in E$ exactly at step n, while $x_{ij}(n) = 0$ otherwise. The variable $y_i(n) = 1$ if node i is repaired at step n, and $y_i(n) = 0$ otherwise. For an edge that has been repaired at the k-th step, it is $x_{ij}(k) = 1$, and $x_{ij}(l) = 0$ for $l \neq k$. We consider working elements as repaired at the 0-th step. For instance, if the node $i \in V_{\mathtt{W}}$ it is $y_i(0) = 1$ and $y_i(n) = 0$ for any other step $n \neq 0$. For all the elements of $(i,j) \in E^*$ it is $x_{ij}(0) = 0$ (initially broken), while if edge $(i,j) \in E_{\mathtt{W}}$ it is $x_{ij}(0) = 1$, and $x_{ij}(n) = 0$ if $n \neq 0$ (working links are considered as links that were repaired at the 0-th stage). Similarly, $y_i(n)$ represents the binary decision to repair node $i \in V$ at the end of the n-th stage. If the node $i \in V_{\mathtt{W}}$ it is $y_i(0) = 1$ and $y_i(n) = 0$ if $n \neq 0$. If the node $i \in V^*$ then $y_i(0) = 0$.

The capacity constraint of the problem is expressed by Equation 2.1(a). If a link (i,j) is still broken at step n, its flow is null, otherwise the flow is bounded by c_{ij}. Notice that if an edge (i,j) is repaired, the corresponding nodes i and j must also be repaired if broken, which implies that $\sum_{k=0}^{n} y_i(k) \geq x_{ij}(n)$, $\forall i,j \in V, \forall n$ as in Equation 2.1(b).

The flow balance constraint is expressed by Equation 2.1(c). In this equation $b_i^h = d_h$ if i is the source of the demand flow h, $b_i^h = -d_h$ if i is the destination, and $b_i^h = 0$ otherwise to balance incoming and outgoing flow.

Finally, Equation 2.1(d) constrains the cost of repairs for each of the N stages, to be limited to the per step budget $B_{\mathtt{repair}}$[2]. Equations 2.1(e-g) denote the domain of the variables of the problem, while Equations 2.1(h-i) initialize the values of the decision variables for the first stage.

We consider the optimization of the accumulative flow over a horizon of N stages. The PDAR optimization problem is therefore formulated in the variables $x_{ij}(n)$, $y_i(n)$, $\alpha_h(n)$ and $f_{ij}^h(n)$ as follows (we omit the statement

[2]This formulation does not consider budget rollover from one repair step to the next in the case of partially depleted budget. This is because we want to use this model to represent limited repair resources, such as vehicles or human personnel.

$\forall n$ in all the constraints for clarity):

$$
\begin{array}{ll}
\text{Max} \sum_{n=1}^{N} \sum_{h \in E_{\mathtt{H}}} d_h \cdot \alpha_h(n) & \\
\text{subject to, for all } n = 1, \ldots, N: & \\
c_{ij} \cdot \sum_{k=0}^{n} x_{ij}(k) \geq \sum_{h=1}^{|E_{\mathtt{H}}|} (f_{ij}^h(n) + f_{ji}^h(n)), \quad \forall (i,j) & (a) \\
\sum_{k=0}^{n} y_i(k) \geq x_{ij}(n), \quad \forall (i,j) & (b) \\
\sum_{j \in V} f_{ij}^h(n) = \sum_{k \in V} f_{ki}^h(n) + b_i^h \cdot \alpha_h(n), \quad \forall (i,h) & (c) \\
\sum_{(i,j) \in E^*} x_{ij}(n) \cdot k_{ij}^e + \sum_{i \in V^*} y_i(n) \cdot k_i^v <= B_{\mathtt{repair}} & (d) \\
f_{ij}^h(n) \geq 0, \quad h \in E_{\mathtt{H}} & (e) \\
y_i(n), x_{ij}(n) \in \{0,1\}, \quad \forall i \in V,\ (i,j) \in E & (f) \\
\alpha_h(n) \in [0,1], \quad h \in E_{\mathtt{H}} & (g) \\
y_i(0) = 0, \text{ if } i \in V^*; \quad y_i(0) = 1, \text{ if } i \in V_{\mathtt{W}} & (h) \\
x_{ij}(0) = 0, \text{ if } (i,j) \in E^*; \quad x_{ij}(0) = 1, \text{ if } (i,j) \in E_{\mathtt{W}} & (i)
\end{array}
\tag{2.1}
$$

Since a simpler instance of the problem PDAR, i.e. the MinR problem, has been proven to be NP-hard in Theorem 1.1.1 in Section 1.1, the PDAR problem is also NP-hard.

2.2 The algorithm CeDAR

In this section, we propose a polynomial algorithm, called *Centrality based Damage Assessment and Recovery (CeDAR)*, to solve the PDAR problem introduced in previous Section 2.1. We consider a progressive monitoring and network recovery in multiple stages, as in Figure 2.1. CeDAR aims at maximizing the accumulative flow over time, as follows:

- prioritizing the repair of network components that can accommodate higher flow, by using a dynamic ranking of broken and unknown elements, based on their centrality with respect to the demand;
- scheduling the repairs of the same-path elements all at once (or in an interrupted sequence, if not allowed by the time based constraint on repair resources) in order to make the repaired components immediately available for flow routing.

For these reasons, CeDAR obtains a high accumulative flow throughout the entire execution period, even when the recovery and monitoring activities are still in progress.

Before to describe in details CeDAR, we briefly focus how CeDAR works. CeDAR determines a repair schedule by privileging the repair of paths whose

nodes have high *demand based centrality*, as we defined in the previous chapter in the designing of ISP (see Definition 1.3 in Section 1.2.2 for details). Furthermore, we extended the definition of dynamic notion of distance introduced in Section 1.2.4, to consider the unknown elements with their cost of repair and link capacity. In this way, when CeDAR makes the decision to repair an element, all successive actions will be performed accordingly. In fact, according to this dynamic notion of distance, the length of a path containing repaired elements will be updated to a lower value, hence the centrality of its nodes will increase, and the path will attract more flow. As result, the algorithm will concentrate demand flows on the repaired components. To make paths immediately available for routing flows, CeDAR aims at repairing entire paths in uninterrupted sequence. To progressively reduce the instance of the problem, CeDAR routes (*prunes*) demands on known paths rather than to continue to place monitors to discover another potentially shortest path. This policy allows CeDAR to drastically reduce the cost for monitoring, achieving at the same time good performance in terms of number of elements repaired, as confirmed in the simulations (Section 2.5).

2.2.1 Definitions and notation

To better understand the algorithm CeDAR in detail, we introduce the following notation and definitions.

Each iteration of CeDAR potentially provides an update of the current view of the status of the network. Hence, subsequent iterations correspond to different stages of the PDAR problem, according to the nomenclature introduced in section 2.1. Notice that some iterations provide long sequences of repair interventions, which may require several steps, within the time constraint on the available repair resources.

At each stage CeDAR performs new repairs and simplifies the problem instance by reducing demand and link capacities according to an operation called *demand pruning*, formalized in Definition 2.2.1 and already introduced in Chapter 1, Section 1.2.6. With $G_{\mathtt{W}}(n)$ we denote the composition of the working graph, and with $d_h(n)$, for $(s_h, t_h) \in E_H(n)$ and $c_{kl}(n)$, for $(k, l) \in E$, we denote the demand and capacities updated at the n-th stage. With $d_h(0)$ and $c_{kl}(0)$ we denote the initial values of demand and capacity (before the disruption).

Depending on the needs of the discussion, the same path is equivalently described as an ordered list of links p, or as a subset of nodes and links, and denoted with $\hat{p}$.

Definition 2.2.1 (Pruning of a demand)**.** *Let us consider a demand of* $x \leq d_h(n)$ *units of flow between the endpoints* s_h *and* t_h*, with* $(s_h, t_h) \in E_H(n)$*, at the current stage* n*. Let* p *be a path between* s_h *and* t_h *in* $G_W(n)$*, that is* $\hat{p} \subset V_W(n) \cup E_W(n)$*, for which* $x \leq min_{(i,j)\in\hat{p}} c_{ij}(n)$*, for all* $(i,j) \in \hat{p}$*.* Pruning x units of demand $d_h(n)$ on path p *consists in the decrease of demand* $d_h(n)$*, so that* $d_h(n+1) = d_h(n) - x$*, and in the corresponding update of the link capacities of* p*:* $c_{ij}(n+1) = c_{ij}(n) - x$*, for all* $(i,j) \in \hat{p}$*.*

The following notion of routable instance, constitutes the core of the termination condition of the CeDAR. We recall the routability test introduced in Chapter 1, Section 1.2.1. When the current demand is routable on the current working graph, without the need of additional repairs, the algorithm CeDAR terminates.

Definition 2.2.2 (Routable demand)**.** *Given a demand graph* $H(n)$ *at stage* n*, and the currently working graph* $G_W(n)$*, with currently updated capacities* $c_{ij}(n)$*, for any* $(i,j) \in E_W(n)$*, we say that* $H(n)$ *is* routable *on* $G_W(n)$ *if the capacity constraints and flow balance equations of the related flow routing problem are satisfied, that is:*

$$\begin{cases} \sum_{h\in E_{H(n)}} (f_{ij}^h(n) + f_{ji}^h(n)) \leq c_{ij}(n) \\ \sum_{j\in V_W(n)} f_{ij}^h(n) = \sum_{k\in V_W(n)} f_{ki}^h(n) + b_i^h(n) \\ f_{ij}^h(n) \geq 0, \ \forall i \in V_W(n), (i,j) \in E_W(n), h \in E_H(n) \end{cases} \tag{2.2}$$

Definition 2.2.3 (Residual capacity graph)**.** *We denote with* $G^{TOT}(n)$ *the supply graph (i.e. containing all broken, working and unknown components), with residual capacities, considering all the pruning actions performed until stage* $n-1$*. Such a graph is shortly called the* residual capacity graph*. Notice that* $G^{TOT}(n)$ *can be obtained from* $G_W(n)$ *repairing all broken nodes and links.*

The following notion of feasible pruning establishes the necessary conditions for the feasibility of a pruning action. Informally, if a pruning action reduces the capacity of the current graph to the point that the current demand is no longer routable, even with complete repairs, then it would determine a non-feasible instance of the problem and therefore should be prohibited.

Definition 2.2.4 (Feasible pruning)**.** *Given a demand graph* $H(n)$ *at stage* n*, and the currently working graph* $G_W(n)$ *with updated capacities* $c_{ij}(n)$*, for* $(i,j) \in E_W(n)$*, we say that the* pruning of x units of demand $d_h(n)$ on path p is feasible*, if after the pruning of* x *on* p*,* $H(n+1)$ *is routable on the residual capacity graph* $G^{TOT}(n+1)$*.*

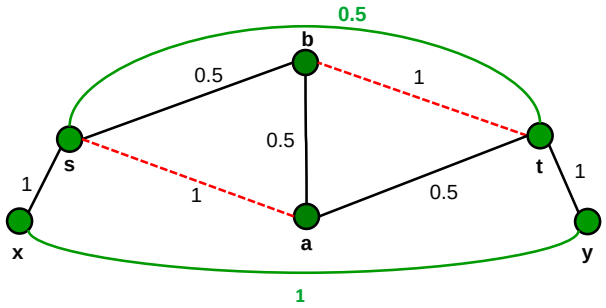


Figure 2.2: Infeasible set of paths for demand flows

Definition 2.2.5 (Infeasible set)**.** *Let $\mathcal{P}$ be a set of paths in the residual capacity graph $G^{TOT}(n)$. $\mathcal{P}$ is an infeasible set for $H(n)$ if for all paths $p \in \mathcal{P}$, and for all the demands $d_h(n)$ in $H(n)$, there is no positive value $\epsilon > 0$ such that pruning of ϵ units of $d_h(n)$ is feasible in p.*

An example of infeasible set is shown in Figure 2.2. In this example, there are two pairs of demand $d_{xy} = \{x, y\}$ and $d_{st} = \{s, t\}$ with a demand of 1 and 0.5 unit of flow respectively, represented with green lines. The black and solid lines represent the working links and the red and dashed lines represents the broken links. The labels on each edge in the graph represent the residual capacity.

The entire demand is routable on $G^{\texttt{TOT}}(n)$, i.e., on the graph of Figure 2.2, after the repair of the links (s, a) and (b, t).

In fact, d_{xy} can be routed, using paths $p^1_{xy} =< x, s, a, b, t, y >$ for 0.5 units of flow, and $p^2_{xy} =< x, s, a, t, y >$ for the remaining 0.5 units of flow, and d_{st} can be routed entirely on the path $p^1_{st} =< s, b, t >$. However, as both the demand pairs have a working path, it may seem intuitive to use it for at least one of them. Nevertheless the current working paths $p^w_{xy} =< x, s, b, a, t, y >$ and $p^w_{st} =< s, b, a, t >$ form an infeasible set. In fact, the pruning of a quantity $\epsilon > 0$ of any of the two demands on its related path, precludes the routability of the remaining demand on $G^{\texttt{TOT}}(n + 1)$, compromising the solution of the problem.

To take care of the partial knowledge, i.e. the nodes and links with unknown status, with the following definition we extend the notion of dynamic path length introduced in the previous chapter, Section 1.2.4.

Definition 2.2.6 (Cost based path length)**.** *Let p be a path in G, that is $\hat{p} \subset V \cup E$. Let $V^p_{B|u}(n)$ and $E^p_{B|u}(n)$ be the sets of nodes and links traversed by p which at the n-th stage are still broken or unknown. We define the* cost

based path length *of p, at the current stage n as follows:*

$$l^{(n)}(p) \triangleq \sum_{(i,j)\in\hat{p}\setminus E^p_{B|u}(n)} \frac{a}{c_{ij}(n)} + \sum_{(i,j)\in E^p_{B|u}(n)} b\cdot\frac{k^e_{ij}}{c_{ij}(n)} + \sum_{i\in V^p_{B|u}(n)} c\cdot k^v_i.$$

For simplicity we consider unitary values of the constants a, b and c, and uniform costs of repair for broken elements $k^e_{ij} = k^e$, and $k^v_i = k^v$ for $(i,j) \in E^p_{B|u}(n)$ and $i \in V^p_{B|u}(n)$, respectively, with $k^e, k^v \gg 1$, for all $(i,j) \in E$ and $i \in V$.

With Definition 2.2.6, the length of a path depends on the number of broken elements, hence varies from stage to stage. Thanks to this dynamic notion of path length, a shortest path selection tends to prioritize paths with fewer broken elements, and links with higher capacities.

We now briefly recall the formulation of demand based centrality, previously introduced in Chapter 1, Section 1.2.2. We use this centrality metric for CeDAR since it consider the problem of flow routing among several nodes.

Definition 2.2.7 (Demand based centrality)**.** *The* demand based centrality $c_d(v)$ *of a node $v \in V$ is defined as:*

$$c_d(v) \triangleq \sum_{(ij)\in E_H} \left(\frac{\sum_{p\in\mathscr{P}^*_{ij}|_v} c(p)}{\sum_{p\in\mathscr{P}^*_{ij}} c(p)} \cdot d_{ij} \right) \tag{2.3}$$

where $\mathscr{P}^*(i,j)$ is the set of the first shortest paths necessary to route the demand (i,j) when considered independently of the other demands, $\mathscr{P}^*_{ij}|_v$ is the set of the paths in $\mathscr{P}^*(i,j)$ traversing v, $c(p)$ is the capacity of path $p \in \mathscr{P}^*(i,j)$, and d_{ij} is the demand flow of the pair $(i,j) \in E_H$.

Notice that, when used by CeDAR the centrality of a node is calculated at each stage to determine how likely the routing of the demand would benefit from the repair of the node. Hence we calculate the value of $c_d(v)$ by considering the instance of the problem at the current stage n. To this purpose, we consider the current demand graph $H^{(n)}$, while the set of paths $\mathscr{P}^*(i,j)$ is calculated in $G^{\mathtt{TOT}(n)}$, and the length of the paths takes account of the current composition of the sets of broken, unknown, and working elements $V_{\mathtt{B}}(n)$, $E_{\mathtt{B}}(n)$, $V_{\mathtt{u}}(n)$, $E_{\mathtt{u}}(n)$ and $V_{\mathtt{W}}(n)$ and $E_{\mathtt{W}}(n)$.

2.2.2 CeDAR in details

In Algorithm 1 we show the details of CeDAR.

We assume that the algorithm has no initial knowledge of the disruption and accumulates information iteratively, through network monitoring.

Initially, in **lines 4-6**, CeDAR repairs the demand endpoints if necessary, and places a software monitor in all of them, to determine their connected working component. CeDAR builds its current view of the working graph $G_{\mathtt{W}}(0)$ with all the nodes and links that were found to be working, and with link capacities as in the original supply graph (before the disruption). If $G_{\mathtt{W}}(0)$ is not sufficient to route all the existing demand flows, CeDAR proceeds with a progressive repair and monitoring of the network, as described in **lines 7-23**.

At stage n of this progressive recovery, in **line 8**, CeDAR computes the set $\mathcal{P}$ that contains, for each demand $d_i \in H(n)$ the corresponding shortest path p_i on $G^{\mathtt{TOT}}(n)$, according to the distance metric given in Definition 2.2.6. Nevertheless $\mathcal{P}$ may constitute an infeasible set for the current demand $H(n)$, according to Definition 2.2.5, tested in **line 9**. In such a case, none of the paths in $\mathcal{P}$ can be used for routing and CeDAR, in **line 10**, resorts to the Equations (2.2) calculated in $G^{\mathtt{TOT}}(n)$, to determine a set of feasible paths $\mathcal{P}_f$.

As $\mathcal{P}_f$ may contain more than one path for each demand pair, in **line 11**, CeDAR builds the new set $\mathcal{P}$ by choosing, for each demand d_i, the shortest path in $\mathcal{P}_f$.

CeDAR only schedules path repairs when the status of all the elements of the path is known. This is meant to *keep unnecessary local interventions at a minimum*. Therefore, in **line 12**, CeDAR looks for the paths $p_i \in \mathcal{P}$, such that $\hat{p_i} \cap (V_u \cup E_u) = \emptyset$ and, with these, it builds the set of paths with known status $\mathcal{P}^k$.

If there is more than one path in $\mathcal{P}^k$ (**line 13**), then in **line 14** CeDAR chooses the path p_i such that $p_i = \arg\max_{p_i \in \mathcal{P}} min_{(k,l) \in p_i} c_{kl}$, namely, the path of maximum capacity, where the capacity of a path is defined as the capacity of the link with minimum capacity. Further ties are addressed by choosing the path of shortest length (not detailed in the pseudocode). CeDAR then schedules the repair of the entire set of broken elements in p_i, which is $\hat{p_i} \cap (V_{\mathtt{B}}(n) \cup E_{\mathtt{B}}(n))$ in **line 15**, and then the pruning, in **line 16**, of the maximum feasible quantity x of $d_i(n)$, on p_i. In **line 17** CeDAR updates the graphs $G_{\mathtt{W}}(n+1), G^{\mathtt{TOT}}(n+1)$ and $H(n+1)$, to keep track of the scheduled repairs and of the updates in the demands and capacities due to the occurred pruning actions.

If all the selected paths of $\mathcal{P}$ contain at least an unknown element (**line 13**), which implies that $\mathcal{P}^k = \emptyset$, in **line 18** CeDAR selects a new node $v_{\mathtt{BC}}$ in which to place a new monitor. To optimize the chance to obtain new information on the area of the network that is of interest for routing the

demand flows, CeDAR selects the node $v_{\mathtt{BC}}$ in the set $V_m(n) \triangleq \{v \in V | v \in V_u(n) \vee \exists w \in V, \text{ s.t. } (v,w) \in E_u(n)\}$ of nodes that are either unknown or have an incident unknown link. Among the nodes of $V_m(n)$, it selects the one with highest demand based centrality: $v_{\mathtt{BC}} = \arg\max_{v \in V_m(n)} c(v)$, according to Definition 2.2.7.

If at the time of the local intervention, the node $v_{\mathtt{BC}}$ is discovered to be broken, it is scheduled for repair in **line 20**, then CeDAR places a monitor in $v_{\mathtt{BC}}$, in **line 21**.

The new repairs and the monitor activity from $v_{\mathtt{BC}}$ require an update of the graphs $G_{\mathtt{W}}(n+1)$ and $G^{\mathtt{TOT}}(n+1)$ and the transition to a new stage.

The algorithm terminates with **line 7** as soon as CeDAR determines that the current demand $H(n)$ is routable over the known working graph $G_{\mathtt{W}}(n)$.

ALGORITHM 1: CeDAR

Input: Supply graph G, demand graph H, broken sets $V_{\mathtt{B}}, E_{\mathtt{B}}$, unknown sets V_u, E_u
Output: Schedule of repairs R

1 Initialize $V_{\mathtt{B}}(0), E_{\mathtt{B}}(0), V_u(0), E_u(0), H(0), R(0)$
2 Build current graphs $G_{\mathtt{W}}(0)$ and $G^{\mathtt{TOT}}(0)$
3 $n \leftarrow 0$
4 **for** $x \in V_H$ **do**
5 If x if broken, append x to $R(n)$ and repair it
6 Monitor from x
7 **while** $H(n)$ *is not routable on* $G_W(n)$ **do**
8 Build the set $\mathcal{P}$ of shortest paths p_i in $G^{\mathtt{TOT}}(n)$, $\forall d_i(n) > 0$
9 **if** $\mathcal{P}$ *is an infeasible set for* $H(n)$ **then**
10 Solve Equations (2.2) in $G^{\mathtt{TOT}}(n)$ to obtain feasible paths $\mathcal{P}_f$
11 Build $\mathcal{P}$ with the shortest path $p_i \in \mathcal{P}_f$, $\forall d_i \in H(n)$
12 $\mathcal{P}^k = \{p_i | p_i \in \mathcal{P} \text{ and } \hat{p_i} \cap (V_u \cup E_u) = \emptyset\}$
13 **if** $\mathcal{P}^k \neq \emptyset$ **then**
14 Choose $p_i = \arg\max_{p_i \in \mathcal{P}^k} min_{(k,l) \in p_i} c_{kl}(n)$
15 Append elements of $\hat{p_i}$ to $R(n)$ and repair them
16 Prune the max feasible x of $d_i(n)$ over p_i in $G(n)$
17 Build the new sets $G_{\mathtt{W}}(n+1), G^{\mathtt{TOT}}(n+1)$ and $H(n+1)$
18 **else**
19 Choose $v_{\mathtt{BC}} = \arg\max_{v \in V_m(n)} c(v)$
20 If $v_{\mathtt{BC}}$ is found broken, append $v_{\mathtt{BC}}$ to $R(n)$ and repair it
21 Deploy a monitor in $v_{\mathtt{BC}}$
22 Build the new graphs $G_{\mathtt{W}}(n+1)$ and $G^{\mathtt{TOT}}(n+1)$
23 $n \leftarrow n+1$

2.3 Properties of CeDAR

As already shown for ISP in the previous chapter, in this section we show the properties of the proposed algorithm. In particular, we focus on the termination, correctness and time complexity of CeDAR.

Theorem 2.3.1 (Termination and correctness of CeDAR)**.** *Let us consider a demand graph $H = (V_H, E_H)$ and a supply graph $G = (V, E)$, which is partially disrupted, such that V_B, E_B are the sets of broken nodes and links, and V_u, E_u are nodes and links of unknown status, and V_W, E_W are the working elements. In a finite number of stages N_{CeDAR}, CeDAR produces a repair schedule R such that the demand H is routable on the* repaired graph *$G^R = (V^R, E^R)$, where $V^R = V_W \cup (R \cap V)$ and $E^R = E_W \cup (R \cap E)$.*

Proof. We first prove that CeDAR terminates in a finite number of stages (termination), then we prove that the demand is routable on the repaired graph (correctness).

Termination. At each stage n, CeDAR selects a set of paths $\mathcal{P}$. If there is at least a path $p \in \mathcal{P}$ such that the status of all the elements of $\hat{p}$ is known, the algorithm enters **lines 14-17**. In this case CeDAR prunes the maximum portion x of a demand d_i on the path p_i, preserving the feasibility of the instance. This requires the solution of an optimization problem with a new variable x. The set of constraints will be the same as in Equations 2.2, on the graph $G^{\texttt{TOT}}(n)$, with the additional equality constraints requiring that the demand d_i be routed for a quantity equal to x on the edges of $\hat{p}$ and for the remaining quantity $d_i - x$ in any other edges, possibly including those of $\hat{p}$. Notice that since the only inequality constraints of this optimization problem are those related to link capacities, every time such optimization is executed, there is a capacity constraint which acts as a *binding constraint* [45]. Given a demand, new pruning decisions will create new binding constraints while previous binding constraints will remain binding. As the number of capacity constraints is equal to the number of links in $G^{\texttt{TOT}}(n)$ it follows that the number of pruning operations for each demand is bounded by $|E^{\texttt{TOT}}(n)|$.

Let us consider instead the case in which none of the paths in $\mathcal{P}$ is completely known, and for each path $p \in \mathcal{P}$ there is at least one unknown status element, so $\forall p \in \mathcal{P}$, $\hat{p} \cap (V_u \cup E_u) \neq \emptyset$. In such a case, the algorithm actions are provided by **lines 18-22**. Every time this happens a new node $v_{\texttt{BC}}$ is selected from $V_m(n)$ which is the set of nodes that are either unknown or have adjacent unknown links. By placing a monitor in $v_{\texttt{BC}}$, according to the assumptions detailed in Section 2.1, we assess the status of (at least) $v_{\texttt{BC}}$ and of all its adjacent links, so the number of elements of $V_m(n)$ gradually de-

creases at each stage. Since this number is lower bounded by 0, the number of monitoring actions is limited by the initial size of V_m.

Correctness. At each stage, CeDAR may either prune a demand, or it may place a monitor and explore its connected component. The first case (**lines 14-17**) CeDAR gradually reduces the total demand preserving the feasibility of the instance, by means of repair and pruning actions, but it requires knowledge of the status of entire paths. In the second case (**lines 18-22**) CeDAR gradually decreases the size of the unknown sets $V_u(n)$ and $E_u(n)$, so it progressively enables more actions of the first kind. Therefore, at each stage new portions of the network are discovered, or a non-infinitesimal demand portion is pruned preserving the feasibility of the problem. As the instance of the problem is feasible by assumption, CeDAR will eventually prune enough demands and repair enough network elements to meet the routability of the demand on the currently repaired graph $G^R = (V^R, E^R)$, where $V^R = V_{\mathtt{W}}(n) \cup (R(n) \cap V)$ and $E^R = E_{\mathtt{W}}(n) \cup (R(n) \cap E)$. □

Theorem 2.3.2 (Time Complexity of CeDAR)**.** *Let G=(V,E) be the graph supply and let H be a feasible demand graph on G. CeDAR has polynomial time complexity.*

Proof. As we discussed in the proof of Theorem 2.3.1, the number of iterations is bounded by the maximum number of demand pruning actions, which is limited by the number of edges in the supply graph, for each demand pair, therefore the number of iterations is $O(|E_{\mathtt{H}}| \times |E|)$. By focusing on the single activities performed at each iteration we have the following analysis.
Determine the routability of current instance (**line 7**). It requires testing the feasibility of the set Equations (2.2). This is known to be polynomial, as detailed in [66, 45].
Determine set of shortest paths $\mathcal{P}$ (**line 8**). It requires the execution of the Dijkstra's algorithm for each demand $d \in E_H$. Therefore it requires $O(|E_H| \cdot (|E| + |V| \log(|V|))$.
Determine if $\mathcal{P}$ *is a conflicting set* (**line 9**). In the worst case, it requires the solution of a linear programming problem (of gradually lower size), to determine whether a non null demand quantity x can be pruned on each of the paths in $\mathcal{P}$ (one for each demand).
Complexity of Pruning **(line 16)**. While the maximum amount of demand x that can be pruned for each demand pair is also calculated in the test to determine whether $\mathcal{P}$ is a conflicting set, there is no need to recalculate it. x units of demand d on the shortest path p and update of the supply and demand graph require linear time in the length of p.

Complexity to find the best candidate (**line 18**). The demand based centrality of each node is determined with the Equation 2.3. In order to calculate the set of paths $\hat{\mathscr{P}}^*_{ij}$, at each iteration we use Dijkstra's algorithm to find the shortest path p between any pair of nodes u and v such that $(u, v) \in E_{\mathtt{H}}$. Let $c(p)$ be the capacity of such a path. If $c(p) \geq d_{u,v}$, where $d_{u,v}$ is the demand requirement between nodes u and v, this path is sufficient, otherwise we consider the residual graph in which we reduce the capacity of p by $c(p)$, and we calculate the next shortest path at the next iteration to satisfy a demand $d_{u,v} - c(p)$, if The resulting complexity is $O(|E_H^{(n)}| \times (|E^{(n)}| + |V^{(n)}| \log(|V^{(n)}|))$, since at each iteration we compute the shortest path using Dijkstra's algorithm and we saturate the capacity of at least one edge. For each selected shortest path, we can update the centrality of its nodes in linear time with respect to the path length. Finding the node with highest centrality requires linear time $O(|V|)$.

In conclusion, as all the actions of each iteration can be performed in polynomial time, and the number of iterations is always polynomial, we can conclude that CeDAR has polynomial time complexity. □

2.4 Heuristics

To the best of our knowledge there is no previous work in the literature that addresses the problem of recovery in the case of incomplete knowledge of the failure extent. To compare the performance of CeDAR, we modified two previous approaches. In fact, since both of them assume perfect knowledge of the disruption, it would be unfair to compare them to CeDAR in a setting with incomplete information, for which CeDAR is specifically designed. For this reason, we modify these approaches to make them able to determine a progressive recovery schedule, where network monitoring is performed in parallel to repairs, and the recovery plan can be progressively adjusted.

2.4.1 Shadow Price Progressive Recovery (ShP)

The work of Wang et al. [86] introduces a progressive recovery approach, which we hereby call the *Shadow Price* (ShP) approach. ShP assumes complete knowledge of the failure which can only affect links and not nodes, and considers limited resource availability to perform simultaneous repairs in a massively disrupted network. The purpose of ShP is to schedule the repairs of the broken network components so as to optimize the weighted sum over time of the flow of every demand pairs. The ShP approach considers the

progressive recovery problem as an MILP problem. By recognizing the NP-hardness of the approach, the authors suggest to use an LP relaxation of the problem and suggest to schedule link repairs according to a decreasing order of the shadow prices of the link capacity constraints.

To make the comparison with CeDAR more fair, we modified ShP as follows. First, as ShP cannot work with broken nodes, we let it assume that all nodes are working, and whenever it selects an edge for repair, its endpoint nodes are also repaired if broken, and a monitor is placed on one of them. Second, we consider a progressive execution of ShP, in which ShP is executed iteratively as a single stage process of repair, and a monitoring activity is performed from the newly repaired nodes at each iteration. Finally, we observe that since ShP aims at maximizing flow, and not at meeting specific flow requirements, it may find solutions in which a large flow of one demand compensates for an insufficient flow of another. We modified the LP problem used by ShP, to include an upper bound on each demand flow equal to its requirement, and stop the algorithm execution as soon as all demand requirements are satisfied. With these three modifications we allow ShP to work also under incomplete knowledge of the failed area and make it more appropriate to meet specific demand requirements.
Why do we expect ShP to perform poorly in terms of accumulative flow over time?

Notice that ShP requires that broken edges have a small residual capacity, to avoid scenarios where all shadow prices are null. This is not realistic, as broken links have null capacity, but is a requirement for the algorithm to work. The values of these residual capacities influence the schedule of repairs. As a consequence ShP does not perform the repair of the components of a same path in an interrupted sequence, which is critical to have high accumulative flow. In the experiments of Section 2.5 we set the residual capacities of broken links to random values as suggested by the authors [86].

2.4.2 Progressive ISP (P-ISP)

In Chapter 1, we proposed a polynomial heuristic called ISP, to solve the problem MinR a simpler instance of PDAR, introduced in Section 1.1, that aims at minimize the cost of repair, while restoring critical demand flows.

However, as discussed, ISP is not designed to work under partial knowledge of the disruption, and does not provide an adaptive schedule of repairs that adjust repair decisions to the incremental knowledge of the network, as required by the PDAR problem. To compare the performance of CeDAR with ISP, we modified the ISP algorithm to work progressively with incremental knowledge of the disruption. We briefly recall how ISP works to better

understand the changes made. For a detailed description of ISP, please refer to Chapter1, Section 1.2.
ISP works by iteratively selecting the next node to repair, called *best candidate*, according to a centrality ranking on the basis of the notion of centrality given in Section 1.2.2. After the repair of this node, ISP selects a demand to *split*, thus creating two smaller demands with the best candidate as a new end-point for both. The algorithm also provides a *pruning* operation, which is similar to the one performed by CeDAR, but works under different enabling conditions based on structural properties of the demand and supply graph (for details see Section 1.2.6).

To make ISP able to work in the case of partial knowledge of the disruption, we designed a progressive variant, hereby called *Progressive ISP (P-ISP)* as follows. First, we assume that network elements of unknown status are broken, and let P-ISP consider them as high cost repair elements. Second, we consider a progressive execution in stages, where at every stage P-ISP executes both repair interventions, monitor deployment, and network probing, according to a stage model similar to the one of Figure 2.1.

We now give more details of this modified variant. When the node s with highest centrality is chosen for repair and demand split, a monitor is first placed on s which probes the nodes of its connected component. Before performing the actual demand split as provided by ISP, P-ISP recalculates the centrality of the nodes on the basis of the new information available. If s is still the best candidate, the split action goes on. Otherwise, a new intervention of repair and monitoring is planned for the new node with highest centrality. This process is iteratively repeated until the new information acquired by monitoring does not determine a new highest ranked node. The pseudo-code of P-ISP is shown in the following Algorithm 2.

Why does ISP perform wrong irreversible decisions?

Although we modified ISP to make it work in the context of incomplete knowledge of the failure, we expect that ISP performs worse than CeDAR. In fact, P-ISP could still make wrong decisions due to the assumption that components with unknown status are broken. This may cause P-ISP to split a demand on a best candidate node which may result an inefficient choice when more knowledge becomes available. Moreover, the split action determines an irreversible routing decision that may compromise the entire solution of the problem. In addition to this, ISP performs repairs one at a time, potentially scheduling successive repairs in distant and unrelated portions of the network, resulting in a low accumulative flow over time. An example of wrong choice due to the partial knowledge of the disruption is shown in Figure 2.3. Figure 2.3 (a) shows a network, in which there is a demand flow between the green nodes s and t, the broken elements are represented in red and with dashed

ALGORITHM 2: Progressive Iterative Split and Prune (P-ISP)

Input: Supply graph G, demand graph H, broken nodes $V_{\mathtt{B}}$ and broken edges $E_{\mathtt{B}}$
Output: Schedule of repairs

for $x \in V_H$ **do**
    Repair x if broken
    Monitor from x
Build current sets $V_{\mathtt{B}}(0), E_{\mathtt{B}}(0), V_u(0), E_u(0)$
Build current graphs $H(0), G_{\mathtt{W}}(0)$ and $G^{\mathtt{TOT}}(0)$
while $H(n)$ *is not routable on* $G_{\mathtt{W}}(n)$ **do**
    while *pruning condition* **do**
        Prune demands satisfying pruning condition on $G(n)$;
        Build the new sets $G_{\mathtt{W}}(n+1), G^{\mathtt{TOT}}(n+1)$ and $H(n+1)$;
    if *there are repairable links* **then**
        Plan interventions for the broken repairable links on $G(n)$;
        Build the new sets $G_{\mathtt{W}}(n+1), G^{\mathtt{TOT}}(n+1)$;
    else
        while *best candidate* v_{BC} *changes* **do**
            Plan an intervention on $v_{\mathtt{BC}}$ on $G(n)$;
            If $v_{\mathtt{BC}} \in V_{\mathtt{B}}(n)$ repair and place a monitor;
            Discover the network from $v_{\mathtt{BC}}$ and update the centrality;
        Find best demand d to split on $v_{\mathtt{BC}}$;
        Calculate the maximum splittable amount d_x;
        Split amount d_x of demand d on $v_{\mathtt{BC}}$;
        Build the new sets $G_{\mathtt{W}}(n+1), G^{\mathtt{TOT}}(n+1)$ and $H(n+1)$;

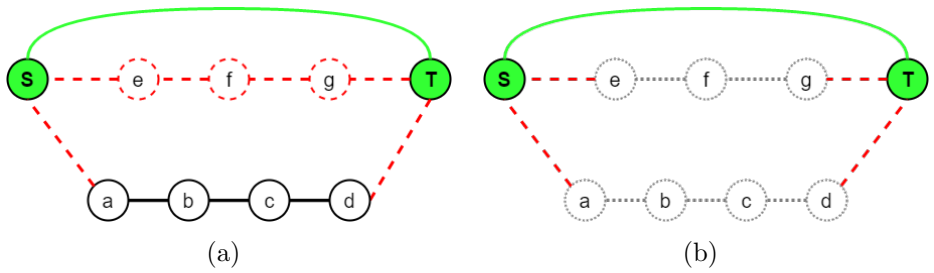


Figure 2.3: Progressive ISP: perfect knowledge (a) - split error due to partial knowledge (b)

lines, and the working elements are black and with solid lines. In a scenario with full knowledge, P-ISP would choose to repair the nodes along the path $p_1 =< s, a, b, c, d, t >$ to minimize the number of repairs. , since it is almost repaired if compared to the path $p_2 =< s, e, f, g, t >$ that is completely destroyed. Hence with this information, P-ISP will perform the split actions on the nodes on path p_1, giving them a higher centrality than the nodes of p_2. However, when the information on the disruption is not available, P-ISP treats the nodes on p_1 and $p_2 =< s, e, f, g, t >$ equally (the same), as shown in Figure 2.3 (b) where gray elements have unknown status. In such a scenario, P-ISP chooses the path p_2 to be repaired, since it potentially requires a lower number of repairs with respect to p_1. The side effect shown in 2.3, due to the partial knowledge, forces P-ISP to perform a higher number of interventions to find a stable best candidate, whose centrality does not change after the monitoring phase.

Why we do not expect P-ISP to show a good accumulative flow over time?

P-ISP repairs links only when there is a demand between their endpoints. In this way P-ISP may repair the network in a way that successive repairs occur in possibly distant and unrelated portions of the network. Although P-ISP eventually repairs all necessary paths to route the entire set of demands, it does not work on one path at a time like CeDAR, but schedule node repairs on the basis of a centrality rank. For this reason, when performed progressively, P-ISP does not perform well in terms of accumulative flow over time.

This result is confirmed in all experiments of Section 2.5, where P-ISP requires high number of interventions for discovering the network.

2.5 Experiments

In this section we study the behavior of the discussed approaches by means of simulations. We consider a real network topology, taken from the CAIDA (Center for Applied Internet Data Analysis) dataset [23]. This dataset includes real topologies describing the connections between backbone/gateway routers of several autonomous systems. We used the topology AS28717, of which we extracted the giant connected component with 825 nodes and 1018 edges, where we set the edge capacities randomly in a range between 20 and 50 units.

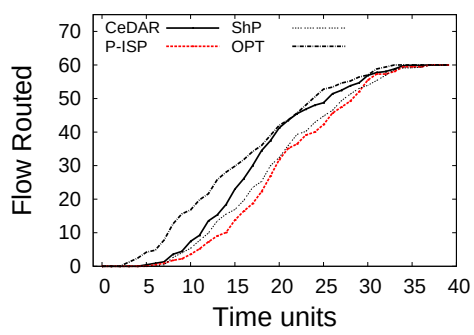
In the following experiments we considered three different scenarios in which we varied the number of demands, the amount of flow for each demand, and the extent of the disruption, randomizing the results for a minimum of 20 runs for each experiment.

In all the experiments, with the only exception being the optimal (OPT) solution, we assume that the initial knowledge of the network state is only partial, and determined by monitoring the network from the demand endpoints. The OPT solution instead is obtained by using complete knowledge of the disruption and solving the NP-hard optimization problem PDAR of Section 2.1.1. Therefore, we underline that OPT is an ideal solution and is considered only as a baseline for comparisons, to evidence the margin of improvement that any algorithm can provide with respect to existing solutions. For this reason we show the comparisons with OPT only in the first scenario.

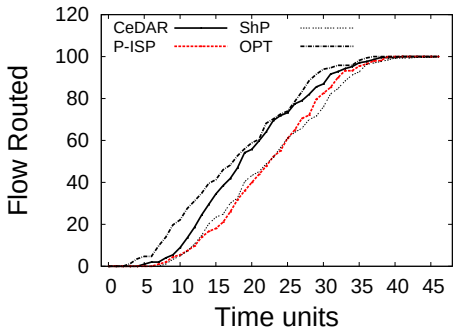
2.5.1 Scenario A: Varying demand intensity

In this scenario we increase the load on the network by varying the amount of flow of 5 uniform demand pairs, with randomly selected endpoints. We generate the network disruption so as to form multiple disconnected portions. To this purpose we generate a geographic distribution of the probability of failure, in the form of a composition of two bi-variate Gaussian distributions, representing two epicenters of maximum disruption probability. The disruption probability gradually decreases with the distance from the epicenters. The extent of the disruption is such that 60% of the network components are broken.

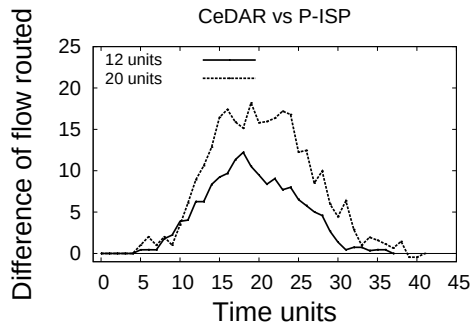
In Figure 2.4 we show the effects of the progressive recovery actions of the three algorithms CeDAR, P-ISP and ShP and of the optimal solution OPT. The figure shows the trend with time of the maximum amount of critical flow that can be routed on the currently repaired supply network. In the figure, the number of repairs grows in proportion with time as we consider that all



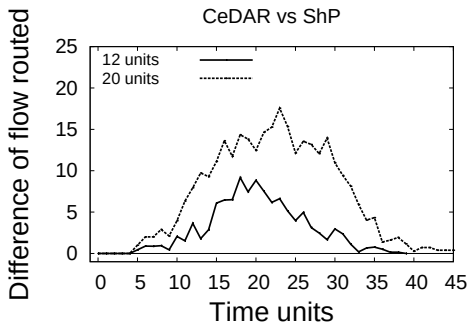
(a) 5 pairs of 12 units of flow



(b) 5 pairs of 20 units of flow



(c) Difference CeDAR – P-ISP



(d) Difference CeDAR – ShP

Figure 2.4: Scenario A. 5 demand pairs with varying demand intensity. Flow routed, 12 (a) and 20 (b) flow units per pair. Flow difference: CeDAR vs. P-ISP (c), CeDAR vs. ShP (d)

the algorithms repair one network element at each time step, to mimic a scenario with limited resources.

We consider two different load settings: a case with moderate flow in Figure 2.4(a) corresponding to 12 flow units for each of the 5 demand pairs, and a case with high flow in Figure 2.4(b), corresponding to 5 demand pairs of 20 flow units each. The figure shows that CeDAR outperforms P-ISP and ShP by routing more flow at each time step, with peaks of about 18 flow units of difference, corresponding to the 30% of the total demand in the case of moderate flow, and to the 18% in the case of high flow. Compared with OPT, CeDAR shows a good approximation in the initial phase, with a difference between the two within the 15% of the total demand, that gradually becomes even lower, until the recovery process is halfway, when the difference between CeDAR and OPT becomes negligible.

Figures 2.4(c) and 2.4(d) emphasize the difference between CeDAR and the other two algorithms by showing how much more flow CeDAR routes in both the considered load settings. For instance, in the case of high load, corresponding to the dashed lines of Figures 2.4(c) and 2.4(d), after about 20 rounds of repairs CeDAR routes an amount of flow that is 20 units higher than P-ISP (see Figure 2.4(c)), and 15 units higher than ShP (see Figure 2.4(d)). It is important to notice that CeDAR does not incur a higher cost of repair than the other two algorithms. In the entire execution period, CeDAR routes more flow than ShP, despite the fact that ShP targets cumulative flow as main objective function. We also recall that, while ShP optimizes the total flow without guaranteeing any fairness among flows,to the point that the optimal flow could be related to one only demand pair, CeDAR aims at guaranteeing the satisfaction of each demand flow requirement. Figure 2.5 considers an experiment where we increased the amount of flow of each of the 5 demand pairs from 4 to 24 flow units. Figure 2.5(a) shows the number of repairs needed to route the entire flow demands. With respect to the number of repairs, CeDAR outperforms ShP and performs the same as P-ISP which instead is specifically meant to optimize repair cost. Notice that the number of repairs performed by CeDAR and P-ISP is close to the optimal OPT, which assumes full knowledge of the status of the network nodes and links. By contrast, ShP needs to repair more network elements than the other algorithms to route the same amount of flow. This is due to the fact that ShP aims at optimizing cumulative flow at each iteration, so it may decide to sacrifice cost, by repairing more elements than strictly necessary for the purpose of satisfying the demand requirements.

Figure 2.5(b) shows that CeDAR deploys a lower number of monitors than ShP and P-ISP. This means that CeDAR is able to perform the necessary monitoring activity with a lower number of monitors, thanks to more focused monitor deployment decisions that aim at obtaining information on portions of the network that are more relevant to the demand requirements.

Figures 2.5(c) and (d) show the number of edges and nodes repaired by the three algorithms. Notice that all the algorithms place a monitor on repaired nodes to obtain maximum benefit from the local interventions. Nevertheless, the number of monitors placed (Figure 2.5(b)) is higher than the number of repaired nodes (Figure 2.5(d)) because the algorithms may need to place monitors on nodes in the unknown area, and may discover that the selected nodes are working properly only after the local inspection.

Finally, we underline that all the three algorithms terminate with no demand loss since the termination condition is the routability test of Equation 2.2 for all of them.

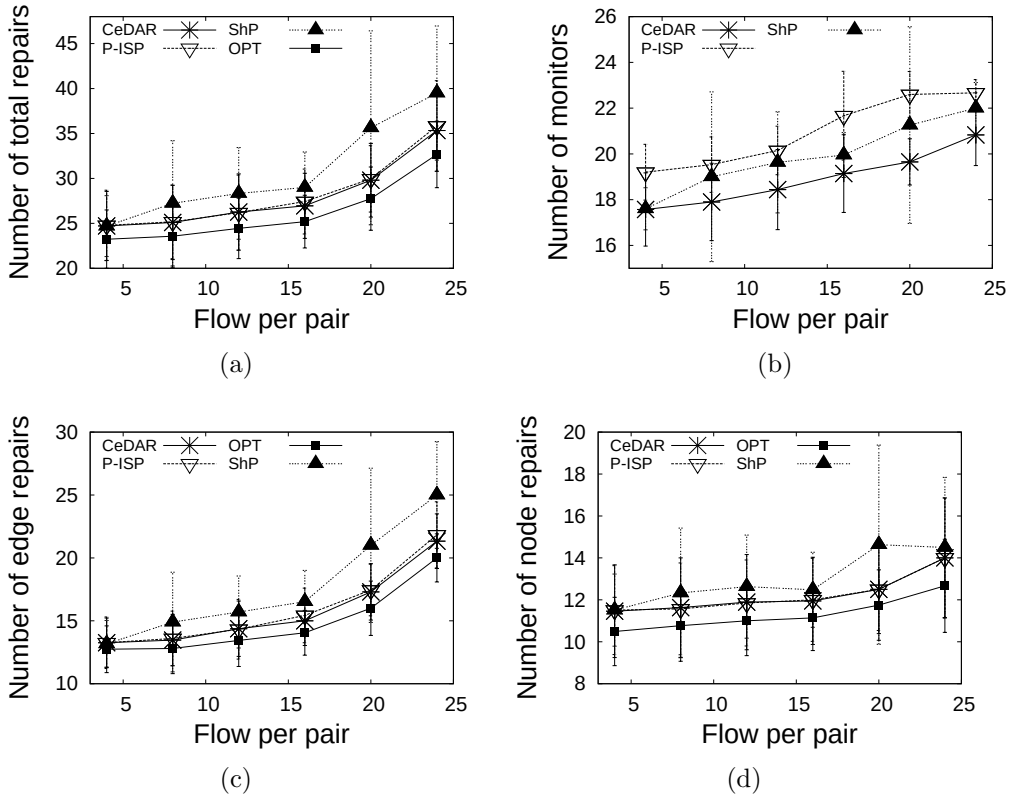
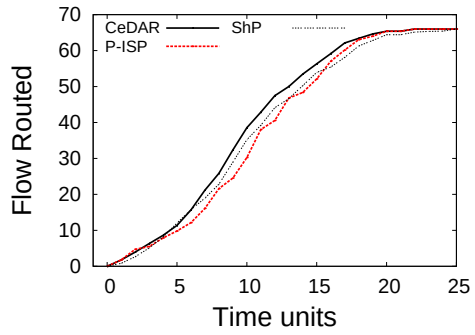


Figure 2.5: Scenario A. 5 demand pairs, with varying demand intensity: total repairs (a), monitors (b). Number of edges (c) and nodes (d) repaired.

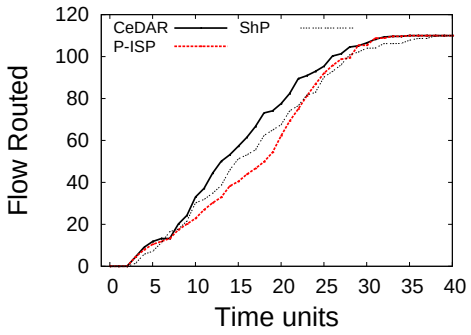
2.5.2 Scenario B: Varying number of demand pairs

In this set of experiments we considered the effect of the demand load by varying the number of critical demand flows in the range from 1 to 6. We consider critical demands of 22 flow units. The setting of this value is determined by the need to have a complex scenario where demands must be routed across multiple paths and potentially generate conflicts with each other in the competition for shared links. Also in this scenario, the disruption is generated according to the composition of two bi-variate Gaussian distributions so that 40% of the network components are broken.

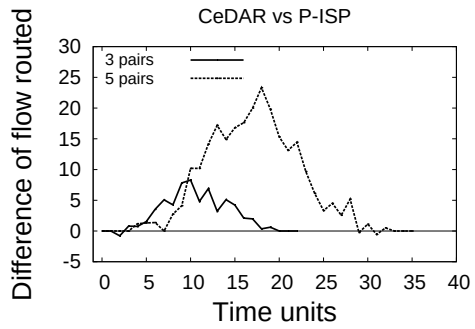
In Figures 2.6(a) and (b) we considered the case of 3 and 5 demand pairs, respectively and show the increase in the amount of routed flow as long as the network is progressively restored. Figure 2.6 shows instead the difference between the flow routed by CeDAR with respect to P-ISP (Figure 2.6(c)) and ShP (Figure 2.6(d)). For instance, with 5 pairs of demand, after about 18 rounds of repairs, CeDAR routes about 25 more units of flow than P-ISP,



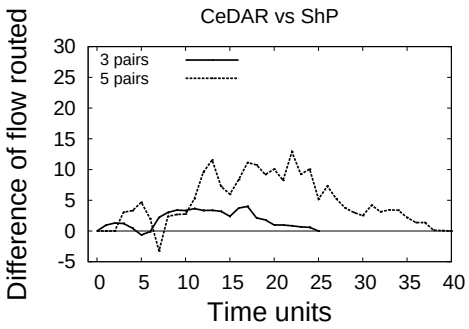
(a) 3 pairs of 22 units of flow



(b) 5 pairs of 22 units of flow



(c) Difference CeDAR – P-ISP

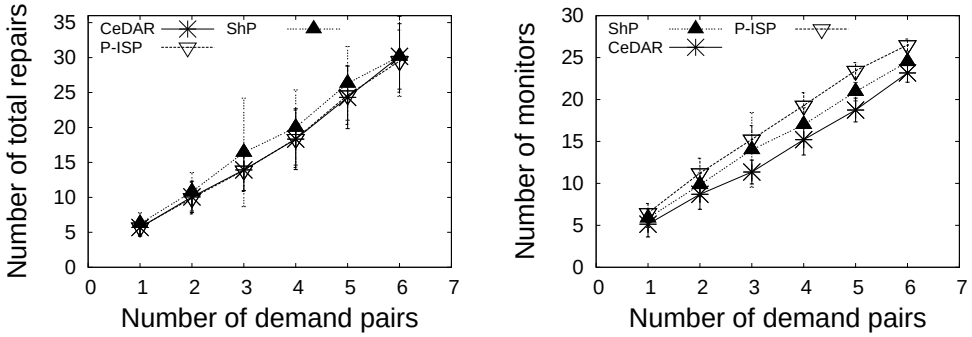


(d) Difference CeDAR – ShP

Figure 2.6: Scenario B: Varying demand pairs. Routed flow for 3 (a) and 5 (b) demand pairs. Flow difference: CeDAR vs. P-ISP (c), CeDAR vs. ShP (d)

corresponding to the 23% of the total demand, and about 11 more units than ShP, which is the 10% of the total demand.

We recall that P-ISP has the explicit objective to minimize the cost of repairs, and it does not address the objective of cumulative flow over time. CeDAR outperforms ShP with regard to the cumulative flow despite the fact that this metric it the objective of ShP. With the last two figures, by varying the number of demand pairs from 1 to 6, we show that CeDAR routes the entire demand with a similar number of repairs as P-ISP, lower than ShP, as shown in Figure 2.7(a) and with a lower number of monitors, as shown in Figure 2.7(b). We conclude that in this scenario, with a number of repairs close to those of P-ISP and lower than ShP, CeDAR achieves a higher cumulative flow than both the other algorithms.



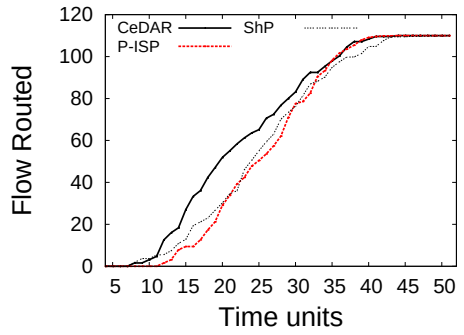
(a) (b)

Figure 2.7: Scenario B. Varying demand pairs: repairs (a), monitors (b)

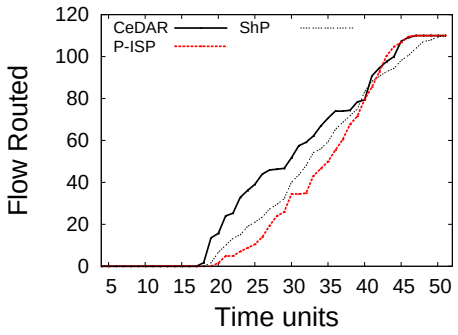
2.5.3 Scenario C: Varying disruption extent

In this last scenario, we investigate the behavior of the three algorithms by varying the extent of the disruption. Similar to the previous experiments we consider two epicenters of bi-variate Gaussian failure distribution. It must be noted that when the network disruption is sparse, the monitoring activity is particularly beneficial. Indeed by placing monitors on nodes of the unknown area of the network, it is likely that these can discover large connected working components of the network, and increase the speed of the recovery process. By contrast, if the extent of the disruption is very large, the monitoring activity is less effective as it is more likely that nodes in the unknown area have broken adjacent links and therefore cannot send monitoring probes to perform the exploration of the surrounding network.

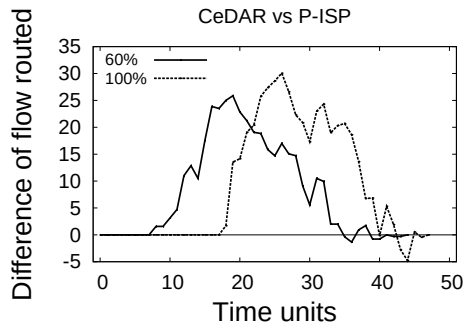
We first consider two different settings, with moderate and complete disruption. The extent is such that 60% of the network elements are broken in the first case, and 100% in the second. We consider 5 demand pairs with a demand of 22 flow units each. In Figure 2.8(a) and Figure 2.8(b) is shown the flow routed by all the algorithm considered. Even in this scenario, CeDAR always routes the demand flows faster than P-ISP and ShP. The difference between the amount of flow routed by CeDAR and the other algorithms is particularly remarkable in both the considered scenarios. Figure 2.8(c) shows the difference between the total flow routed by CeDAR and P-ISP. In the case of complete disruption (dashed line), after 28 time units, the flow routed by CeDAR is 30 units higher than with P-ISP, corresponding to about 27% of the total demand. Similar to these results, Figure 2.8(d) shows that in the case of complete disruption, after about 20 time units, CeDAR routes 23 more units of flow than P-ISP, corresponding to about the 21% of the total demand.



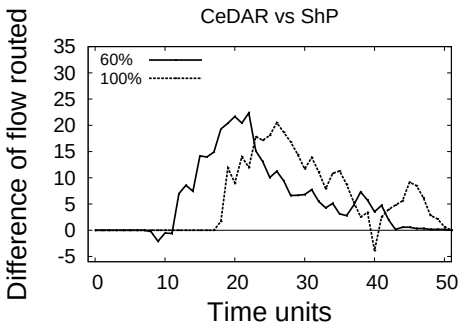
(a) Disruption of 60%



(b) Disruption of 100%



(c) Difference CeDAR – P-ISP



(d) Difference CeDAR – ShP

Figure 2.8: Scenario C: Varying disruption extent. Flow routed for 60% (a) and 100% (b) of disruption. Flow difference: CeDAR vs. P-ISP (a), CeDAR vs. ShP (b)

The figure shows analogous results for the case of moderate disruption (solid line), where the accumulative flow of CeDAR is even higher as it allows to route flow sooner, i.e., after about 7 time units, compared to the case of complete disruption (dashed line), in which it requires 18 time units before we start seeing a positive flow routed. It is evident that in the case of moderate disruption there is more room for prioritizing the repair of the network elements that can ensure a higher value of the cumulative flow over time, while in the case of large disruption it is more likely that the first recovery interventions will not be sufficient to accommodate any demand flow nor to create enough working paths for monitoring.

The study of this scenario confirms the results discussed for Scenario A and B. The higher cumulative flow routed by CeDAR is obtained through a better scheduling of repairs and monitor placement. A more detailed study, conducted by varying the disruption from 40% to 100% evidences

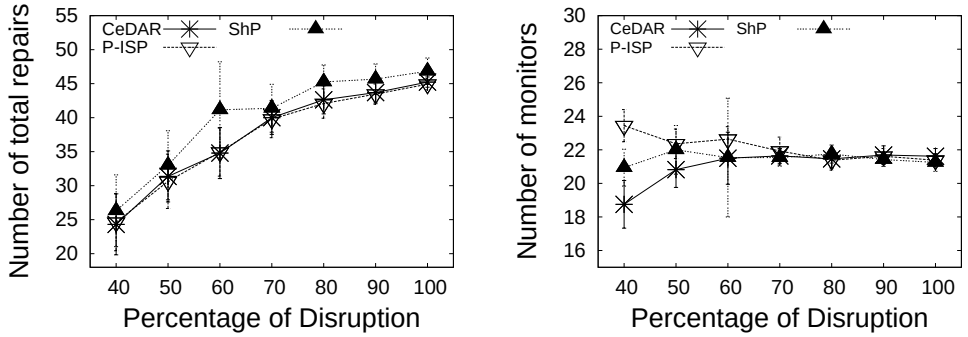
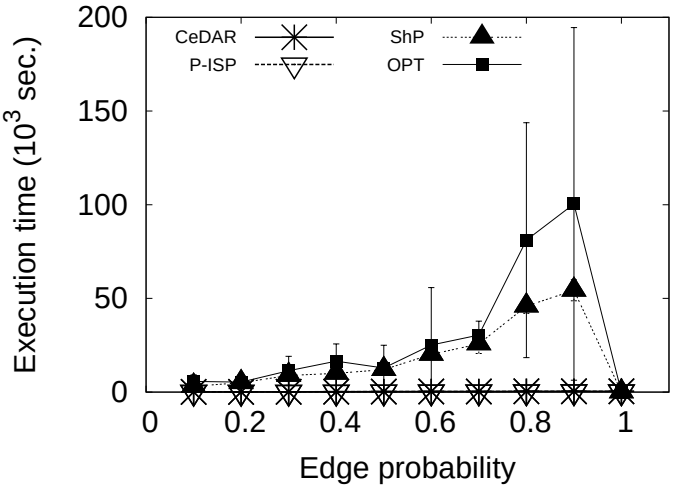


Figure 2.9: Scen. C. Varying disruption: repairs (a), monitors (b)

that CeDAR performs a number of repairs close to P-ISP and lower than ShP, as shown in Figure 2.9(a) while Figure 2.9(b) shows that CeDAR also requires a lower number of monitors. Notice that in the case of large disruption the number of monitors coincides with the number of repaired nodes as most of the nodes selected to host monitors in the unknown area, are found to be broken. Also in this case, we recall that all algorithms terminate according to the routability test of Equation 2.2, and therefore ensure complete satisfaction of the demand requirements.

2.5.4 Scenario D: Execution time comparison

In this last section, we compare the approaches in terms of execution time. We performed experiments in which we varied the problem size and we tested the algorithms on different topologies. We noticed that the average node degree affects the computation time of the algorithms more than other aspects. We generated an Erdos-Renyi random graph with 100 nodes, and we progressively increased the average node degree to study how it affects the performance of the algorithms. Since in this set of experiments, we are considering the execution time more than the demand flow routed, we used low demand flows and large links capacities (only connectivity requirement). The experimental results on the execution time are shown in Figure 2.10. When the average node degree is about 30, i.e. p=0.3 in the picture, the optimal solution OPT of the PDAR problem requires more than 3 hours of execution time, while ShP takes about 2.5 hours, P-ISP and CeDAR less than 5 minutes. For a degree of 80, i.e. p=0.8 in the picture, the computation time increases significantly to about 22.5 hour for the optimal, about 12.5 hours for ShP and less than 10 minutes for P-ISP and CeDAR. The dif-



(a)

Figure 2.10: Erdos-Renyi topology. Varying edge probability p. Execution time (a).

ference between the optimal and the heuristics becomes much more evident when the algorithms need to be executed for larger networks. These results show the excellent scalability of our algorithm CeDAR with respect to previous approaches. Notice that, even the progressive version of our ISP has a similar behavior in terms of execution time (as confirmed by Section 1.5.3). However , as discussed in the previous scenarios, P-ISP doesn't achieve good performance in the scheduling of repairs to make the demand flows available faster.

Conclusions

In this first part of the thesis, we considered, for the first time, the problem of how to efficiently restore critical services in a communication network after large scale failures. In the first chapter, we assumed to have a complete knowledge of the disruption and we studied the problem of how to recover the critical services with minimum cost of repairs. We modeled this problem, named Minimum Recovery (MinR), as a Mixed Integer Linear Programming (MILP) problem, and show it is NP-hard. We proposed ISP, an efficient heuristic to solve MinR, based on a novel demand based centrality metric. ISP makes use of this metric to iteratively select the best nodes for repair, and concentrate the flow on them by means of split actions. It additionally prunes demand flows if they can be satisfied by the currently repaired supply network. We also proposed several greedy heuristics. Experimental results on real and synthetic topologies show that ISP outperforms other approaches in number of repairs and in execution time. In particular, it achieves a number of repairs close to the optimum without incurring any demand loss.

In the second chapter, we removed any assumption on the knowledge of the disruption. In this new scenario, we studied for the first time, the problem of progressive recovery a communication network after large scale failure under incomplete knowledge of the damage extent. We model the problem of Progressive Damage Assessment and network Recovery (PDAR), which is shown to be NP-hard. We proposed CeDAR, an efficient heuristic to solve PDAR, that performs joint repair and monitor interventions, to progressively restore critical services. We compared CeDAR with previous approaches modified to deal with incomplete knowledge. Experimental results on real topologies show that CeDAR outperforms the previous approaches with a significantly higher accumulative flow over time and comparable repair cost. Furthermore, CeDAR requires an execution time in the order of few minutes even on giant network, while the optimal solution to PDAR could require up to 25 hours to be solved.

In our future work, we will consider other aspects:

- *different technologies*: such as wired and wireless links, to achieve more efficient recovery strategies for the given disaster scenario;
- *network tomography*: we will incorporate network tomography techniques to infer the damaged area.
- *failure propagation*: we will consider the temporal failure propagation even during the recovery phase.

Part II

Mobile Wireless Sensor Networks

Introduction

In recent years, technological developments have made possible the design and realization of *mobile sensors*, i.e. devices with sensing, communication, computation and locomotion capabilities. Thanks to these capabilities, the sensors are able to accomplish complex tasks, such as cooperative monitoring, autonomous deployment, and dynamic reorganization of the network if certain events occur. To perform their actions, mobile sensors are endowed with several different *sensing* units that allow them to monitor a wide range of phenomena such as temperature, humidity, geographical location, pressure, solar energy, wind power, presence/absence of certain kinds of objects, etc. Furthermore, mobile sensors are able to coordinate with each other and coordinates their movements to self-deploy over an Area of Interest (AoI). Differently from static sensor networks, mobile wireless sensor networks are suitable to monitor inaccessible areas, unknown or hostile to man, such as areas in which there are gas leaks, the presence of radioactivity, rescue operations, forest fire detection etc. In fact, the inaccessible and often hazardous scenarios typical of mobile sensor networks applications, impedes the manual sensors positioning [80]. For these reasons, often these sensors are airborne or released in a safe area in proximity of the AoI. From this initial configuration, which does not meet target requirements, mobile sensors can use their locomotion capabilities to achieve the desired deployment. Therefore there is the need of distributed algorithms for the coordination of movements.

The problem of the deployment of mobile sensors was already studied in literature. The proposed approaches can be classified as follows:

- approaches based on *Virtual Forces*, are inspired by the molecular interactions of particles. Each sensor exerts a virtual force on the others, that can be attractive or repulsive, depending on the distance. The resulting force on the sensor determines its movement;
- approaches based on *Computational Geometry*, use geometrical structures, such as *Voronoi diagrams* and *Delaunay triangulations*, in order to guide the sensors movement;
- approaches based on *Geometric Pattern*, aim at deploying sensors according to a pre-defined pattern. Each sensor moves from its initial position to some key position in a pattern.

Usually, the mobile sensors are not equipped with tamper-resistant hardware (tamper-proof) and are hence subject to manipulations by an attacker.

Indeed, it is possible for an attacker to compromise (via hardware or software) a number of sensors and use them to performs attacks on the network. Although in the literature there are several works on the safety of mobile sensor networks, it has not been paid attention on the security issues of the deployment of mobile sensor networks. Compromising some sensors of the network, the attacker could deploy them over the AoI to prevent the natural deployment of not compromised (legitimate sensors), precluding the objectives of coverage. Only recently, in [9], the authors study for the first time these vulnerabilities for approaches based on Virtual Forces introducing a new type of attack, called *Opportunist Movement (OM attack)*, which aims to undermine the deployment of mobile sensor networks. The authors show how such an attack is able to limit the coverage of the network and propose a solution for the safe deployment of the sensors.

In this part, we analyze the vulnerabilities of the deployment of mobile sensor networks for approaches based on Voronoi diagrams in the presence of an attacker performing an OM attack. In a previous work [10], the authors only show an experimental result on the vulnerabilities of Voronoi deployment approach again the OM attack. They also proposed a new deployment algorithm based on Voronoi diagrams, called SecureVor, to counterattack the OM attack. Starting from this experimental result, we significantly extend the previous work in [10]. In particular, we give a novel geometric characterization and a formal proof of the efficacy of the OM attack [9] against this deployment approach, showing that the attack can seriously compromise coverage. We show that *during* the deployment of the network the OM attack is more effective against Voronoi based solutions than against the virtual force approach. In fact, with Voronoi based solutions, the efficacy of the attack depends only on the perimeter of the area that the attacker wants to keep uncovered, and there is no gain in increasing the number of legitimate sensors deployed as occurs for Virtual Forces based solutions.

On the basis of the geometric characterization described above, we propose a new algorithm *Secure Swap Deployment (SSD)*, which is designed to counteract the OM attack. SecureVor [10] works under the assumption that the transmission radius is at least four times larger than the sensing radius. Under this operative setting, which is common to most outdoor application scenarios, a sensor can determine the legitimacy of its neighbors movements and communications.
Instead, SSD is designed to work in the same operative setting as the original *VOR* algorithm presented in [85], i.e., $R_{\mathtt{tx}} > 2R_{\mathtt{s}}$, so that it is complementary to SecureVor. SSD exploits sensor positions swaps to verify the neighbors behavior.

We show that both algorithms can defeat the OM attack in their respective operative settings, and we formally prove that both terminate in a finite time.

We perform extensive simulations to study the performance of SecureVor and SSD in comparison with the original solution VOR. The results show that both algorithms are able to successfully neutralize the OM attack and achieve coverage of the AoI at the expense of a small overhead in terms of energy consumption and deployment time. SecureVor is more effective when the transmission radius is sufficiently large with respect to the sensing radius, while SSD is preferable when such an assumption does not hold.

The original contributions of this part are:

- For the first time, we point out and formally prove the vulnerabilities of Voronoi-based deployment algorithms, giving a geometric characterization of possible attack configurations.
- We propose a new secure deployment algorithms called SSD, which successfully counteract the OM attack, in different and complementary operative settings of SecureVor [10].
- We show that both algorithms have a guaranteed termination, show through simulation that both defeat the OM attack.
- Through simulations, we highlight the efficacy of the algorithms in providing full coverage, even in the presence of an OM attack, under a wide range of operative conditions, at the expense of a moderate increase in energy consumption and deployment time.
- This work is published on *IEEE Transactions on Mobile Computing* (http://ieeexplore.ieee.org/document/7401077/) [14].

Related works

In the following, we briefly recall the works on the security problems of ad-hoc networks.

Communication. In [27] the authors propose three new mechanisms for key establishment using the framework of pre-distributing a random set of keys to each node. In [33], the authors use certain deployment knowledge that is available a priori to propose a novel random key pre-distribution scheme that exploits deployment knowledge and avoids unnecessary key assignments. In [84], Wander et al. quantify the energy cost of authentication and key exchange based on public-key cryptography on an 8-bit microcontroller platform.

False position claims. In [61] , the authors study the security issues of geographic routing (GR) protocols. In particular, they propose a location verification algorithm to address the attacks falsifying the location information. Furthermore, they also propose approaches for trust-based multi-path routing, aiming to defeat attacks on GR. In [25], Capkun et al. propose and analyze a new approach for securing localization and location verification in wireless networks based on hidden and mobile base stations.

Sybil attack. In the Sybil attack, a node illegitimately claims multiple identities. The work [72] analyzes the threat posed by the Sybil attack to wireless sensor networks and demonstrates that the attack can be exceedingly detrimental to many important functions of the sensor network such as routing, resource allocation, misbehavior detection, etc. Furthermore, the authors establish a classification of different types of the Sybil attack and then propose several novel techniques to defend against it.

In addition to these well known vulnerabilities of wireless sensors networks, mobile sensors suffer of other kind of security issues. In fact, lack tamper-proof hardware allows an adversary to capture several nodes, extract their cryptographic material and reprogram them according to its malicious goal. The reprogrammed sensors, hereafter called *malicious* sensors, may perform several attacks to damage the network, exploiting the specific vulnerabilities of the *deployment algorithm* in use. All the previous solutions for deploying mobile sensors fall in to one of three major families: approaches based on virtual force models [50, 62, 93, 41, 58], on the formation of patterns [11, 87], or on computational geometry techniques [85, 63, 12]. However none of this work focus on the security issues of the deployment algorithm. Only recently, the vulnerabilities of the virtual force approach for sensor deployment have been considered [9]. In this work, the authors introduce a simple attack tailored for mobile sensor deployment algorithms, called the *Oppor-*

tunistic Movement (OM) attack that aims to impede the natural spread of the network. Using a small set of malicious sensors, the attacker can influence the deployment of legitimate sensors by exploiting the coordination mechanism of the self-deployment approach. Malicious nodes may coordinate with each other to reduce the area in which the legitimate sensors are deployed, thus creating a non monitored zone.

While the work in [9] shows the detrimental effects of the OM attack against the virtual force approach, in [10] the authors provide only an experimental evidence of similar vulnerabilities in computational geometry approaches, and in particular in the Voronoi approach to mobile sensor deployment [85, 12].

In the present work we significantly extend the previous results on the vulnerabilities of Voronoi based approaches. In fact, we provide, for the first time in the literature, an analytic study of the vulnerabilities of such an approach and we formally prove the efficacy of the OM attack again the Voronoi based approaches. Furthermore, we propose a new secure algorithm SSD to counterattack the OM attack and guarantee the coverage goals in a complementary scenario with respect to SecureVor [10].

Chapter 3

On the Vulnerabilities of Voronoi-based Approaches to Mobile Sensor Deployment

3.1 Vulnerabilities of the Voronoi approach

In this section, we study how the OM attack presented in [9] is able to compromise the coverage goals of deployment algorithms based on the Voronoi approach. We first recall the deployment protocol VOR proposed by Wang et al. in [85], which is one of the most cited in the literature on Voronoi based deployment algorithms. After, we focus on the detrimental effects of the OM attack on the VOR algorithm. We conclude this section with a novel geometrical analysis that formally proves the effectiveness of the OM attack on the Voronoi based deployment approaches.

3.1.1 Background on the Voronoi approach

The Voronoi approach (VOR) to mobile sensor deployment has been introduced in [85]. It makes use of Voronoi diagrams to guide sensor movements within the AoI. According to [85], sensors communicate within a distance $R_{\mathtt{tx}}$ (*communication radius*), they sense over a circular area of radius $R_{\mathtt{s}}$ (*sensing radius*), with $R_{\mathtt{tx}} > 2R_{\mathtt{s}}$. Nodes can move in any direction inside the AoI, are endowed with low cost GPS, and are loosely synchronized.

VOR is executed in a distributed manner at each node and is round based. At each round t any sensor s broadcasts its position coordinates, and determines its set of neighbors $N_{\mathtt{tx}}^{(\mathtt{t})}(s)$, i.e. the sensors located within its communication radius. It then calculates its Voronoi polygon $V^{(\mathtt{t})}(s)$. Sensor s determines its next destination according to one of two movement criteria: the *Farthest Vertex* (FV) and the *MiniMax* (MM) [85].

According to FV a sensor s moves along the segment connecting its position and the farthest vertex of its polygon. Its destination is a point on this segment at distance $R_{\mathtt{s}}$ from the farthest vertex.

According to MM, the destination of s is the point that minimizes the maximum distance from the vertices of $V^{(\mathtt{t})}(s)$, which is the center of the minimum circle enclosing its polygon.

Regardless of the adopted movement criterion, a sensor s moves to its destination only if its movement provides a better coverage of $V^{(\mathtt{t})}(s)$, otherwise it remains still.

Furthermore, according to [85], s can traverse a maximum distance per round $d_{\mathtt{max}} = R_{\mathtt{tx}}/2 - R_{\mathtt{s}}$, to take into account possible inaccuracies in the distributed construction of Voronoi polygons, which may be due to the limited transmission radius.

3.1.2 The Opportunistic movement attack

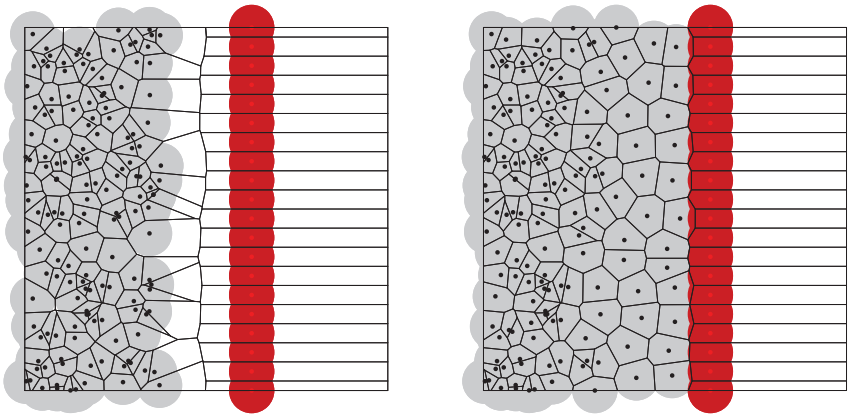
The original work [85] does not address the security vulnerabilities of the VOR approach. Since sensors lack tamper-proof hardware, an adversary may capture some nodes, and extract their cryptographic related information and reprogram them. Such malicious sensors may not be recognized by legitimate sensors as they are able to send valid messages containing a valid ID, and make use of legitimate cryptographic information. The attacker can thus exploit these corrupted nodes to perform malicious attacks to prevent a successful network deployment. For instance, the attacker can be interested in creating a non monitored area around a zone of interest, or isolating a part of the network. To pursue its goal, the attacker utilizes a set of malicious nodes that are able to collude with each other by performing coordinated movements and communications in order to influence the movements of the legitimate sensors.

The OM attack introduced in [9] aims at reducing the network coverage. To this purpose, malicious sensors initially form an *attack configuration* over the AoI. From such a configuration, malicious nodes start the attack by moving according to the adversary strategy, but communicating according to the communication protocol provided by the deployment algorithm.

The OM attack is a general attack which can be performed in different manners, depending on the movement strategy of malicious sensors. A particularly effective strategy is the Barrier Opportunistic Movement (BOM), in which malicious sensors form a linear barrier over the AoI [9, 10].

As provided by the OM attack, malicious sensors periodically communicate their positions at the beginning of each round in a legitimate way. By contrast, they move according to the attacker strategy. In particular, the malicious sensors forming the barrier may move towards legitimate sensors or remain still, in order to prevent legitimate sensors from spreading over uncovered areas.

In Figure 3.1, we show an example of a BOM attack. The red circular areas represent the sensing disks of the malicious nodes performing the BOM attack. The grey circles are the sensing ranges of the legitimate sensors that are spreading over the AoI according to VOR. The two figures 3.1(a) and (b) represent the initial and the final deployment, respectively. A barrier of malicious sensors is initially deployed over the AoI as in Figure 3.1(a), limiting the movements of legitimate sensors that will be able to spread only in the area limited by the barrier, as in Figure 3.1(b). The malicious nodes remain still, forming a barrier that prevents further movements of the legitimate sensors. In fact, the legitimate sensors that come in proximity with the barrier nodes stop moving. They do not move towards and across



(a) (b)

Figure 3.1: BOM Attack. (a) Initial deployment and (b)final deployment under VOR.

the barrier because, from the information received by malicious nodes, they derive that there is no way and no necessity to improve their local coverage.

In this work, we give a geometric characterization of the vulnerabilities of the Voronoi approach to mobile sensor deployment. Moreover, we exploit the information derived from the geometric characterization to design a novel algorithm SSD, which is able to neutralize the attack and works in complementary scenario with respect to SecureVor [10]. In fact, SecureVor is designed for an operative setting in which the communication radius is at least four times greater than the sensing radius, i.e. $R_{\mathtt{tx}} > 4R_{\mathtt{s}}$. SSD, instead, is designed to work in the same operative settings as the VOR approach, i.e. $R_{\mathtt{tx}} > 2R_{\mathtt{s}}$.

Similarly to [9, 10], in order to highlight the strength of the BOM attack, in this work we do not consider other attacks which may be performed in conjunction with BOM. Our goal is to show how the BOM attack, alone, can produce detrimental effects in terms of coverage to VOR based solutions.

3.1.3 Efficacy of the BOM attack against the Voronoi approach

In this section we formally analyze the vulnerabilities of VOR against the BOM attack. We refer the reader to [32] for a brief survey of the properties of Voronoi tessellations.

We consider the diagram of Figure 3.2, where a Cartesian reference models the AoI. Malicious sensors are evenly deployed along the axis $x = 0$, with

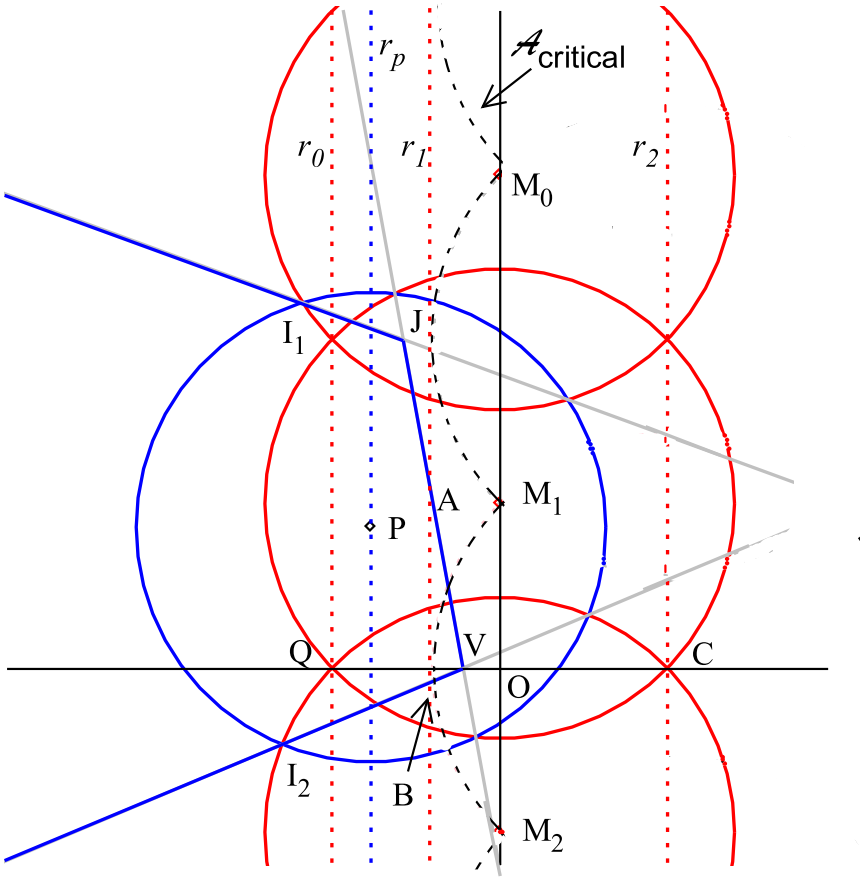


Figure 3.2: A legitimate sensor in P approaches a d-spaced barrier (sensors deployed in $(0, d/2 + k \cdot d)$, with $k \in \mathbb{N}$). As P is on the left of line r_1, it does not cross the barrier, as long as $d \leq \sqrt{3} R_s$.

step size d, occupying the positions $(0, d/2 + k \cdot d)$, with $k \in \mathbb{N}$. We hereby call such a configuration a *d-spaced barrier* of malicious sensors.

We define $\Delta(R_s, d) \triangleq \sqrt{R_s^2 - d^2/4}$, also referred to as Δ. Let w be the width of the overlapping region between two adjacent malicious sensors. In Figure 3.2, $w = d(C, Q) = 2\Delta$, where $d(\cdot, \cdot)$ is the Euclidean distance between two points. Notice that such a width is larger than R_s if $d \leq \sqrt{3} \cdot R_s$.

We use the following notation. We denote with $\mathcal{L}(\ell)$ and $\mathcal{R}(\ell)$ the half-planes at the left and right side of the line ℓ, respectively, where ℓ is a generic line of equation $x = x_\ell$. For brevity, we will use the same notation for a point $P = (x_p, y_p)$, denoting with $\mathcal{L}(P)$ and $\mathcal{R}(P)$, the half-planes $\mathcal{L}(x = x_p)$ and $\mathcal{R}(x = x_p)$, respectively.

Let us consider the lines r_0, r_1 and r_2, with equations $x = -\Delta$, $x = -R_s + \Delta$, and $x = \Delta$, respectively. Notice that if $d \leq \sqrt{3} \cdot R_s$, the line

r_1 falls between the lines r_0 and r_2. We will prove that the line r_1 acts as a *frontline* of the barrier, precluding legitimate sensors from traversing it, independently of the moving criterion adopted by the Voronoi algorithm VOR.

Given a sensor s positioned in P, we denote with $V(s)$ its Voronoi polygon, and with $\mathcal{C}(s)$ its sensing circle. The following Lemma 3.1.1 recalls a general property of Voronoi polygons that is necessary for the following discussion. It is a specific case of a more general theorem given in [12] (Theorem 3.1) and states that if a point of $V(s)$ is covered by any sensor, then it is also covered by s.

Lemma 3.1.1 (Theorem 3.1 of [12])**.** *Let us consider N sensors s_i, $i = 1, \ldots, N$, with positions $P_i = (x_i, y_i)$, sensing circles $\mathcal{C}(s_i)$ and sensing radius R_s. Let $V(s_i)$ be the Voronoi polygon of s_i. For all k and $j = 1, 2, \ldots, N$, $V(s_k) \cap \mathcal{C}(s_j) \subseteq \mathcal{C}(s_k)$.*

Let us denote with $\mathcal{A}_{critical}$ the locus of points determined by this equation:

$$\mathcal{A}_{critical} = \bigcup_{k \in \mathbb{N}} \{(x, y) | (x - \Delta)^2 + (y - k \cdot d)^2 = R_s^2,\ x \leq 0\}. \tag{3.1}$$

The shape of this locus is a periodic sequence of circular segments, along the y-axis, as depicted in Figure 3.2.

Extending the previous notation, we denote with $\mathcal{L}(\mathcal{A}_{critical})$ and with $\mathcal{R}(\mathcal{A}_{critical})$ the regions on the left and right side of $\mathcal{A}_{critical}$, respectively.

Lemma 3.1.2 (Frontline)**.** *Let us consider a legitimate sensor s, positioned in $P = (x_p, y_p)$, with $P \in \mathcal{L}(\mathcal{A}_{critical})$, with a d-spaced barrier of malicious sensors, with $d \leq \sqrt{3} \cdot R_s$. It holds that $V(s) \cap \mathcal{R}(r_0) \subseteq \mathcal{C}(s)$.*

In other words, the portion of the Voronoi polygon of s located on the right side of the line r_0 is completely covered, and is covered by s itself.

Proof. Consider the diagram of Figure 3.2. By contradiction, let us consider a point $Z \in V(s) \cap \mathcal{R}(r_0)$ and assume that Z is not in $\mathcal{C}(s)$. As a consequence of Lemma 3.1.1, Z is not covered by any sensor. Since the region between the lines r_0 and r_2 is covered by [at least] the barrier sensors, Z must be in $\mathcal{R}(r_2)$. Therefore, as Voronoi polygons are convex, $V(s)$ must have an uncovered vertex V_z in $\mathcal{R}(r_2)$.

Let us now remove from the diagram every sensor but s itself and the two closest barrier sensors m_1 and m_2, positioned in the points $M_1 = (0, d/2)$ and $M_2 = (0, -d/2)$. We obtain a new Voronoi polygon $V'(s)$ for s, such that $V(s) \subseteq V'(s)$. As s does not cover the vertex V_z, it also does not cover the unique vertex of the bigger enclosing polygon $V'(s)$. Let us denote with V

such a vertex, which is the unique intersection of the perpendicular bisectors of the segments $\overline{PM_1}$, $\overline{PM_2}$ and $\overline{M_1M_2}$, as shown in Figure 3.2.

$$V = (x_v, y_v) = \left(\frac{x_p^2 + y_p^2 - d^2/4}{2 \cdot x_p}, 0 \right).$$

As V is not covered, it must also be located in $\mathcal{R}(r_2)$, therefore $x_v > \Delta$, from which, recalling that $x_p < 0$, we derive: $(x_p - \Delta)^2 + y_p^2 \leq d^2/4 + \Delta^2 = R_s^2$.

Therefore the vertex V, generated when the sensor s is in $\mathcal{L}(\mathcal{A}_{critical})$, would actually be uncovered only if the sensor s were also located in $\mathcal{R}(\mathcal{A}_{critical})$, which is a contradiction. This implies that $Z \notin \mathcal{R}(r_2)$, concluding the proof that if $Z \in V(s) \cap \mathcal{R}(r_0)$ then Z must also belong to $\mathcal{C}(s)$.

□

The above Lemma 3.1.2 admits the following special case that follows by considering that r_1 is tangential to $\mathcal{A}_{critical}$.

Lemma 3.1.3. *Let us consider a legitimate sensor s, positioned in $P = (x_p, y_p)$, with $P \in \mathcal{L}(r_1)$, and a d-spaced barrier of malicious sensors, with $d \leq \sqrt{3} \cdot R_s$. It holds that $V(s) \cap \mathcal{R}(r_1) \subseteq \mathcal{C}(s)$.*

Lemma 3.1.3 is necessary to demonstrate that no sensor located in $\mathcal{L}(r_1)$ crosses the barrier with a Voronoi based movement. Therefore a minimum distance threshold

$$d_T(R_s, d) \triangleq R_s - \sqrt{R_s^2 - d^2/4}$$

can be defined, so that no sensor located at a distance higher than d_T from the barrier can cross it by means of a pure Voronoi based movement. We study the two criteria separately.

Theorem 3.1.4. *Let us consider a network of mobile sensors, with sensing radius R_s, being deployed according to the FV criterion. Let us consider a d-spaced barrier of malicious sensors with step $d \leq \sqrt{3} \cdot R_s$. No legitimate sensor located at a distance longer than $d_T(R_s, d)$ from the barrier, is able to traverse it.*

Proof. Let us consider a Cartesian reference so that the barrier is deployed along the axis $x = 0$, in the positions $(0, d/2 + k \cdot d)$, with $k \in \mathbb{N}$. Due to symmetry we consider the left side only.

Consider a legitimate sensor s, located at a distance higher than $d_T(R_s, d)$ from the barrier, on the left side of it. It follows that $P_s \in \mathcal{L}(r_1)$, where r_1 is the line of equation $x = -d_T(R_s, d)$. Thanks to Lemma 3.1.3, we can assert that the Voronoi polygon $V(s)$ of the sensor s does not have any uncovered vertex on the right side of the line r_1. Therefore, given the rules of the FV

criterion described in Section 3.1.1, either the sensor s does not move, or its destination D is also in the left side of the line r_1, that is $D \in \mathcal{L}(r_1)$. Since any movement of s, even if performed in multiple steps, will carry s from its current position P_s in $\mathcal{L}(r_1)$ to a destination D which is also in $\mathcal{L}(r_1)$, and since the region $\mathcal{L}(r_1)$ is convex, all the paths traversed by s are internal to $\mathcal{L}(r_1)$ and s never crosses the line $x = 0$ at which the barrier is deployed. □

We proceed with the analysis of the MiniMax criterion. We recall that the MiniMax point of a polygon is the center of its smallest enclosing circle.

Lemma 3.1.5. *Let $\mathcal{P}$ be a convex polygon with N vertices, and let $\mathcal{E}_\mathcal{P}$ be the minimum enclosing circle of $\mathcal{P}$. Every arc of 180° degrees in $\mathcal{E}_\mathcal{P}$ must traverse at least one vertex of the polygon $\mathcal{P}$.*

Proof. The minimum enclosing circle $\mathcal{E}_\mathcal{P}$ of a polygon $\mathcal{P}$ has at least two vertices of $\mathcal{P}$ on its boundary. As discussed in [32] we have two cases. Case (1): only two vertices of $\mathcal{P}$ are on the boundary of $\mathcal{E}_\mathcal{P}$, and they are antipodal. In such a case the two vertices divide $\mathcal{E}_\mathcal{P}$ into two half-circles. Case (2): more than two vertices of $\mathcal{P}$ are on the boundary of $\mathcal{E}_\mathcal{P}$, and three of these vertices form a non-obtuse triangle (or $\mathcal{E}_\mathcal{P}$ would not be minimal). In this case, the center of $\mathcal{E}_\mathcal{P}$ would coincide with the circumcenter of such a triangle and the angular distance between any two vertices would be less than or equal to 180° degree. It follows that in both cases every arc of the circumference whose length is 180° degree must contain at least one vertex of the polygon $\mathcal{P}$. □

We now give a characterization of the possible positions of the MiniMax point of $V(s)$ on the basis of the position of the sensor s with respect to the barrier.

Lemma 3.1.6. *Let us consider a d-spaced barrier of malicious sensors, with $d \leq \sqrt{3}R_s$, along the y-axis of a Cartesian reference and consider a legitimate sensor s positioned in $P \in \mathcal{R}(r_0) \cap \mathcal{L}(\mathcal{A}_{critical})$. If the Voronoi polygon $V(s)$ is not completely covered then its MiniMax point $M \in \mathcal{L}(P)$.*

Proof. Let us refer to Figure 3.2. As the lines r_0 and r_2 cross the intersection points between pairs of sensing circles of barrier sensors, the region $\mathcal{R}(r_0) \cap \mathcal{L}(r_2)$ is completely covered by the barrier sensors. The width of such a region is $w = 2\Delta$. Since $d \leq \sqrt{3}R_\mathtt{s}$ then $w \geq R_\mathtt{s}$. For vertical periodicity and horizontal symmetry in the construction, let us only consider the case with $0 \leq y_p \leq \frac{d}{2}$.

We initially neglect the presence of other sensors but s and the barrier sensors located in $M_0 = (0, \frac{3d}{2})$, $M_1 = (0, \frac{d}{2})$, and $M_2 = (0, \frac{-d}{2})$. As sensor s

approaches the barrier, $\mathcal{C}(s)$ can have a non null intersection with the barrier sensors, generating two vertices of the Voronoi polygon $V(s)$ (drawn in blue in Figure 3.2). As the closest barrier sensors are M_1 and M_2, the vertex V generated with these sensors is the closest to the barrier. Due to Lemma 3.1.2, as $s \in \mathcal{R}(r_0) \cap \mathcal{L}(\mathcal{A}_{critical})$, the portion of its Voronoi polygon $V(s)$ located in $\mathcal{R}(r_0)$ is completely covered and is covered by s. Therefore the uncovered points of $V(s)$ lie in the region $\mathcal{L}(r_0)$.

Let us denote with I_1 and I_2 the intersection points of $V(s)$ with the boundary of the sensing circle $\mathcal{C}(s)$. The uncovered points of $V(s)$ must be located beyond the arc $\overset{\frown}{I_1 I_2}$, in the left region of the segment $\overline{I_1 I_2}$. P is more distant than $R_{\mathtt{s}}$ from the uncovered points whereas its distance from all the vertices in $\mathcal{R}(P)$ is lower than $R_{\mathtt{s}}$.

We will now prove that no point of $V''(s) \triangleq V(s) \cap \mathcal{R}(P)$ can be the MiniMax of $V(s)$. We proceed by contradiction. Assume that $X \in V''(s)$ is the MiniMax of $V(s)$, and $V(s)$ has uncovered points. The requirement given by Lemma 3.1.5, establishes that X be the center of a circle which crosses at least one vertex of $V(s)$ every 180° degrees.

As the angle formed by I_1 and I_2, with any point of $V''(s)$ at the right hand side of the segment $\overline{I_1 I_2}$ is wider than 180° degrees, Lemma 3.1.5 states that an enclosing circle centered in X must cross one of the two vertices V and J, formed with M_1 and M_2 in addition to one or more uncovered vertices at the left side of the arc $\overset{\frown}{I_1 I_2}$, for a total of two or three vertices. Furthermore, the enclosing circle must cross the circumference $\mathcal{C}(s)$ in two points (in order to include an external region), which requires X to be in $\mathcal{L}(P)$[1].

In order to finish the proof we recall that in all this reasoning we neglected the presence of other sensors besides s, and the three barrier sensors located in M_0, M_1 and M_2. The argument remains valid even when considering other sensors, as they would cover additional portions of the AoI and of $V(s)$, and the potentially uncovered portion of the arc $\overset{\frown}{I_1 I_2}$ could only be smaller, leaving even wider angles at its right than we considered in the first part of the proof. Therefore, although having additional sensors may reduce the size of $V(s)$, when coverage of $V(s)$ is incomplete and s is in $P \in \mathcal{R}(r_0) \cap \mathcal{L}(\mathcal{A}_{critical})$, the MiniMax point would be in $\mathcal{L}(P)$. □

[1]This is because: 1) an enclosing circle bigger than $\mathcal{C}(s)$ and centered in $\mathcal{R}(P)$ would not touch any vertex in an angle wider than 180° degrees, contradicting Lemma 3.1.5. Therefore if X had a radius $R_X > R_{\mathtt{s}}$, it would be in $\mathcal{L}(P)$; 2) an enclosing circle of the same size as $\mathcal{C}(s)$ or even smaller is also possible, but in order for it to include points that are external to $\mathcal{C}(s)$ on the left side of the arc $\overset{\frown}{I_1 I_2}$, in addition to both vertices in $\mathcal{R}(P)$ it must be centered in $X \in \mathcal{L}(P)$.

Theorem 3.1.7. *Let us consider a network of mobile sensors, with sensing radius R_s, being deployed according to the MiniMax criterion. Let us consider a d-spaced barrier of malicious sensors, with step $d \leq \sqrt{3} \cdot R_s$. No legitimate sensor located at a distance longer than $d_T(R_s, d)$ from the barrier, is able to traverse it.*

Proof. Thanks to symmetry, we can consider the only left side of our reference plane.

Consider a legitimate sensor s, located at a distance higher than $d_T(R_s, d)$ from the barrier, on the left side of it. Then $P_s \in \mathcal{L}(r_1)$. If P_s is located in $\mathcal{L}(r_0)$, it may have uncovered portions of its polygon $V(s)$ in the half plane $\mathcal{R}(P)$. Therefore its MiniMax point can also be in $\mathcal{R}(P)$. Nevertheless, by analyzing the coordinates of the point V, we derive that $V \in \mathcal{L}(r_1)$. Therefore, the whole polygon $V(s)$ and so its MiniMax point M, are also in $\mathcal{L}(r_1)$.

By contrast, if $P_s \in \mathcal{R}(r_0) \cap \mathcal{L}(r_1)$, Lemma 3.1.6 allows us to conclude that the MiniMax point M resides in $\mathcal{L}(r_1)$ or the polygon is completely covered and no movement occurs. Therefore, the destination D is also in the left side of the line r_1. Since any movement of s, even if performed in multiple steps, will carry s from its current position P_s in $\mathcal{L}(r_1)$ to a destination D which is also in $\mathcal{L}(r_1)$, and since the region $\mathcal{L}(r_1)$ is convex, all the path traversed by s must be internal to $\mathcal{L}(r_1)$. Therefore s never crosses the line $x = 0$ at which the barrier is deployed. □

The above theorems show that the number of malicious sensors necessary to impede complete coverage of an area only depends on the perimeter of the area, regardless of the number of legitimate sensors deployed.

Notice also that the OM attack has no impact on an already deployed network which provides full coverage of the AoI.

Theorem 3.1.8. *Under VOR, once legitimate sensors have achieved full coverage of the AoI, the OM attack cannot cause the movement of any sensors.*

Proof. Let us consider a legitimate sensor s with neighbors $N(s)$. Since the AoI is completely covered, $V(s)$ is also completely covered, hence s does not move. When the OM attack starts, s has a set of neighbors $\widehat{N}(s)$, which may include some additional malicious sensors, and a polygon $\widehat{V}(s)$. Since $N(s) \subseteq \widehat{N}(s)$ then $\widehat{V}(s) \subseteq V(s)$, thus $\widehat{V}(s)$ is also completely covered, hence s does not move, in agreement with the rules described in Section 3.1.1. □

3.2 The SecureVor algorithm

In this Section we introduce a first preliminary result called SecureVor [10], a secure Voronoi-based deployment algorithm designed to counterattack the BOM attack explained in Section 3.1.2.

SecureVor is designed on the basis of the adversary model introduced in Section 3.1. It assumes a signature protocol to verify the exchanged messages, and an algorithm to verify position claims of nodes within the communication range R_{tx} [34, 90][2]. SecureVor assumes that $R_{tx} > 4R_s$ and sets $d_{max} = R_{tx}/4 - R_s$. We relax this assumption with the algorithm SSD, discussed in Section 3.3. Notice that, we do not require the communication range of a sensor to be a perfect disk. Indeed, there can be anisotropies provided that a sensor is able to communicate with all sensors located at a distance up to $4R_s$ from itself. Finally, similar to previous works [85, 12] on mobile sensor deployment, we assume that nodes are endowed with consumer grade GPS[3] and that they are loosely synchronized.

SecureVor provides a method to recognize malicious sensors and detect malicious movements when the deployment is based on VOR. It can be applied to both moving strategies FV and MiniMax. The idea of SecureVor is to detect malicious nodes by verifying the compliance of their movements to the rules of the deployment algorithm in use. This verification activity allows each sensor to formulate its own list of *trusted* and *untrusted* sensors. Each sensor will ignore untrusted neighbors and use only the information exchanged with trusted ones to determine future movements.

In order to let sensors reciprocally verify each other's movement, at the beginning of each round every sensor s is required to declare the set of its trusted neighbors, namely the set of sensors that it will use to determine its polygon. Notice that, a sensor determines this set only on the basis of its local observation, since SecureVor does not require transitive trust among sensors. The neighbor sensors of s locally calculate the polygon of s, based on its stated set, and verify whether its movement is in compliance with

[2] Location verification can be achieved by using dedicated hardware and/or previously deployed anchor nodes. Sensors can autonomously verify position claims if they are equipped with a radar system [34, 90]. These radars conform to our requirements as they are inexpensive, low power and provide object detection up to 20m distance. Alternatively, Ultra Wide Band systems [39] and anchor nodes can be used for location verification through Verifiable Multilateration (VM) [24]. In this case, anchor nodes are responsible for the location verification and advertise false location claims when detected. Using VM, a sensor incurs in a constant communication overhead for each anchor it communicates with.

[3] Low-cost, consumer grade GPS currently available provide accuracy in the orders of few decimeters [17] and have a cost around 200$ per unit [67].

the deployment algorithm or not. If a malicious movement is detected, s is marked as untrusted and ignored by its neighbors thereafter. SecureVor, and similarly SSD, could be extended with reputation-based techniques [83, 57].

Let N be the set of sensors to be deployed. We recall from Section 3.1.1 that we denote by $N_{\mathtt{tx}}^{(\mathtt{t})}(s)$ the neighbors of s, that is the set of sensors that are, at round t, at a distance less then the communication radius $R_{\mathtt{tx}}$ from s. The sets $N_{trusted}^{(\mathtt{t})}(s)$ and $N_{untrusted}^{(\mathtt{t})}(s)$ keep track, for a sensor s, of the set of sensors that s considers as *trusted* and *untrusted*, respectively, until round t. These sets are updated at each round. According to SecureVor, a sensor s only considers neighbors at a distance less than $R_{tx}/2$ as potential neighbors to calculate its own polygon. We refer to such neighbors at a round t as $Q^{(\mathtt{t})}(s)$. This choice enables s to be in communication with the sensors considered by its neighbors in $Q^{(\mathtt{t})}(s)$ to determine their polygon. Among the nodes in $Q^{(\mathtt{t})}(s)$, s takes into account only the sensors that it considers as trusted in order to determine its polygon. We define the set of sensors that s actually considers at round t as $N_{SV}^{(\mathtt{t})}(s) = Q^{(\mathtt{t})}(s) \bigcap N_{trusted}^{(\mathtt{t})}(s)$. $N_{SV}^{(\mathtt{t})}(s)$ may be empty if s has no trusted neighbor in its proximity at round t. In such a case, $V^{(\mathtt{t})}(s)$ is the whole AoI. Finally, the position of sensor s at the current round is denoted with $pos^{(\mathtt{t})}(s)$. Table 1 summarizes the adopted notation.

Notation	**Description**
$V^{(\mathtt{t})}(s)$	Polygon of s
$N_{\mathtt{tx}}^{(\mathtt{t})}(s)$	Neighbors of s (distance $\leq R_{tx}$)
$Q_{\mathtt{tx}}^{(\mathtt{t})}(s)$	Neighbors of s (distance $\leq R_{tx}/2$)
$N_{trusted}^{(\mathtt{t})}(s)$	Sensor s trusted neighbors until round t
$N_{untrusted}^{(\mathtt{t})}(s)$	Sensor s untrusted neighbors until round t
$pos^{(\mathtt{t})}(s)$	Position of s
$\widehat{pos}^{t}(s)$	Expected position of s
$N_{SV}^{(\mathtt{t})}(s)$	Sensors considered by s to build $V^{(\mathtt{t})}(s)$

Table 3.1: Summary of adopted notation. All notations refer to round t.

3.2.1 SecureVor in detail

SecureVor is round based similar to VOR. In particular, it comprises four phases, namely: *Position communication*, *Movement verification*, *Trusted neighbors communication* and *Coverage evaluation and movement*. Notice that we do not consider localization errors of the GPS positioning system

or of the location verification algorithm. SecureVor can be extended to take these aspects into account with the same approach described in Section 8.4 of [9].

The pseudo-code is shown as Algorithm SecureVor.

ALGORITHM SecureVor, node s at round t.

```
   // Position communication:
1  Broadcast pos^(t)(s);
2  Receive and verify neighbor positions;
3  Determine the sets N_tx^(t)(s) and Q^(t)(s);
   // Movement verification:
4  if t = 0 then
5      N_untrusted^(t)(s) = ∅;
6      N_trusted^(t)(s) = N;
7  else
8      N_untrusted^(t)(s) = N_untrusted^(t-1)(s) ∪ (Q^(t-1)(s) \ N_tx^(t)(s));
9      for q ∈ Q^(t)(s) s.t. q ∉ N_untrusted^(t)(s) do
10         if (s ∉ N_trusted^(t-1)(q) ∨ N_trusted^(t-1)(q) ⊈ N_tx^(t-1)(s)) then
11             N_untrusted^(t)(s) ← q;
12         Calculate V^(t-1)(q);
13         Calculate pos^t(q);
14         if pos^t(q) ≠ pos^t(q) then N_untrusted^(t)(s) ← q;
15         ;
16     N_trusted^(t)(s) = N \ N_untrusted^(t)(s);
17     N_SV^(t)(s) = Q^(t)(s) ⋂ N_trusted^(t)(s);
   // Trusted neighbors communication:
18 Broadcast the list of nodes in N_SV^(t)(s);
19 Receive N_SV^(t)(z) from any z ∈ Q^(t)(s);
   // Coverage evaluation and movement:
20 Calculate V^(t)(s) on the basis of N_SV^(t)(s);
21 if V^(t)(s) is completely covered then do not move;
22 ;
23 else Determine destination point and move accordingly. ;
```

Position communication (lines 1-3)
At the beginning of a round each sensor communicates its position to the neighbors through a signed message and determines the sets $N_{\mathtt{tx}}^{(\mathtt{t})}(s)$ and $Q^{(\mathtt{t})}(s)$, which are the set of communication neighbors of s and the set of nodes located at less than $R_{tx}/2$ from s, respectively.

Movement verification (lines 4-16)
In this phase, a sensor s verifies the movements of its neighbors to deter-

mine $N^{(t)}_{trusted}(s)$, $N^{(t)}_{untrusted}(s)$ and ultimately $N^{(t)}_{SV}(s)$. At the first round, $N^{(t)}_{trusted}(s) = N$ and $N^{(t)}_{untrusted}(s) = \emptyset$ **(lines 4-6)**.

The set of untrusted neighbors at round $t > 1$, $N^{(t)}_{untrusted}(s)$, contains all the sensors that were determined as untrusted in any of the previous rounds $N^{(t-1)}_{untrusted}(s)$ plus the sensors that were in $Q^{(t-1)}(s)$ and that are no longer in communication with s at the current round **(line 8)**[4]. Other sensors that are detected as malicious in the current round are added to $N^{(t)}_{untrusted}(s)$ **(lines 9-16)** as explained in the following.

A sensor s verifies, for each sensor q in $Q^{(t-1)}(s)$, not yet in $N^{(t)}_{untrusted}(s)$, the correctness of its movement in the previous round[5]. The first check that s performs for a sensor q, in order to verify the correctness of its movement, is on the truthfulness of the set $N^{(t-1)}_{SV}(q)$ **(lines 12-13)**. Two inconsistencies can be detected by s.

First inconsistency: the sensor q may have maliciously omitted s itself in the set of its trusted neighbors. Since s knows that it has behaved correctly according to the moving strategy, q must include s in its trusted set.

Second inconsistency: the sensor q may have fabricated the presence of some sensors in $N^{(t-1)}_{SV}(q)$ which are not physically located in its proximity to justify its movement. Sensor s can detect such malicious behavior because, according to SecureVor, a sensor q must select the sensors in $N^{(t-1)}_{SV}(q)$ among those in $Q^{(t-1)}(q)$. In order to be in $N^{(t-1)}_{SV}(q)$, a sensor must be at a distance at most $R_{tx}/2$ from q which implies that it is at a distance at most R_{tx} from s, being q at a distance at most $R_{tx}/2$ from s ($q \in Q^{(t-1)}(s)$). More formally $N^{(t-1)}_{SV}(q) \subseteq Q^{(t-1)}(q) \subseteq N^{(t-1)}(s)$.

If an inconsistency is detected, q is marked as untrusted and will be ignored by s hereafter. If no inconsistency is detected, the sensor s verifies whether q has moved according to the nodes belonging to $N^{(t-1)}_{SV}(q)$ **(lines 14-16)**. To this aim, s calculates the polygon of q at the previous round $V^{(t-1)}(q)$ on the basis of $N^{(t-1)}_{SV}(q)$ and $pos^{(t-1)}(q)$. Sensor s then compares the current position $pos^{(t)}(q)$, which q has just broadcast in the previous phase, with the expected position of q at the current round, $\widehat{pos}^{(t)}(q)$, calculated considering the polygon $V^{(t-1)}(q)$ and $pos^{(t-1)}(q)$. If $pos^{(t)}(q)$ is different from $\widehat{pos}^{(t)}(q)$, sensor s marks q as untrusted.

[4]SecureVor imposes that a sensor travels a maximum distance $d_{max} = R_{tx}/4 - R_s$. Hence even if two sensors, at a distance at most $R_{tx}/2$, move in opposite directions, they will stop at a distance from each other less than $R_{tx}/2 + 2(R_{tx}/4 - R_s)$ which is less than R_{tx}. This means that $Q^{(t-1)}(s) \subseteq N^{(t)}_{tx}(s)$, so if a sensor in $Q^{(t-1)}(s)$ is not in $N^{(t)}_{tx}(s)$, s can mark it as untrusted.

[5]Notice that, the trustworthiness of the sensors belonging to $Q^{(t)}(s) \setminus Q^{(t-1)}(s)$ will be evaluated at the next round.

Trusted neighbors communication (lines 19-20)
In this phase each sensor s broadcasts a signed message containing the IDs of the nodes belonging to the set $N_{SV}^{(t)}(s)$ calculated in the previous phase. This information enables the neighbors of s to verify its movement at the next round.
Coverage evaluation and movement (lines 21-23)
This phase is the same as the original VOR approach described in Section 3.1, except that each sensor s calculates its Voronoi polygon $V^{(t)}(s)$ on the basis of the sensors in $N_{SV}^{(t)}(s)$. Furthermore s looks for a destination point p within a distance $d_{max} = R_{tx}/4 - R_s$ instead of $d_{max} = R_{tx}/2 - R_s$.

3.3 The SSD algorithm

In this section we describe the SSD algorithm, designed to work in scenarios for which the hardware available at the sensor nodes does not satisfy the requirement on the transmission radius of SecureVor. In particular, unlike SecureVor which requires $R_{\mathtt{tx}} > 4R_{\mathtt{s}}$, SSD works under the same assumption of the original VOR algorithm, i.e. $R_{\mathtt{tx}} > 2R_{\mathtt{s}}$. Except for the transmission radius, SSD adopts the same assumptions of SecureVor discussed in Section 3.1.1.

The algorithm SSD explicitly aims at solving the blocked movement situation geometrically characterized in Section 3.1.3, in which a legitimate sensor does not move towards uncovered regions because it is in front of a barrier of malicious sensors.

Because this algorithm works under the relaxed assumption $R_{\mathtt{tx}} > 2R_{\mathtt{s}}$, sensors are not able to verify the movement of their neighbors only on the basis of message exchanges. This is because the communication range is too small to let a sensor verify whether its neighbors are behaving consistently with what should be their Voronoi polygon. Hence a sensor is not able, on the basis of messages alone, to distinguish a blocked movement situation (under attack) from a normal condition in which it cannot contribute a better coverage. In both cases the polygon of the sensor is completely covered and the sensor is not required to move to increase coverage of its polygon.

For these reasons, SSD provides temporary position swaps among pairs of neighbors to be performed when sensors are stationary and potentially in a blocked movement situation. We show a high level pseudocode in Algorithm SSD. As in the case of SecureVor, SSD requires each sensor s to maintain a list of trusted neighbors at time t: $N_{trusted}^{(t)}(s)$. In the next section we describe SSD in detail, making use of a similar nomenclature to the one introduced for SecureVor in Section 3.2.

3.3.1 SSD in detail

As in the case of SecureVor, according to SSD, each sensor s updates the list of trusted neighbors $N^{(t)}_{trusted}(s)$ at each round t. Such a set initially includes all the network nodes N **(line 2)**. At round t, s calculates its Voronoi polygon $V^{(t)}(s)$ by taking account only of the sensors in the set $N^{(t)}_{\mathtt{SSD}}(s)$, which is defined as the set of sensors in its radio proximity that s considers as trusted, i.e. $N^{(t)}_{\mathtt{SSD}}(s) = N^{(t)}_{trusted}(s) \bigcap N^{(t)}_{\mathtt{tx}}(s)$ **(lines 6-7)**.

If $V^{(t)}(s)$ is completely covered, s should remain still. Nevertheless this situation may occur in the presence of an attack. Therefore SSD provides the following mechanism to perform legitimacy checks of the behavior of its Voronoi neighbors. In order to determine the presence of malicious sensors, the sensor s selects one of its Voronoi neighbors and temporarily swaps its position with it. In order to prevent conflicting requests, these are generated at random times in a given time interval and served according to a FIFO discipline.

We now describe the process and conditions that result from a legitimate sensor being bound by a malicious barrier. When sensors are spread from a safe location, a legitimate sensor encounters a barrier that was initially far from it. When a sensor is blocked by some malicious sensors of the barrier, its Voronoi polygon is determined by new neighbors which were not previously observed, or by neighbors with which s forms a vertex that was uncovered in any previous round. If the sensor s moves towards a steady barrier, it converges to a position in which its polygon has vertices at the boundary of the sensing regions of a barrier sensor and therefore of the sensor s itself; this occurs because initially the sensor s forms uncovered vertices with barrier sensors, and it performs additional movements of smaller and smaller size, until it stops due to complete coverage. Figure 3.3 shows a legitimate sensor at the left of a malicious barrier, which is in a blocked movement situation, forming two vertices V and W which are both newly covered and located at the boundary of the sensing region.

We call any of the Voronoi neighbors of s resulting from this scenario the *vertex neighbor* of s[6].

Sensor s invites one of its new vertex neighbors, let it be j, to perform a swap of positions **(lines 8-10)**. The purpose of this swap is to let s perform a legitimacy check of j in order to calculate and verify its expected future movement.

[6]Notice that in order to provide convergence in a finite number of steps both VOR and SSD provide a movement threshold which prevents infinitesimal movements. Such a threshold is also kept into account in the definition of a vertex neighbor.

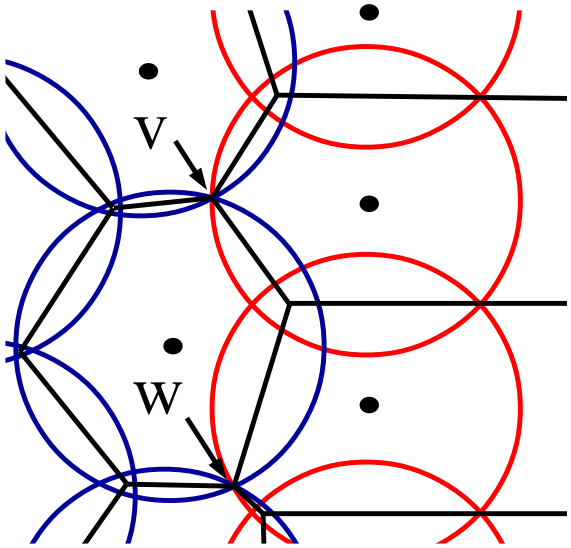


Figure 3.3: Boundary vertices V and W, between a legitimate sensor and two barrier sensors.

As j itself calculates its Voronoi polygon on the basis of its set of neighbors $N_{\mathtt{SSD}}^{(\mathtt{t})}(j)$, the position swap requires also j to send this set to s in order to let s be able to properly calculate the expected future movement of sensor j **(line 11)**. After this information exchange, s moves to the position currently held by j **(line 12)**, while j is required to move towards the position previously held by s. This last movement of j is required to ensure that the position of s is continuously covered and that the movements of other neighbors of s can be correctly verified. Sensor s exploits its location verification capabilities to verify if j honored the position swap protocol, otherwise it removes j from its local list of trusted sensors, therefore $N_{trusted}^{(\mathtt{t})}(s) \leftarrow N_{trusted}^{(\mathtt{t})}(s) \backslash \{j\}$ **(lines 14-15)**.

Once in the position previously held by sensor j, the sensor s sends a neighbor discovery message. The discovered list of communication neighbors of j is hereby denoted with $\widehat{N}_{\mathtt{tx}}^{(\mathtt{t})}(j)$. Sensor s and j send the list of communication trusted neighbors to each other. After this message exchange, thanks to its location verification capabilities, s verifies the consistency of the list of neighbors received by j, namely $N_{\mathtt{SSD}}^{(\mathtt{t})}(j)$ **(line 17)**, with the list of neighbors it observed while in the place of j, namely $\widehat{N}_{\mathtt{tx}}^{(\mathtt{t})}(j)$ **(line 13)**. If such consistency check fails, or if the trusted set of j does not include s, s does not return to its original position and continues the algorithm execution from the former position of sensor j. If otherwise the consistency check

ALGORITHM SSD, executed by node s at round t.

if *t=0* **then**
  $N^{(t)}_{trusted}(s) \leftarrow N$;
Exchange position msgs, determine $N^{(t)}_{tx}(s)$;
`// Movement verification:`
if (swapped with j at time $(t-1)$) $\wedge$ $(pos^{(t)}(j) \neq \widehat{pos}^{(t-1)}(j)$) **then**
  $N^{(t)}_{trusted}(s) \leftarrow N^{(t)}_{trusted}(s) \backslash \{j\}$;
Let $N^{(t)}_{SSD}(s) \leftarrow N^{(t)}_{trusted}(s) \bigcap N^{(t)}_{tx}(s)$;
Update $V^{(t)}(s)$ based on $N^{(t)}_{SSD}(s)$;
`// Coverage evaluation and Swap Agreements:`
if $V^{(t)}(s)$ is covered $\wedge$ # of new vertex neighbors ≥ 2 **then**
  Select a Vertex Neighbor j;
  Send *swap_request* to j;
  Receive $N^{(t)}_{SSD}(j)$ from j and send $N^{(t)}_{SSD}(s)$;
  Move to $pos^{(t)}(j)$ and send neighbor discovery msg;
  Receive position msgs and determine $\widehat{N}^{(t)}_{tx}(j)$;
  if (j did not reach $pos^{(t)}(s)$) **then**
    $N^{(t)}_{trusted}(s) \leftarrow N^{(t)}_{trusted}(s) \backslash \{j\}$;
  else
    if $(N^{(t)}_{SSD}(j) \subseteq \widehat{N}^{(t)}_{tx}(j))$ **then**
      Calculate $V^{(t)}(j)$ on the basis of $N^{(t)}_{SSD}(j)$;
      Calculate $\widehat{pos}^{(t)}(j)$;
      move to $pos^{(t)}(s)$;
`// Voronoi's Movement Phase`
else
  Move according to VOR criterion;

succeeds, sensor s is now able to calculate the Voronoi polygon of sensor j and the expected movement that j should perform **(lines 18-19)**. Sensors s and j can now return to their original positions **(line 20)**.

After this temporary swap activity, the algorithm SSD proceeds with the execution of the regular activities provided by the Voronoi approach **(lines 21-22)**. During the next movement phase, s verifies the movement of j using the location verification capabilities. If, during the next movement phase, sensor j fails to perform the expected movement calculated by s, it is removed from the trusted list of sensor s to be used at the next round **(lines 4-5)**. From now on, the sensor s will consider j as untrusted and will ignore it and adapt its Voronoi polygon and coverage as if j did not exist.

SSD provides some additional mechanisms to prevent more complex behaviors of malicious sensors. For example, malicious sensors could refuse to fulfill swap requests pretending to be involved in other position swap activities (with other malicious sensors). In order to prevent this behavior, first, SSD requires that a sensor which refuses a swap request provide a proof of the previous swap agreement (signed messages of both involved parties). Second, according to SSD, a random permutation $P^{(t)}$ of the sensor IDs is generated at each round, using the round counter t as a seed. This permutation is common to all sensors, and it establishes a priority in the position swap activities. In particular, sensors with higher priority at the current round have precedence in swapping, thus preventing two malicious sensors to continuously swap only between themselves. In addition, SSD allows the same pair of sensors to swap positions only once. Although a higher number of swaps per pair would increase the accuracy of detection of malicious sensors, this would be at the expense of energy for movements. Furthermore, by limiting the number of swaps, we prevent malicious sensors from extinguishing the batteries of legitimate neighbor sensors demanding unnecessary swaps. Note that, for the sake of simplicity the pseudo code in Algorithm SSD does not address the additional mechanisms described above, nor does it cover possible synchronization issues, and the case of s receiving swap requests from other sensors, which is treated according to the permutation priority described above.

3.4 Algorithm properties

In this section we provide a theoretical analysis of SecureVor and SSD. We hereafter denote with L and M the set of legitimate and malicious sensors, respectively. Hence, the total number of sensors deployed over the AoI is $|N| = |L| + |M|$.

3.4.1 Properties of SecureVor

We first study the capability of SecureVor to counteract the OM attack. Notice that, if a malicious node m moves in compliance to VOR it cannot be detected, since it is actually behaving as a legitimate sensor. Nevertheless, such movements are unlikely to meet the attacker goals. We define a *malicious movement* of a malicious sensor as a movement which is not in compliance with the deployment rules. Furthermore, given a malicious sensor $m \in M$ performing a malicious movement at round t, we define the set L_m^t as the set of legitimate sensors whose movement can be influenced by the malicious movement of m.

Lemma 3.4.1. *Given a malicious sensor $m \in M$ performing a malicious movement at round t, if $L_m^t \neq \emptyset$ then m is marked as untrusted by at least one sensor in L_m^t at round $t+1$.*

Proof. Since m can influence the movement of the sensors in L_m^t, such sensors consider m as trusted at the current round. Furthermore, since we assume that a node considers only sensors at a distance $R_{tx}/2$ to determine its polygon, $\forall s \in L_m^t \; d(s,m) < R_{tx}/2$ thus s is able to verify if $N_{SV}^{(\mathtt{t})}(m)$ is inconsistent. As a result, according to the assumptions made in Section 3.1, the only degree of freedom that m has in order to try to justify its malicious movement without being detected lies in the selection of the nodes to be advertised in $N_{SV}^{(\mathtt{t})}(m)$. Notice that all nodes in L_m^t are legitimate and are at a distance less than $R_{tx}/2$ from m, thus such sensors should be included in the trusted set of m. If m does not include one or more of them in $N_{SV}^{(\mathtt{t})}(m)$, such sensors mark m as untrusted at round $t+1$ and the assertion is valid.

If, on the contrary, m includes all sensors in L_m^t in $N_{SV}^{(\mathtt{t})}(m)$, such sensors are in communication range with m at round $t+1$ since $\frac{R_{tx}}{2} + 2d_{max} < R_{tx}$. As a result, sensors in L_m^t are able to verify the correctness of the current movement of m at the next round. Since m is performing the OM attack, its malicious movement is detected and thus all sensors in L_m^t mark m as untrusted at round $t+1$. □

We now prove that SecureVor terminates in a finite time. To this purpose, we show that at each round, either at least a malicious sensor is detected, or the overall coverage provided by legitimate sensors increases. We define a network state as follows.

Definition 3.4.2. *A* network state *under SecureVor is a vector $S_{SV} =< c_1, \ldots, c_{|M|}, s_1, \ldots, s_{|L|}, m_1, \ldots, m_{|M|} >$ where c_j is the number of legitimate sensors which consider the malicious sensor $m_j \in M$ as untrusted, $s_i \in L$ for $i = 1, \ldots, |L|$ and $m_j \in M$ for $j = 1, \ldots, |M|$.*

We define a function $f_{\mathtt{SV}} : \mathbb{N}^{|M|} \times L^{|L|} \times M^{|M|} \to \mathbb{N} \times \mathbb{R}_+$ such that given a network state $S_{\mathtt{SV}}$, $f_{\mathtt{SV}}(S_{\mathtt{SV}}) = (\sum_{j=0}^{|M|} c_j, A_{total})$, where A_{total} is the size of the area covered by legitimate sensors in $S_{\mathtt{SV}}$. Given two network states $S^1_{\mathtt{SV}}, S^2_{\mathtt{SV}}$ we say that $f_{\mathtt{SV}}(S^1_{\mathtt{SV}}) \prec f_{\mathtt{SV}}(S^2_{\mathtt{SV}})$ according to the lexicographic order. Notice that, the function $f_{\mathtt{SV}}$ is upper-bounded by the pair $(|L||M|, AoI)$. In the following, in order to prove the convergence of SecureVor, we show that at each round the value of such function increases.

Theorem 3.4.3. *The algorithm SecureVor converges.*

Proof. Let us consider a generic state change from round t to round $t+1$. We want to show that $f_{\mathtt{SV}}(S^{(\mathtt{t})}_{\mathtt{SV}}) \prec f_{\mathtt{SV}}(S^{(\mathtt{t+1})}_{\mathtt{SV}})$. We recall that, for a malicious sensor $m \in M$ performing a malicious movement at round t, L^t_m is the set of legitimate nodes whose movement can be influenced by the malicious movement of m. We consider two cases:

Case 1: $\exists m_j \in M$ s.t. $L^t_{m_j} \neq \emptyset$.
Thanks to Lemma 3.4.1 we know that there exist at least one legitimate sensor at round $t+1$ that marks m_j as untrusted. As a result, $c_j[S^{(\mathtt{t})}_{\mathtt{SV}}] < c_j[S^{(\mathtt{t+1})}_{\mathtt{SV}}]$, hence $f(S^{(\mathtt{t})}_{\mathtt{SV}}) \prec f(S^{(\mathtt{t+1})}_{\mathtt{SV}})$.

Case 2: $\forall m_j \in M$, $L^t_{m_j} = \emptyset$.
In this case no malicious movement influences the movement of legitimate sensors. As a result no malicious sensor is detected at round $t+1$, hence $\forall$ $j = 1, \ldots, |M|$, $c_j[S^{(\mathtt{t+1})}_{\mathtt{SV}}] = c_j[S^{(\mathtt{t})}_{\mathtt{SV}}]$. Notice that, if no malicious sensor is detected SecureVor lets sensors deploy according to the rules of VOR. Under VOR, if in a specific round at least one sensor moves, the provided coverage increases (as shown in the proof of Theorem 4.1 of [12]), so also in this case it holds that $f_{\mathtt{SV}}(S^{(\mathtt{t})}_{\mathtt{SV}}) \prec f_{\mathtt{SV}}(S^{(\mathtt{t+1})}_{\mathtt{SV}})$. As the function $f_{\mathtt{SV}}()$ is upper-bounded and it increases at each round of the algorithm execution, we can conclude that SecureVor converges.

□

The above theorem proves that SecureVor converges, nevertheless the increase in coverage may be infinitesimal and the algorithm may require an infinite number of rounds to terminate.

Corollary 3.4.4. *The algorithm SecureVor terminates if movements are allowed only if they provide a coverage increase which exceeds a positive minimum threshold ϵ.*

The introduction of ϵ ensures fast termination and power saving, at the expense of a small loss in the coverage extension.

3.4.2 Properties of SSD

Similarly to SecureVor, to prove the termination of SSD we first show that it converges. We consider a static barrier of malicious sensors performing the BOM attack.

Definition 3.4.5. *A* network state *under SSD is a vector* $S_{SSD} =< a_1, \ldots, a_{|L|}, s_1, \ldots, s_{|L|}, m_1, \ldots, m_{|M|} >$ *where* a_j *is the number of swaps performed by the legitimate sensors* s_j*,* $s_i \in L$ *for* $i = 1, \ldots, |L|$ *and* $m_j \in M$ *for* $j = 1, \ldots, |M|$*.*

We define a function $f_{\texttt{SSD}} : \mathbb{N}^{|L|} \times L^{|L|} \times M^{|M|} \to \mathbb{N} \times \mathbb{R}_+$ such that given a network state $S_{\texttt{SSD}}$, $f_{\texttt{SSD}}(S_{\texttt{SSD}}) = (\sum_{j=0}^{|L|} a_j, A_{total})$, where A_{total} is the size of the area covered by legitimate sensors in S. Given two network states $S^1_{\texttt{SSD}}, S^2_{\texttt{SSD}}$ we say that $f_{\texttt{SSD}}(S^1_{\texttt{SSD}}) \prec f_{\texttt{SSD}}(S^2_{\texttt{SSD}})$ according to the lexicographic order.

Notice that, the function $f_{\texttt{SSD}}()$ is upper-bounded by the pair $(|L| \times (|L| - 1 + |M|), AoI)$, since legitimate sensors are allowed to swap at most once with another sensor, and the maximum area that can be covered is the whole AoI. Similarly to the case of SecureVor, we show that during the unfolding of SSD the value of such function increases.

Theorem 3.4.6. *The algorithm SSD converges.*

Proof. Let us consider a generic network state $S^{(\texttt{t})}_{\texttt{SSD}}$ at round t. We want to show that either the algorithm has terminated at round t, or there exists $k \in \mathbb{N}$ s.t. $f_{\texttt{SSD}}(S^{(\texttt{t})}_{\texttt{SSD}}) \prec f(S^{(\texttt{t+k})}_{\texttt{SSD}})$. We consider two cases:
Case 1: $\exists s_i \in L$ that performs a movement at round t.
Legitimate sensors deploy according to the rules of the VOR approach. Since under VOR if a sensor moves then the overall coverage increases [12], this holds also under SSD. As a result, if at least one sensor moves then $f_{\texttt{SSD}}(S^{(\texttt{t})}_{\texttt{SSD}}) \prec f_{\texttt{SSD}}(S^{(\texttt{t+1})}_{\texttt{SSD}})$.
Case 2: $\nexists s_i \in L$ that performs a movement at round t.
This case occurs if no sensor can move and increase the coverage of its polygon, hence sensors will also not move at subsequent rounds. As a result, either the algorithm has terminated, or the network state may change as a consequence of a swap. Let us consider a sensor s_i which wants to exchange with s_j. s_i may not be able to exchange with s_j at round t, due to their priority in $P^{(\texttt{t})}$. However, the random generation of permutations ensures that there eventually exists $k \in \mathbb{N}$ s.t. in $P^{(\texttt{t+k})}$, s_i has higher priority than s_j, and the swap can be performed. In this case, the number of swaps increases from state $S^{(\texttt{t})}_{\texttt{SSD}}$ to $S^{(\texttt{t+k})}_{\texttt{SSD}}$, and in particular $a_i^{(\texttt{t+k})} = a_i^{(\texttt{t})} + 1$. As a result, $f_{\texttt{SSD}}(S^{(\texttt{t})}_{\texttt{SSD}}) \prec f_{\texttt{SSD}}(S^{(\texttt{t+k})}_{\texttt{SSD}})$.

As $f_{\texttt{SSD}}()$ is upper-bounded and it increases at each round, SSD converges. □

Similarly to SecureVor, to prove the termination of SSD we include a positive threshold $\epsilon > 0$, which prevents infinitesimal increase in coverage, as stated by the following corollary.

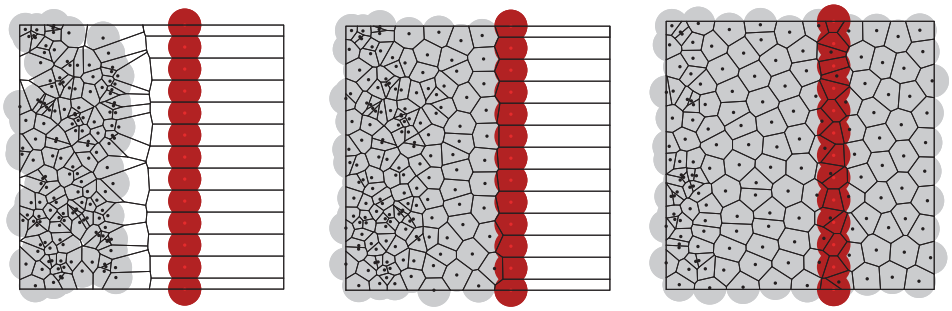
Corollary 3.4.7. *The algorithm SSD terminates provided that movements are allowed only if they enable a coverage increase greater than a threshold* $\epsilon > 0$.

Unlike SecureVor, we cannot formally prove that every time a malicious node is encountered in SSD it is detected, due to the limited information available at each sensor as a consequence of the smaller transmission radius than with SecureVor. In particular, we cannot exclude that a malicious sensor m is not detected during a swap, because its polygon is actually fully covered by some legitimate sensors that crossed the barrier in a previous round. Nevertheless, the experiments show that overall, SSD thwarts the OM attack. Hence, in Section 3.5 the effectiveness of SSD in defeating the OM attack is shown through extensive experiments, which also demonstrate the capability of SSD to achieve full coverage of the AoI.

3.5 Experimental results

In this section we provide an analysis of the performance of SecureVor and SSD. To this purpose, we developed a simulator on the basis of the Wireless module of the Riverbed Opnet simulation environment [94]. In the simulations we considered a squared AoI of size 80m×80m. Sensors can move at a maximum speed of 1m/s. We set the threshold ϵ for minimum coverage increase to 0.001, for both SecureVor and SSD. We investigated several scenarios which consider different settings of R_{tx} and R_s.

In the first scenario (Scenario A), we consider a setting favorable to SecureVor, i.e. such that $R_{\texttt{tx}} > 4R_{\texttt{s}}$. In the second scenario (Scenario B), we consider instead a setting for which SSD is designed, i.e. $4R_{\texttt{s}} > R_{\texttt{tx}} > 2R_{\texttt{s}}$. The third scenario (Scenario C) is devoted to a sensitivity analysis of both algorithms to the setting of the transmission radius. While all these scenarios consider a static BOM attack, the last experimental scenario (Scenario D) considers a BOM attack with a mobile barrier.



(a) (b) (c)

Figure 3.4: Scenario A: Initial deployment of 150 legitimate sensors and 13 malicious sensors (a), final deployment of VOR (b), and SecureVor (c).

3.5.1 Scenario A: SecureVor setting

In this scenario we set $R_{tx} = 30$m and $R_s = 5$m, and investigate the performance of SecureVor. Under this setting, the maximum moving distance d_{max} is 2.5m. Malicious sensors perform the BOM attack by periodically advertising their position during the Position communication phase while remaining still. In order to avoid being easily detected by the surrounding legitimate sensors, each malicious sensor m, advertises a trusted set $N^{(t)}_{trusted}(m) = Q^{(t)}(m)$. Legitimate sensors are randomly deployed on the left side of the AoI.

We compare the performance of SecureVor with respect to the results obtained by the original VOR algorithm in the same setting. In order to evaluate the overhead introduced by SecureVor, we also show the behavior of VOR when all sensors are legitimate and expand freely (VOR-Free in the figures) without a barrier.

Before showing the results, we provide an example of the detrimental effect of the BOM attack in this scenario with 150 legitimate sensors and 13 malicious sensors. Figure 3.4(a) shows the initial deployment, while Figures 3.4 (b) and (c), show the final deployments achieved by VOR and SecureVor, respectively. Under VOR legitimate sensors are not able to cross the barrier, resulting in a significant loss of coverage. On the contrary, under SecureVor legitimate sensors detect malicious sensors, and are able to cross the barrier and achieve full coverage of the AoI.

In the experiments we set the number of malicious sensors to 13 and we increase the number of legitimate sensors from 60 to 240. Figure 3.5(a) shows the coverage of the AoI achieved by the considered algorithms. Legitimate sensors under VOR are not able to cross the barrier of malicious sensors, no

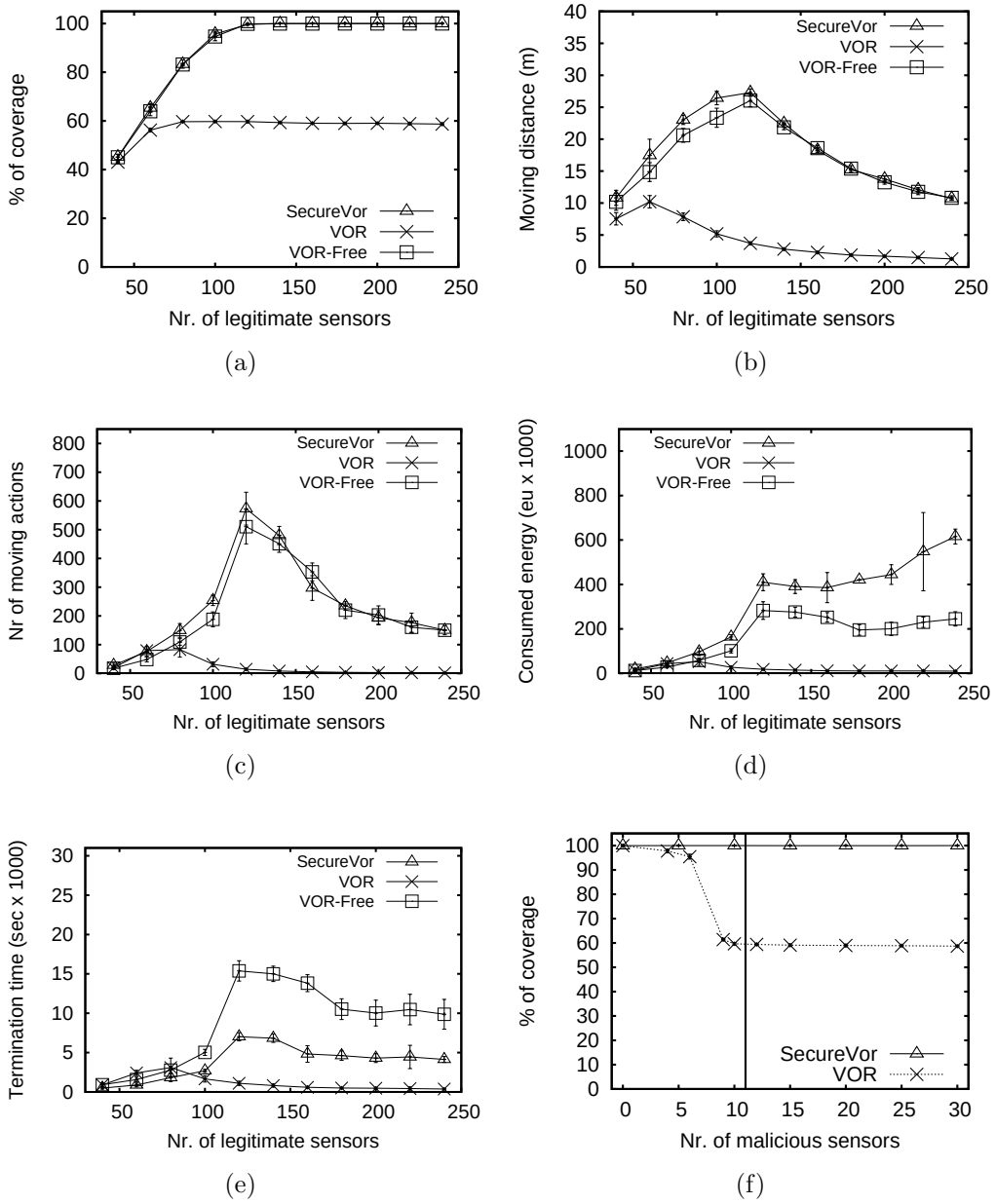


Figure 3.5: Scenario A: coverage of the AoI (a), traversed distance (b), number of movements (c), consumed energy (d), termination time (e). Coverage achieved with 140 legitimate sensors (f).

matter how many legitimate sensors are deployed. Therefore the coverage is at most 60%.

On the contrary SecureVor, thanks to its security policy, detects and ignores malicious sensors and successfully covers the AoI. Note that, SecureVor achieves the same coverage of VOR-Free, that is the original Voronoi algorithm with no attack. This shows that SecureVor completely defeats the attack and maximizes the coverage.

Since under VOR sensors are not able to spread over the AoI when the attack is in place, this algorithm achieves lower values of all the considered performance metrics, such as traversed distance and consumed energy, with respect to the other algorithms. This does not imply superior performance of this algorithm, but just the inability to cover the AoI. For this reason, in the following we do not discuss its results although we show them in the figures.

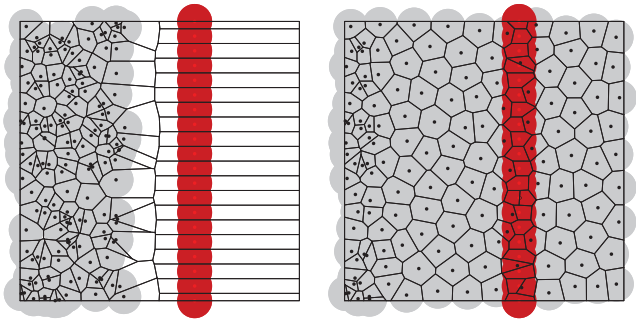
Figure 3.5(b) shows the average distance traversed by sensors. SecureVor introduces a very small overhead in terms of traversed distance with respect to VOR-Free. The peak in the traversed distance of all approaches is a common behavior of mobile sensors deployment algorithms since, when few sensors are available, all sensors move in order to contribute to the achievement of the final coverage. Instead, when more sensors are available, the average traversed distance decreases, since only sensors detecting a coverage hole are allowed to move.

Figure 3.5(c) shows the average number of moving actions. This is an important metric to evaluate mobile sensor deployment algorithms, since a sensor consumes a high amount of energy to start and stop a movement. Similar considerations with respect to the traversed distance and the peaks in the Figures discussed above can be made. SecureVor introduces a small overhead in terms of number of movements due to the reduced traversed distance per round which results in an higher number of movements to traverse the same distance.

We now show results related to sensor energy consumption. We adopt the energy cost model commonly used in the literature for mobile sensors [11, 85, 8]. In particular, receiving a message costs 1 energy units (eu), sending a message 1.125eu, traversing one meter costs 300eu and starting/stopping a movement costs as one meter of movement. We consider a cumulative energy consumption metric which takes into account all the above contributions.

Figure 3.5(d) shows the obtained results. All algorithms incur in a higher communication cost as the sensor density increases. Such an overhead is higher under SecureVor because of the additional messages required to communicate the trusted neighbor set. The energy consumption under VOR-Free is 43% less energy with respect to SecureVor.

The termination time is shown in 3.5(e). SecureVor shows a shorter termination time with respect to VOR-Free. This is due to the shorter maximum traversed distance of SecureVor which allows shorter movements that are



(a) (b)

Figure 3.6: Scenario B: Initial deployment (a), and final deployment under SSD (b).

forbidden by VOR. As a result, under VOR sensors move only when a long movement is possible, thus resulting in cascade movements which lengthens the termination time. On the contrary, shorter movements enable sensors to move more in parallel, resulting in a lower termination time for SecureVor.

In order to further study the performance of the considered algorithms, we performed some experiments by setting the number of legitimate sensors to 140 and by increasing the number of malicious sensors from 0 to 30. Figure 3.5(f) shows the achieved coverage. The vertical line represents the minimum number of malicious sensors for which the distance d between them is less than $\sqrt{3}R_s$. As proven in Theorems 3.1.4 and 3.1.7, legitimate sensors are not able to cross the barrier if d is less than or equal to such a value. These experiments show that legitimate sensors do not cross the barrier even when a small number of malicious sensors is present. SecureVor is not affected by the number of malicious sensors deployed, since legitimate sensors are able to detect malicious sensors and cover the AoI.

3.5.2 Scenario B: SSD setting

In this section we consider a setting for which SSD is designed, that is where $4R_{\mathtt{s}} > R_{\mathtt{tx}} > 2R_{\mathtt{s}}$. In particular, we set $R_{\mathtt{tx}} = 12$m, and $R_{\mathtt{s}} = 5$m. Therefore, the maximum moving distance $d_{\mathtt{max}}$ is 1m.

Similar to the previous scenario, we study the performance of SSD in presence of the BOM attack performed by 20 malicious sensors and by increasing the number of legitimate sensors deployed. We compare the performance of SSD to the original VOR algorithm and with the same algorithm in absence of the attack (VOR-Free).

Figure 3.6(a) show an instance of this scenario with 150 legitimate sensors. Figure 3.6(b) shows the final deployment achieved by SSD. Even with limited transmission radius, legitimate sensors are able to detect malicious nodes thanks to position swaps, and ultimately achieve full coverage.

Figure 3.7(a) shows the coverage of the AoI achieved by the considered approaches. VOR achieves similar results as in the previous scenario, with legitimate sensors unable to cross the barrier. On the contrary, SSD successfully defeats the attack and enables legitimate sensors to cover the AoI, achieving the same coverage of VOR-Free. Similarly to the previous scenario, we do not discuss the performance of VOR in the following.

Figure 3.7(b) shows the average distance traversed by sensors. The figure evidences the additional traversed distance of SSD with respect to VOR-Free, due to the position swaps necessary to detect malicious sensors. As mentioned for Scenario A, the peak in the traversed distance occurs in correspondence to the minimum number of legitimate sensor necessary to achieve full coverage.

Figure 3.7(c) shows the average number of start and stop actions. SSD shows a lower number of starts and stops with respect to VOR-Free. This apparently surprising result is due to the swap activity. In particular, when a legitimate sensor swaps with a malicious sensor, the malicious sensor is detected and the legitimate sensor does not move back to its original position. As a result, such sensor performed a longer movement, whose length is not limited by the parameter $d_{\texttt{max}}$, nor it is affected by small local position adjustments.

Figure 3.7(d) shows the overall consumed energy. Such a measure includes both the communication and the movement costs. SSD performs better than VOR-Free in this case, thanks to the fewer number of start and stop actions, that dominate the energy consumption.

The termination time is shown in Figure 3.7(e). The results detailed in this figure reveal a longer termination time of SSD compared to VOR-Free. This is due to the longer round length required to include swap activities during every iteration of the algorithm. Nevertheless, this increased termination time allows SSD to defeat the attack, even in the restricted case of the communication radius.

We finally show in Figure 3.7(f) the average number of swaps per sensor under SSD. The number is relatively low, with a peak of eight swaps when the number of sensors deployed is close to the minimum to achieve full coverage. As the number of sensor increases, the number of swaps rapidly decreases.

We performed additional experiments varying the number of malicious sensors. We do not show them as they look very similar to those obtained for SecureVor and detailed in Figure 3.5 (d). These results confirm that

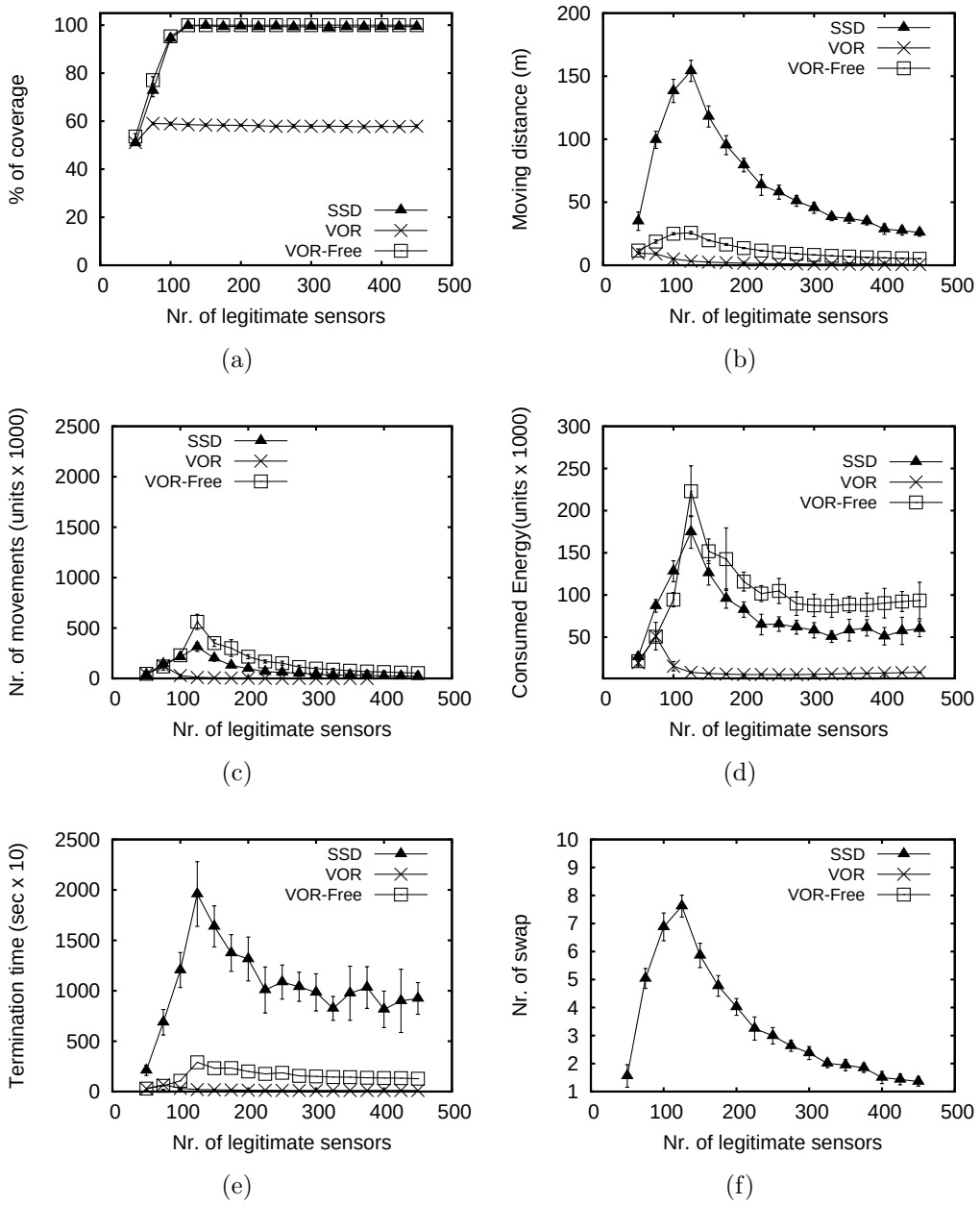


Figure 3.7: Scenario B: coverage of the AoI (a), traversed distance (b), number of movements (c), consumed energy (d), termination time (e), average number of swaps (f).

the performance of SSD also, is not significantly affected by the number of malicious sensors.

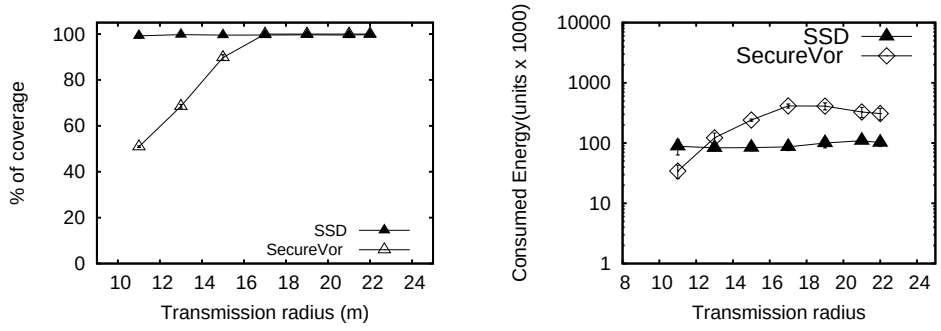
3.5.3 Scenario C: Transmission radius sensitivity analysis

In this section we perform a sensitivity analysis to compare SecureVor and SSD under various settings of the transmission radius. We recall that SecureVor assumes that $R_{\mathtt{tx}} > 4R_{\mathtt{s}}$, while SSD is designed for the more restricted scenario in which $4R_{\mathtt{s}} > R_{\mathtt{tx}} > 2R_{\mathtt{s}}$. In these experiments we increase $R_{\mathtt{tx}}$ from the setting of SSD to the setting of SecureVor, and compare the performance of the algorithms.

To enable SecureVor to work even when $R_{\mathtt{tx}} < 4R_{\mathtt{s}}$, we define a *virtual sensing radius* $R_{\mathtt{vs}}$, for which $R_{\mathtt{tx}} > 4R_{\mathtt{vs}}$. Using this modification, legitimate sensors deploy as if the sensing radius were the virtual radius $R_{\mathtt{vs}}$. This allows legitimate sensor to detect malicious sensors. However the drawback of this setting is a denser deployment, so more sensors are needed to achieve full coverage.

In this experimental scenario, we consider 200 sensors with sensing radius $R_{\mathtt{s}} = 5$m, while we let the transmission radius $R_{\mathtt{tx}}$ vary from 11 meters up to 22 meters. As the maximum allowed distance for SecureVor depends on the virtual radius according to the equation $d_{\mathtt{max}} = \frac{R_{\mathtt{tx}}}{4} - R_{\mathtt{vs}}$, we fix the maximum moving distance $d_{\mathtt{max}}$ for SecureVor to 0.5m, and let the value of $R_{\mathtt{vs}}$ grow according to the equation $d_{\mathtt{max}} = \frac{R_{\mathtt{tx}}}{4} - R_{\mathtt{vs}}$. Therefore $R_{\mathtt{vs}} = \min\{R_{\mathtt{s}}, (\frac{R_{\mathtt{tx}}}{4} - d_{\mathtt{max}})\}$. Under such a setting, when $R_{\mathtt{tx}}$ spans from 11 meters to 22 meters, $R_{\mathtt{vs}}$ correspondently grows from 2.25 meters to 5 meters. A further increase in $R_{\mathtt{tx}}$ would not cause any increase in the virtual radius, which would be the same as the real sensing radius. This last setting is what SecureVor requires to work at its best, deploying sensors at the density required by VOR.

Figure 3.8, shows the coverage achieved by the two algorithms when the transmission radius $R_{\mathtt{tx}}$ increases. As we can see, SSD always reaches full coverage of the AoI, independently of the setting of $R_{\mathtt{tx}}$. By contrast, SecureVor is unable to complete the coverage when working with transmission radius lower than 17m, because the corresponding virtual sensing radius is too short to cover the area with only 200 sensors. These results highlight the benefit of using SSD when the assumptions required of SecureVor are not met by the available hardware. Although SecureVor can be used with minor modifications, its performance can be significantly penalized. The results of



(a) (b)

Figure 3.8: Scenario C: coverage of the AoI (a) and consumed energy (b).

the other performance metrics have similar trends to those shown in Figure 3.5.

Figure 3.8 (b), shows the energy consumed by the two algorithms. This figure highlights that SecureVor consumes less energy than SSD only when the transmission radius is lower than 12m, when SecureVor does not perform complete coverage. By contrast, when the transmission radius is larger than 12m, and even in the most favorable setting to SecureVor, the energy consumption with SSD is lower than with SecureVor. This reveals the superiority of SSD even in the scenarios which are more favorable to SecureVor.

3.5.4 Scenario D: Mobile barrier attack

The last set of experiments introduces a more complex attack, in which malicious sensors initially form a barrier, and then start moving towards legitimate sensors. Malicious movements are perpendicular to the barrier and are of length $d_{\texttt{max}}$, according to the algorithm rules. A malicious sensor never breaks the barrier when moving, therefore it only moves if it can maintain a distance lower than $\sqrt{3}R_{\texttt{s}}$ with neighbor malicious sensors. A malicious sensor stops moving as soon as it reaches a distance lower than $2R_{\texttt{s}}$ from at least one legitimate sensor. In this setting we use a setting of transmission and sensing radius suitable for both SecureVor and SSD.

Figure 3.9(a) shows the initial deployment, with 150 legitimate sensors and a moving barrier of 20 malicious sensors. This attack can severely compromise the coverage provided by legitimate sensors under VOR, which in fact terminates the execution as shown in Figure 3.9(b). Malicious sensors successfully confine legitimate sensors in a small portion of the AoI. Legitimate sensors do not cross the barrier because the mutual distance between

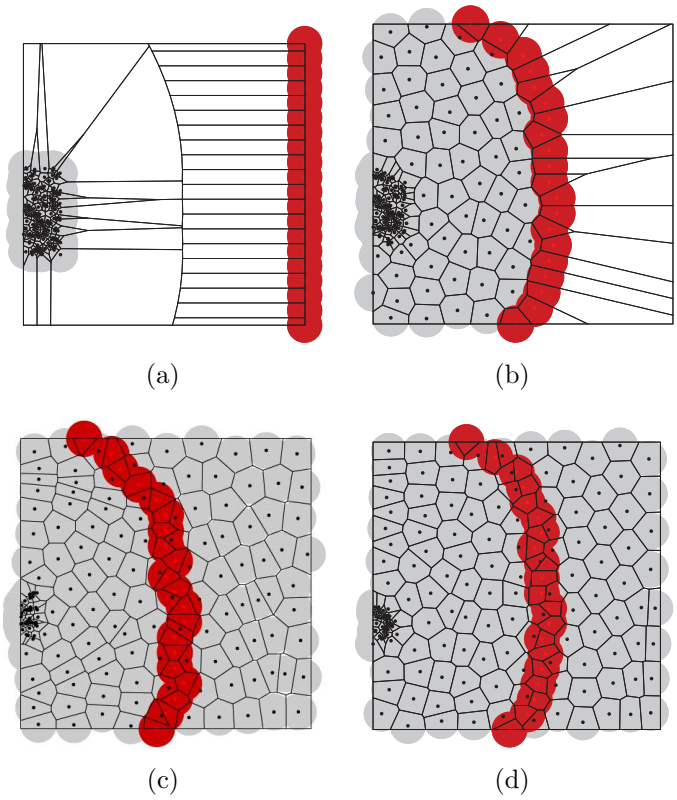


Figure 3.9: Scenario D. Dynamic barrier: initial deployment (a), final deployment with VOR (b), SecureVor (c) and SSD (d).

malicious sensors is always lower than $\sqrt{3}R_s$, which also confirms the theoretical results described in Section 3.1.3.

Under SecureVor legitimate sensors discover the malicious movements resulting from the barrier movement. Consequently, legitimate sensors are able to detect and ignore malicious sensors, and achieve full coverage as shown in Figure 3.9(c). SSD requires a minor modification to work under the attack of a dynamic barrier, with particular focus on the concept of *vertex neighbor*. When a legitimate sensor approaches the barrier, the new vertex can be located in a circular corona of the boundary, and not exactly on the boundary, to take into account possible movements of the barrier sensors. With such a modification, SSD successfully lets legitimate sensors discover barrier sensors and ignore them, achieving full coverage as shown in Figure 3.9(d).

We conducted additional experiments with more complex configurations, such as multiple barriers or barriers of various irregular shapes. Results show that both SecureVor and SSD are able to defeat such attacks.

3.6 Conclusions

In this part, we focused on the application of wireless sensor networks in critical, hostile to man and dangerous scenarios. We paid attention to the security issues of the deployment protocol of mobile wireless sensor networks. In particular, we addressed the vulnerabilities of one of the most acknowledged approaches to mobile sensor deployment: the Voronoi based approach. We considered a recently proposed attack to mobile sensor networks, the OM attack, and characterized the geometric conditions under which such an attack is effective if the network adopts the Voronoi approach to deployment.

We analyzed the previous solution SecureVor [10] and we developed a new algorithm called Secure Swap Deployment (SSD) to counteract the OM attack. The algorithms work in complementary operative settings. SecureVor assumes that the transmission radius is sufficiently large to allow a sensor to locally analyze its neighbors movements. Conversely, SSD relaxes this assumption and sensors rely on position swaps to verify the correct behavior of their neighbors. Both allow legitimate sensors to determine the malicious nature of their neighbors by observing their movements. We formally proved that SecureVor is able to defeat the OM attack, and that both SecureVor and SSD have a guaranteed termination. Additionally, we performed an extensive experimental analysis that confirmed that with these algorithms the network achieves its monitoring goals even in the presence of an attack, at the expense of a small overhead in terms of movements and deployment time.

In our future work, we will consider other aspects, such as:

- *temporal legitimate behavior*: we will study the performance of the proposed solution in the presence of an attacker that initially adopts a legitimate behavior according to the protocol of deployment. After a time window, the malicious sensors start to perform the attack to the network.

- *presence of obstacles*: we will consider a scenario in which there is the presences of obstacles of different shapes in the AoI. The major challenge in this scenario is to correctly distinguish the obstacles and the malicious sensors that aim to impede the natural deployment of the network.

- *deployment approaches*: we will study how this kind of attack shown in this part can influence the performance of other categories of deployment algorithms (Delaunay triangulations, Geometric Pattern etc.,).

Part III

Smart Grid

Introduction

The term grid is used for an electricity system that generally support operations such as electricity generation, electricity transmission, electricity distribution, and electricity control. A *smart grid (SG)*, also called smart electrical/power grid, intelligent grid, is an enhancement of the 20th century power grid. The traditional power grids are generally used to carry power from a few central generators to a large number of users or customers. In contrast, the SG uses two-way flows of electricity and information to create an automated and distributed advanced energy delivery network. By utilizing modern information technologies, the SG is capable of delivering power in more efficient ways and responding to wide ranging conditions and events. In fact, the SG is able to respond to events that occur anywhere in the grid, such as power generation, transmission, distribution, and consumption, and adopt the corresponding strategies. For instance, once a medium voltage transformer failure event occurs in the distribution grid, the SG may automatically change the power flow and recover the power delivery service. More specifically, the SG can be regarded as an electric system that uses information, two-way, cyber-secure communication technologies, and computational intelligence in an integrated fashion across electricity generation, transmission, distribution and consumption to achieve a system state that is safe, secure, reliable, resilient, efficient, and sustainable. In order to realize this new grid paradigm, the U.S. Energy Independence and Security Act of 2007 directed the National Institute of Standards and Technology (NIST) provided a conceptual model [70], as shown in Figure 3.10, which can be used as a reference for the various parts of the electric system where SG standardization works are taking place. This conceptual model divides the SG into seven domains. Each domain encompasses one or more SG actors, including devices, systems, or programs that make decisions and exchange information necessary for performing applications. A brief descriptions of the domains and actors are given in [70].

In this part, we focused on the *power grids contingency analysis* in Smart Grids to study the impact of potential component failures. For instance, the *N-1* contingency analysis refers to the ability to predict the behavior of the electrical grid in response to a single random failure. Predicting the behavior of the power grid after a failure is significantly relevant, since contingencies could results in cascading failures. Although these events occur at a low probability, cascading failures are critical issues in power systems operation since can evolve into large-scale blackouts. A report by the U.S. Executive Office of the President estimates that between 2003 and 2012, 679

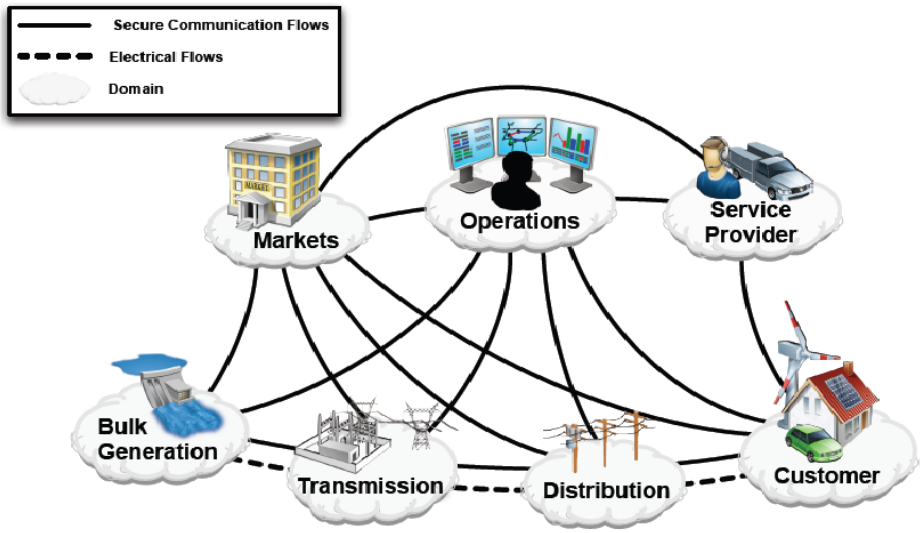


Figure 3.10: NIST Smart Grid Framework

large scale power outages occurred in the U.S., each affecting at least 50,000 customers [36, 18, 19]. The economical impact of cascading failures is also significant, as the costs range from 18 to 33 billion dollars per year [36].

The cause of such cascading failures is often the relative primitiveness of *contingency management*. On one hand, management often relies on the judgments of human operators, who decide on possible countermeasures based on their experience. On the other hand, such operators can only see the high-voltage transmission levels of the grid, with little outlook on the adjustable loads at the end-users' level. Although during natural disaster there could be a relationship between failures in the power grids and failures in communication networks, we focus the attention on the contingency analysis of the smart grids. In fact, this work particularly examines contingency cases where one or more of the system components are unexpectedly down but the system balance is still achieved due to the strict reliability criteria. Even when the system restores its balance after a contingency, however, there is a risk of cascading failures as with the 2003 North American blackout case [36]. In fact, other lines can approach their maximum limits, and eventually drive the system over the critical point beyond a stable state. Therefore, precautionary measures to avoid cascading failures are necessary.

This work proposes a novel framework to alleviate this type of risks by adjusting a large number of end-use loads. The assumption is that, for such emergency cases, curtailing some non-critical loads to prevent cascading failures yields greater aggregate utility than leaving the lights on and having cascading failures later. Therefore, in this work, all controlled loads, which

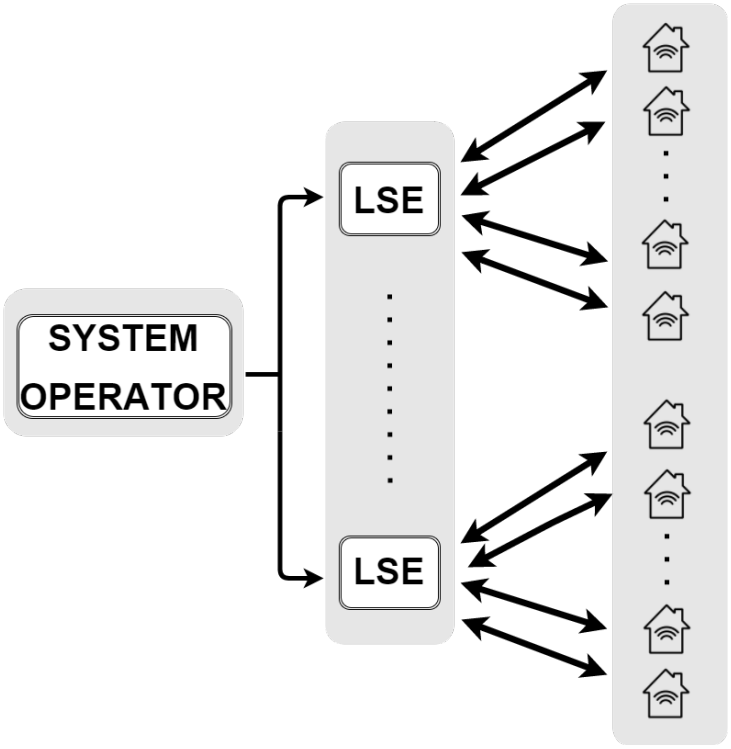


Figure 3.11: Overview of the proposed system.

exclude critical loads such as life-support devices, are assumed to be curtailable within a short time period, e.g., 1 hour. The end-user loads are controlled by smart devices, realized through the emerging paradigm of the Internet of Things [5, 91]. According to this paradigm, smart devices are equipped with communication, computation and storage capabilities, and they are connected to a *smart home management system* through wireless access points. Each smart device controls an appliance such as a space heater, an air conditioner, a refrigerator, etc.

The framework comprehensively involves the system operator, the load serving entities (LSEs), and the end-users' smart systems. The overview of the system is shown in Figure 3.11. The system operator prevents cascading failures by completing the following tasks after achieving the stable, but still potentially risky, state of the power system: 1) identify the components that violate the predetermined reliability criteria, and if there is any, 2) calculate the load adjustment at different locations (i.e., buses) to alleviate the additional stress in those particular components. Since it is assumed that the system is in balance, the total amount of load curtailment at each

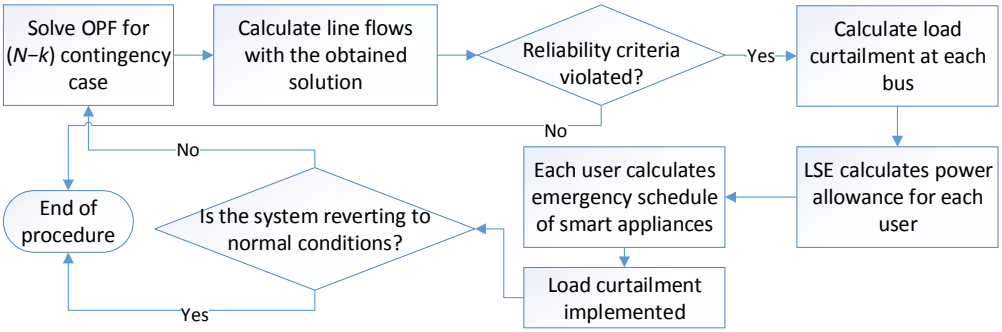


Figure 3.12: The flowchart of the contingency management framework

bus is efficiently calculated by a novel approach using the linearized network equation.

When an LSE is notified of a load curtailment amount at its load bus, it solves a mixed integer linear optimization problem that maximizes the aggregate utility, i.e. the sum of its end-users' *utility*. In this part *utility* is defined as a quantifiable measure of user satisfaction from using a certain appliance. To improve scalability of the LSE's problem, an approximated convex problem of the mixed integer optimization is solved, using an efficient heuristic based on regression techniques.

The solution to the LSE's problem is the individual load curtailments of the LSE's users. The smart home management system then calculates an *emergency schedule*, which defines the best set of appliances that the user is allowed to use. This schedule minimizes the impact of the curtailment on the user's habits, while satisfying the power allowance requested by the LSE. The calculation of the emergency schedule requires the knowledge of the future user interaction with appliances. To predict this interaction, the framework uses the WRAP (Welch-based Reactive Appliance Prediction) algorithm. WRAP uses smart devices to monitor the user habits and to predict the appliance usage following the contingency.After load curtailments, the system operator evaluates the system condition, and if the system is not reverting to the normal condition, the procedure is repeated from the beginning. The flowchart of the framework is shown in Figure 3.12.

Extensive simulations are performed on the IEEE New England 39-bus test system, using real power consumption datasets, to validate the benefits of this framework. The results show that the proposed method is effective in calculating the load curtailments needed after a contingency, ensuring that the lines operate within their capacity margins. Additionally, the WRAP algorithm achieves highly accurate predictions of the appliance usage. Fi-

nally, the regression-based heuristic performed by the LSEs closely matches the results achieved by solving the original mixed integer programming problem. As a result, the proposed framework is effective in keeping the system stable during contingencies, preventing cascading failures while maximizing the aggregate user utility.

The main contributions of this work are the following:

- A comprehensive framework for contingency management using smart appliances based on the paradigm of the Internet of Things;
- A novel and efficient method to enable system operators to calculate the load curtailment needed in order to keep the system in a safe state after a contingency;
- An efficient heuristic to distribute such curtailments across end-users;
- The Welch-based Reactive Appliance Prediction (WRAP) algorithm to predict utilization of each appliance by a user;
- Extensive simulations on realistic system settings to validate the proposed approach in managing contingencies while maximizing users' utility.
- This work is published on *IEEE Transactions on Smart Grid* (http://ieeexplore.ieee.org/document/7425266/)[30].

Related Works

There have been efforts in managing contingencies with adjustable demand. The work in [43] considers dispatch of load curtailment at the system-level operation alongside with generation, based on the bids submitted by the customers. Presumably these customers represent the load serving entities, but the dynamics or attributes of the individual load models are missing. In [28], demand response is used in place of spinning reserves to restore the frequency post contingencies. In [52], authors use demand response for efficient use of transformers during contingencies. Instead, [75] corrects voltages adjusting post contingency demands. The last three works are novel in terms of using demand to manage contingencies, but have different purposes from this work, where we alleviate congestions post contingencies.

There are also many works on predicting users' power consumption by appliance. Work [15] proposes an algorithm to identify the individual consumptions of residential appliances. It works at a coarser granularity since it is able to predict only the appliances future states (on/off) to compute the future energy load requested. In [16], authors describe an approach to predict daily energy consumption of large groups of customers, but they consider only electrical heating and cooling as appliances. In [54], user discomfort is minimized with a Q-learning algorithm, which computes the optimal set of appliances to switch off during the system peaks. However, this approach assumes that the importance of an appliance to a user is known in advance.

Our work takes a step further from the literature on users' demand and utility, and proposes an effective algorithm that exploits smart appliances to *predict* and maximize the users' utility using historical data. Another novel aspect of the proposed work is in modeling the system comprehensively from the power transmission grid all throughout the end-users equipped with smart home management systems. The objective is not only to manage contingencies at the system level, which existing literature has studied extensively, but to maximize the aggregate utility of all users, when a certain amount of capacity is requested from the system operator.

Chapter 4

Managing Contingencies In Smart Grids Via The Internet Of Things

In this chapter we study the problem of how to prevent cascading failures in the smart grid balancing the users' load. We present this problem from the point of view of each agent involved in the framework: the system operator, the load serving entities and the end-users. At the same time, we present and discuss the proposed solutions for each of them. Extensive simulations on real users' load traces and on real systems show the accuracy and the efficiency of the proposed framework. The chapter is organized as follows. Section 4.1 describes the problem of the system operator, while Sections 4.2 and 4.3 address the users' and LSE's problems, respectively. The simulation results are presented in Section 4.4, and Section 4.5 concludes this part.

4.1 The Problem of a System Operator

The objective of the system operator in general is to keep the system reliable at the least cost. After one or more lines failed, the power flows in other lines can approach their limits. Therefore, in this work the system operator's objective is to find the load curtailment that can alleviate the line flows of these additional lines at risk to prevent cascading failures.

The relationship between the active power flows in the lines and the active power injected into each bus can be linearized with a power transfer distribution factor (PTDF) matrix H. $HP = F$ where P is a vector of active power injection at each node except the slack bus, and F is a vector of active power flow in each line [88, 79]. Therefore, demand curtailment ΔP_D, defined as a vector with the adjustable demand buses as its components, that yields the line power flow difference ΔF can be calculated by solving

$$H_C C_D \Delta P_D = \Delta F_C \tag{4.1}$$

where ΔF_C is the line adjustment vector with only the congested lines selected from F, whose length is equal to N_L, number of congested lines. C_D is a bus-demand connection matrix with a dimension (the number of total buses in the system N_B)-by-(the number of demand buses N_D), whose element is 1 when the bus (row) is a demand bus (column) and 0 otherwise. H_C is extracted from H with only the rows of the congested lines, thus N_L-by-N_B. Usually since $N_D > N_L$, the solution to this equation can be obtained as $\Delta P_D = H^+ \Delta F$ where H^+ is the Moore-Penrose pseudoinverse of H.

This calculation gives a solution with the minimal norm among the many solutions of (4.1). Therefore, the sum of the solution, or the total system load adjustments, can be negative, resulting in the need for more generation to balance the supply and demand. In order to avoid this, the solution is

sought so that the load adjustments in the system sums up to zero, i.e.,

$$\sum_{n=1}^{N_D} \Delta P_D(n) = 0. \tag{4.2}$$

Concatenating (4.1) and (4.2) yields an augmented power flow equation

$$\begin{bmatrix} H \\ \mathbf{1}_{N_D}^T \end{bmatrix} \Delta P_D = \begin{bmatrix} \Delta F \\ 0 \end{bmatrix}, \text{ or } \tilde{H} \Delta P_D = \Delta \tilde{F} \tag{4.3}$$

where $\mathbf{1}_n$ denotes an n-length column vector that has 1 as all its elements. The solution can be obtained in the same way by solving $\Delta P_D = \tilde{H}^+ \Delta \tilde{F}$.

The solution to (4.3) can include negative load adjustments, which means that some load buses need to *increase* their consumption. If the system operator decides that this is unreasonable or if it is technically infeasible, then the system operator can take the nonnegative solution $\Delta P_D|_+$ where $\Delta P_D(l)|_+ = \max[0, \Delta P_D(l)]$ for all l, and resolve the power flows with this adjusted solution. It should be noted that the power flows with this solution may result in power flow adjustment smaller than the target ΔF_C. However, since the relationship between the load curtailment and the power flows is linear, the load curtailment solution can be simply scaled by the factor of the desirable power flow adjustment.

4.2 The Problem of a User

This section discusses the problem solved by the smart home energy management system of a user. As a contingency occurs, the system operator sends each LSE the load curtailment $\Delta P_D(l)$ for each bus l. Let $M_{\max}^l$ be the power allowance that the LSE is allotted for Bus l. Then the LSE distributes $M_{\max}^l$ across its users, calculating the individual power allowance $M_1^l, \ldots, M_{N_l}^l$ for each of its N_l users so that $\sum M_i^l = M_{\max}^l$. Finally, the users' smart home management systems schedule the smart appliances to be used. Since most variables and parameters for an LSE's problem are defined in the users' problem, to improve readability, the problem solved by the users is first presented, followed by how an LSE distributes $M_{\max}^l$ in Section 4.3.

In the proposed framework, a smart home has n smart devices $d_1, \ldots, d_n$. According to the paradigm of the Internet of Things, smart devices are equipped with communication, computation and storage capabilities. In particular, they are connected to the smart home management system through wireless access points deployed in the smart home. The framework exploits their computation and storage capabilities to monitor and learn the user

habits. Note that, critical appliances, such as life support medical devices, are not considered by the framework for load curtailment. Appliances are generally classified as *critical* and *flexible*. Critical appliances, such as life support machines, need to be always operational. On the contrary, flexible appliances can be turned off if needed. In the following, only flexible appliances are considered. The smart home management system identifies critical appliances, and excludes them from the load curtailment procedure. For each appliance, the framework defines a time-dependent *importance factor*, according to the user's usage preference and patterns, which may vary depending on the time of day, the season of year, and the user's habits. The goal is to use smart devices to learn the importance factors of the appliances for each user u during normal system conditions. To calculate such factors, the framework considers *time slots* $\tau_1, \tau_2, \ldots$ of arbitrary length, set to one hour in this work. Let $\lambda^u_{i,j} \in [0,1]$ be the fraction of time that user u uses the appliance i in time slot j. The importance factor $\gamma^u_{i,j}$ of appliance i is defined as:

$$\gamma^u_{i,j} = \frac{\lambda^u_{i,j}}{\sum_{h=1}^{n} \lambda^u_{h,j}}. \tag{4.4}$$

Therefore, $\gamma^u_{i,j}$ represents the relative usage time of appliance i with respect to the other appliances during time slot j.

The approach assumes that the importance factor $\gamma^u_{i,j}$ measures the contribution of appliance i to the utility of user u during the time slot τ_j. Therefore, given a set of appliances A, the utility that results from using such appliances in time slot j for user u is $\sum_{d_i \in A} \gamma^u_{i,j}$. As soon as the LSE informs the user u's smart home management system of the new power allowance M_u, an *emergency schedule* that determines the set of appliances that can be used is calculated so that it maximizes the *user utility*.

4.2.1 Optimal emergency schedule

Using the importance factors defined previously, the framework makes use of the following optimization problem to determine the emergency schedule of a user u. The problem is solved by the smart home management system, which receives a power allowance M_u from the LSE (the calculation of M_u is described in Section 4.3). The description considers a load curtailment during time slot τ. Let $x_i \in \{0,1\}$ be a decision variable, where $x_i = 1$ if appliance i is allowed to be used during the emergency schedule, and $x_i = 0$ otherwise. Additionally, let $e_1, \ldots, e_n$ be the maximum power rating of the appliances. For simplicity, the following discussion assumes the state of an

appliance is either ON or OFF. Then the optimal scheduling problem is:

$$\underset{x_i}{\text{maximize}} \sum_{i=1}^{n} \gamma_{i,\tau}^{u} x_i \tag{4.5}$$

$$\text{subject to} \sum_{i=1}^{n} x_i e_i \leq M_u \tag{4.6}$$

where the values of $\gamma_{i,\tau}^{u}$ are calculated according to (4.4), and are estimated as described in the next subsection.

This optimization problem is clearly NP-hard. However, since smart homes generally have a limited number of appliances, the problem can be solved in a short time optimally, or through standard heuristics [31]. The resulting schedule is enforced by the smart home management system, which restricts the use to only the selected appliances.

4.2.2 Learning Algorithm for Importance Factors

In order to solve (4.5), the values of time-dependent importance factors $\gamma_{i,\tau}^{u}$'s need to be known. However, since they represent the user's future behavior, they can only be predicted. This section describes the Welch-based Reactive Prediction (WRAP) algorithm, executed by the smart home management system. Note that these factors cannot be determined simply by looking at historical data, since the use of appliances may present short- and long-term variations. As an example, if a user is generally at home at a specific time slot, but if on a certain day he is not, then the use of appliances in that day is likely to significantly differ. Similarly, the user lifestyle may change during the year, presenting long-term variations. For these reasons, WRAP makes use of a statistical change detection mechanism based on the Welch's t-test [89] to predict the importance factors.

WRAP is based on the assumption that the fraction of time $\lambda_{i,j}^{u}$, during which user u uses appliance i in time slot τ_j, is distributed over multiple days as a Gaussian random variable. The means and standard deviations of $\lambda_{i,j}^{u}$'s may change over time. The results in Section 4.4 prove that this assumption enables accurate estimation of the importance factors and maximization of user utility.

The smart home management system keeps track of the historical usage of each appliance. In particular, for each time slot τ_j the system calculates the *historical mean* $\mu_{i,j}^{H}$ and the *historical variance* $\sigma_{i,j}^{H}$. At the end of each time slot, these historical values are updated with the newly observed values.

Short-term change detection

WRAP adopts a change detection mechanism based on the Welch's t-test [89], to achieve high accuracy and reactivity, i.e., the ability to react to changes in the usage pattern. In particular, the idea of detecting short-term changes is to verify if the most recent utilization of an appliance is *unusual* with respect to the historical data.

Consider an emergency period occurring at time period τ_j, hence the importance factors need to be predicted for τ_j. Let $< \mu_{i,j}^H, \sigma_{i,j}^H >$ be the historical distribution for appliance i during τ_j, and $< \mu_i^{W_S}, \sigma_i^{W_S} >$ be the distribution over only a recent time window W_S, e.g. the last 60 minutes. In order to detect if there is a change in user behavior, WRAP determines whether the distribution $< \mu_{i,j}^H, \sigma_{i,j}^H >$ and $< \mu_i^{W_S}, \sigma_i^{W_S} >$ belong to the same population (*null hypothesis*) or not (*alternative hypothesis*).

The Welch's t-test defines a parameter t, which depends on the two distributions [31]. WRAP performs the test for each appliance d_i and calculate the value of t_i as follows:

$$t_i = \frac{\mu_{i,j}^H - \mu_i^{W_S}}{\sqrt{\frac{\sigma_{i,j}^H}{n_H} + \frac{\sigma_i^{W_S}}{n_{W_S}}}} \tag{4.7}$$

where n_H and n_{W_S} are the numbers of samples used to calculate the historical distributions and the distributions over W_S, respectively. For each t_i it is possible to estimate the degree of freedom ν_i as follows [31]:

$$\nu_i \approx \frac{(\frac{\sigma_{i,j}^H}{n_H} + \frac{\sigma_i^{W_S}}{n_{W_S}})^2}{\frac{(\sigma_{i,j}^H)^2}{{n_H}^2(n_H-1)} + \frac{(\sigma_i^{W_S})^2}{(n_W)^2(n_{W_S}-1)}} \tag{4.8}$$

The test can verify if a change has occurred with a given probability. In particular, it is possible to determine if the alternative hypothesis is verified with probability α. To this purpose, given t_i and ν_i of appliance d_i, Student's t distribution tables give the value β_i, such that if $t_i > \beta_i$ then a change has occurred with probability α. WRAP has $O(1)$ complexity, since prediction, change detection and distributions update can be performed in constant time.

Prediction by the WRAP algorithm

WRAP exploits the property of Gaussian random variables that the minimum mean square error estimate is equal to its mean [47]. The actual mean used for the estimation of an appliance d_i can be either the historical mean,

or the mean in the recent time window W_S, depending on whether a change is detected for appliance d_i or not. As soon as the LSE alerts the smart home management system of a new power allowance, WRAP verifies whether a short-term change has occurred for each appliance d_i. If no change is detected for d_i for time slot τ_j, the algorithm uses the historical mean, i.e., $\lambda_{i,j} = \mu^H_{i,j}$. If otherwise a change is detected, then the algorithm uses the distribution of the most recent time window W_S, i.e., $\lambda_{i,j} = \mu_i^{W_S}$. Given the values of $\lambda_{i,j}$ for each appliance, (4.4) is used to calculate the importance factors $\gamma_{i,j}$, which are then used to calculate the emergency schedule described in Section 4.2.1.

Long-term change detection

In order to detect long-term changes in the user behavior, WRAP uses a similar approach based on Welch's t-test. In particular, we compare the historical distribution $< \mu^H_{i,j}, \sigma^H_{i,j} >$ with the recent time window distribution $< \mu_i^{W_L}, \sigma_i^{W_L} >$. W_L can be set to several weeks. This test is performed periodically, e.g. daily or weekly. If a change is detected, the historical distribution no longer represents the current usage pattern of an appliance, and hence the new recent set of samples in W_L constitutes the historical distribution.

4.3 The Problem of a Load Serving Entity

This section describes how LSEs calculate the power allowance M_u for each user u, given the power allowance that resulted from the load curtailment at Bus l requested by the system operator $\Delta P_D(l)$. The following description focuses on a single LSE and, without loss of generality, assumes one LSE is responsible for all users at one bus, to drop the LSE index l. Let $M_{\max}$ be the maximum power allowance resulting from the curtailment for the considered LSE. Intuitively, not all users need the same level of power, since the same capacity may result in different values of utility for different users.

The goal of the LSE is to calculate the individual power allowance M_u for each user $u = 1, \ldots, N$, such that $\sum_u M_u = M_{\max}$, and the *aggregate user utility* is maximized. Aggregate user utility is defined as the sum of the utilities of all users served by the LSE. To this purpose, after the LSE receives a load curtailment request, the LSE inquires its users' smart home management systems and receives from them the importance factors for the current time slot predicted by WRAP. The LSE then needs to solve the

following optimization problem for time slot τ:

$$\underset{x_i, M_u}{\text{maximize}} \sum_{u=1}^{N} \sum_{i=1}^{n_u} \gamma_{i,\tau}^{u} x_i^u \tag{4.9}$$

$$\text{subject to} \sum_{i=1}^{n_u} x_i^u e_i^u \leq M_u \;\forall\; u; \qquad \sum_{u=1}^{N} M_u = M_{\max} \tag{4.10}$$

where the user u has n_u appliances, e_i^u is the maximum power rating of appliance d_i^u, and $x_i^u = 1$ when u is allowed to use d_i^u, and 0 otherwise. Here, unlike the user's problem in (4.5), the power allowances M_u are variables to be determined, and the importance factors $\gamma_{i,\tau}^u$ are given by the smart home management systems.

Although the problem above is similar to the emergency schedule optimization in (4.5), it can suffer severe scalability issues because now the dimension of the problem is multiplied by a large number of users. For this reason, the framework exploits the following *regression based heuristic*, which relaxes the problem into a convex optimization.

4.3.1 Regression-based heuristic

Consider a specific user u, and let $M_{\max}^u$ be the maximum consumption that u can generate if he utilizes all his appliances at the same time, i.e. $M_{\max}^u = \sum_{i=1}^{n_u} e_i$. Setting $M_u = M_{\max}^u$ would obviously maximize the utility of user u. Since the LSE is aware of the importance factors $\gamma_{i,j}^u$ for each appliance i, it is also able to solve the optimization problem in (4.5). In fact, according to the heuristic, the LSE solves K instances of the problem for user u using different power allowance levels. At the k-th instance, it sets the power allowance to $\alpha_k M_{\max}^u$, where $\alpha_k \in [0, 1]$ and it is increased at each instance. Note that solving these problems, although NP-hard, is feasible thanks to the limited number of appliances per user.

Let $\delta_1, \ldots, \delta_K$ be the optimal solutions of such instances, where δ_k is the solution for α_k. The pairs (α_k, δ_k), $k = 1, \ldots, K$ are used to infer a continuous function $H_u : \mathbb{R}^+ \to [0, 1]$, which relates the power allowance M_u to the utility achieved by user u. A *regression* technique is adopted to approximate this function. Since $H_u(\cdot)$ is monotonically increasing, power law regressions can be used, that is $H(\cdot)$ can be approximated as $H(M) = \alpha M^\beta$, where $\alpha, \beta \in \mathbb{R}$ [44], and $\beta \leq 1$ to ensure that $H_u(\cdot)$ is concave.

The LSE calculates the functions $H_u(\cdot)$ for each of the N users, and solves the following optimization problem:

$$\underset{M_u}{\text{maximize}} \sum_{u=1}^{N} H_u(M_u) \tag{4.11}$$

$$\text{subject to} \sum_{u=1}^{N} M_u = M_{\max}; \qquad M_u \leq M_{\max}^u \ \forall u \tag{4.12}$$

The problem is a relaxation in the continuous domain of problem (4.9), and it returns an assignment of the maximum loads M_u to the users. Note that the problem does not solve for the decision variables x_u^i explicitly, which is the key for the reduction in complexity. Since the objective function is concave, the problem can be solved using standard convex optimization techniques [46].

Section 4.4.3 shows that the regression-based heuristic coupled with WRAP are able to provide a aggregate user utility close to what would be achieved by solving the NP-hard problem in (4.9) with the perfect knowledge of the importance factors of the users' appliances.

4.4 Simulation Results

The framework is tested on the IEEE 39-bus system modeled after the ISO New England system with 10 generators, 46 lines, and 21 nonzero load buses [4, 53]. The one-line diagram of this system is shown in Figure 4.1. All nonzero load buses are assumed to have capability of load curtailment with the Internet of Things technologies, but not all these buses are necessarily subjected to a load curtailment in contingency cases. We use synthetic (randomly generated) and real traces (dataset) to model user appliance usage. Real traces are taken from the data repository Tracebase [76], which collects the power consumption of various electrical appliances, with a resolution of several samples per second. Some of the considered appliances and their maximum power ratings are listed in Table 4.2.

4.4.1 The system operator's problem

The following simulations concern contingencies where a single line has failed. First, the one-component failure cases were identified by running the DC optimal power flow (OPF) problems with each line taken out. As long as the line failure did not isolate a generator bus with the rest of the system, the solution existed, and only theses cases were studied in this work. When a

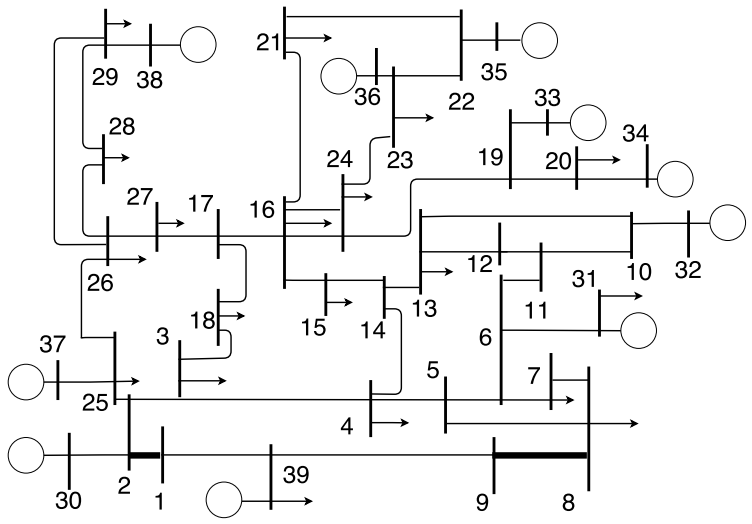


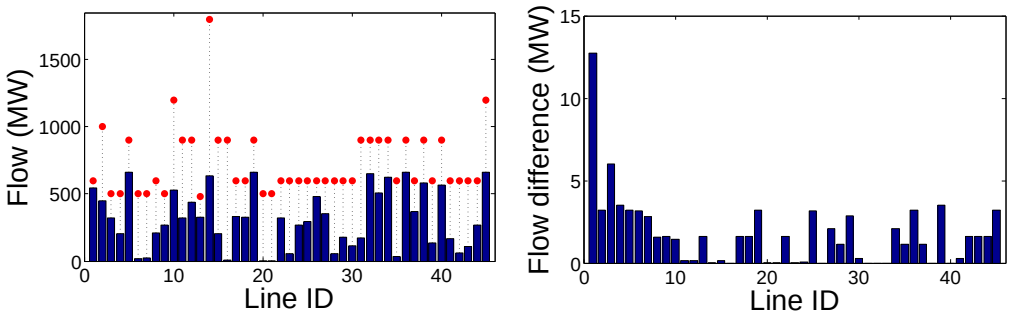
Figure 4.1: The IEEE 39-bus system used in the simulations.

generator is isolated as a result of a line failure, it changes the topology of the system, which changes the network matrix and the PTDF matrix H of the system.

This work focuses on the cases where the power flow solution exists even after one line failed. The case where Line 16 that connects Buses 8 and 9 (marked as a thick line in Figure 4.1) has failed is presented. The resulting active power flows in the lines as a result of the DC OPF for this case are shown in Figure 4.2(a).

As can be seen from the figure, Line 1 (between Buses 1 and 2, marked as a thick line in Figure 4.1) resulted in a power flow very close to the limit, and the system operator may decide to reduce this line flow. The system operator can set the criterion in advance for which s/he decides to take actions and apply load curtailment. Once the criterion has been violated and load curtailment is deemed appropriate, then ΔP_D is calculated for all the adjustable load buses, as described in Section 4.1. In this work 10% margin of the line MVA rating is used as the reliability criterion. The difference of the absolute power flows before and after the load curtailment is depicted in Figure 4.2(b).

The total amount of system load to be curtailed as a result is 16.04 MW, in order to reduce 6.37 MW of power flow in Line 1. Some generators were able to reduce their output due to this curtailment, and the output reduction from each of Generators 1, 3, 6, 9, and 10 was 3.25 MW, with the other generators' output unchanged. The amounts of load curtailment by bus is shown in Table 4.1. All the other nonzero load buses that are not shown in



(a) (b)

Figure 4.2: (a) Active power flows in the lines after Line 16 is out with the red markers denoting the line flow limits; (b) Active power flow difference in all lines before and after load curtailments

Table 4.1: Load curtailment by bus

Bus ID	Curtailment (MW)	Bus ID	Curtailment (MW)
1	15.9	8	0.00492
4	0.0246	15	0.0355
7	0.000189	18	0.0315

the table did not have any curtailment. The other single-line failure cases yielded comparable results.

4.4.2 The users' problem

This subsection first studies the prediction accuracy of the WRAP algorithm. Subsequently, it analyzes the user utility achieved by the emergency schedule calculated with the prediction provided by WRAP.

Accuracy of WRAP

Synthetic traces are first used for simulations since changes in the data pattern can be manipulated at specific time instants. This shows the benefits of WRAP's change detection mechanisms in a controlled setting. Then the performance in real settings is analyzed using the real traces from the data repository Tracebase [76].

Synthetic Traces
To generate synthetic traces, the length of each time slot is set to 1 hour.

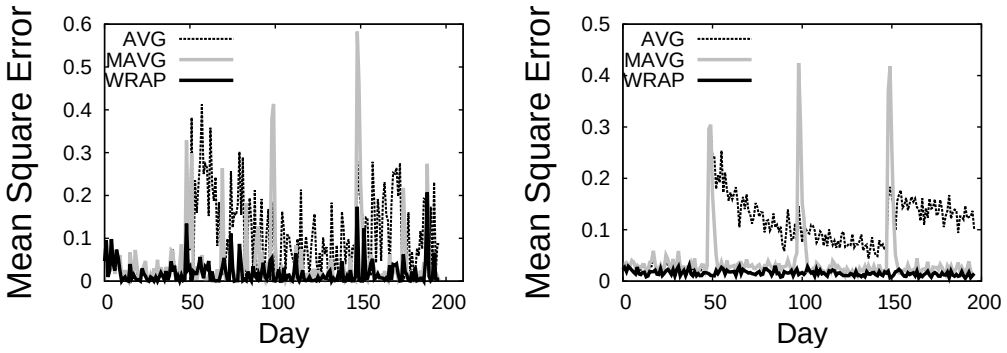


Figure 4.3: Prediction error: single time slot (a), average of all time slots (b).

For each appliance d_i and time slot j, the utilization $\lambda_{i,j} \in [0, 1]$ is randomly generated. This represents the fraction of time that d_i is utilized on average during time slot j. Then, to simulate the variability of the user behavior, for each day the actual utilization at τ_j is generated using a Gaussian's distribution with mean $\lambda_{i,j}$ and variance σ.

The traces consider 200 days of observations, simulating a change in the user behavior by selecting a new value of $\lambda_{i,j}$, for each appliance d_i, every 50 days. Additionally, $\sigma = 0.2$, which according to the experiments, well approximates the realistic variability in user habits.

Since WRAP considers each appliance independently, the following experiment focuses on a single device. In particular, WRAP predicts the utilization $\lambda_{i,j}$ at the time slot j of the $(k+1)$-th day, using the values of $\lambda_{i,j}$ generated for the same time slot of the previous k days. The results show the accuracy of the predicted values in terms of the mean square error with respect to the actual values in the traces.

WRAP is compared with two other standard prediction techniques. *Average* (AVG): this approach predicts the next $(k+1)$-th value of $\lambda_{i,j}$ as the average of all the previous k values. *Moving Average* (MAVG): this scheme predicts the next $(k+1)$-th value of $\lambda_{i,j}$ as the average of the last observed w values of $\lambda_{i,j}$, where w is the size of the time window. To ensure the reactivity of the approach, we set w equal to 5 days.

Figure 4.3 (a) shows the results for the three approaches for a single time slot, while Figure 4.3 (b) shows the average across all the time slots of a day. Before the first change occurs, all the three approaches perform similarly, since in the synthetic traces, the appliance utilization is drawn from the same distribution. However, after 50 days, a change in the usage pattern

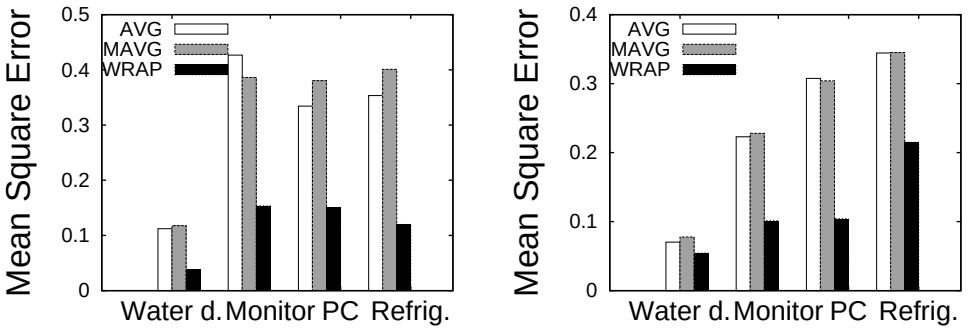


Figure 4.4: Prediction error: single day (a), average over 30 days (b).

occurs. The error of AVG suddenly increases, since it keeps using all the previous dataset. MAVG, instead, is more reactive, but it still incurs high errors for a few days after the change. Additionally, it often overreacts to the fluctuation in user appliance utilization during stationary periods.

WRAP is able to promptly react to the changes and achieves significantly lower error than the other approaches, thanks to the short- and long-term change detection mechanisms. In fact, as soon as a change occurs, the short-term mechanism detects the change and predicts using only the most recent observed values. When the change in the user habits persists, the long-term mechanism eventually discards the former knowledge and only considers the observed values after the change.

Real Traces

These experiments consider as appliances PC, refrigerator, monitor, and water dispenser, from the data repository Tracebase [76]. Note that these are the only appliances for which the repository provides at least 30 days of data. Similar experiments are performed as with the synthetic traces, in which k days of observation are used to predict the $(k+1)^{\text{th}}$.

Figure 4.4(a) focuses on the 17^{th} day and it shows the performance of the approaches averaging the mean square errors across all the time slots of that day. The results show that for some appliances, such as the water dispenser, all approaches incur in a low estimation error due to the regular usage pattern. Differently, for other appliances, the results for AVG and MAVG strictly depend on the considered appliance. In particular, AVG outperforms MAVG for appliances, such as refrigerator and PC, which have a stationary utilization pattern, with minor short term variations. On the contrary, for appliances with non-stationary pattern, such as monitor, AVG is worse than MAVG, since the most recent days are more representatives of

the future utilization. WRAP, always achieves the lowest error with respect to the other approaches, thanks to its adaptability to short and long term variations. Figure 4.4(b) shows the average error over 30 days of predictions. The results confirm that our approach achieves the best performance. Note that, the difference between AVG and MAVG is smoothed by averaging over several days.

User utility of emergency schedule

This section compares WRAP, AVG, and MAVG in terms of the individual user utility achieved by the corresponding emergency schedule. The emergency schedule defines the appliances that are allowed to be ON, and it is calculated by solving the optimization problem in (4.5) with the predicted importance factors and power allowance given by the LSE. The optimal schedule (OPT) is also shown for comparison, which is calculated by solving the optimization problem with the actual importance factors, assuming perfect knowledge of the user future behavior. The user utility of a schedule is calculated as the sum of the actual importance factors of the appliances allowed by the schedule.

The experiments consider real traces for 12 appliances. A contingency occurs on Day 16, and the previous 15 days are used for the prediction. Recall that, as discussed in Section 4.2, given a power allowance M_u for user u, the emergency schedule is a set of appliances that u is allowed to use, and it is calculated by solving the optimization problem in (4.5). The user utility is defined as the sum of the actual importance factors of the appliances allowed by the schedule. Therefore, more accurate predictions result in higher user utility. Figures 4.5(a) and (b) show the utility of a single user for time slots 11 a.m. and 2 p.m., respectively, under different power allowances given by the LSE (x-axis). The accuracy of WRAP allows to perform very close to the optimal, unlike the other methods. Note that only OPT yields a monotonically increasing utility with respect to the power allowance. This is because of the inaccuracy in predicting the importance factors by the other methods.

4.4.3 The LSE's problem

As described in Section 4.3, an LSE receives a load curtailment from the system operator, and it calculates the power allowance for its users to maximize the aggregate user utility. The LSE executes the regression-based heuristic to efficiently approximate the optimal solution of (4.9). This section studies the accuracy of the approximation provided by the heuristic. The experi-

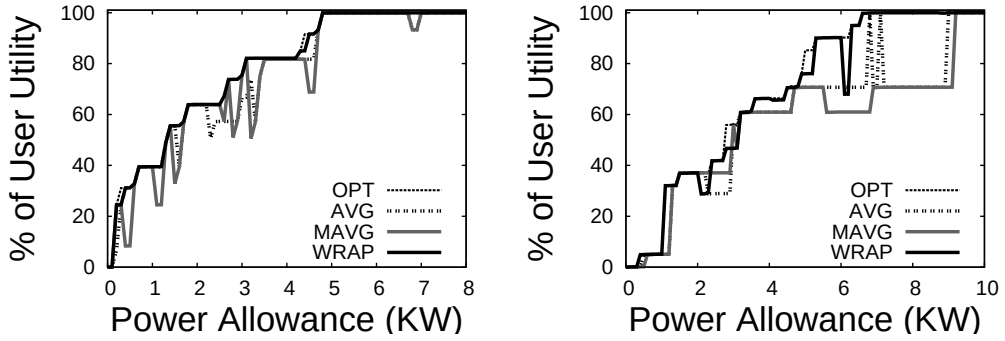


Figure 4.5: Utility of a single user for time slot 11 a.m. (a) and 2 p.m. (b)

ments consider 8,000 users with 12 appliances each, using the real traces. Since the traces do not provide data for multiple users, the available data are replicated to represent the number of users in the simulations.

Figure 4.6 shows the aggregate user utility (i.e. the sum of individual users' utilities), expressed as the percentage of maximum utility achievable with no curtailment. The experiments show the performance of the regression heuristic (WRAP-Reg) given varying power allowances to the LSE (x-axis). The results are compared with the other two solutions: the optimal solution (OPT) assuming perfect knowledge of the future user behavior, which is the solution to the NP-hard optimization problem defined in (4.9); and an approximated solution (WRAP-OPT) where the LSE solves the same NP-hard problem but uses WRAP to predict the importance factors.

WRAP-OPT well approximates the optimal solution, again validating the accuracy of the prediction algorithm. However, this approach still requires to solve an NP-hard problem, and is thus not applicable in scenario where the LSE has a large number of users. Instead, WRAP-Reg achieves similar results close to the optimal, with significantly lower complexity. It should be noted that different prediction techniques would only impact the aggregate user utility, and not the load curtailment quantities at the system level.

Now the specific case of the IEEE 39-bus system presented in Section 4.4.1 is considered. Recall that in this case scenario, Line 16 fails. The system operator, to prevent a cascading failure, notifies the LSE at Bus 1 that a curtailment of 15.9 MW is needed. It is assumed that the LSE's users' demand was 40 MW before the curtailment, therefore the power allowance at the LSE after curtailment is 24.1 MW.

The experiments compare OPT, WRAP-Reg and MAVG in this scenario to distribute the 24.1 MW to the users. The results of AVG are omitted

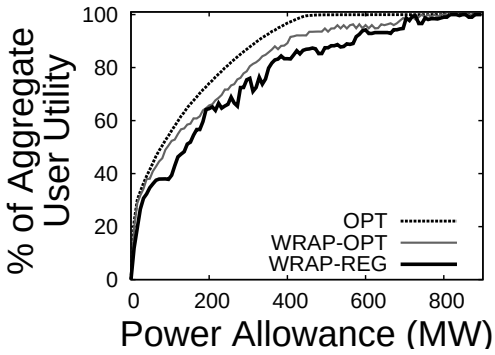


Figure 4.6: Aggregate utility achieved by all methods with 8000 users

since it performs similar to MAVG. To calculate the power allowances M_u's, OPT optimally solves (4.9), while MAVG evenly distributes the power among users, i.e. $M_u = M_{\max}/N \ \forall u$. To calculate the emergency schedule, OPT uses the actual importance factors, while MAVG uses the factors predicted with this strategy. WRAP-Reg adopts WRAP to predict the importance factors and the regression heuristic to distribute the load.

Table 4.2 shows the solution of the LSE problem, and the emergency schedules, for a specific user affected by the curtailment of the LSE at Bus 1. The LSE gives a power allowance to the user of 3,273 W under OPT, 3,267 W under Wrap-Reg, and 2,750 W under MAVG. The corresponding utility is 99.93%, 99.93% and 33.17%, respectively. Note that the goal of the framework is to maximize the *aggregate* user utility, not necessarily the utility of an individual user. In this scenario, OPT achieves 83.1% aggregate utility, while WRAP-Reg 73.6% and MAVG 47%. The inaccuracy of MAVG in estimating the importance factors negatively affects both the power allowance distribution and the emergency schedule, and ultimately the aggregate user utility. Conversely, the high accuracy of WRAP-Reg is able to achieve only 10% less utility than OPT, which however assumes perfect knowledge of the future users' behavior and has higher complexity.

4.5 Conclusions

In this chapter we study for the first time how to prevent large-blackout in smart grids where one or more of the system components fail. In particular, we proposed a framework for contingency management that comprehensively involves the system operator, the LSEs and the end-users. The framework enables the system operator to prevent subsequent failures by relieving lines possibly overloaded after the contingency. This is achieved through flexible

Table 4.2: An example of the appliance schedule of a single user

Appliance	Power rating (W)	OPT (3273 W)	WRAP (3267 W)	MAVG (2750 W)
Coffee maker	1305	ON	ON	OFF
Monitor	106	ON	ON	OFF
PC	166	OFF	OFF	ON
Refrigerator	1075	ON	ON	ON
Dish washer	2132	OFF	OFF	OFF
Water dispenser	360	ON	ON	OFF
User utility	-	99.64%	99.59%	33.17%

loads at the user level realized with the emerging paradigm of the Internet of Things. The framework provides efficient algorithmic solutions to: 1) determine the curtailment at each bus, 2) calculate the resulting power allowance for each user and, 3) predict the user's near-future behavior to minimize the impact of the curtailment on the user utility. Results on the New England 39-bus test system, using real traces, show that the framework is effective in keeping lines within their capacity margins, with minimal impact on the user utility.

Conclusions

In this dissertation we have studied some critical and emergency scenarios of different computer networks. We have highlighted several dangerous and unpredictable environmental conditions that are able to seriously compromise the safety of network applications in modern society. In particular, we focused on the vulnerabilities of the network communications, wireless sensor networks and smart grids.

In the first Part of this thesis, we analyzed the detrimental effects of a massive disruption on a communication network. Since our society heavily depends on communication networks to support critical services, it is important that such infrastructures are repaired quickly. In this emergency scenario, we focus on the problem of how efficiently restoring sufficient resources in a communications network to support the demand of mission critical services (e.g. government offices, police stations, fire stations, power plants, hospitals etc.) after a large scale disruption. In Chapter 1, we modeled the *recovery problem (MinR)* that aims to recover the damaged infrastructure in order to minimize the cost of the recovery actions. We formally proved that the problem is NP-hard and through experimental simulations showed that it may require up to 27 hours to find the optimal solution on a relative small size network. In order to approximate a solution to the MinR problem, we proposed an original and polynomial time heuristic called *Iterative Split and Prune* (ISP) to recover the network efficiently with a solution close to the optimal. To select the node to be repaired, we introduced a new notion of betweenness centrality, called *demand centrality*, to relate the importance of a node with respect to the demand flows. ISP minimizes the repairs by concentrating flows towards the areas of the network already repaired and it *prunes* the demand flows which can be routed on the currently repaired network. We proved some properties of ISP algorithm, such as the correctness, the time complexity and the termination. Finally, experimental results shown that ISP performs very close to the optimal in terms of number of elements repaired and outperforms previous heuristic approaches.

However, all the approaches to MinR, even ISP, assume to have a perfect knowledge of the disrupted area, i.e. they need to know the exact sets of nodes and failed links. Since this information is not always available, a complete and detailed damage assessment could taken long time for extensive monitoring and local inspections. Due to the emergency scenario, it is fundamental that recovery interventions start as soon as possible even if knowledge of the damage extension is incomplete. For this purpose, in Chapter 2, we studied the problem of *progressive restoration* of the mission critical services in condition of partial knowledge. In this new scenario, for the first time, we formulated the problem of Progressive Damage Assessment and network Recovery (PDAR), which aims at progressively restoring critical services in the

shortest possible time, under constraints on the availability of recovery resources and partial knowledge on the disruption. Since the PDAR problem is proven to be NP-hard, we proposed a novel polynomial time algorithm called Centrality based Damage Assessment and Restoration (CeDAR), which dynamically schedules repair interventions, local inspections and remote probing of network components, with the objective to restore critical services in the shortest possible time with efficient use of recovery resources. CeDAR restores critical demand flows iteratively by planning repair schedules which are based on the current global view of the network, maximizing at the same time the accumulative service flow during the recovery process. As did for ISP, we proven the correctness, the time complexity and the termination properties of CeDAR. To compare the performance of CeDAR, we extended two previous approaches to work in a scenario of incomplete knowledge. Through extensive simulations, we shown that CeDAR recovers the network with minimum cost of repairs, minimum number of local inspections to discover the network's status and with the highest flow restored over time, compared to the other approaches in all the experimental scenarios.

In the second Part, we moved the attention to the field of the wireless sensor networks and on their application to monitor an area of interest. In particular, we focused on the vulnerabilities of the deployment algorithms for Mobile Wireless Sensors Network (MWSN) based on Voronoi diagrams to coordinate mobile sensors and guide their movements. Recents works, show how an attacker can easily prevent the network from achieving its coverage goals by taking control of few nodes. The first contribution to this problem is to give a geometric characterization of possible attack configurations, proving that a simple attack consisting of a barrier of few compromised sensors can severely reduce network coverage creating a non monitored area subject to malicious actions by an attacker. Based on this analysis, we propose a new secure deployment algorithm named SSD (Secure Swap Deployment). This algorithm allows a sensor to detect compromised nodes by analyzing their movements, under a different and complementary operative settings with respect to a previous proposed solution named SecureVor [10]. We shown that the proposed algorithm is effective in defeating a barrier attack, achieving the total coverage of the AoI and has guaranteed termination. We performed extensive simulations showing that SecureVor and SSD have better robustness and flexibility, excellent coverage capabilities and deployment time, even in the presence of an attack.

In the last Part III, we studied the detrimental effect of systems failures in the smart grid. Since these failures could lead to a cascading failures phenomenon and could induce a large-scale blackout, we have shown how much important is to perform preventive actions. We proposed a novel framework

to alleviate this type of risks by adjusting a large number of end-use loads through the paradigm of the Internet of Things. Differently from previous works, for the first time, we designed a framework that include all the domains involved in this scenario: the System Operator (SO), the load serving entities (LSEs), and the end-users' smart systems. From the point of view of the system operator, we formulated a novel approach for the linearized network equation to compute the load curtailment needed to prevent the cascading failures. When an LSE is notified of a load curtailment amount, it estimates the individual load curtailments of each users, in order to maximize end-users' satisfaction. Finally, the smart home management system calculates the best set of appliances that the user is allowed to use and that minimizes the impact of the curtailment on the user's habits, while satisfying the power allowance requested by the LSE. To predict the future user interaction with appliances we developed the WRAP (Welch-based Reactive Appliance Prediction) algorithm. The results shown that the proposed framework is effective in keeping the system stable during contingencies, preventing cascading failures while maximizing the aggregate user utility.

Acknowledgements

There are many people that I would like to thank, since they have contributed to achieve this important target.
First of all, my acknowledgments go to my advisor Prof. Novella Bartolini, that guided me through these years. As advisor, she allowed me to understand the world of research through her expert and wise guide, supporting me, correcting me, pushing me to go beyond one's limits and teaching me that you can always improve. Her influence on my life goes beyond the academic role as advisor that she did during my Phd. In fact, in addition to what I mentioned above, she represents an example of strength, intelligence, determination, selflessness, kindness and congeniality. With her, I found a special friend, and I know that everything that she did, it was for my own good. In a person's life, there are few people that strongly influence your personality, and she is one of them.
A special thank goes to Prof. Simone Silvestri of Missouri University of Science and Technology, that helped me through these years allowing me to grow up as researcher by stimulating my ideas.
I wish to thank the Prof. Thomas F. La Porta of Pennsylvania State University and Prof. Sajal K. Das of Missouri University of Science and Technology for the opportunity and the honor to work with them and be part of their labs.
Special thanks are due to everyone who have always been there for me. In particular. I want to express my gratitude to my family who have given me the opportunity to complete this path, supporting my choices. I wish to thank my aunt Rita, that has been a second mom for all this time and without her patience and generosity nothing it would have been possible. Thanks to my relatives and friends to have shared precious moments together. All these people never missed the opportunity to show their love and their presence in the hardest and lonely moments.

Bibliography

[1] Gurobi. *"http://gurobi.com".Last accessed on November 2015.*

[2] The internet topology zoo. *"http://www.topology-zoo.org/".Last accessed on May 2015.*

[3] A. Arab, A. Khodaei, Z. Han, and S. K. Khator. Proactive recovery of electric power assets for resiliency enhancement. *IEEE Access*, 3:99–109, 2015.

[4] T. Athay, R. Podmore, and S. Virmani. A practical method for the direct analysis of transient stability. *Power Apparatus and Systems, IEEE Transactions on*, (2):573–584, 1979.

[5] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Elsevier Computer networks*, 54(15):2787–2805, 2010.

[6] B. Awerbuch and R. Khandekar. Greedy distributed optimization of multi-commodity flows. *Distributed Computing*, 21(5):317–329, 2009.

[7] S. R. Bailey. Disaster preparedness and resiliency. In *Guide to Reliable Internet Services and Applications*, pages 517–543. Springer, 2010.

[8] N. Bartolini, G. Bongiovanni, T. La Porta, and S. Silvestri. On the security vulnerabilities of the virtual force approach to mobile sensor deployment. *IEEE INFOCOM*, 2013.

[9] N. Bartolini, G. Bongiovanni, T. La Porta, and S. Silvestri. On the vulnerabilities of the virtual force approach to mobile sensor deployment. *IEEE Trans. on Mobile Computing*, 13(11):2592–2605, 2014.

[10] N. Bartolini, G. Bongiovanni, T. L. Porta, S. Silvestri, and F. Vincenti. Voronoi-based deployment of mobile sensors in the face of adversaries. In *Communications (ICC), 2014 IEEE International Conference on*, pages 532–537. IEEE, 2014.

[11] N. Bartolini, T. Calamoneri, E. G. Fusco, A. Massini, and S. Silvestri. Push & pull: autonomous deployment of mobile sensors for a complete coverage. *Wireless Networks*, 16(3):607–625, 2010.

[12] N. Bartolini, T. Calamoneri, T. La Porta, and S. Silvestri. Autonomous deployment of heterogeneous mobile sensors. *IEEE Trans. on Mobile Computing*, 10(6):753 –766, 2011.

[13] N. Bartolini, S. Ciavarella, T. La Porta, and S. Silvestri. Network recovery after massive failures. *IEEE DSN*, June 2016.

[14] N. Bartolini, S. Ciavarella, S. Silvestri, and T. L. Porta. On the vulnerabilities of voronoi-based approaches to mobile sensor deployment. *Transactions on Mobile Computing, IEEE*, 15(12):3114–3128, 2016.

[15] K. Basu, V. Debusschere, and S. Bacha. Residential appliance identification and future usage prediction from smart meter. In *IEEE IECON*, 2013.

[16] K. Basu, V. Debusschere, and S. Bacha. Residential appliance identification and future usage prediction from smart meter. In *IEEE IECON*, 2013.

[17] T. Beran, R. B. Langley, S. B. Bisnath, and L. Serrano. High-accuracy point positioning with low-cost gps receivers. *Navigation*, 54(1):53–63, 2007.

[18] U.S.-Canada Power System Outage Task Force. Final report on the august 14, 2003 blackout in the United States and Canada: Causes and recommendations. Technical report, Natural Resources Canada and U.S. Department of Energy, April 2004.

[19] U.S.-Canada Power System Outage Task Force. Final report on the implementation of the task force recommendations. Technical report, Natural Resources Canada and U.S. Department of Energy, September 2006.

[20] P. Bonacich. Power and centrality: A family of measures. *American journal of sociology*, pages 1170–1182, 1987.

[21] R. Bowden, H. X. Nguyen, N. Falkner, S. Knight, and M. Roughan. Planarity of data networks. In *Teletraffic Congress (ITC), 2011 23rd International*, pages 254–261. IEEE, 2011.

[22] U. Brandes. A faster algorithm for betweenness centrality*. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[23] L. J. CAIDA. Ca,cooperative association for internet data analysis (caida),.

[24] S. Capkun and J.-P. Hubaux. Secure positioning of wireless devices with application to sensor networks. *IEEE INFOCOM 2005*, 3:1917–1928, 2005.

[25] S. Capkun, K. Rasmussen, M. Cagalj, and M. Srivastava. Secure location verification with hidden and mobile base stations. *IEEE Trans. on Mobile Computing*, 7(4):470–483, 2008.

[26] A. Chakrabarti, L. Fleischer, and C. Weibel. When the cut condition is enough: a complete characterization for multiflow problems in series-parallel networks. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 19–26. ACM, 2012.

[27] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. *IEEE Symposium on Security and Privacy*, 2003.

[28] L.-R. Chang-Chien, L. N. An, T.-W. Lin, and W.-J. Lee. Incorporating demand response with spinning reserve to realize an adaptive frequency restoration plan for system contingencies. *Smart Grid, IEEE Transactions on*, 3(3):1145–1153, 2012.

[29] S. Ciavarella, N. Bartolini, H. Khamfroush, and T. L. Porta. Progressive damage assessment and network recovery after massive failures. *IEEE Proceedings of the International Conference on Computer Communications (IEEE INFOCOM 2017)*.

[30] S. Ciavarella, J. Joo, and S.Silvestri. Managing contingencies in smart grids via the internet of things. *Transactions on Smart Grid, IEEE*, 7(4):2134–2141, 2016.

[31] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[32] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. Computational geometry: Algorithms and applications. *Springer-Verlag Berlin Heidelberg*, 2008.

[33] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. *IEEE INFOCOM*, 2004.

[34] P. K. Dutta, A. K. Arora, and S. B. Bibyk. Towards radar-enabled sensor networks. *ACM IPSN*, pages 467–474, 2006.

[35] P. ERDdS and A. R&WI. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[36] Executive Office of the President. Economic benefits of increasing electric grid resilience to weather outages. 2013.

[37] S. Ferdousi, F. Dikbiyik, M. Tornatore, and B. Mukherjee. Progressive datacenter recovery over optical core networks after a large-scale disaster. *IEEE DRCN*, 2016.

[38] L. Fleischer, J. Könemann, S. Leonardi, and G. Schäfer. Simple cost sharing schemes for multicommodity rent-or-buy and stochastic steiner tree. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 663–670. ACM, 2006.

[39] R. J. Fontana, E. Richley, and J. Barney. Commercialization of an ultra wideband precision asset location system. *IEEE UWST*, pages 369–373, 2003.

[40] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[41] M. Garetto, M. Gribaudo, C.-F. Chiasserini, and E. Leonardi. A distributed sensor relocation scheme for environmental control. *IEEE MASS*, 2007.

[42] N. Garg and J. Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

[43] L. Goel, V. P. Aparna, and P. Wang. A framework to implement supply and demand side contingency management in reliability assessment of restructured power systems. *Power Systems, IEEE Transactions on*, 22(1):205–212, 2007.

[44] F. A. Graybill and H. K. Iyer. Regression analysis. *Duxbury Press*, 1994.

[45] B. Guenin, J. Könemann, and L. Tunçel. *A gentle introduction to optimization*. Cambridge University Press, 2014.

[46] O. Güler. *Foundations of optimization*. Springer Science & Business Media, 2010.

[47] A. Gut. *An intermediate course in probability.* Springer, 2009.

[48] P. Hage and F. Harary. Eccentricity and centrality in networks. *Social networks*, 17(1):57–63, 1995.

[49] M. Hauptmann and M. Karpinski. A compendium on Steiner tree problems. *"http://theory.cs.uni-bonn.de/info5/steinerkompendium".Last accessed on May 2015.*

[50] N. Heo and P. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Trans. on Syst., Man and Cyb.*, 35(1):78–92, 2005.

[51] H. Ho and A. Sumalee. Optimal recovery plan after disaster: Continuum modeling approach. *Journal of Transportation Engineering*, 140(8):04014034, 2014.

[52] M. Humayun, A. Safdarian, M. Z. Degefa, and M. Lehtonen. Demand response for operational life extension and efficient capacity utilization of power transformers during contingencies. *Smart Grid, IEEE Transactions on*, 30(4):2160–2169, 2015.

[53] Illinois Center for a Smarter Electric Grid, Information Trust Institute, University of Illinois at Urbana-Champaign. IEEE 39-Bus System. `http://publish.illinois.edu/smartergrid/ieee-39-bus-system/`, 2015. [Online; accessed January 30, 2017].

[54] A. T. Kaliappan, S. Sathiakumar, and N. Parameswaran. Flexible power consumption management using Q learning techniques in a smart home. In *IEEE CEAT*, 2013.

[55] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, 2011.

[56] A. Kumar, A. Gupta, and T. Roughgarden. A constant-factor approximation algorithm for the multicommodity rent-or-buy problem. In *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pages 333–342. IEEE, 2002.

[57] B. Lagesse, M. Kumar, and M. Wright. Arex: An adaptive system for secure resource access in mobile p2p systems. In *IEEE P2P*, pages 43–52, 2008.

[58] M. Lam and Y. Liu. Two distributed algorithms for heterogeneous sensor network deployment towards maximum coverage. *IEEE ICRA*, pages 3296–3301, 2008.

[59] W. M. Leavitt and J. J. Kiefer. Infrastructure interdependency and the creation of a normal disaster the case of hurricane katrina and the city of new orleans. *Public works management & policy*, 10(4):306–314, 2006.

[60] E. E. Lee II, J. E. Mitchell, and W. A. Wallace. Restoration of services in interdependent infrastructure systems: A network flows approach. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(6):1303–1317, 2007.

[61] K. Liu, N. Abu-Ghazaleh, and K. Kang. Location verification and trust management for resilient geographic routing. *Elsevier JPDC*, 67(2):215–228, 2007.

[62] K. Ma, Y. Zhang, and W. Trappe. Managing the mobility of a mobile sensor network using network dynamics. *IEEE Trans. on Paral. and Distr. Syst.*, 19(1):106–120, 2008.

[63] M. Ma and Y. Yang. Adaptive triangular deployment algorithm for unattended mobile sensor networks. *IEEE Trans. on Computers*, 56(7):946–847, 2007.

[64] T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005.

[65] S. Martello and P. Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[66] N. Megiddo. *On the complexity of linear programming*. IBM Thomas J. Watson Research Division, 1986.

[67] Memsic. Mts420/400 datasheet.

[68] D. Mendonça. Decision support for improvisation in response to extreme events: Learning from the response to the 2001 world trade center attack. *Decision Support Systems*, 43(3):952–967, 2007.

[69] R. Miller. Hurricane katrina: Communications & infrastructure impacts. Technical report, DTIC Document, 2006.

[70] National Institute of Standards and Technology. . NIST framework and roadmap for smart grid interoperability standards, release 1.0. `http://www.nist.gov/public_affairs/releases/upload/smartgrid_interoperability_final.pdf`. [Online; accessed January 30, 2017].

[71] Y. Nemoto and K. Hamaguchi. Resilient ict research based on lessons learned from the great east japan earthquake. *IEEE Communications Magazine*, 52(3):38–43, 2014.

[72] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. *ACM IPSN*, 2004.

[73] H. Okamura and P. D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31(1):75–81, 1981.

[74] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[75] A. Rabiee, A. Soroudi, B. Mohammadi-Ivatloo, and M. Parniani. Corrective voltage control scheme considering demand response and stochastic wind power. *Power Systems, IEEE Transactions on*, 29(6):2965–2973, 2014.

[76] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz. On the Accuracy of Appliance Identification Based on Distributed Load Metering Data. In *IFIP SustainIT*, 2012.

[77] T. Sakano, Z. M. Fadlullah, T. Ngo, H. Nishiyama, M. Nakazawa, F. Adachi, N. Kato, A. Takahara, T. Kumagai, H. Kasahara, et al. Disaster-resilient networking: a new vision based on movable and deployable resource units. *IEEE Network*, 27(4):40–46, 2013.

[78] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

[79] F. C. Schweppe, M. C. Caramanis, R. D. Tabors, and R. E. Bohn. *Spot pricing of electricity*. Springer Science & Business Media, 2013.

[80] G. Sibley, M. Rahimi, and G. Sukhatme. Mobile robot platform for large-scale sensor networks. *IEEE ICRA*, pages 1143–1148, 2002.

[81] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford. Network architecture for joint failure recovery and traffic engineering. *ACM SIGMETRICS*, June 2011.

[82] A. Todimala and B. Ramamurthy. Approximation algorithms for survivable multicommodity flow problems with applications to network design. *IEEE INFOCOM*, April 2006.

[83] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *USENIX NSDI*.

[84] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. *IEEE PerCom*, 2005.

[85] G. Wang, G. Cao, and T. La Porta. Movement-assisted sensor deployment. *IEEE Trans. on Mobile Computing*, 5(6):640–652, 2006.

[86] J. Wang, C. Qiao, and H. Yu. On progressive network recovery after a major disruption. In *INFOCOM, 2011 Proceedings IEEE*, pages 1925–1933. IEEE, 2011.

[87] Y.-C. Wang, C.-C. Hu, and Y.-C. Tseng. Efficient placement and dispatch of sensors in a wireless sensor network. *IEEE Trans. on Mobile Computing*, 7(2):262–274, 2008.

[88] P. Wei, Y. Ni, and F. F. Wu. Decentralised approach for congestion management and congestion price discovering. *IEE Proceedings-Generation, Transmission and Distribution*, 149(6):645–652, 2002.

[89] B. L. Welch. The generalization of student's' problem when several different population variances are involved. *JSTOR Biometrika*, 34(1/2):28–35, 1947.

[90] G. Yan, S. Olariu, and M. C. Weigle. Providing vanet security through active position detection. *Elsevier Computer Communications*, 31(12):2883–2897, 2008.

[91] A. Zanella, N. Bui, A. P. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 2014.

[92] Q. Zheng, G. Cao, T. La Porta, and A. Swami. Optimal recovery from large-scale failures in ip networks. *IEEE ICDCS*, 2012.

[93] Y. Zou and K. Chakrabarty. Sensor deployment and target localization in distributed sensor networks. *ACM Trans. on Emb. Comp. Syst.*, 3(1):61–91, 2003.

[94] Opnet technologies inc. *http://www.opnet.com.*