Modeling, Analysis, and Efficient Resource Allocation in Cyber-Physical Systems and Critical Infrastructure Networks

by

Arun Das

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved October 2016 by the
Graduate Supervisory Committee:

Arunabha Sen, Chair
Guoliang Xue
Georgios Fainekos
Chunming Qiao

ARIZONA STATE UNIVERSITY

December 2016

ABSTRACT

The critical infrastructures of the nation are a large and complex network of human, physical and cyber-physical systems. In recent times, it has become increasingly apparent that individual critical infrastructures, such as the power and communication networks, do not operate in isolation, but instead are part of a complex interdependent ecosystem where a failure involving a small set of network entities can trigger a *cascading event* resulting in the failure of a much larger set of entities through the failure propagation process.

Recognizing the need for a deeper understanding of the interdependent relationships between such critical infrastructures, several models have been proposed and analyzed in the last few years. However, most of these models are over-simplified and fail to capture the complex interdependencies that may exist between critical infrastructures. To overcome the limitations of existing models, this dissertation presents a new model – the *Implicative Interdependency Model* (IIM) that is able to capture such complex interdependency relations. As the potential for a failure cascade in critical interdependent networks poses several risks that can jeopardize the nation, this dissertation explores relevant research problems in the interdependent power and communication networks using the proposed IIM and lays the foundations for further study using this model.

Apart from exploring problems in interdependent critical infrastructures, this dissertation also explores resource allocation techniques for environments enabled with cyber-physical systems. Specifically, the problem of efficient path planning for data collection using mobile cyber-physical systems is explored. Two such environments are considered: a Radio-Frequency IDentification (RFID) environment with mobile "*Tags*" and "*Readers*", and a sensor data collection environment where both the *sensors* and the *data mules* (data collectors) are mobile.

Finally, from an applied research perspective, this dissertation presents RAPTOR, an advanced network planning and management tool for mitigating the impact of *spatially correlated*, or *region based faults* on infrastructure networks. RAPTOR consolidates a wide range of studies conducted in the last few years on region based faults, and provides an interface for network planners, designers and operators to use the results of these studies for designing robust and resilient networks in the presence of spatially correlated faults.

To the loving memory of my grand-uncle, ***Satyabrata Sarkar (Gopal)***,

who was the embodiment of patience, humility, and gratefulness

ACKNOWLEDGMENTS

This dissertation presents a portion of my work carried out at Arizona State University's Network Science Lab, aptly named the *Laboratory of Networked Existence*. The work outlined in this dissertation could never have been possible without the constant help, support and encouragement of several people who I wish to acknowledge.

First and foremost, I would like to express my sincere gratitude to my adviser, Prof. Arunabha Sen for giving me an opportunity to work at the Network Science Lab, and for fostering a research environment that encouraged ideas, accommodated criticism, overcame failure, and celebrated success. Prof. Sen's clarity of thought, attention to detail, and ability to transform pressing real world problems into addressable mathematical abstractions has immensely influenced my thought process and understanding of the world.

I am also extremely grateful to my dissertation committee members Prof. Guoliang Xue, Prof. Georgios Fainekos, and Prof. Chunming Qiao for their support, encouragement and insightful feedback towards my research. I have been a student of both Prof. Xue's and Prof. Fainekos's classes and have enjoyed and gained from those learning experiences. I have also had the opportunity to work on a research project with Prof. Qiao and that experience was very enriching.

At Arizona State University, I would like to thank Brent Sebold and Ken Mulligan from the Fulton Schools Startup Center for inculcating the *Entrepreneurial Mindset* in me. In their own unique ways, Brent and Ken have instilled in me the desire for creating value in the world, and have made me realize the need and importance of failure. I would also like to thank Christina Sebring, Araxi Hovhannessian, Monica Dugan, Pamela Dunn, Theresa Chai, Brint MacMillan, and Mike McAllister from the School of Computing, Informatics, and Decision Systems Engineering for helping me in countless ways during my time as a doctoral student.

I would like to thank my collaborators and lab mates Anisha Mazumder, Chenyang Zhou, Zahra Derakhshandeh and Joydeep Banerjee with whom I have had the opportunity to work on several research problems and have enjoyed their company inside and outside the lab. I would also like to thank the Network Science Lab alumni of Dr. Sujogya Banerjee, Dr. Shahrzad Shirazipourazad and Harsh Vacchani, for their insight and experience of working at the Network Science Lab.

I started my doctoral studies when my son Aakash was just six months old, and I am extremely certain that my doctoral experience would not have been enriching nor fulfilling without the constant support and encouragement from my family. I would like to express my sincere gratitude to my wife Chandrani for always being there for me during the highs and lows of life. I would like to thank my wife's parents, Debajyoti and Kalpana, for always being available to help us out. A special thanks goes to Sukhendu Kundu for encouraging me to step out of my comfort zone and pursue a doctoral degree. I would also like to thank Samir and Debjani for their insight, support and encouragement towards making my doctoral studies successful. I would also like to express my love and sincere gratitude to Aditya, Jolly, and Abhijit – my father, mother, and brother, for always encouraging and supporting me in whatever I do in life and for believing that my best is yet to come.

Finally, I would like to thank my friends who have been a source of immense support and enjoyment for all these years. I would like to thank the *Vivekanada Park* family who are my roots back home – Sougata, Samantak, Rahul, Rambo, Sukanta, Manas and Somenath. I would also like to thank my childhood friends Pritha, Samieta, Chirag, Shankar, Sayantini, and Sanglap for bearing with my idiosyncrasies all these years. Here in Arizona, I would like to thank Prithwish, Tiyasa, Rohit, Shantanu, Abhinandan, Tanmoy, Jinia, Sabarna, Sayantan, Arpan, Veer, Dheeraj, Niranjan, Shiva, Dharav, Arun Sundar, Ragav and David who all made my stay in Arizona extremely enjoyable.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The critical infrastructures of the nation have immense impact in all aspects of society, hence their robustness and resiliency are a major area of study. Although research on fault analysis for individual critical infrastructure systems have been extensive, such as studies on robustness of communication networks, most of such studies have considered these systems in isolation. However, in the real world, critical infrastructures such as the power grid and the communication network belong to a much larger ecosystem of interdependent systems whose symbiotic relationships not only affect the well being of the individual systems, but also impact the socio-economic and political environments that are built around these systems. For example, the communication networks of today are highly dependent on the power infrastructure for their operation, and as has been periodically reported, the power network can be very susceptible to large scale failures.

For instance, in March 2014, the Wall Street Journal, based on a study conducted by the Federal Energy Regulatory Commission (FERC), reported that "The U.S. could suffer a coast-to-coast blackout if saboteurs knocked out just *nine* of the country's 55,000 electric-transmission substations on a scorching summer day". The FERC study concluded that "coordinated attacks in each of the nation's three separate electric systems could cause the entire power network to collapse". The study also revealed that "a small number of the country's substations play an outsize role in keeping power flowing across large regions, and knocking out *nine of those key substations could plunge the country into darkness for weeks, if not months*" [1]. It can be comprehended that the effects of such an attack on the power infrastructure will severely impact the communication network, which in turn will

compound the crisis and jeopardize other critical infrastructures such as financial markets, emergency services, healthcare facilities, and military establishments.

In light of such circumstances, the last few years has seen a heightened awareness in the research community that the critical infrastructures are closely coupled together, and that there is a compelling need for a deeper understanding of the interdependent relationships between such infrastructures. For instance, in the case of the interdependent power and communication networks, power grid entities, such as SCADA systems that control power stations and sub-stations, are reliant on the communication network to send and receive control signals. On the other hand, communication network entities, such as routers and base stations are reliant on electric power. Understanding the impact of cascading failures in the power grid, becomes even more complex when the coupling between the power grid and communication network entities are considered. This coupling, or interdependence, allows not only entities in the power network, such as generators and transmission lines, to trigger power failure, but also communication network entities, such as routers and optical fiber lines, can potentially trigger failures in the power grid. Thus, it is imperative that the interdependencies in this complex network ecosystem be well understood, so that preventive measures can be undertaken to avoid catastrophic failures in interdependent critical infrastructures such as the *Interdependent Power-Communication Network* (IPCN).

This dissertation highlights the existing studies in the area of interdependent critical infrastructures and outlines limitations of the interdependency models proposed in these studies. To overcome the limitations of existing models, this dissertation presents the *Implicative Interdependency Model* (IIM) and describes some of the studies conducted using the IIM. Specifically, the problems of identifying the $\mathcal{K}$ most vulnerable nodes of an interdependent network, and identifying the smallest pseudo target set of entities for a directed attack in interdependent networks are discussed in this dissertation. Each of these problems

are formulated in the IIM setting, the computational complexities are analyzed, and solution techniques are proposed. The efficacy of the solutions are evaluated using the power and communication network data of Maricopa County, Arizona.

Apart from exploring problems in interdependent critical infrastructures, this dissertation also explores resource allocation techniques for environments enabled with cyber-physical systems. Specifically, the problem of efficient path planning for data collection using mobile cyber-physical systems is explored. In this context, two application environments are considered – a Radio-Frequency IDentification (RFID) environment with mobile "*Tags*" and "*Readers*", and a sensor data collection environment where both the sensors and the *data mules* (data collectors) are mobile. In both these environments, there are mobile entities that provide or supply data, (tags and sensors), and mobile entities that collect or consume the provided data (readers and data mules). Although similarities exist between these two environments, however, the data collection requirements may differ considerably in these environments. For instance, in a RFID environment a tag provides a fixed amount of data which is relatively uniform across all the tags available in the environment. Whereas, in a sensor network, each sensor may have different data collection and storage capabilities and thus each sensor may require varying amounts of data to be supplied to the data mules. Additionally, the data provided by a tag in a RFID environment must necessarily be read in its entirety by a single reader. However, in a sensor network environment, depending upon the computational resources available in the environment, the data provided by a sensor may be required to be read by a single data mule in its entirety, or multiple mules may partially read the data from the sensor and reconstruct them after collection. Due to such differences in the two environments we study the path planning problem in these two environments separately.

Finally, from an applied research perspective, this dissertation presents RAPTOR, an advanced network planning and management tool for mitigating the impact of *spatially correlated*, or *region based faults* on infrastructure networks. This tool consolidates a wide range of theories and solutions developed in the last few years on region based faults, and provides an interface for network planners, designers and operators to use these results and techniques for designing robust and resilient networks in the presence of spatially correlated faults. This dissertation describes the novel concepts developed to design networks that are robust against region based faults, and then describes how these concepts have been incorporated into the tool. As a service to the networking research community, one of the goals of this dissertation is to bring to the attention of the community the existence of RAPTOR as a tool that consolidates a large body of work on spatially correlated failures, and as a tool that can be used by the community to meet the needs for robust network design against spatially correlated failures.

The remainder of this dissertation is organized as follows: In Chapter 2 a brief overview of the existing interdependent multilayer network models are outlined, and in Chapter 3 the proposed Implicative Interdependency Model (IIM) is presented. In the IIM setting, the $\mathcal{K}$ most vulnerable nodes identification problem is addressed in Chapter 4, and the smallest pseudo target set identification problem is addressed in Chapter 5. In Chapters 6 and 7 the dissertation addresses the reader minimization problem in RFID systems, and the data mule minimization problem in sensor networks respectively. In Chapter 8 the RAPTOR tool is presented. Finally, in Chapter 9 this dissertation is concluded, and possible future work stemming from this dissertation is outlined.

Chapter 2

INTERDEPENDENT MULTILAYER NETWORK MODELING

Recognizing the need for a deeper understanding of the interdependent relationships between a multilayered network, in the last few years, significant efforts have been made by the research community to achieve this goal. Accordingly, a number of models have been proposed and analyzed [2]–[10]. This Chapter provides an overview of some of these models and highlights the limitations of existing models.

Motivated by the 2003 electricity blackout in Italy, in [2] Buldyrev et al. proposed a graph based interdependency model with a power network graph A, and a communication network graph B. The authors assume that (i) the number of nodes in the power network is equal to the number of nodes in the communication network (i.e. $|A| = |B|$), and (ii) there exists a one-to-one dependency between a node in the power network and a node in the communication network. The model also makes an implicit assumption that the power (communication) nodes are homogeneous with respect to functionality, i.e. there is no distinction between power-plant, sub-station or load nodes (or cell towers and routers). The authors describe a cascading failure process (the rules by which the nodes and edges of the graph is removed), and using this model, they compute the percolation threshold for existence of a giant connected component. The assumptions about this one-to-one dependency between the power and communication network nodes, and the homogeneous nature of the nodes are unrealistic as the model fails to capture the complex interdependencies that may exist between the entities of the network. The authors also opine in a subsequent paper (in [3]), that a single node in one network may be dependent on multiple nodes in the other network.

In [5], Rostato et al. model the power flow in the power grid, and the data flow in the communication network separately. They then analyze the effect of failures in the communication network, caused by failures in the power grid using a coupling model between the two infrastructures. The authors construct graphs for the power grid and communication network from the Italian high voltage electric transmission network (HVIET), and the high-bandwidth backbone of the Italian Internet network (GARR). For the power network, the model considers the DC power flow model [11], and for the communication network, a probabilistic packet routing model is considered for sending data packets from randomly generated source and destination nodes. A dependency between the two networks is setup by associating a node from the communication network to the closest load node from the power network (in terms of Euclidean distance). It may be noted that the dependency considered in this model is one directional, i.e. for a communication node to be operational it is dependent on a power network node, however, the power network node is not dependent on the communication node for its survival. In the event of a failure, a load re-dispatching process is initiated on the power network, and a communication network node remains operational as long as the load node it is connected to is dispatched with power greater than a computed threshold. Although the model proposed in [5] is realistic to a point, the dependency model is a one way dependency model and fails to represent the interdependency that may exist between the power and communication networks.

Although a number of interdependency models have been proposed and analyzed in the recent past [2]–[9], however, most of these models are over simplified and fail to capture complex interdependencies that may involve a combination of conjunctive and disjunctive relationships between network entities. For instance, suppose the power network entities such as *power generators, transmission lines* and *substations* are denoted by the set $A = \{a_1, a_2, \ldots, a_n\}$ and the entities of the communication network, such as *routers,*

fiber optic lines and *base stations* are denoted by the set $B = \{b_1, b_2, \ldots, b_m\}$. Due to the topological design of the power-communication networks, it may so happen that an entity a_i is *operational* if (i) the entities b_j *and* b_k *and* b_l are operational, *or* (ii) b_m *and* b_n are operational, *or* (iii) b_p is operational. Graph based interdependency modeling, such as in [2]–[9] cannot capture such interdependencies involving conjunctive and disjunctive terms. In Chapter 3 the *Implicative Interdependency Model* (IIM) is proposed that is able to capture such complex interdependencies using Boolean Logic and overcomes the limitations of existing graph based approaches.

Chapter 3

IMPLICATIVE INTERDEPENDENCY MODEL (IIM)

The *Implicative Interdependency Model* (IIM) is an entity based model that allows representation of complex dependency relations between entities of multilayer network systems. The dependent relationships between the network entities are represented using Boolean Logic and are termed as *Implicative Interdependency Relations* (IDRs). For instance, in a sample IPCN, if the power network entities are the set of A type entities, $A = \{a_1, \ldots, a_n\}$ and the communication network entities are the set of B type entities, $B = \{b_1, \ldots, b_m\}$. If power network entity a_i is *operational* if (i) the entities b_j *and* b_k *and* b_l are operational, *or* (ii) b_m *and* b_n are operational, *or* (iii) b_p is operational, the corresponding IDR would be of the form $a_i \leftarrow b_j b_k b_l + b_m b_n + b_p$. It may be noted that the IDRs only provide a *necessary condition* for entities (such as a_i) to be *operational*. In other words, a_i may *fail* independently and may be *inoperable* even when the conditions given by the corresponding IDR are *satisfied*.

Table 1 outlines a set of IDRs representing a sample IPCN where the power network and communication network entities are represented by the sets $A = \{a_1, a_2, a_3, a_4\}$ and $B = \{b_1, b_2, b_3\}$ respectively. The IDRs represent a set of necessary Boolean conditions that need to be satisfied for an entity to be operational. In Table 1, entity b_1 is operational if either a_2 is operational, or both a_1 and a_3 is operational. It may be noted that although in the IDRs of this example, A (B) type entities appear on either the left hand side or the right hand side of an IDR, the IIM does not require that A (B) type entities appear only on one side of an IDR. In other words, an IDR can also be of the form $a_i \leftarrow a_q b_j b_k b_l + a_r b_m b_n + b_p + a_s$

implying that A (B) type entities may depend on A (B) type entities. The conjunction of entities, such as $a_r b_m b_n$, is also referred to as a *minterm*.

Power Network	**Comm. Network**
$a_1 \leftarrow b_1 b_2$	$b_1 \leftarrow a_1 a_3 + a_2$
$a_2 \leftarrow b_1 + b_2$	$b_2 \leftarrow a_1 a_2 a_3$
$a_3 \leftarrow b_1 + b_2 + b_3$	$b_3 \leftarrow a_1 + a_2 + a_3$
$a_4 \leftarrow b_1 + b_3$	$--$

Table 1: A sample Interdependent Power-Communication Network (IPCN)

The interdependencies expressed through IDRs govern the failure cascade process in IIM, where the failure of a set of entities can trigger further failures due to the interdependencies shared between the entities. This cascading failure process is illustrated with the help of an example: For the IPCN system of Table 1, Table 2 shows a time-stepped cascading failure of entities triggered by the failure of $\{a_2, b_3\}$ at time step 0.

Entities	Time Steps (t)						
	0	1	2	3	4	5	6
a_1	0	0	1	1	1	1	1
a_2	1	1	1	1	1	1	1
a_3	0	0	0	0	1	1	1
a_4	0	0	0	0	1	1	1
b_1	0	0	0	1	1	1	1
b_2	0	1	1	1	1	1	1
b_3	1	1	1	1	1	1	1

Table 2: Time Stepped Failure Propagation in a Multilayer Network. A value of 1 denotes entity failure.

In Table 2, the cascading failure process initiated by the failure of a subset of A type entities at time step 0 (denoted by A_d^0), and a subset of B type entities (denoted by B_d^0), till the system reaches its final steady state is shown diagrammatically in Figure 1. Accordingly,

an interdependent multilayer network can be viewed as a "closed loop" control system as shown in Figure 2. Finding the steady state after an initial failure in this case is equivalent to computing the *fixed point* [12] of a function $\mathcal{F}(.)$ such that $\mathcal{F}(A_d^p \cup B_d^p) = A_d^p \cup B_d^p$, where p represents the number of steps when the system reaches the steady state. In the sample cascade of Table 2, at time step 4 the system reaches a steady state after the initial failure of $\{a_2, b_3\}$ at time step 0.

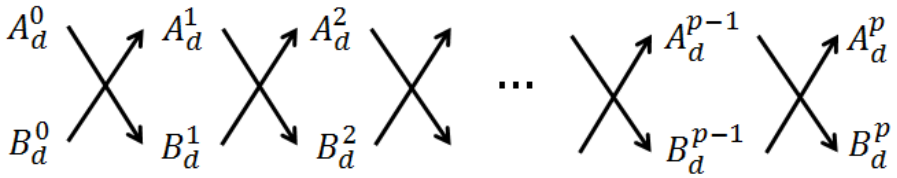


Figure 1: Cascading failures reach steady state after p time steps

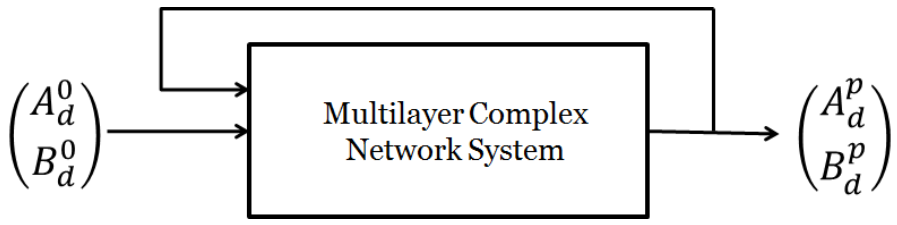


Figure 2: Interdependent multilayer network as a closed loop control system

The dependency relationships, or IDRs, used to represent the interdependent system can be formed either by careful analysis of the underlying system as was done in [10], or by consultation with the subject matter experts of the complex systems.

Chapter 4

$\mathcal{K}$-MOST VULNERABLE NODES IDENTIFICATION PROBLEM

4.1 Introduction

In this Chapter, the IIM (described in Chapter 3) is used to model the interdependent relationships between critical infrastructures and techniques are proposed to identify the $\mathcal{K}$ most "vulnerable" nodes of the interdependent network system.

The set of $\mathcal{K}$ entities in a multilayer interdependent network is defined to be most vulnerable, if failure of these $\mathcal{K}$ entities triggers the failure of the largest number of other entities. The goal of the $\mathcal{K}$ *Most Vulnerable Nodes* ($\mathcal{K}$MVN) problem is to identify this set of nodes, given an interdependent network with A and B type entities, where $n = |A|, m = |B|$. This is equivalent to identifying $A_d^0 \subseteq A$, $B_d^0 \subseteq B$, that *maximizes* $|A_d^p \cup B_d^p|$, subject to the constraint $|A_d^0 \cup B_d^0| \leq \mathcal{K}$. For solving the $\mathcal{K}$MVN problem the following assumptions are made: (i) an entity $e_i \in A \cup B$ can fail by itself and not due to its dependencies, only at time step 0. Any failures after time step 0 occur due to the cascade effect of entities that failed at time step 0, (ii) dependent entities immediately fail in the next time step, i.e. if $e_i \leftarrow e_j e_k$, and e_k fails at time step $p-1$, then e_i fails at p, and (iii) time step $p = n+m-1$ is a sure end of any failure cascade that begins at time step 0 as there are at most $n+m$ entities and it's assumed that entities cannot become operational once they fail.

It is shown that the $\mathcal{K}$MVN problem can be solved in polynomial time for some special cases, whereas for some others, the problem is NP-complete. A technique is provided utilizing Integer Linear Programming to compute the solution to the $\mathcal{K}$MVN problem. Fi-

nally, experiments are conducted using the proposed technique on real data collected from the power grid and communication network that span Maricopa County of Arizona.

4.2 Computational Complexity and Algorithms

As elaborated in Chapter 3, in IIM the dependent relationships between the network entities are represented using Boolean Logic and are termed as *Implicative Interdependency Relations* (IDRs), where the power (communication) network entities are denoted by the set A (B). Based on the number and the size of the minterms in the IDRs, the IDRs can be divided into four different cases as shown in Table 3. The algorithms for finding the $\mathcal{K}$ most vulnerable nodes in the interdependent multilayer networks and the computational complexity for each of the cases are discussed in the following sections.

Case	No. of Minterms	Size of Minterms
Case I	1	1
Case II	1	Arbitrary
Case III	Arbitrary	1
Case IV	Arbitrary	Arbitrary

Table 3: Types of Implicative Dependency Relations (IDRs)

4.2.1 Case I: Problem Instance with One Minterm of Size One

In this case, an IDR will have the following form: $x_i \leftarrow y_j$ where x_i and y_j are elements of the set A (B) and B (A) respectively. In the example $a_i \leftarrow b_j$, $x_i = a_i$, $y_1 = b_j$. It may be noted that a conjunctive IDR of the form $a_i \leftarrow b_j b_k$ can also be written as two separate IDRs $a_i \leftarrow b_j$ and $a_i \leftarrow b_k$. However, such cases are considered in Case II (Section 4.2.2)

and is excluded from consideration in Case I. The exclusion of such IDRs implies that the entities that appear on the LHS of an IDR in Case I are unique. This property enables the development of a polynomial time algorithm for the solution of the $\mathcal{K}$ most vulnerable node problem for this case. The algorithm is presented next.

Algorithm 1: Case I Optimal Algorithm for the $\mathcal{K}$MVN problem

Data:
1. Set of network entities $A \cup B$, with $n = |A|$ and $m = |B|$
2. A set S of IDRs of the form $y \leftarrow x$, where $x, y \in A \cup B$
3. An integer $\mathcal{K}$

Result: A set $V' \subset A \cup B$ where $|V'| = \mathcal{K}$, such that failure of entities in V' at time step 0 results in failure of the largest number of entities by time step $p = n + m - 1$.

1 **begin**
2 Construct a directed graph $G = (V, E)$, where $V = A \cup B$. For each IDR $y \leftarrow x$ in S, where $x, y \in A \cup B$, introduce a directed edge $(x, y) \in E$;
3 For each node $x_i \in V$, construct a transitive closure set C_{x_i} as follows: If there is a path from x_i to some node $y_i \in V$ in G, then include y_i in C_{x_i}. As $|A| + |B| = n + m$, there will be $n + m$ transitive closure sets $C_{x_i}, 1 \leq i \leq (n + m)$. Each x_i is termed as the *seed entity* for the transitive closure set C_{x_i};
4 Remove all the transitive closure sets which are proper subsets of some other transitive closure set;
5 Sort the *remaining transitive closure sets* C_{x_i}, where the rank of the closure sets is determined by the cardinality of the sets. The sets with a larger number of entities are ranked higher than the sets with a fewer number of entities;
6 Construct set V' by selecting the *seed entities* of the top $\mathcal{K}$ transitive closure sets. If the number of *remaining transitive closure sets* is less than $\mathcal{K}$ (say, $\mathcal{K}'$), arbitrarily select the remaining entities;
7 **return** V'

Time complexity of Algorithm 1: Step 2 takes $O(n + m + |S|)$ time, and Step 3 can be executed in $O((n + m)^3)$ time. Step 4 takes at most $O((n + m)^2)$ time. As Step 5 sorts at most $|S|$ entries, a standard sorting algorithm takes $O(|S| log |S|)$ time. Selecting $\mathcal{K}$ entities in Step 6 takes $O(\mathcal{K})$ time. Since $|S| \leq n + m$, hence the overall time complexity of Algorithm 1 is $O((n + m)^3)$.

Theorem 1 *For each pair of transitive closure sets C_{x_i} and C_{x_j} produced in Step 3 of Algorithm 1, either $C_{x_i} \cap C_{x_j} = \emptyset$ or $C_{x_i} \cap C_{x_j} = C_{x_i}$ or $C_{x_i} \cap C_{x_j} = C_{x_j}$, where $x_i \neq x_j$.*

Proof: Consider, if possible, that there is a pair of transitive closure sets C_{x_i} and C_{x_j} produced in Step 3 of Algorithm 1, such that $C_{x_i} \cap C_{x_j} \neq \emptyset$ and $C_{x_i} \cap C_{x_j} \neq C_{x_i}$ and $C_{x_i} \cap C_{x_j} \neq C_{x_j}$. Let $x_k \in C_{x_i} \cap C_{x_j}$. This implies that there is a path from x_i to x_k ($path_1$) as well as there is a path from x_j to x_k, ($path_2$). Since, $x_i \neq x_j$ and $C_{x_i} \cap C_{x_j} \neq C_{x_i}$ and $C_{x_i} \cap C_{x_j} \neq C_{x_j}$, there is some x_l in the $path_1$ such that x_l also belongs to $path_2$. Without loss of generality, let us consider that x_l be the first node in $path_1$ such that x_l also belongs to $path_2$. This implies that x_l has an in-degree greater than 1. This in turn implies that there are two IDRs in the set of IDRs S such that x_l appears in the LHS of both. This is a contradiction because this violates a characteristic of the IDRs in Case I. Hence, the initial assumption is incorrect and the theorem is proved. ■

Theorem 2 *Algorithm 1 gives an optimal solution for the problem of selecting $\mathcal{K}$ most vulnerable entities in a interdependent multilayer network system with Case I dependencies.*

Proof: Consider that the set V' returned by Algorithm 1 is not optimal and the optimal solution is V_{OPT}. Let $x_i \in A \cup B$ be an entity such that $x_i \in V_{OPT} \setminus V'$. This implies that either, (i) C_{x_i} was removed in Step 4, or (ii) $|C_{x_i}|$ is less than the cardinalities of all the transitive closure sets with seed entities $x_j \in V'$, as Algorithm 1 did not select x_i. Hence, in both cases, replacing any entity $x_j \in V'$ by x_i reduces the total number of entities rendered inoperable. Thus, the number of inoperable entities induced by the failure of entities in V_{OPT} is less than the number of inoperable entities induced by the failure of the entities in V', hence V_{OPT} is not optimal contradicting the initial assumption and proving the theorem. ■

4.2.2 Case II: Problem Instance with One Minterm of Arbitrary Size

In this case, an IDR in general will have the following form: $x_i \leftarrow \prod_{k=1}^{q} y_j$ where x_i and y_j are elements of the set A (B) and B (A) respectively, q represents the *size* of minterm. In the example $a_i \leftarrow b_j b_k b_l$, $q = 3$, $x_i = a_i$, $y_1 = b_j$, $y_2 = b_k$, $y_3 = b_k$.

4.2.2.1 Computational Complexity

It is now shown that the $\mathcal{K}$MVN problem in a interdependent multilayer network system is NP-complete in Case II. The problem is formally stated as follows:

INSTANCE: Given (i) the set A and B representing the entities of the power and communication networks respectively with $n = |A|$, $m = |B|$, (ii) a set of IDRs between A and B type entities in the form $x_i \leftarrow \prod_{k=1}^{q} y_j$, and (iii) integers $\mathcal{K}$ and $\mathcal{L}$.

QUESTION: Is there a subset of A and B type entities of size at most $\mathcal{K}$ whose failure at time step 0, triggers a cascade of failures resulting in failures of at least $\mathcal{L}$ entities at time step $p = n + m - 1$?

Theorem 3 *The $\mathcal{K}$MVN problem is NP-complete.*

Proof: It is proved that the $\mathcal{K}$MVN problem is NP-complete by providing a transformation for the set cover (SC) problem. An instance of the set cover problem is specified by a universal set $S = \{s_1, s_2, ..., s_n\}$, and a set of subsets $S' = \{S_1, S_2, ..., S_m\}$ where $S_i \subseteq S, \forall i, 1 \leq i \leq m$, and a positive integer Q. In the set cover problem one wants to know whether there exists a subset of $S'' \subseteq S'$ such that $\bigcup_{S_i \subseteq S''} S_i = S$ and $|S''| \leq Q$. From an instance of the SC problem an instance of the $\mathcal{K}$MVN problem is constructed in the following way: For every element $s_i \in S$, entity a_i is added to set A. For each subset $S_j \in S'$,

entity b_j is added to set B. For each subset $S_j \in S'$, that contains s_i (say S_u, S_v, S_w), an IDR of the form $a_i \leftarrow \prod_{s_i \in S_j} b_j, \forall S_j \in S'$ is created ($a_i \leftarrow b_u b_v b_w$). Finally, the parameters $\mathcal{K}$ and $\mathcal{L}$ entities are set to $\mathcal{K} = Q$ and $\mathcal{L} = |S| + Q$. It may be noted that in this construction no IDRs are introduced for elements in set B, implying that these entities do not depend on any other entity for their survival, but can fail independently. It can now easily be verified that the instance of the SC problem has a set cover of size Q, iff in the created instance of the $\mathcal{K}$MVN problem the failure of $\mathcal{K}$ entities at time step 0 triggers a failure of $\mathcal{L}$ entities by time step $p = n + m - 1$. ■

4.2.2.2 Optimal Solution with Integer Linear Programming

For Case II, the optimal solution to the $\mathcal{K}$MVN problem can be computed using Integer Linear Programming (ILP). To construct the ILP, a binary indicator variable x_i (y_i) is used to capture the state of the entities a_i (b_i). Where x_i (y_i) is 1 when a_i (b_i) is in a failed state and 0 otherwise. Since it is required to find the set of $\mathcal{K}$ entities whose failure at time step 0 triggers a cascading failure resulting in the failure of the largest number of entities, the objective of the ILP can be written as follows: $\max \sum_{i=1}^{n} x_i + \sum_{i=1}^{m} y_i$. It may be noted that the variables in this objective function do not have any notion of *time*. However, cascading failure takes place in time steps, a_i can trigger failure of b_j at time step 1, which in turn can trigger failure of a_k at time step 2 and so on. Accordingly, in order to capture the cascading failure process, a notion of time is required to be introduced into the indicator variables of the ILP. As mentioned earlier, if the cardinality of A and B type entities are n and m respectively, the steady state must be reached by time step $n + m - 1$ (cascading process starts at time step 0). Accordingly, $n + m$ versions of the variables x_i and y_i are introduced, i.e., $x_i[0], \ldots, x_i[n + m - 1]$ and $y_i[0], \ldots, y_i[n + m - 1]$. To indicate the state of entities a_i

and b_i at times $p = \{0, \ldots, n+m-1\}$. The objective of the ILP is now altered to:

$$\max \sum_{i=1}^{n} x_i[n+m-1] + \sum_{i=1}^{m} y_i[n+m-1] \tag{4.1}$$

Subject to the constraint that no more than $\mathcal{K}$ entities can fail at time 0:

Constraint 1: $\sum_{i=1}^{n} x_i[0] + \sum_{i=1}^{m} y_i[0] \leq \mathcal{K}$

In order to ensure that the cascading failure process conforms to the dependency relations in the IDR additional constraints must be imposed:

Constraint 2: If an entity fails at time step p it should remain in the failed state for all subsequent time steps. That is $x_i[t] \geq x_i[t-1], \forall t, 1 \leq t \leq n+m-1$. A similar constraint also applies to $y_i[t]$.

Constraint 3: Translate Case II type IDRs into a linear constraint that captures the dependency relationship. For instance, the IDR $a_i \leftarrow b_j b_k b_l$ can be translated into a linear constraint in the following way $x_i[t] \leq y_j[t-1] + y_k[t-1] + y_l[t-1], \forall t, 1 \leq t \leq n+m-1$.

The optimal solution to the $\mathcal{K}$MVN problem for a interdependent multilayer network system represented by Case II type IDRs can be found by solving the above ILP.

4.2.3 Case III: Problem Instance with an Arbitrary Number of Minterms of Size One

Case III type IDRs have the following form: $x_i \leftarrow \sum_{j=1}^{q} y_j$ where x_i and y_j are elements of the set A (B) and B (A) respectively, q represents the *number* of minterms in the IDR. In the example $a_i \leftarrow b_j + b_k + b_l$, $q = 3$, $x_i = a_i$, $y_1 = b_j$, $y_2 = b_k$, $y_3 = b_l$.

4.2.3.1 Computational Complexity

It is now shown that a special case of the problem instances with an arbitrary number of minterms of size one is same as the *Subset Cover* problem (defined below), which is

proven to be NP-complete. Let $Implication\_Set(A)$ be the set of all Case III type IDRs where a_i is on the LHS of an IDR for all $a_i \in A$, and $Implication\_Set(B)$ be the set of all IDRs where b_j is on the LHS of an IDR for all $b_j \in B$. Now consider a subset of the set of problem instances with an arbitrary number of minterms of size one where (i) either $Implication\_Set(A) = \emptyset$ or $Implication\_Set(B) = \emptyset$, and (ii) IDRs of entities of network A (B) do not contain minterms from the same network A (B), i.e. if an IDR for entity $a_i \in A$ ($b_j \in B$) exists, then the minterms of the IDR for a_i (b_j) contain only entities from network B (A).

Let A' be the set of all entities $a_i \in A$ such that a_i is an element on the LHS of an IDR in $Implication\_Set(A)$. The set B' is defined accordingly. If $Implication\_Set(B) = \emptyset$ then $B' = \emptyset$. In this case, failure of any $a_i, 1 \leq i \leq n$ type entities will not cause failure of any $b_j, 1 \leq j \leq m$ type entities. Since an adversary can cause failure of only $\mathcal{K}$ entities, the adversary would like to choose only those $\mathcal{K}$ entities that will cause failure of the largest number of entities. In this scenario, there is no reason for the adversary to attack any $a_i, 1 \leq i \leq n$ type entities as they will not cause failure of any $b_j, 1 \leq j \leq m$ type entities. On the other hand, if the adversary attacks $\mathcal{K}$ b_j type entities, not only do those $\mathcal{K}$ b_j type entities will fail, some a_i type entities will also fail due to the IDRs in $Implication\_Set(A)$. As such, the goal of the adversary will be to carefully choose $\mathcal{K}$ $b_j, 1 \leq j \leq m$ type entities that will destroy the largest number of a_i type entities. In its abstract form, the problem can be viewed as the Subset Cover problem.

Subset Cover Problem:

INSTANCE: A set $S = \{s_1, \ldots, s_m\}$, a set $\mathcal{S}$ of r subsets of S, i.e., $\mathcal{S} = \{S_1, \ldots, S_r\}$, where $S_i \subseteq S, \forall i, 1 \leq i \leq r$, integers p and q.

QUESTION: Is there a p element subset S' of S ($p < m$) that completely covers at least q

elements of the set $\mathcal{S}$? (A set S' is said to be *completely covering* an element $S_i, \forall i, 1 \leq i \leq r$ of the set $\mathcal{S}$, if $S' \cap S_i = S_i, \forall i, 1 \leq i \leq r$.)

The set S in the subset cover problem corresponds to the set $B = \{b_1, \ldots, b_m\}$, and each set $S_i, 1 \leq i \leq r$ corresponds to an IDR in the $Implication\_Set(A)$ and comprises of the b_j's that appear on the RHS of the IDR. The goal of the problem is to select a subset B'' of B that maximizes the number of S_i's completely covered by B''.

Theorem 4 *The Subset Cover problem is NP-complete.*

Proof: The Subset Cover problem is proven to be NP-complete by giving a transformation from the well known Clique problem. It may be recalled that an instance of the Clique problem is specified by a graph $G = (V,E)$ and an integer K. The decision question is whether or not a clique of size at least K exists in the graph $G = (V,E)$. It is shown that a clique of size K exists in graph $G = (V,E)$ iff the Subset Cover problem instance has a p element subset S' of S that completely covers at least q elements of the set $\mathcal{S}$.

From an instance of the Clique problem, an instance of the Subset Cover problem is created in the following way. Corresponding to every vertex $v_i, 1 \leq i \leq n$ of the graph $G = (V,E)$ ($V = \{v_1, \ldots, v_n\}$), an element in the set $S = \{s_1, \ldots, s_n\}$ is created. Corresponding to every edge $e_i, 1 \leq i \leq m$, m subsets of S is created, i.e., $\mathcal{S} = \{S_1, \ldots, S_m\}$, where S_i corresponds to a two element subset of nodes, corresponding to the end vertices of the edge e_i. The parameters $p = K$ and $q = K(K-1)/2$ are set up. Next it is shown that in the instance of the subset cover problem created by the above construction process, a p element subset S' of S exists that completely covers at least q elements of the set $\mathcal{S}$, iff the graph $G = (V,E)$ has a clique of size at least K.

Suppose that the graph $G = (V,E)$ has a clique of size K. It is clear that in the created instance of the subset cover problem, there will be $K(K-1)/2$ elements in the set $\mathcal{S}$, that will be completely covered by a K element subset of the set S. The K element subset of S

corresponds to the set of K nodes that make up the clique in $G = (V, E)$ and the $K(K-1)/2$ elements in the set $\mathcal{S}$ corresponds to the edges of the graph $G = (V, E)$ that corresponds to the edges of the clique. Conversely, suppose that the instance of the Subset Cover problem has K element subset of S that completely covers $K(K-1)/2$ elements of the set $\mathcal{S}$. Since the elements of $\mathcal{S}$ corresponds to the edges in G, in order to completely cover $K(K-1)/2$ edges, at least K nodes (elements of the set S) will be necessary. As such, this set of K nodes will constitute a clique in the graph $G = (V, E)$. ■

4.2.3.2 Optimal Solution with Integer Linear Programming

By associating binary indicator variables x_i and y_i to capture the state of the entities a_i and b_i, to construct the ILP, an identical procedure used in Case II is applicable to Case III type IDRs barring one exception. Unlike the linear constraint constructed in Constraint 3 of Case II, for a given IDR of the form $a_i \leftarrow b_{j_1} + b_{j_2} + \cdots + b_{j_g}$ in Case III a linear constraint of the form $g \times x_i[t] \leq \sum_{h=1}^{g} y_{j_h}[t-1], \forall t, 1 \leq t \leq n+m-1$ is constructed.

The optimal solution to the $\mathcal{K}$MVN problem for a interdependent multilayer network system represented by Case III type IDRs can be found by solving the modified ILP.

4.2.4 Case IV: Problem Instance with an Arbitrary Number of Minterms of Arbitrary Size

4.2.4.1 Computational Complexity

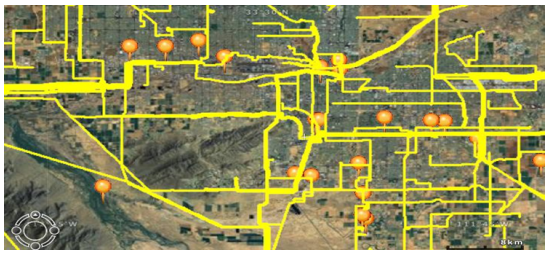
Since both Case II and Case III are special cases of Case IV, the computational complexity of finding the $\mathcal{K}$ most vulnerable nodes in the interdependent multilayer network represented by Case IV type IDRs is NP-complete as well.

4.2.4.2 Optimal Solution with Integer Linear Programming

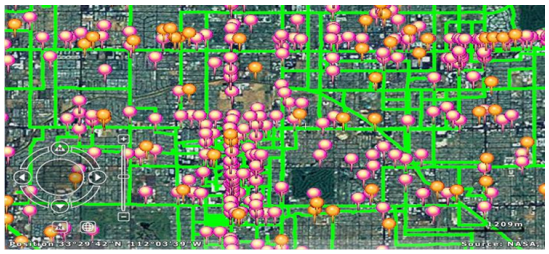
The optimal solution to this version of the problem can be computed by combining the techniques developed for the solution of the versions of the problems considered in Cases II and III. For e.g., an IDR $a_1 \leftarrow b_j b_k b_l + b_m b_n + b_p$ can be written as $a_1 \leftarrow c_1 + c_2 + c_p$ and treated as Case III type IDRs, where $c_1 \leftarrow b_j b_k b_l, c_2 \leftarrow b_m b_n$ can be treated as Case II type IDRs.

4.3 Experimental Results

Using the proposed IIM model the $\mathcal{K}$MVN problem was studied on data collected for Maricopa county, the most densely populated county of Arizona with approximately 60% of Arizona's residential population. More specifically, the objective of the experiments was to find whether some geographical regions of Maricopa county were more vulnerable to failure than some other regions. The data for the multilayered network was obtained from different sources. The power network data was obtained from Platts (www.platts.com).



(a) Snapshot of Power Network in Maricopa County obtained from Platts (www.platts.com)



(b) Snapshot of Communication Network in Maricopa County obtained from GeoTel (www.geo-tel.com)

Figure 3: Snapshots of the Power and Communication Network data of Maricopa County, Arizona

The power network data set consisted of 70 power plants and 470 transmission lines. The communication network data set was obtained from GeoTel (www.geo-tel.com). The communication network data consisted of $2,690$ cell towers and $7,100$ fiber-lit buildings as well as $42,723$ fiber links. Snapshots of the power network data and communication network data are shown in Figure 3.

Figure 3(a) provides a snapshot of the power network data, the orange markers show locations of power plants while the yellow continuous lines represent transmission lines. Figure 3(b) provides a snapshot of the communication network, the pink markers show the location of fiber-lit buildings, the orange markers show the location of cell towers, while the green continuous lines represent the fiber links. In this data set, "load" in the Power Network was sub-divided into Cell towers and Fiber-lit buildings. Other load bearing entities, such as residential houses, were not considered as part of these experiments. Thus in the power network, three types of Power Network Entities (PNEs) were considered - Generators (a_1 type entities), Load – consisting of Cell towers and Fiber-lit buildings (a_2 type entities), and Transmission lines (a_3 type entities). For the communication network, three types of Communication Network Entities (CNEs) were considered - Cell Towers

(b_1 type entities), Fiber-lit buildings (b_2 type entities) and Fiber links (b_3 type entities). Fiber-lit buildings were also considered as communication network entities as these buildings house routers which are communication network equipment. From this data set two networks A and B were constructed using the following rules:

It was considered that a PNE is dependent on a set of CNEs for being in an operable state ("active") or being in an inoperable state ("inactive"). Similarly, a CNE was also dependent on a set of PNEs for being in a active or inactive state. For simplicity, in the experiments it was considered that the IDRs had at most two minterms and the size of each minterm is at most two.

Generators ($a_{1,i}, 1 \leq i \leq p$, where p is the total number of generators): It was assumed that each generator ($a_{1,i}$) is dependent on the nearest Cell Tower ($b_{1,j}$) or the nearest Fiber-lit building ($b_{2,k}$) and the corresponding Fiber link ($b_{3,l}$) connecting $b_{2,k}$ and $a_{1,i}$. Thus, the IDR was setup as: $a_{1,i} \leftarrow b_{1,j} + b_{2,k} \times b_{3,l}$

Load ($a_{2,i}, 1 \leq i \leq q$, where q is the total number of loads): It was assumed that the loads in the power network do not depend on any CNE.

Transmission Lines ($a_{3,i}, 1 \leq i \leq r$, where r is the total number of transmission lines): It was assumed that the transmission lines do not depend on any CNE.

Cell Towers ($b_{1,i}, 1 \leq i \leq s$, where s is the total number of cell towers): It was assumed that the cell towers depend on the nearest pair of generators and the corresponding transmission line connecting the generator to the cell tower. Thus, the IDR was setup as: $b_{1,i} \leftarrow a_{1,j} \times a_{3,k} + a_{1,j'} \times a_{3,k'}$

Fiber-lit Buildings ($b_{2,i}, 1 \leq i \leq t$, where t is the total number of fiber-lit buildings): It was assumed that the fiber-lit buildings depend on the nearest pair of generators and the

corresponding transmission lines connecting the generators to the cell tower. Thus, the IDR was setup as: $b_{2,i} \leftarrow a_{1,j} \times a_{3,k} + a_{1,j'} \times a_{3,k'}$

Fiber Links ($b_{3,i}, 1 \leq i \leq u$, where u is the total number of fiber links): It was assumed that the fiber links do not depend on any PNE.

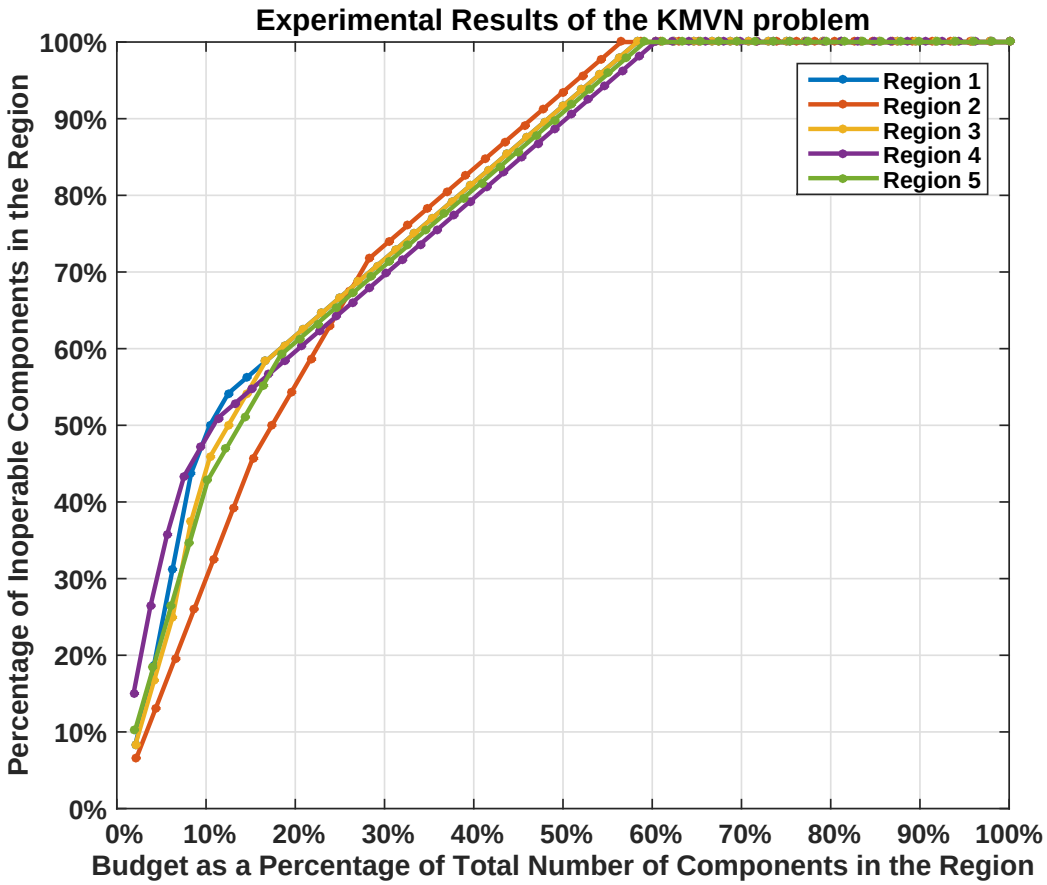


Figure 4: Experimental results of failure vulnerability across five regions of Maricopa county, Arizona

Because of resource limitations on available experimental equipment, the experiments considered five non-overlapping circular geographical regions of Maricopa County. IBM CPLEX Optimizer 12.5 was used to run the formulated ILP's on the experimental data set.

The results of these experiments are shown in Figure 4. It was observed that in each of the regions, there was a specific budget threshold beyond which each additional increments in the budget resulted in the failure of only one additional entity. The reason for this behavior is that the assumption that entities such as the transmission lines and the fiber links are not dependent on any other entities. These "immune" entities begin independently failing as more budget becomes available, but they can no longer impact the well being of other entities as these other entities have already failed. It was also observed that in this data set, all entities of the two networks can be destroyed with a budget of approximately 60% of the number of entities of the two networks A and B.

Chapter 5

SMALLEST PSEUDO TARGET SET IDENTIFICATION PROBLEM

5.1 Introduction

As noted in the previous chapters, the interdependent relationships between the entities of the IPCN implies that a failure involving a small set of entities can trigger a *cascading event* that results in the failure of a much larger set of entities. This creates a potential scenario where an adversary with an intent to jeopardize a specific set of entities E', or *real targets*, now no longer needs to destroy E' directly. Instead, the adversary can take advantage of the cascading failure process by identifying a smaller set of entities E'', or *pseudo targets*, whose failure eventually leads to the failure of E' due to the cascade. Thus, the objective of the adversary is to identify the smallest set of pseudo targets E'' whose failure eventually causes E' to fail. In this chapter this problem is referred to as the *Smallest Pseudo Target Set Identification Problem* (SPTSIP), and in the IIM setting, the problem is categorized in four classes. It is shown that one class of the problem is solvable in polynomial time, whereas for others it is NP-complete. For the second class of the problem an *approximation algorithm* is provided, and for the most general form of the problem an optimal solution using an ILP, and a heuristic solution is provided. Finally, the efficacy of the heuristic is evaluated using power and communication network data of Maricopa County, Arizona.

5.2 Problem Formulation and Computational Complexity Analysis

In this section the *Smallest Pseudo Target Set Identification Problem* (SPTSIP) is formally stated in the IIM setting, and its complexity is analyzed for different types of dependency relations. The SPTSIP is formulated as follows:

The Smallest Pseudo Target Set Identification Problem

INSTANCE: Given:

(i) the set A and B representing the entities of the power and communication networks respectively with $n = |A|$, $m = |B|$

(ii) a set of dependency relations or IDRs, between A and B type entities

(iii) the set of *real targets* $E' \subseteq A \cup B$

(iv) positive integer K

QUESTION: Is there a subset $E'' \subseteq A \cup B$ of *pseudo targets*, with $|E''| \leq K$, whose failure at time step 0, triggers a cascade of failures resulting in failure of the real target set E' by time step $p = n + m - 1$?

The assumptions for the SPTSIP are outlined as follows: First, it is assumed that an entity $e_i \in A \cup B$ can fail by itself and not due to its dependencies, only at time step 0. Any failures after time step 0 occur due to the cascade effect of entities that failed at time step 0. Second, it is assumed that dependent entities immediately fail in the next time step, i.e. if $e_i \leftarrow e_j e_k$, and e_k fails at time step $p-1$, then e_i fails at p. Third, time step $p = n + m - 1$ is a sure end of any failure cascade that begins at time step 0 as there are at most $n + m$ entities and it is assumed that entities cannot become operational once they fail. Finally, the pseudo target set E'' does not have to be unique.

To analyze the complexity of the SPTSIP, the type of IDRs encountered in IPCNs is

categorized in terms of the number of minterms they contain, and the size of each minterm. The complexity of each of these categories is analyzed as follows:

5.2.1 Case I: Problem Instance with One Minterm of Size One

For Case I the IDR's are represented as: $x_i \leftarrow y_j$, where x_i and y_j are elements of the set A (B) and B (A) respectively. In the example $a_i \leftarrow b_j$, $x_i = a_i$, $y_1 = b_j$. A conjunctive implication of the form $a_i \leftarrow b_j b_k$ can be written as two separate IDRs $a_i \leftarrow b_j$ and $a_i \leftarrow b_k$. However, this case is considered in Case II and not in Case I. This exclusion implies that the entities that appear on the left hand side of an IDR in Case I are unique. For Case I, Algorithm 2 presents a polynomial time algorithm for the solution of the SPTSIP.

Algorithm 2 Time Complexity: Since $|A| + |B| = n + m$. Step 2 takes $O(n + m + |S|)$ time. Step 3 can be executed in $O((n+m)^3)$ time. Step 4 takes at most $O((n+m)^3)$ time. The while loop in Step 6 takes at most $O(n(n+m))$. Therefore the overall complexity of Algorithm 2 is $O((n+m)^3)$.

Theorem 5 *For each pair of transitive closure sets C_{x_i} and C_{x_j} produced in Step 3 of Algorithm 2, either $C_{x_i} \cap C_{x_j} = \emptyset$ or $C_{x_i} \cap C_{x_j} = C_{x_i}$ or $C_{x_i} \cap C_{x_j} = C_{x_j}$, where $x_i \neq x_j$.*

Proof: The theorem is proved by contradiction, assume there exists a pair of transitive closure sets C_{x_i} and C_{x_j} such that $C_{x_i} \cap C_{x_j} \neq \emptyset$, $C_{x_i} \cap C_{x_j} \neq C_{x_i}$ and $C_{x_i} \cap C_{x_j} \neq C_{x_j}$. Let $x_k \in C_{x_i} \cap C_{x_j}$, this implies that there exists a path $P1$ from x_i to x_k, as well as a path $P2$ from x_j to x_k. Thus there exists some x_l such that $x_l \in P1$ and $x_l \in P2$. Without loss of generality assume that x_l is the first node in $P1$, also, as $x_l \in P2$, x_l has an in-degree greater than 1. This implies that there is more than one IDR in set S such that x_l appears on the left hand side of these IDRs. This is a contradiction as it violates the definition of Case I type IDRs and hence the theorem is proved. ■

Algorithm 2: Case I Optimal Algorithm for SPTSIP

Data:
1. Set of network entities $A \cup B$, with $n = |A|$ and $m = |B|$
2. A set S of IDRs of the form $y \leftarrow x$, where $x, y \in A \cup B$
3. A set of *real targets* E'

Result: The smallest set of *pseudo targets* E'' such that if E'' fails at time step 0, the real target set E' fails by time step $p = n + m - 1$

1 **begin**
2 Construct a directed graph $G = (V, E)$, where $V = A \cup B$. For each IDR $y \leftarrow x$ in S, where $x, y \in A \cup B$, introduce a directed edge $(x, y) \in E$;
3 For each node $x_i \in V$, construct a transitive closure set C_{x_i} as follows: If there is a path from x_i to some node $y_i \in V$ in G, then include y_i in C_{x_i}. As $|A| + |B| = n + m$, there will be $n + m$ transitive closure sets $C_{x_i}, 1 \leq i \leq (n + m)$. Each x_i is termed as the *seed entity* for the transitive closure set C_{x_i};
4 Remove all the transitive closure sets which are proper subsets of some other transitive closure set;
5 $E'' \leftarrow \emptyset$;
6 **while** $E' \neq \emptyset$ **do**
7 For entity $e_j \in E'$, find set C_{x_i} such that $e_j \in C_{x_i}$;
8 Include *seed entity* x_i in E'';
9 $E' \leftarrow E' \setminus C_{x_i}$;
10 **return** E''

Theorem 6 *Algorithm 2 gives an optimal solution for the SPTSIP in a multilayer network for Case I type IDRs.*

Proof: Theorem 5 proves that every pair of the transitive closure sets created in Step 3 of Algorithm 2 are either disjoint or is a proper subset of the other, in Step 4 of the algorithm all transitive closure sets that are proper subsets of some other transitive closure set are removed. This implies that the remaining sets are all necessarily disjoint, and for every $e_i \in E'$, e_i belongs to exactly one transitive closure set. This necessitates that the seed entity x_k of the transitive closure set C_{x_k} that e_i belongs to, must be included in the solution. This is done in the while loop of Step 6. To prove the optimality claim it is required to show that the number of seed entities chosen by the algorithm is minimum. If it is assumed that

the number of seeds chosen is not minimum, then some C_{x_i} chosen by the algorithm must necessarily be a proper subset of another closure. This contradicts Theorem 5, and hence Algorithm 2 always returns the optimal solution. ■

5.2.2 Case II: Problem Instance with One Minterm of Arbitrary Size

For Case II the IDR's are represented as:
$x_i \leftarrow \prod_{k_1=1}^{l} y_{k_1} \prod_{k_2=1}^{q} x_{k_2}$ (with $x_i \neq x_{k_2} \forall x_{k_2}, 1 \leq k_2 \leq q$), where x_i, x_{k_2} are elements of set A (B) and y_{k_2} is an element of set B (A). The size of the minterm is given as $l+q$. In the example $a_r \leftarrow b_u b_v a_s$. $l+q=3$, $x_i = a_r, y_1 = b_u, y_2 = b_v$ and $x_1 = a_s$.

Theorem 7 *The SPTSIP for Case II is NP Complete*

Proof: The SPTSIP for Case II is NP-complete is proved by giving a transformation for the Set Cover (SC) problem [13]. An instance of the set cover problem is specified by a universal set $S = \{s_1, \ldots, s_n\}$ and a set of subsets S', $S' = \{S_1, \ldots, S_m\}$, where $S_i \subseteq S, \forall i, 1 \leq i \leq m$. In the set cover problem one wants to know whether there exists a subset of $S'' \subseteq S'$ such that $\bigcup_{S_i \subseteq S''} S_i = S$ and $|S''| \leq Q$, for some specified integer Q. From an instance of the SC problem an instance of the SPTSIP is created in the following way: For every $s_i \in S$, an IDR of the form $s_i \leftarrow \prod_{s_i \in S_j} S_j$ is created for all $S_j \in S'$. The real target set E' is set equal to S and $K = Q$. It can now easily be verified that the instance of the SC problem has a set cover of size Q, iff in the created instance of SPTSIP the failure of K entities at time step 0 triggers a cascade of failures resulting in the failure of all entities in the set E' by time step $p = n+m-1$. ■

The following definition is now outlined:

Definition: *Kill Set of a set of Entities* $\mathcal{P}$: The *Kill Set* of a set of entities $\mathcal{P}$, denoted by $KillSet(\mathcal{P})$, is the set of all entities in the multilayer network (including $\mathcal{P}$) that fail by $p = n+m-1$ time steps as a consequence of: (i) the failure of $\mathcal{P}$ entities at time step 0, and (ii) the interdependency relationships (IDRs) shared between the entities of the network.

In Algorithm 3 an approximation algorithm is presented for the SPTSIP with Case II type IDRs.

Algorithm 3: Case II Approx. Algorithm for SPTSIP

Data:
1. Set of network entities $A \cup B$, with $n = |A|$ and $m = |B|$
2. A set of IDRs of the form $y \leftarrow \prod_{i=1}^{q} x_i$, where $x_i, y \in A \cup B, \forall 1 \leq i \leq q$
3. A set of *real targets* E', with $M = |E'|$

Result: Set of entities $E'' \subseteq A \cup B$ such that failure of E'' entities in time step 0 results in failure of E' entities by time step $p = n+m-1$.

```
begin
    U ← ∅;
    DEP_i ← ∅, S_i ← ∅,   ∀i = 1,...,M;
    KillSet_j ← ∅,   ∀j = 1,...,n+m;
    foreach e_i ∈ E' do
        foreach entity e_j ∈ IDR e_i ← ∏_{j=1}^{q} e_j do
            DEP_i ← DEP_i ∪ {e_j};
        U ← U ∪ {i};
        S_i ← U ∪ {i};
    foreach e_i ∈ A ∪ B do
        KillSet_i ← KillSet(e_i);
        for d = 1 to M do
            if KillSet_i ∩ DEP_d ≠ ∅ then
                S_i ← S_i ∪ {d};
    E'' ← ∅;
    while U ≠ ∅ do
        Select S_i, i = 1,...,M that maximizes |S_i ∩ U|;
        E'' ← E'' ∪ {e_i};
        U ← U \ S_i;
    return E''
```

Theorem 8 *The approximation solution produced by Algorithm 3 for Case II type IDRs is at most $O(ln(M))$ times the optimal, where $M = |E'|$*

Proof: Algorithm 3 implements a greedy approach for solving a set cover problem. The set cover problem is setup the following way: First, in Steps 5-9, for each entity $e_i \in E'$ dependency DEP_i is constructed as the set of entities out of which at least one entity must fail for e_i to fail, thus "unsatisfying" the dependency. Step 9 accounts for the dependency DEP_i getting unsatisfied due to failure of e_i itself.

The universe U contains the indexes of each of these M dependencies. Next, in Steps 10-14, for each entity $e_i \in A \cup B$ the $KillSet(e_i)$ is computed and set S_i is constructed that contains the index of the dependencies in $DEP_j, j = 1,\ldots,M$ that has a non-empty intersection with $KillSet(e_i)$. This implies that with the failure of e_i and the ensuing failure propagation, DEP_j gets unsatisfied. With the universe set of U and the subsets $S_i, i = 1,\ldots,M$, the greedy technique for set cover is used in Steps 16-19 that yields a known approximation factor of $O(ln(M))$ times the optimal solution [14]. ■

Algorithm 3 Time Complexity: To construct the dependencies in Steps 5-9, at most M IDRs will be traversed each with at most $n+m$ entities hence these steps take $O(M(n+m))$ time. In Steps 10-14 computing the kill set of each of the $n+m$ entities and comparing it to each of the M dependencies of maximum size $n+m$ requires $O(M(n+m)^3)$ time. And finally, the greedy set cover in Steps 16-19 takes $O(M\ log(n+m))$. Overall the complexity of Algorithm 3 is $O(M(n+m)^3)$.

5.2.3 Case III: Problem Instance with an Arbitrary Number of Minterms of Size One

For Case III an IDR has the following form:
$x_i \leftarrow \sum_{k_1=1}^{l} y_{k_1} + \sum_{k_2=1}^{q} x_{k_2}$ (with $x_i \neq x_{k_2} \forall x_{k_2}, 1 \leq k_2 \leq q$), where x_i, x_{k_2} are elements of set A (B) and y_{k_2} is an element of set B (A). The size of the minterm is given as $l+q$. In the example $a_r \leftarrow b_u + b_v + a_s$. $l+q=3$, $x_i = a_r, y_1 = b_u, y_2 = b_v$ and $x_1 = a_s$.

Theorem 9 *The SPTSIP for Case III is NP Complete*

Proof: The SPTSIP for Case III is proved to be NP-complete by giving a transformation for the Vertex Cover (VC) problem [13]. An instance of the vertex cover problem is specified by an undirected graph $G = (V,E)$ and an integer R. In the vertex cover problem, one wants to know whether there is a subset $V' \subseteq V$ such that $|V'| \leq R$, and for every edge $e \in E$, at least one end vertex of e is in V'. From an instance of the VC problem an instance of the SPTSIP is created in the following way: From the graph $G = (V,E)$ for each vertex $v_i \in V$ that has adjacent nodes (say) v_j, v_k and v_l, an IDR $v_i \leftarrow v_j + v_k + v_l$ is created. The real target set E' is set equal to V and $K = R$. It can now be verified that the instance of the VC problem has a vertex cover of size R, iff in the created instance of SPTSIP the failure of K entities at time step 0 triggers a cascade of failures resulting in the failure of all entities in the set E' by time step $p = |V| - 1$. ■

5.2.4 Case IV: Problem Instance with an Arbitrary Number of Minterms of Arbitrary Size

This is the general case, where IDRs have arbitrary number of minterms of arbitrary size.

Theorem 10 *The SPTSIP for Case IV is NP Complete*

Proof: As both Case II and Case III are special cases of Case IV, the SPTSIP for Case IV is NP-Complete as well. ■

5.3 Algorithms for the SPTSIP

In this section an optimal solution for the SPTSIP using Integer Linear Programming (ILP) is proposed along with a polynomial time heuristic solution.

5.3.1 Optimal Solution for the SPTSIP

An optimal solution for the SPTSIP is formulated with an ILP that uses two variables x_{it} and y_{jt}. Where $x_{it} = 1$, when entity $a_i \in A$ is in a failed state at time step t, and 0 otherwise. And, $y_{jt} = 1$, when entity $b_j \in B$ is in a failed state at time step t, and 0 otherwise.

The objective function can now be formulated as follows:

$$\min \sum_{i=1}^{n} x_{i0} + \sum_{j=1}^{m} y_{j0} \tag{5.1}$$

Where $n = |A|$ and $m = |B|$. The constraints are as follows:

Failure Consistency Constraints: $x_{it} \geq x_{i(t-1)}, \forall t, 1 \leq t \leq p$, these constraints ensure that if an entity a_i fails at time step t, it continues to remain in a failed state for all subsequent time steps. A similar constraint applies for y_{it} variables.

Failure Propagation Constraints: These constraints govern the failure cascade process caused by the dependencies shared between the network entities. The constraints were

elaborated in Section 4.2.2.2, an overview is outlined of these constraints here for consistency. For any Case IV type IDRs of the form $a_i \leftarrow b_j b_k b_l + b_v b_u + b_q$ the subsequent steps are followed to model the failure propagation:

Step 1: Transform the IDR to a disjunctive form of size one minterms, i.e. $a_i \leftarrow c_1 + c_2 + b_q$.

Step 2: For each of the c type minterms create constraints to model the failure cascade for individual c type minterms, i.e. for $c_1 \leftarrow b_j b_k b_l$ introduce $c_{1t} \leq y_{j(t-1)} + y_{k(t-1)} + y_{l(t-1)}, \forall t, 1 \leq t \leq p$.

Step 3: For each transformed IDR from Step 1, for example $a_i \leftarrow c_1 + c_2 + b_q$, introduce a constraint of the form $N \times x_{it} \leq c_{1(t-1)} + c_{2(t-1)} + b_q, \forall t, 1 \leq t \leq p$, where N is the number of minterms in the transformed IDR, in this example $N = 3$.

Prior to the transformation of Step 1, if an IDR does not contain any disjunctions (Case II), then Step 3 is skipped, or if it does not contain any conjunctions (Case III), then Step 2 is skipped.

Real Target Set Failure Constraints: $x_{ip} = 1, \forall a_i \in E'$, and $y_{ip} = 1, \forall b_i \in E'$, these set of constraints ensure that all entities of the real target set E' are in a failed state at time step p.

Adhering to the above constraints, the objective in (5.1) minimizes the total number of entities that need to fail at time step 0 so that E' entities fail by time step p.

5.3.2 Heuristic Solution for the SPTSIP

Before the heuristic solution is presented, the following definition is first outlined:

Definition: *Kill Impact of a set of Entities* $\mathcal{P}$: The Kill Impact of a set of entities $\mathcal{P}$, denoted by $KillImpact(\mathcal{P})$, is defined as the contribution of $\mathcal{P}$ entities in causing the failure of entities in E'. It may be noted that any entity $e_i \in E'$ can fail due to two reasons: (i) when e_i itself fails at time step 0, or (ii) when at least one entity in all the minterms of e_i's IDR

fail in some time step. $KillImpact(\mathcal{P})$ captures these two aspects by computing the impact of failure of $\mathcal{P}$ entities on E' based on: (i) the number of entities that fail in E' at time step p when $\mathcal{P}$ entities fail at time step 0, and (ii) the number of minterms in the IDR of each entity $e_i \in E'$ that get affected at time step p when $\mathcal{P}$ entities fail at time step 0. For a given set of $\mathcal{P}$ entities, and the set of minterms $MT_i = \{mt_1, mt_2, \ldots, mt_{|MT_i|}\}, mt_j \subseteq A \cup B$, for each entity $e_i \in E'$, to compute $KillImpact(\mathcal{P})$, $impact_i$ is first computed as the impact of failure of $\mathcal{P}$ entities on e_i as follows:

If $e_i \in KillSet(\mathcal{P})$, $impact_i = 1$, else if $e_i \notin KillSet(\mathcal{P})$:

$$impact_i = \frac{\left|\bigcup_{mt_j \cap KillSet(\mathcal{P}) \neq \emptyset} mt_j\right|}{|MT_i|}, \quad \forall mt_j \in MT_i \tag{5.2}$$

$KillImpact(\mathcal{P})$ is then computed as follows:

$$KillImpact(\mathcal{P}) = \frac{\sum_{i=1}^{|E'|} impact_i}{|\mathcal{P}|} \tag{5.3}$$

In Algorithm 4, a heuristic technique is presented to solve the SPTSIP for the general case of the problem. The general approach for Algorithm 4 is to greedily select a set of entities that provide the maximum benefit towards reaching the objective of failing E'. In Steps 5-7, for each entity $e_j \in A \cup B$, how frequently e_j appears in all minterms in the set of IDRs and $KillImpact(e_j)$ is computed. Next, in Steps 8-14, for each entity $e_i \in E'$, each of the minterms of e_i's IDR is examined and the highest frequency entity of each minterm is selected to construct set k_i and then $KillImpact(k_i)$ is computed. In Step 15 the most impactful set of entities is chosen from the total $KImpact$ sets constructed. Intuitively, this selection of a higher kill impact set for inclusion implies more failures in the target set. Also, since the objective is to minimize the size of the entities selected, a set with the largest impact to size ratio is preferred. Finally, the algorithm proceeds to update E'' and $failed$ set of entities, and prunes the IDR set and minterm set in Steps 16-19. This greedy

Algorithm 4: Case IV Heuristic for SPTSIP

Data:
1. Set of network entities $A \cup B$, with $n = |A|$ and $m = |B|$
2. A set S of IDRs of type Case IV (general case)
3. A set of *real targets* E'

Result: A set of *pseudo targets* E'' such that when E'' fails at time step 0, the real target set E' fails by time step $p = n + m - 1$

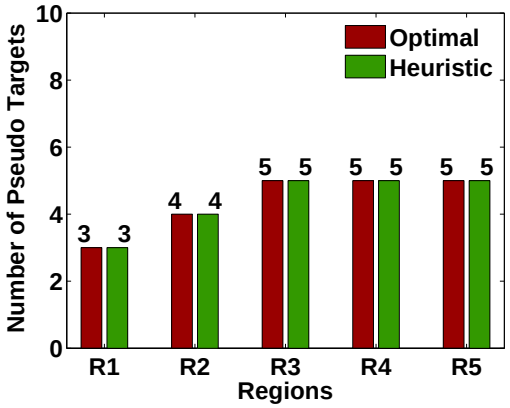
begin
  $E'' \leftarrow \emptyset$, $failed \leftarrow \emptyset$;
  while $E' \not\subseteq failed$ **do**
    $KImpact \leftarrow \emptyset$;
    foreach *entity* $e_j \in A \cup B$ *and* $e_j \notin failed$ **do**
      Compute $frequency_j$ as the number of times e_j appears in a minterm for all IDRs in S;
      $KImpact \leftarrow KImpact \cup (e_j, KillImpact(e_j))$;
    foreach *entity* $e_i \in E'$ *and* $e_i \notin failed$ **do**
      Let idr in S be the IDR of entity e_i;
      $k_i \leftarrow \emptyset$;
      foreach *minterm* MT *in* idr **do**
        Select entity $e_j \in MT$ with largest $frequency_j$ from all entities in MT;
        $k_i \leftarrow k_i \cup e_j$;
      $KImpact \leftarrow KImpact \cup (k_i, KillImpact(k_i))$;
    Select tuple $(failSet, failVal) \in KImpact$ where $failVal \geq val, \forall (set, val) \in KImpact$;
    $E'' \leftarrow E'' \cup failSet$;
    $failed \leftarrow KillSet(E'')$;
    Remove IDR of entity e_k from S, $\forall e_k \in failed$;
    For each IDR in S remove all minterms that contain entity e_k, $\forall e_k \in failed$;
  return E''

selection process repeats until $E' \subseteq failed$. The heuristic ensures that for every iteration of the while loop in Step 3 the E'' set increases in such a way that at least one additional entity in E' fails than the previous iteration, thus moving closer to the objective. Algorithm 4 runs in polynomial time, specifically it runs in $O(M(n+m)^4)$ time, where $M = |E'|$. In Section 5.4, the experiments show that Algorithm 4 almost always produces the optimal result.

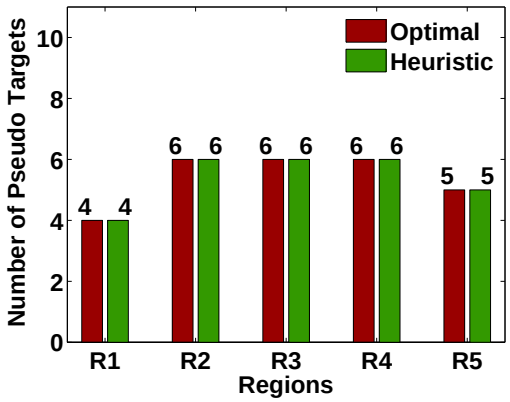
5.4 Experimental Results

The experimental results for the SPTSIP is presented in this section along with a comparison between the optimal solution (computed using an ILP), and the proposed heuristic algorithm is also presented. The experiments were conducted on power and communication network data of Maricopa County, Arizona. The power network data was obtained from Platts (www.platts.com), and the communication network data obtained from Geo-Tel (www.geo-tel.com). This data consisted of 70 power plants, 470 transmission lines, 2,690 cell towers, 7,100 fiber-lit buildings and 42,723 fiber links. Five non-intersecting geographical regions were identified, and from the consolidated power and communication network data of each region, interdependencies between the network entities were setup using the rules outlined in Section 4.3. For continuity, an overview of these rules are also outlined here: For each generator to be operational, either (i) the nearest cell tower must be operational, or (ii) the nearest fiber-lit building and the fiber link connecting the generator to the fiber-lit building must be operational. For each fiber-lit building and cell tower to be operational, at least one of the two nearest generators and the connecting transmission lines must be operational. The transmission lines and the fiber links have no dependencies.

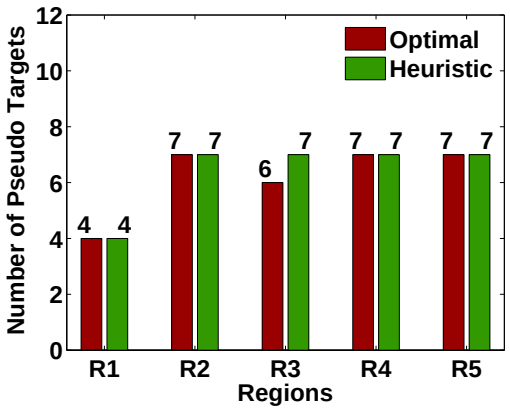
The optimal solutions were obtained by solving Integer Linear Programs using the IBM CPLEX Optimizer 12.5. For each of the five regions $R1$ through $R5$, real target sets of entities of sizes 5, 10, 15 and 20 were chosen from the set of all power and communication entities of that region. For each real target set the optimal and heuristic solutions were computed, and these results are presented in Figure 5. The experiments showed that for the five regions considered, in the worst case the heuristic solution differed from the optimal by a factor of 0.16, in the best case was equal to the optimal, and on an average was within a factor of 0.02 of the optimal solution.



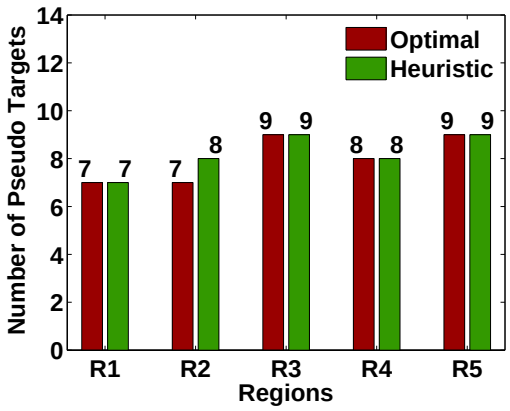
(a) Real Target Set Size: 5



(b) Real Target Set Size: 10



(c) Real Target Set Size: 15



(d) Real Target Set Size: 20

Figure 5: Comparison of optimal and heuristic approaches for computing pseudo targets (E''), for given real targets (E') of sizes 5, 10, 15 and 20, on five geographical regions of Maricopa County, Arizona.

Chapter 6

READER PATH PLANNING FOR TAG ACCESS IN MOBILE RFID SYSTEMS

6.1 Introduction

RFID systems, comprising of *readers* and *tags*, are used extensively for identification of objects with unique identifiers. In order to support the complex needs of RFID dependent business sectors, such as Supply Chain Management and Transportation, a RFID system is expected to allow readers fast and accurate access to tags available in the environment. The operating environment of RFID systems can be divided into three classes: (i) *both readers and tags are stationary*, (ii) *either tags or readers are mobile*, and (iii) *both readers and tags are mobile*. A number of studies have been undertaken in the last few years that consider the scenarios where both tags and readers are stationary, or either the tags or the readers are mobile.

Although, the environment where both the tags and readers are mobile is not widespread in the industry today, given that current warehouse infrastructures, such as the ones owned by Amazon that contain 15 miles of conveyor belts [15], and low cost drones are readily available, it is believed that it is only a matter of time when drone mounted RFID readers will be used in large warehouses [16] to read the RFID tags of items moving through a complex layout of conveyor belts spanning a wide area. In such a setting, the trajectories of the tags are known (as they are constrained to move on conveyor belts, whose layout is known). In this environment, where both tags and readers are mobile and the trajectories of the tags are known, one may want to compute the (i) minimum number of readers, and (ii) their trajectories, that will ensure that all tags available in the deploy-

ment area are read within a pre-specified time – a problem that may be viewed as a *path planning problem* for *mobile readers*. Although a number of studies have been undertaken in the last few years that consider the scenarios where both tags and readers are stationary [17], [18], or either the tags or the readers are mobile [19]–[22], the *Reader Minimization Problem* (RMP) in the RFID context where both readers and tags are mobile, has not been studied in the past.

6.2 Reader Minimization Problem

The goal of the Reader Minimization Problem (RMP) is to find the fewest number of mobile readers and their trajectories that are sufficient to read all mobile tags in the deployment area within a specified period of time. In this problem it is assumed that a central controller has knowledge of: (i) the number of tags in the deployment area, (ii) the tag trajectories, (iii) location of tags at time $t = 0$, (iv) the speed of movement of tags, and (v) the maximum speed of the readers. It may be noted that today's large warehouses contain miles of conveyor belts that can carry RFID tagged items. For example, as reported in [15], Amazon's warehouse in Baltimore, USA, contains 15 miles of conveyor belts. In this setting, (a) the layout of the conveyor belts, (b) the speed of the conveyor belts, and (c) the rate of uploading of items on the conveyor belt, are all known, thus justifying the above assumptions. With this information, a centralized controller can compute the minimum number of readers and their trajectories required to read all tags given the readers' maximum speed.

The RMP is elaborated with the help of an example: Figure 6 shows the trajectories of six mobile tags (shown with thin colored lines) on a two dimensional deployment area, and their corresponding locations at times $t = 0$ to $t = 14$. For this example, it is assumed

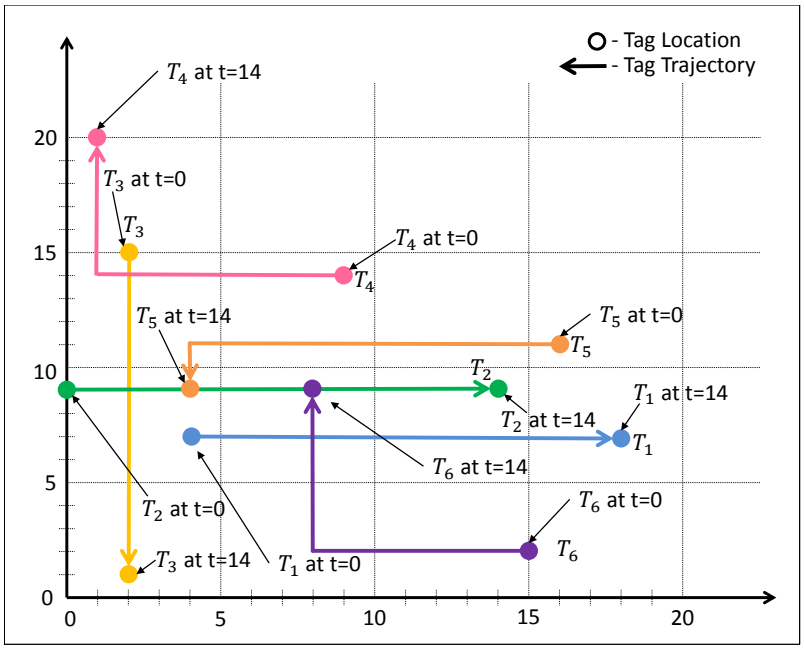


Figure 6: Location of tags $T_1, \ldots, T_6$, their trajectories from time [0-14].

that (i) speed of both tags and readers are 1 unit/sec (where the unit of measure can be feet, meter, etc.), (ii) the reader sensing range is 1 unit, i.e. a reader can read a tag only when the distance between the reader and tag is at most 1 unit, and (iii) all tags must be read by time $t = 14$. The goal of the RMP is to find the minimum number of readers and their trajectories, required to read all tags.

As shown in Figure 7, for this example, only one mobile reader is sufficient to read all mobile tags. The trajectory of the mobile reader is shown with a thick red line. The locations where the reader reads the tags (*rendezvous points*), are shown with black rectangles in Figure 7 and in bold letters in Table 4. For instance, at time $t = 0$, when reader R_1 is at location $(3,7)$, it reads tag T_1 which is at location $(4,7)$ at time $t = 0$. Similarly, when R_1 is at location $(7,9)$ at time $t = 14$, it reads T_6 which is at location $(8,9)$ at time $t = 14$.

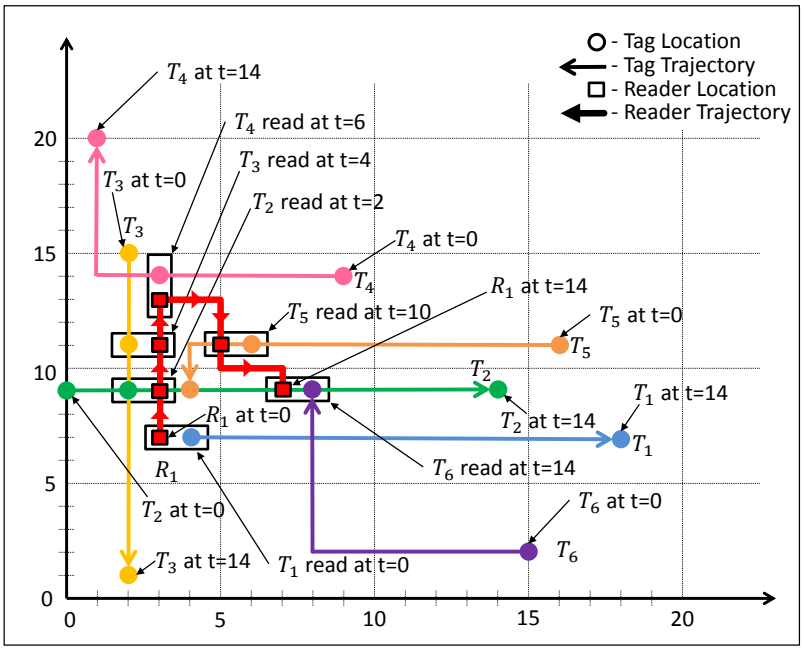


Figure 7: A single reader (R_1) is sufficient to read all tags of Fig. 6 within time step 14. Trajectory of R_1 is shown with a thick red line and the *rendezvous points* where the reader reads the tags are shown in black rectangles.

It is now shown that even when all the tags are stationary, finding the minimum number of stationary readers to read all tags is computationally hard.

Theorem 11 *Reader Minimization Problem is NP-complete.*

Proof: The special case of the RMP where both tags and readers are stationary is equivalent to the Geometric Disk Cover Problem which is known to be NP-complete [23]. ■

It is only conceivable that when both the tags and readers are mobile, computation of the minimum number of readers needed to read all the tags within a specified time and finding the reader trajectories will be considerably harder than the scenario when both the tags and readers are stationary.

Tags/ Readers	Time Steps (t)							
	0	1	2	3	4	5	6	7
R_1	**(3,7)**	$(3,8)$	**(3,9)**	$(3,10)$	**(3,11)**	$(3,12)$	**(3,13)**	$(4,13)$
T_1	**(4,7)**	$(5,7)$	$(6,7)$	$(7,7)$	$(8,7)$	$(9,7)$	$(10,7)$	$(11,7)$
T_2	$(0,9)$	$(1,9)$	**(2,9)**	$(3,9)$	$(4,9)$	$(5,9)$	$(6,9)$	$(7,9)$
T_3	$(2,15)$	$(2,14)$	$(2,13)$	$(2,12)$	**(2,11)**	$(2,10)$	$(2,9)$	$(2,8)$
T_4	$(9,14)$	$(8,14)$	$(7,14)$	$(6,14)$	$(5,14)$	$(4,14)$	**(3,14)**	$(2,14)$
T_5	$(16,11)$	$(15,11)$	$(14,11)$	$(13,11)$	$(12,11)$	$(11,11)$	$(10,11)$	$(9,11)$
T_6	$(15,2)$	$(14,2)$	$(13,2)$	$(12,2)$	$(11,2)$	$(10,2)$	$(9,2)$	$(8,2)$
Tags/ Readers	Time Steps (t)							
	8	9	10	11	12	13	14	15
R_1	$(5,13)$	$(5,12)$	**(5,11)**	$(5,10)$	$(6,10)$	$(7,10)$	**(7,9)**	–
T_1	$(12,7)$	$(13,7)$	$(14,7)$	$(15,7)$	$(16,7)$	$(17,7)$	$(18,7)$	–
T_2	$(8,9)$	$(9,9)$	$(10,9)$	$(11,9)$	$(12,9)$	$(13,9)$	$(14,9)$	–
T_3	$(2,7)$	$(2,6)$	$(2,5)$	$(2,4)$	$(2,3)$	$(2,2)$	$(2,1)$	–
T_4	$(1,14)$	$(1,15)$	$(1,16)$	$(1,17)$	$(1,18)$	$(1,19)$	$(1,20)$	–
T_5	$(8,11)$	$(7,11)$	**(6,11)**	$(5,11)$	$(4,11)$	$(4,10)$	$(4,9)$	–
T_6	$(8,3)$	$(8,4)$	$(8,5)$	$(8,6)$	$(8,7)$	$(8,8)$	**(8,9)**	–

Table 4: (x,y) coordinates of reader R_1, and tags $T_1,\ldots,T_6$ during time steps $t=0$ to $t=14$ following the trajectories shown in Fig. 7

6.3 RMP Solution Approach

In the following sections it is shown how the RMP can be solved by the generalization of the well-known minimum flow problem [24] on a graph (called the *RMP Graph*) created from an instance of the RMP. It may be noted that an instance of the RMP comprises of (i) the number of tags in the deployment area, (ii) the tag trajectories, (iii) location of tags at time $t=0$, (iv) the speed of movement of tags, and (v) the maximum speed of the readers. Although movement of the tags are primarily in a two or three dimensional space, the solution approach described below considers that the tag movements are restricted to a one dimensional space (i.e., tags move on a line in both directions). The explanation of the RMP graph construction process from the RMP instances where tag movement is restricted to one dimensional space is presented due to its simplicity. Later in the chapter it

is explained how the concepts described for the one dimensional scenario can be extended to two or three dimensions.

Consider a set of n tags $\mathcal{A} = \{a_0, \ldots, a_{n-1}\}$ moving on a one dimensional space (i.e., a line) over time instances $0, \ldots, T$. It may be noted that although the movement of tags is restricted to one dimension, there is no restriction on the direction of their movement, i.e. tags can move left and/or right and can change directions arbitrarily. Let $p(a_i,t) = x(a_i,t)$ be the location of tag a_i at time instance t where $x(a_i,t)$ denotes the x-coordinate of a_i at time t. Assume that a tag can be *read* by a reader R_j only if the distance between them is less than sensing radius of the reader r. In the following section the RMP Graph construction process is described.

6.4 RMP Graph Construction

In this section, the RMP graph construction process is described through an example, where the movements of the tags are restricted to a one dimensional space (i.e., the tags can move only *left* or *right* on a *straight line*). As noted earlier, this restriction is imposed only to explain the graph construction process in a simple way. Once the construction process is understood, the same principle can be followed for constructing the RMP graph where the tags are moving in a two or a three dimensional space. It may recalled that each tag is required to be read by at least one reader before the tag movement period is over. The goal of the RMP is to read all tags using as few readers as possible. Although the RMP is a *continuous time domain* problem, (as the mobile tags and readers can be anywhere in the deployment area at a given time), the proposed solution discretizes both time and space. Time is discretized into equal intervals of length δ, and space into equal *units* of size ε. Depending on the spatial dimension of the problem instance, i.e., one, two, or three

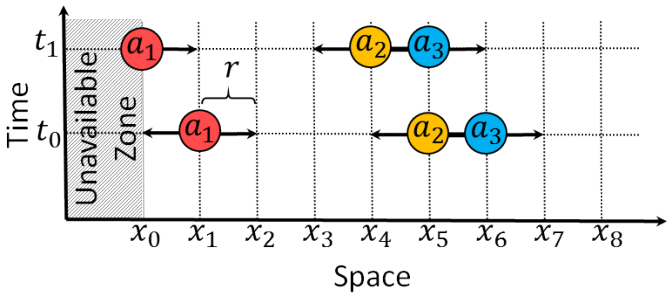


Figure 8: Locations of three tags on one dimensional space (line) at two different instances of time

dimensions, the unit will imply an *interval* of length ε, a *square* with length of each side ε, and a cube with length of each side ε, respectively. It may be noted that, although in the following sub-sections the discussion is focused on a one dimensional space, the proposed techniques are applicable to higher dimensions as well.

As shown in Figure 8, this example has three tags and the movement period comprises of two time steps. The tag a_1 is at location x_1 at time t_0 and in location x_0 at time t_1. Similarly, the tag a_2 is at location x_5 at time t_0 and in location x_4 at time t_1 and the tag a_3 is at location x_6 at time t_0 and in location x_5 at time t_1. Although in this example, all the tags are moving to the left direction at the same speed, the tags are neither required to move in the same direction nor at the same speed. As noted earlier, a reader can read a tag only if the tag is within the reader's sensing radius r. In the example of Figure 8 it is assumed that $r = \varepsilon$. It can thus be observed that in order to read tag a_1, there must be a reader at (x_0,t_0) or (x_1,t_0) or (x_2,t_0) or (x_0,t_1) or (x_1,t_1), where (x_i,t_j) indicates location x_i at time t_j. Using similar reasoning it can be concluded that in order to read tag a_2, there must be a reader at (x_4,t_0) or (x_5,t_0) or (x_6,t_0) or (x_3,t_1) or (x_4,t_1) or (x_5,t_1). Lastly, in order to read tag a_3, there must be a reader at (x_5,t_0) or (x_6,t_0) or (x_7,t_0) or (x_4,t_1) or (x_5,t_1) or (x_6,t_1).

The RMP graph $G = (V, E)$ is a directed graph and is constructed in the following way. It may be noted that for each tag $a_i, 1 \leq i \leq n$ to be read by a reader R_j, the distance between the reader and the tag cannot exceed the sensing range of the reader. Corresponding to each tag $a_k, 1 \leq k \leq n$, there exists a set of $LT_k = (location, time)$ pairs, denoted by the variables $(X_{k,i}, T_{k,i})$, and a reader must be in at least one of the locations at the specific time to be able to read the tag a_k. In other words, at least one element of the set LT_k (say, $(X_{k,i}, T_{k,i})$) must be *chosen* in order to *satisfy* the requirement that the tag a_k must be read. In the example of Figure 8:

$$LT_1 = \{(X_{1,1}, T_{1,1}), (X_{1,2}, T_{1,2}), \ldots, (X_{1,5}, T_{k,5})\},$$
$$LT_2 = \{(X_{2,1}, T_{2,1}), (X_{2,2}, T_{2,2}), \ldots, (X_{2,6}, T_{k,6})\},$$
$$LT_3 = \{(X_{3,1}, T_{3,1}), (X_{3,2}, T_{3,2}), \ldots, (X_{3,6}, T_{k,6})\}$$

Let LT_k be denoted by $(location, time)$ pair variables as follows: $LT_k = \{(X_{k,1}, T_{k,1}), (X_{k,2}, T_{k,2}), \ldots, (X_{k,p_k}, T_{k,p_k})\}$. Corresponding to each $LT_k, 1 \leq k \leq n$, in the graph $G = (V, E)$, there are, (i) $X_{k,i}$ type nodes, (ii) $T_{k,i}$, type nodes, and (iii) a directed edge from the node $X_{k,i}$ to the node $T_{k,i}$. It may be noted that the $(location,\ time)$ pair values of the $(X_{i,j}, T_{i,j})$ variable need not be *unique* and that the variable pairs $(X_{i,j}, T_{i,j})$ and $(X_{k,l}, T_{k,l})$ may represent the same $(location,\ time)$ value pair. In the event when two $(X_{i,j}, T_{i,j})$ variables have the same $(location,\ time)$ pair values, only one pair of nodes representing the $(location,\ time)$ pair values are created in the graph $G = (V, E)$. In the example of Figure 8, variables $(X_{2,2}, T_{2,2})$ and $(X_{3,1}, T_{3,1})$ have the same $(location,\ time)$ pair values of (x_5, t_0), thus only one pair of nodes corresponding to location x_5 at time t_0 will be created in G. This graph construction is shown in Figure 9. The $X_{i,j}$ type nodes are referred to as *location nodes*, and $T_{i,j}$ type nodes are referred to as *time nodes*. In addition to these nodes, one source node S and one sink node D is also created in G. In addition to the directed edges of type $X_{k,i} \rightarrow T_{k,i}$, three additional types of edges in G are also included:

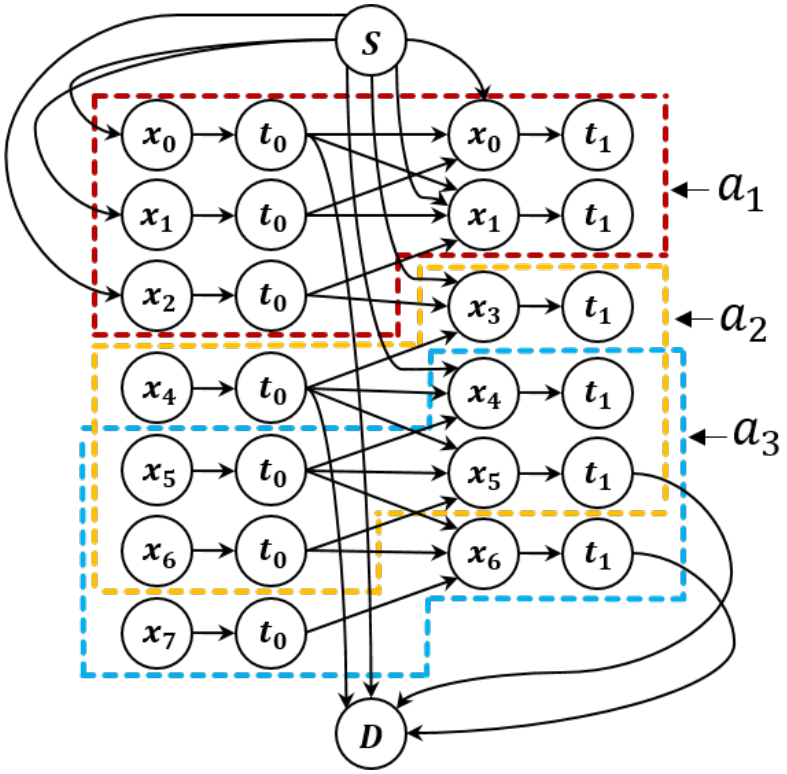


Figure 9: RMP graph $G = (V, E)$ constructed from the problem instance of Fig. 8

1. Mobility edges: If a reader located at x_a at time t_b, can move to a location x_c at time t_d, then in the graph $G = (V, E)$, a directed edge from the node t_b to x_c is added. It may be noted that whether the reader can move from location x_a at time t_b to a location x_c at time t_d, depends on (i) the distance between the locations x_a and x_b, (ii) the time interval between t_c and t_d, and (iii) the maximum speed of the reader.
2. Source edges: There is a directed edge from the source node S to all location nodes.
3. Sink edges: There is a directed edge from all time nodes to the sink node D.

The RMP graph $G = (V, E)$ constructed for the problem instance with three tags in Figure 8 is shown in Figure 9. It may be noted that not all source, sink and mobility edges are shown in Figure 9 for the sake of clarity in the diagram.

6.5 Minimum Flow Problem, Generalized Minimum Flow Problem, and Their Relation to the RMP

In this section the minimum flow problem [24] is defined, and then a generalized version of the problem is stated. Finally, it is shown how the RMP can be solved by solving the generalized minimum flow problem on the RMP graph.

Minimum Flow Problem (MFP): Given a capacitated network $G = (V,E)$ with a non-negative capacity $c(i,j)$ and with a non-negative lower bound $l(i,j)$ associated with each edge (i,j) and two special nodes, a source node S and a sink node D, a flow is defined to be a function $f : E \to \mathbb{R}^+$ satisfying the following conditions:

$$\sum_{j\in V} f(i,j) - \sum_{j\in V} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S,D \\ -F, & i = D \end{cases}$$

$$l(i,j) \leq f(i,j) \leq c(i,j)$$

for some $F \geq 0$ where F is the value of the flow f. The MFP finds a flow f which minimizes F.

Generalized Minimum Flow Problem (GMFP): The generalized version of the MFP is similar to the MFP, except that the lower bound on the flow requirement $l(i,j)$ is no longer associated with an edge (i,j), but associated with a set of edges $E_k \subseteq E$ of the graph $G = (V,E)$, and is denoted by l_k. Formally, the problem can be stated as follows:
Given a capacitated network $G = (V,E)$ with a non-negative capacity $c(i,j)$ associated with each edge (i,j), a set of subsets E' of the edge set E (i.e. $E' = \{E_1,\ldots,E_p\}$, where $E_k \subseteq E, 1 \leq k \leq p$), a lower bound on the flow requirement l_k associated with each $E_k, 1 \leq k \leq p$,

and two special nodes, a source node S and a sink node D. A flow is defined to be a function $f : E \to \mathbb{R}^+$ satisfying the following conditions:

$$\sum_{j\in V} f(i,j) - \sum_{j\in V} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S, D \\ -F, & i = D \end{cases}$$

$\forall E_k, 1 \leq k \leq p, \exists l_k$, a lower bound of flow in E_k, implying that there must exist at least one edge $(i,j) \in E_k$ such that $l_k \leq f(i,j) \leq c(i,j)$

for some $F \geq 0$ where F is the value of the flow f. The generalized minimum flow problem is to determine a flow f for which F is minimized. It may be noted that when $|E_k| = 1, 1 \leq k \leq p$, the GMFP reduces to MFP.

Solving the RMP: It is now shown how solving the GMFP on the RMP graph solves the RMP. First, for the RMP graph $G = (V,E)$ constructed in Section 6.4, the capacity $c(i,j)$ for all edges $(i,j) \in E$ is set to 1. Second, as noted above, an instance of the GMFP has a set of subsets E' of the edge set E (i.e. $E' = \{E_1, \ldots, E_p\}$), where $E_k \subseteq E, 1 \leq k \leq p$, with a lower bound on the flow requirement l_k associated with each E_k. If l_k is the lower bound of flow in E_k, the GMFP requires that there must exist at least one edge $(i,j) \in E_k$ such that $l_k \leq f(i,j) \leq c(i,j)$. In the RMP graph $G = (V,E)$, E_k is set to $E_k = LT_k, 1 \leq k \leq p$. For the example of Figure 8, since LT_1 is $\{(x_0,t_0),(x_1,t_0),(x_2,t_0),(x_0,t_1),(x_1,t_1)\}$, E_1 is set to $E_1 = \{(x_0 \to t_0),(x_1 \to t_0),(x_2 \to t_0),(x_0 \to t_1),(x_1 \to t_1)\}$. The lower bound of flow requirement in $E_k, 1 \leq k \leq p$ is set to 1, i.e., $l_k = 1$. In this example, at least one edge in the edge set $\{(x_0 \to t_0),(x_1 \to t_0),(x_2 \to t_0),(x_0 \to t_1),(x_1 \to t_1)\}$ must have a flow of *one unit*. It is assumed that two readers cannot be at the same location at the same time.

In Figure 9, the directed edge set E_1, E_2 and E_3 corresponding to three tags, a_1, a_2 and a_3 are shown enclosed in three rectangular boxes, colored red, yellow and blue respectively.

Theorem 12 *Any valid flow of the GMFP provides the minimum number of mobile readers needed to read all the mobile tags, and it also provides the trajectory that the mobile readers need to follow in order to read all the tags.*

Proof: If the minimum number of readers needed to read all the tags is m, there exists m flows (paths) from the source to the sink node in G. Suppose that the location-time pairs of reader $R_i, 1 \leq i \leq m$ is given by, $(l_{i,1}, t_{i,1})$, $(l_{i,2}, t_{i,2})$, $\ldots$, (l_{i,q_i}, t_{i,q_i}). Being at these locations at these times, enables the readers to read all the tags. A *flow* of *one unit* (or a path) from the source node S to the sink node D can be constructed in the following way. A path from S to D will be $S \rightarrow l_{i,1} \rightarrow, \ldots, l_{i,q_i} \rightarrow D$. Since such a path from S to D can be constructed for every reader R_i, there will be m unit flows from S to D.

If the solution to the GMFP is m unit flows from S to D, then m readers are sufficient to read all the tags in the deployment area. Because of the way the RMP graph is constructed, each unit flow corresponds to a path from S to D where the intermediate nodes are of the type $X_{k,i}$ and $T_{k,i}$ and edges are either of the form $location \rightarrow time$ or $time \rightarrow location$. Suppose that there is a flow from S to D of the form $S \rightarrow l_a \rightarrow t_b \rightarrow l_c \rightarrow t_d \rightarrow l_e \rightarrow t_f \rightarrow D$. From this flow, a trajectory can be constructed for a reader, where it moves from location l_a at time t_b to location l_c at time t_d to location l_e at time t_f, reading some of the tags. Since m flows are sufficient to satisfy the lower bound constraints imposed on the graph by each tag, it can be concluded that m readers are sufficient to read all the tags. ■

The RMP is solved by solving the GMFP using Integer Linear Programming (ILP). The objective of the ILP is as follows:

$$minimize\ F$$

subject to,

$$\sum_{j \in V} f(i,j) - \sum_{j \in V} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S, D \\ -F, & i = D \end{cases} \tag{6.1}$$

$$\forall E_k, 1 \leq k \leq p, \text{ if the edge } (i,j) \in E_k, \sum f(i,j) \geq 1 \tag{6.2}$$

$$\forall (i,j) \in E, f(i,j) \leq c(i,j) \tag{6.3}$$

$$\forall f(i,j) = 0/1 \tag{6.4}$$

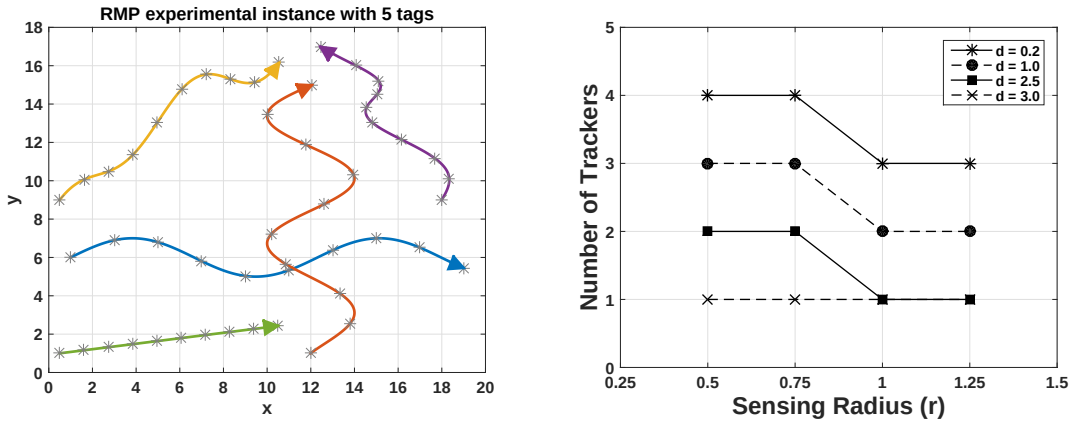
6.6 Extension to Higher Dimensions and Other Computational Complexity Issues

Extension to higher dimensions: In the preceding sections, although an explanation for the graph construction process was provided with an example where tag movement was restricted to one dimension, the proposed solution technique for the RMP is not restricted to only one dimensional movement of the tags. A critical component of the graph is the directed edges of the form *location* $\rightarrow$ *time* node pairs. If the locations of the tags are restricted to one dimension, *location* $\rightarrow$ *time* node pair takes the form *(x)* $\rightarrow$ *t*, where *x* is the location and *t* is the time. If the locations of the tags are restricted to two or three dimensions, *location* $\rightarrow$ *time* node pairs will take the form *(x, y)* $\rightarrow$ *t* or *(x, y, z)* $\rightarrow$ *t*, i.e., the location will be specified by two or three dimensional coordinates. However, such a representation will not affect the generalized minimum flow based technique to solve the RMP. As reported in Section 6.7, the experiments consider mobile readers and tags in a two dimensional space.

Impact of discretization: It may be noted that although the RMP is a *continuous time domain* problem (as the mobile tags and readers can be anywhere in the deployment area at a given time), the proposed solution technique discretizes both time and space. Time is discretized into equal intervals of length δ and space into equal intervals of length ε. This discretization allows the possibility of degradation of the quality of the solution, in the sense that the absolute minimum number of readers needed to read all tags may not be found. Accuracy of the solution can be improved if smaller values of the discretization parameters δ and ε are chosen. However, it may also be noted that smaller values of δ and ε is also going to increase the number of nodes and edges of the RMP graph, thereby increasing the complexity of finding the solution to the GMFP in the RMP graph. This is particularly true if the GMFP for the RMP graph is computed using an ILP. Thus, the proposed technique offers a direct trade-off between the *quality of the solution* (accuracy) and the *cost of the solution* (computation time).

6.7 Experimental Results

In this section, the effect of different parameters of the RMP, such as the reader sensing radius (r), and the speed of the reader (d) is investigated on the number of readers required to read all the tags. Experimental results are also presented to examine the effects of the granularity of the discretization parameters ε and δ on the total number of readers required to read all tags. For the experiments, a RMP instance with five mobile targets in a 2-dimensional deployment area over the time interval [0-10] were considered. The tag trajectories considered for the experiments are shown in Figure 10(a). The trajectories considered for each of the five tags in the problem instance were given by different para-



(a) Tag Trajectories

(b) RMP Experimental Results

Figure 10: (a) Trajectories of 5 tags in the time interval [0-10], (b) Number of readers vs. sensing range (r) with varying reader speed d in the time interval [0-10], $\varepsilon = 0.5, \delta = 1.0$

metric equations, thus, the speed of the tags were not constant or uniform. IBM CPLEX Optimizer 12.5 was used to solve the ILP to compute the minimum number of readers required to read all five tags. The impact of different reader parameters was investigated, namely, the sensing-range radius r and the speed of readers d, on the total number of readers required. The granularity of the discretized deployment area was specified by setting $\varepsilon = 0.5, \delta = 1.0$, and the observation time was [0-10]. The reader speed d was varied from 0.2-3.0, and the sensing-range r from 0.5-1.25. Figure 10(b) shows a portion of the results that, for a given reader speed, demonstrates the change in the number of readers required when the sensing range is varied. The experiments show that for a given reader speed, increasing the sensing range lowers the number of readers required to read all tags.

The variables δ and ε, used to discretize time and space were also varied in the experiments. Observations show that smaller values of δ and ε allow the proposed solution technique to yield better results, i.e. a lesser number of readers may be required to read all tags, an observation that is intuitively correct – a smaller value of ε increases the granularity of the deployment space, thus a larger number of locations are considered by the technique.

This may result in fewer number of required readers to read all tags. Also, smaller values of δ increases the number of possible locations the readers can be at a particular time step, which may also lead to a fewer number of required readers. Although smaller values of δ and ε can increase the accuracy of the solution, the cost of computation also increases considerably. For the example in Figure 10(a), different values of δ and ε were examined, specifically $\delta = 0.5, 1, 2, 4, 8$ and $\varepsilon = 0.5, 1$ with $r = 1, d = 1$ in the time interval [0-8]. For the considered set of tag trajectories, the change of values for δ and ε had no impact on the minimum number of readers required to read all tags, in this case 3. However, the computation time significantly increased when smaller values of δ and ε were used.

Chapter 7

PATH PLANNING FOR MOBILE SENSOR DATA COLLECTION USING DATA MULES

7.1 Introduction

In the prevalent literature, "*data mules*" are referred to mobile devices that travel to, and collect data from sensors located at sparsely dispersed points in a deployment area. The mules then subsequently bring back the collected data to a central collection point [25], [26]. From an energy saving perspective, data mules offer an attractive alternative to the sensor data collection process carried out by multi-hop forwarding techniques. Data mules travel to the vicinity of sensors in the deployment area and once within the communication range of the sensors, start collecting data from these sensors. Since the amount of data stored in different sensors may vary, the data collection time required by the mule for each sensor may also vary. Although data collection using data mules may result in energy savings, it might also result in increased delay (or latency) for data collection. Accordingly, a number of studies have also been undertaken to find intelligent paths for the mules with the objective of minimizing the delay [27].

Although sensor data collection problems using data mules have been studied fairly extensively in the literature, in most of these studies, while the mule is *mobile*, the sensors are assumed to be *stationary*. The objective of a majority of these studies is to *minimize* the time needed by the mule to collect data from all the sensors and return to the central collection point. The problem studied in this dissertation has two major differences with earlier studies. First, in this study it is assumed that *both* mules and sensors are *mobile*.

Second, minimizing the data collection time is not attempted, instead, the objective is to minimize the *number of mules* required to collect data from all sensors, subject to the constraint that the entire data collection process has to be *completed within some pre-specified time*. This problem is termed as the *Mule Minimization Problem* (MMP). It may be noted that stationary sensors can be viewed as a special case of mobile sensors, hence a solution technique for the MMP is equally applicable to both mobile and stationary sensors. Some of the specifics of the MMP are outlined below.

In the MMP, it is assumed that a central controller has the knowledge of: (i) the number of sensors in the deployment area, (ii) the trajectories of their movement, (iii) their location at every instance of time during the data collection period $\mathcal{T}$, (iv) their speed, and (v) the amount of data available on each sensor. From this information, a centralized controller computes (i) the minimum number of mules required to read data from all sensors within the pre-specified time $\mathcal{T}$, and (ii) the trajectories that the mules should follow in order to accomplish this task. The problem is illustrated with the help of an example: Figure 11 shows the trajectories of six mobile sensors on a two dimensional deployment area and their locations at various instances of time. It is assumed that the speed of the sensors are uniformly 1 unit/sec, (where the unit of measure can be feet, meter, etc.). The location of the sensors moving at this speed at various instances of time between the time interval [0-46] are also shown in Figure 11, however, to retain clarity, not all locations of each sensor are shown.

The goal of the MMP is to find the minimum number of mules needed to collect data from all sensors within a given pre-specified time $\mathcal{T}$, where $\mathcal{T}$ denotes the maximum allowable time step within which the data collection process must end. It may be noted that $\mathcal{T}$ does not include travel time from the central collection center to the start of the data collection process, (the location at the first instance of time when data is transferred between a

sensor and mule), and also excludes the travel time to the collection center from the end of the data collection process, (the location at the instance of time when data from all sensors have been read by one or more mules).

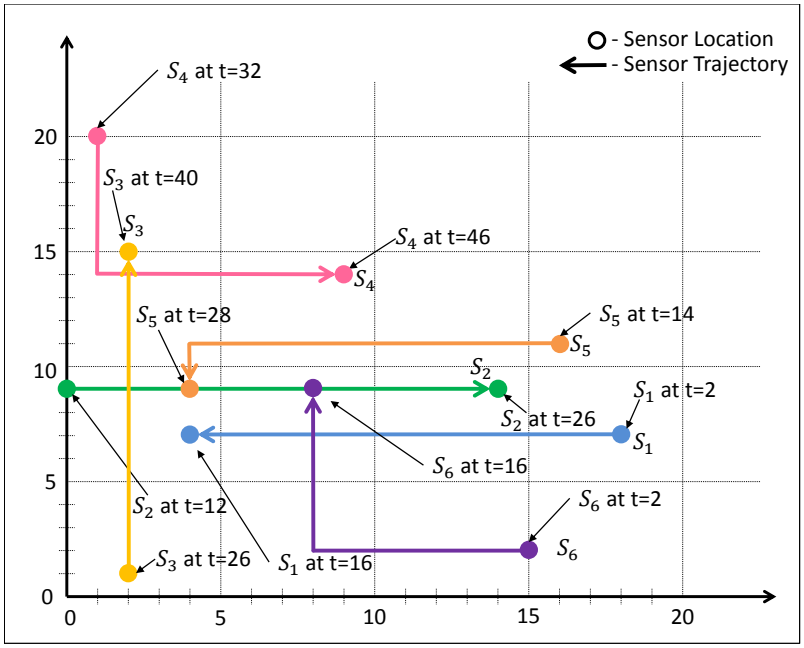


Figure 11: Locations and trajectories of six sensors $S_1, \ldots, S_6$ between the time interval [0-46]

For the sake of simplicity, for the example of Figure 11 it is assumed that (i) a mule can collect data from a sensor only when the distance between the mule and the sensor is at most one unit, (ii) the speed of the mule is equal to that of the sensors, i.e. 1 unit/sec, and (iii) the rate of data transfer between the sensor and the mule is 1 unit/sec (where the unit of measure can be megabytes, kilobytes, etc.). The solution to the problem of Figure 11 is shown in Figure 12, where only one mule is sufficient to collect all data from six sensors within the pre-specified time of $\mathcal{T} = 46$. In Figure 12, the trajectory of the mobile data mule is shown with a thick red line, and the locations where the mule collects data from the

sensors are shown with hatched rectangles. Specifically, the mule collects 3 units of data from S_6 during time interval [10-13], 2 units of data from S_1 during time interval [13-15], 2 units of data from S_2 during time interval [17-19], 3 units of data from S_5 during time interval [23-26], 4 units of data from S_3 during time interval [33-37], and finally 6 units of data from S_4 during time interval [38-44] at a uniform data transfer rate of 1 unit/sec.

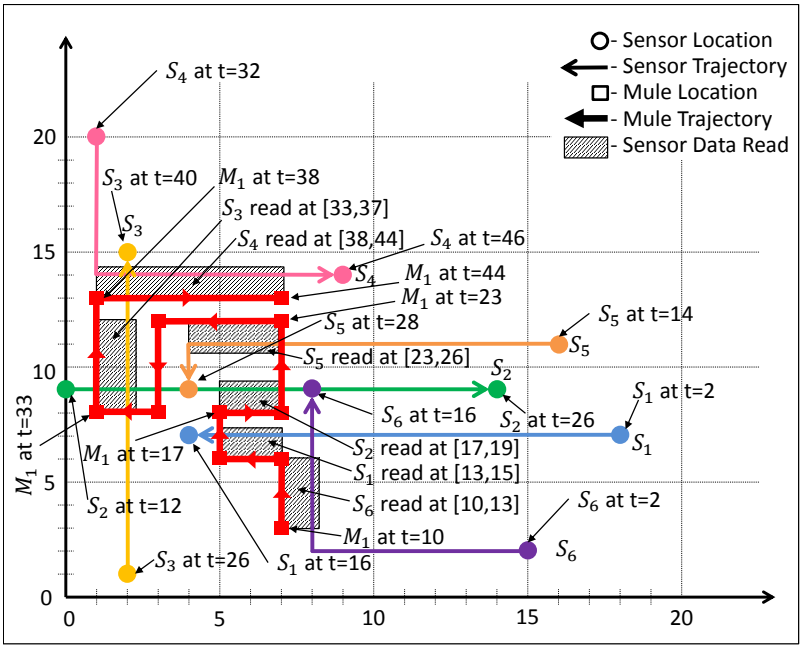


Figure 12: A single mule is sufficient to read data from all six sensors of Fig. 11 within time step $\mathcal{T} = 46$. The trajectory of the mule is shown with a thick red line and the locations where the mule collects data from the sensors are shown with hatched rectangles.

As shown in Figure 11, the model allows different sensors to have different amounts of data to transfer. This implies that each sensor may require different amounts of time to transfer such data to a mule. This raises an important question with respect to the available data collection infrastructure of the mobile data mules: Whether fragmented data collection from sensors is allowed or not? In other words, should a single mule collect all available data from a sensor, or can multiple mules collect fragments of the available data from a

sensor that can then be consolidated by a central system? If multiple mules are allowed to pick up fragments of data from a sensor, then there must exist some synchronization between the mules to determine which mule picks up which part of the data. Additionally, the mules must also possess a level of *intelligence* to facilitate such synchronization by reading parts of the sensor data at specific intervals of time. This dissertation considers both versions of the problem: one in which fragmented data collection is allowed, i.e. mules have sufficient intelligence to allow synchronization and are capable of reading specific parts of the sensor data, and the other, when fragmented data collection is disallowed, i.e. mules do not possess such intelligence and it is necessary that a single mule read a sensor's data in its entirety. As discussed in Section 7.3, the complexity of the solution for the second version of the problem is considerably higher than the first.

7.2 Related Work

As stated earlier, the mule minimization problem with a constraint on the data collection time has not been studied in the past. Most of the previous work either consider different problems and assumptions, or focus on similar issues but with different goals and objectives. In [28], [29] the authors focus on the problem of choosing the path of a data mule that traverses through a sensor field with sensors generating data at a given rate. To this purpose, the authors of [28] designed heuristic algorithms to find a path that minimizes the buffer overflow at each sensor node, that they later extended to multiple data mules and viewed the problem as a vehicle routing problem (VRP) [30]. In these works, however, it was assumed that data mules need to travel to the sensor nodes' exact location to collect data (i.e., excluding remote communication). This assumption facilitates TSP-type formulations for their problem and makes the data mule path selection problem similar to

a packet routing problem, such as the one studied in [31]. However, these formulations under-utilize communication capabilities, as data mules can use wireless communication to collect data from nodes without having to visit their exact locations.

Zhao and Ammar in [32] studied the problem of optimally controlling the motion of a data mule in mobile ad-hoc networks. A data mule, called a *message ferry*, mediates communications between sparsely deployed stationary nodes. They considered remote communication, but path selection was done based on a TSP-like formulation. They extended their work to multiple data mules in [33] and presented heuristic algorithms. In [27], the authors proposed to adapt the motion of the mule to minimize the full delay of data gathering. Ma and Yang [34] discussed the path selection problem under different assumptions. Their objective was to maximize the network lifetime, which is defined as the time until the first node dies (i.e. minimum of the lifetime of all nodes). They considered remote wireless communication and also multi-hop communication among nodes. When the path of a mule is given, they showed the problem of maximizing the network lifetime is formulated as a flow maximization problem that has a polynomial time algorithm. Choosing the mule's path is done by their heuristic algorithm that uses a divide and conquer approach to find a near optimal path for each part of the path.

Other approaches like the one in [35] are also inspired from vehicle networks to transfer data. They are called carry-and-forward techniques and offer more opportunistic data delivery. They can thus neither guarantee any QoS, nor limit the number of mules. Finally, as mentioned earlier, these existing works do not consider mobile sensors. One of the only studies that consider mobile sensors is [36], but this work assumes a given number of mules and does not try to minimize this number and instead proposes the best coverage possible at a given time using the available mules.

7.3 Mule Minimization Problem

The proposed technique for solving the Mule Minimization Problem (MMP) is to first transform the problem into a network flow problem, and then utilize Integer Linear Programming (ILP) to solve the network flow problem. As indicated in Section 7.1, this dissertation considers two variants of the problem: (i) when the mules have sufficient intelligence that allow fragmented data collection, i.e. the task of reading data from a single sensor can be distributed to different mules, and (ii) when the mules do not possess such intelligence, and the entirety of a single sensor's data must be read by a single mule. In Section 7.3.1 the solution for the MMP where mules have such intelligence is presented, and in Section 7.3.2, the technique from Section 7.3.1 is extended to address the scenario where the mules do not possess such intelligence.

7.3.1 Fragmented Data Collection – Mules with Intelligence

Although the MMP is a *continuous time domain* problem (as the mobile sensors and mules can be anywhere in the deployment area at a given time), the approach for solving the MMP is to discretize both time and space. Similar to the discretization process for the Reader Minimization Problem of Chapter 6, for the MMP, time is discretized into equal intervals of length δ, and space into equal *units* of size ε. Depending on the spatial dimension of the problem instance, i.e., one, two, or three dimensions, the unit will imply an *interval* of length ε, a *square* with length of each side ε, and a cube with length of each side ε, respectively.

In the subsequent discussion a problem instance of the MMP is formulated on a one dimensional space, i.e., sensors and mules are allowed to move either left or right on a

line. This formulation and the subsequent explanation of the proposed technique for a one dimensional space is presented due to its simplicity and brevity. Once the underlying principles of the solution is established, extension to higher dimensions is straightforward as the same principles apply.

The problem is formally setup as follows: Consider a set of n mobile sensors $\mathcal{A} = \{a_1, \ldots, a_n\}$ moving on a one dimensional plane (i.e., a line) over time instances $0, \ldots, \mathcal{T}$. It may be noted that although the movement of sensors is restricted to one dimension, there is no restriction on the direction of their movement, i.e. they can move either left and/or right, and can change directions arbitrarily. Let $p(a_i,t) = x(a_i,t)$ be the location of sensor a_i at time instance t where $x(a_i,t)$ denotes the x-coordinate of a_i at time t. For this problem instance, it is assumed that data from a sensor a_i can be *collected* by a mule M_j only if the distance between a_i and M_j is within the communication range of the mule denoted by r.

It is now shown that even when all the tags are stationary, finding the minimum number of stationary readers to read all tags is computationally hard. It may be noted that the proof is structurally similar to Theorem 11 of Chapter 6.

Theorem 13 *Mule Minimization Problem is NP-complete.*

Proof: The problem instance of the MMP where all sensors and data mules are stationary is equivalent to the Geometric Disk Cover Problem which is known to be NP-complete [23]. ■

It is only conceivable that when both the sensors and the data mules are mobile, computation of the minimum number of mules needed to read all the tags within a specified time and finding the trajectories of the mules will be considerably harder than the scenario when both the sensors and mules are stationary.

To find the solution for the MMP, the problem is first transformed to a *generalized version of the minimum flow problem* on a directed graph $G = (V,E)$. In this formulation, individual flows correspond to a path from the source to the destination node in $G = (V,E)$. The number of flows provides the number of mules required, and each path corresponds to the required trajectory of a mule as it moves through the deployment area collecting data from the sensors. Before proceeding with this transformation, the *Minimum Flow Problem* (MFP) [24] and its generalized version – GMFP, is presented below.

Minimum Flow Problem (MFP): Given a capacitated network $G = (V,E)$ with a non-negative capacity $c(i,j)$ and with a non-negative lower bound $l(i,j)$ associated with each edge (i,j) and two special nodes, a source node S and a sink node D, a flow is defined to be a function $f : E \rightarrow \mathbb{R}^+$ satisfying the following conditions:

$$\sum_{j \in V} f(i,j) - \sum_{j \in V} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S, D \\ -F, & i = D \end{cases}$$

$$l(i,j) \leq f(i,j) \leq c(i,j)$$

for some $F \geq 0$ where F is the value of the flow f. The MFP finds a flow f for which F is minimized.

Generalized Minimum Flow Problem (GMFP): The generalized version of the MFP is similar to the MFP, except that the lower bound on the flow requirement $l(i,j)$ is no longer associated with an edge (i,j), but associated with a set of edges $E_k \subseteq E$ of the graph $G = (V,E)$, and is denoted by l_k. Formally, the GMFP problem can be stated as follows: Given a capacitated network $G = (V,E)$ with a non-negative capacity $c(i,j)$ associated with each edge (i,j), a set of subsets E' of the edge set E (i.e. $E' = \{E_1, \ldots, E_p\}$, where $E_k \subseteq E, \forall k, 1 \leq k \leq p$), a lower bound on the flow requirement l_k associated with each $E_k, 1 \leq k \leq p$, and two special nodes, a source node S and a sink node D. A flow is defined

to be a function $f : E \rightarrow \mathbb{R}^+$ satisfying the following conditions:

$$\sum_{j \in V} f(i,j) - \sum_{j \in V} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S, D \\ -F, & i = D \end{cases}$$

$\forall E_k, 1 \leq k \leq p, \exists l_k$, a lower bound of flow in E_k, implying that the total flow on the set of edges in E_k is such that $\sum_{(i,j) \in E_k} f(i,j) \geq l_k$, and $f(i,j) \leq c(i,j), \forall (i,j) \in E_k$

for some $F \geq 0$, where F is the value of the flow f. The GMFP finds a flow f for which F is minimized.

It may be noted that when $|E_k| = 1$, $\forall k, 1 \leq k \leq p$, and $p = |E|$, the GMFP reduces to MFP.

7.3.1.1 MMP Graph Construction

The MMP graph construction process is outlined with an example where the movements of the sensors are restricted to a one dimensional space, i.e. the sensors are allowed to move either *left* or *right* on a *straight line* and are allowed to change directions arbitrarily. This restriction is imposed only to explain the graph construction process in a lucid way. Once the construction process is established, the same principles can be followed for constructing the MMP graph when the sensors and readers move in a two or three dimensional space. It may be recalled that data from each sensor can be collected by one or more mules, within a pre-specified data collection time $\mathcal{T}$, and the goal of the MMP is to collect data from all sensors with as few mules as possible within time $\mathcal{T}$.

Figure 13 illustrates this example, this example has three sensors and a pre-specified data collection time $\mathcal{T} = 2$, each sensor has one unit of data that must be read by a mule and the rate of data transfer available to all mules is one unit of data per time step. As seen

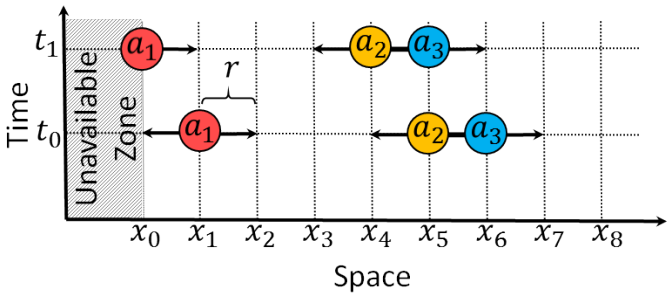


Figure 13: Locations of three sensors on a one dimensional space (line) at two different instances of time

in Figure 13, sensor a_1 is at location x_1 at time t_0 and in location x_0 at time t_1. Similarly, the sensor a_2 is at location x_5 at time t_0 and in location x_4 at time t_1, and sensor a_3 is at location x_6 at time t_0 and at location x_5 at time t_1. Although in this example, all sensors are moving left at the same speed, the sensors are free to move in either direction and at different speeds. A mule can collect data from a sensor only if the sensor is within the communication range r. For this example, it is assumed that $r = \varepsilon$, implying that in order to collect one unit of data from sensor a_1, there must be a mule at (x_0,t_0) or (x_1,t_0) or (x_2,t_0) or (x_0,t_1) or (x_1,t_1), where (x_i,t_j) indicates location x_i at time t_j. Using a similar reasoning it can be concluded that in order to collect data from sensor a_2, there must be a mule at (x_4,t_0) or (x_5,t_0) or (x_6,t_0) or (x_3,t_1) or (x_4,t_1) or (x_5,t_1). Also, in order to collect data from sensor a_3, there must be a mule at (x_5,t_0) or (x_6,t_0) or (x_7,t_0) or (x_4,t_1) or (x_5,t_1) or (x_6,t_1).

The MMP graph $G = (V,E)$ is a directed graph and is constructed in the following way: It may be noted that for a mule M_j to collect data from a sensor $a_i, 1 \leq i \leq n$ the distance between the mule and the sensor cannot exceed the communication range r. For this reason, in the above example, to collect data from sensor a_1, there must be a mule at (x_0,t_0) or (x_1,t_0) or (x_2,t_0) or (x_0,t_1) or (x_1,t_1). Corresponding to each sensor $a_k, 1 \leq k \leq n$,

there exists a set of potential $LT_k = (location, time)$ pairs of the form $(X_{k,i}, T_{k,j})$, and a mule must be in at least one of these locations at the specific time to be able to collect one unit of data from the sensor a_k.

If the rate of data transfer is assumed to be uniform for all mules at μ units of data per δ unit of time, then if d_k units of data have to be collected from sensor a_k, then at least $d'_k = \lceil \frac{d_k}{\mu} \rceil$ elements of the set LT_k must be *chosen* in order to *satisfy* the requirement that d_k units of data are collected from sensor a_k. For the example of Figure 13, it is assumed that $\delta = 1$, $\mu = 1$, and $d_1 = d_2 = d_3 = 1$. The potential $LT_k = (location, time)$ pairs of the example of Figure 13 is as follows:

$$LT_1 = \{(X_{1,1}, T_{1,1}), (X_{1,2}, T_{1,2}), \ldots, (X_{1,5}, T_{k,5})\},$$
$$LT_2 = \{(X_{2,1}, T_{2,1}), (X_{2,2}, T_{2,2}), \ldots, (X_{2,6}, T_{k,6})\},$$
$$LT_3 = \{(X_{3,1}, T_{3,1}), (X_{3,2}, T_{3,2}), \ldots, (X_{3,6}, T_{k,6})\}$$

To construct the MMP graph $G = (V, E)$, for each $LT_k = \{(X_{k,1}, T_{k,1}), \ldots, (X_{k,p_k}, T_{k,p_k})\}$, $1 \leq k \leq n$, the following are introduced in G: (i) a node representing $X_{k,i}$, (ii) a node representing $T_{k,i}$, and (iii) a directed edge from the node representing $X_{k,i}$ to the node representing $T_{k,i}, \forall i, 1 \leq i \leq p_k$. It may be noted that the $(X_{i,j}, T_{i,j})$ pair need not be *unique* and that the pairs $(X_{i,j}, T_{i,j})$ and $(X_{k,l}, T_{k,l})$ may represent the same $(location,\ time)$ pair. In case of non-unique $(X_{i,j}, T_{i,j})$ pairs, only one pair of nodes are created in the graph $G = (V, E)$. In the above example, the pairs $(X_{2,2}, T_{2,2}) = (X_{3,1}, T_{3,1}) = (x_5, t_0)$ thus only one pair of nodes corresponding to location x_5 at time t_0 will be created in G. This graph construction is shown in Figure 14. The nodes representing $X_{i,j}$'s are referred to as *location nodes*, and the nodes representing $T_{i,j}$'s are referred to as *time nodes*. In addition to these nodes, a source node S and a sink node D is also added to G. Apart from the directed edges of type $X_{k,i} \rightarrow T_{k,i}$ for each $(location,\ time)$ pair, the following three additional types of edges are also introduced in G:

1. *Mobility edges*: If a mule located at x_a at time t_b, can move to a location x_c at time t_d, then in the graph $G = (V,E)$, a directed edge from the node t_b to x_c is added. It may be noted that whether the mule can move from location x_a at time t_b to a location x_c at time t_d, depends on (i) distance between locations x_a and x_c, (ii) time interval between t_b and t_d, and (iii) the speed of the mule.
2. *Source edges*: A directed edge from the source node S is added to all location nodes.
3. *Sink edges*: A directed edge from all time nodes to the sink node D is introduced.

The capacity $c(i,j)$ of edge $(i,j) \in E$ is set to 1 for all edges in $G = (V,E)$.

As discussed earlier, an instance of the GMFP has a set of subsets E' of the edge set E (i.e. $E' = \{E_1, \ldots, E_p\}$, where $E_k \subseteq E, \forall k, 1 \leq k \leq p$), with a lower bound on the flow requirement l_k associated with each E_k. If l_k is the lower bound of flow in E_k, the GMFP requires that there must exist at least l_k edges $(i,j) \in E_k$ such that $\sum_{(i,j)\in E_k} f(i,j) \geq l_k$ and $f(i,j) \leq c(i,j)$. In the graph $G = (V,E)$, E_k is set to $E_k = LT_k, \forall k, 1 \leq k \leq p$. In the above example, since LT_1 is $\{(x_0,t_0),(x_1,t_0),(x_2,t_0),(x_0,t_1),(x_1,t_1)\}$, E_1 is set to $E_1 = \{(x_0 \rightarrow t_0),(x_1 \rightarrow t_0),(x_2 \rightarrow t_0),(x_0 \rightarrow t_1),(x_1 \rightarrow t_1)\}$. The lower bound of the flow requirement for each $E_k, 1 \leq k \leq p$ is set to d'_k, i.e., $l_k = d'_k$, where d'_k is the number of time units required to collect d_k data units from sensor a_k, at μ units of data per δ unit of time. In this example if $\delta = 1$, $\mu = 1$ and $d_1 = 3$, then at least three edges in the edge set $\{(x_0 \rightarrow t_0),(x_1 \rightarrow t_0),(x_2 \rightarrow t_0),(x_0 \rightarrow t_1),(x_1 \rightarrow t_1)\}$ must have a flow of *one unit*. The GMFP graph $G = (V,E)$ constructed for the problem instance with three sensors in Figure 13 is shown in Figure 14. The directed edge set E_1, E_2 and E_3, corresponding to three sensors, a_1, a_2 and a_3 are shown enclosed in three rectangular boxes, colored red, yellow and blue respectively.

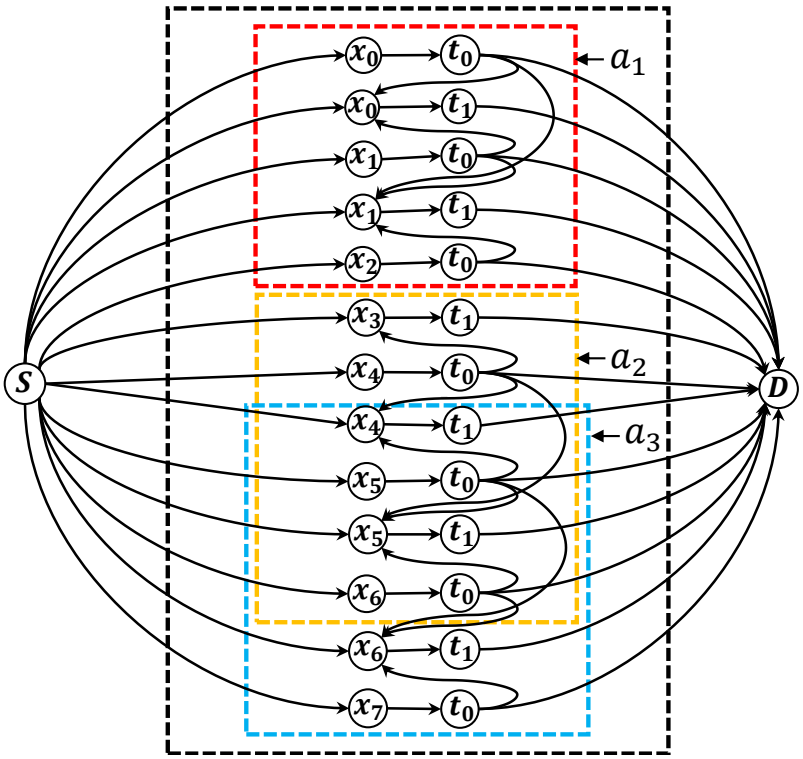


Figure 14: Mules with intelligence – MMP graph $G = (V, E)$ constructed from the instance of the problem shown in Fig. 13

7.3.1.2 Solution of MMP

The MMP problem is solved by solving the GMFP using Integer Linear Programming (ILP). The ILP formulation is as follows:

$$\textit{minimize } F$$

subject to,

$$\sum_{j \in V} f(i,j) - \sum_{j \in V} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S, D \\ -F, & i = D \end{cases}$$

$$\forall E_k, 1 \leq k \leq p, \text{ if edge } (i,j) \in E_k, \sum f(i,j) \geq d'_k$$

$$\forall (i,j) \in E, f(i,j) \leq c(i,j)$$

$$\forall f(i,j) = 0/1$$

It is now proven that the minimum number of mules required to collect data from all the sensors is equal to the solution of the GMFP in graph $G = (V,E)$, and the trajectories of the mules can be constructed from the solution of the GMFP.

Theorem 14 *Any valid flow of the GMFP provides the minimum number of mules needed to collect data from all the mobile sensors within the specified data collection time $\mathcal{T}$. The solution also provides the trajectory that the mules need to follow in order to collect data from the sensors.*

Proof: If the minimum number of mules needed to collect data from all the sensors is m, there exists m flows (paths) from the source to the destination node in G. Suppose that the location-time pair of mule $M_i, 1 \leq i \leq m$ is given by, $(l_{i,1}, t_{i,1}), (l_{i,2}, t_{i,2}), \ldots, (l_{i,q_i}, t_{i,q_i})$. Being at these locations at these times, enables the mules to collect μ units of data from the sensors. A *flow* of *one unit* (or a path) for the source node S to the destination node D can be constructed in the following way: A path from S to D will be $S \rightarrow l_{i,1} \rightarrow, \ldots, l_{i,q_i} \rightarrow D$. Since such a path can be constructed from S to D for every mule M_i, there will be m unit flows from S to D.

If the solution to the GMFP is m unit flows from S to D, then m mules are sufficient to collect data from all the sensors in the deployment area. Because of the way the constraints are set up, each unit flow corresponds to a path from S to D where the intermediate nodes are of the type $X_{k,i}$ and $T_{k,i}$ and edges are of the form $location \rightarrow time$. Suppose that there

is a flow from S to D of the form $S \rightarrow l_a \rightarrow t_b \rightarrow l_c \rightarrow t_d \rightarrow l_e \rightarrow t_f \rightarrow D$. From this flow, a trajectory of a mule can be constructed, where the mule moves from location l_a at time t_b to location l_c at time t_d to location l_e at time t_f, collecting a portion of the data to be collected from the sensors. Since m flows are sufficient to satisfy the lower bound constraints imposed on the graph by each sensor (which is equal to the amount of data to be collected from each sensor), it can be concluded that m mules are sufficient to collect all data from all sensors. ■

7.3.2 Unfragmented Data Collection – Mules without Intelligence

As discussed earlier, if multiple mules are allowed to pick up fragments of data from a sensor, then a central synchronization mechanism must exist and mules must be capable of reading portions of a sensor's data. The previous section (Section 7.3.1) addressed this scenario, and in this section a scenario is addressed where such synchronization capabilities are unavailable and a single mule is responsible for collecting a sensor's data in its entirety. First, it is noted note that if the amount of data to be collected from a sensor is more than μ units then the solution proposed in Section 7.3.1 may not be able to guarantee that the entire data from a sensor will be collected by a single mule. The following example illustrates this point.

Consider a scenario where data has to be collected from two sensors S_1 and S_2. Sensor S_1 has 2μ units of data to provide, and the sensor S_2 has μ units of data to provide. Suppose that due to the locations of the sensors, their speeds of movements, and the data collection threshold time $\mathcal{T}$, there are only two (*location*, *time*) pairs $(l_1,t_1),(l_2,t_2)$ where data collection from S_1 is feasible. Similarly, there are two (*location*, *time*) pairs (l_2,t_2) and (l_3,t_3) where data collection from S_2 is feasible. Suppose also, that due to the speed of

movement of the mules, it is possible for a mule to travel from location l_1 to l_2 within time interval [t_1-t_2] and also to travel from location l_2 to l_3 within time interval [t_2-t_3]. In addition, suppose that $\mathcal{T}$ is at least as large as the time interval between [t_1-t_2] and [t_2-t_3], but is less than the time interval [t_1-t_3]. To illustrate the example further, suppose that $t_1 = 1$, $t_2 = 2$, $t_3 = 1$ and $\mathcal{T} = 2$. In this case there can be two optimal solutions:

1. *Solution* 1: Mule M_1 collects 2μ units from S_1 from l_1 at time t_1, and l_2 at time t_2, and mule M_2 collects μ units from S_2 from l_3 at time t_1.
2. *Solution* 2: Mule M_1 collects μ units from S_1 from l_1 at time t_1. Mule M_2 collects μ units from S_2 from l_3 at time t_1 and μ units from S_1 from l_2 at time t_2.

Clearly, in Solution 1, only one mule collects the entire data (2μ units) from S_1, but in Solution 2, one mule collects only half the data (μ unit) from S_1 and the other mule collects the rest. However, there is no way for the optimal solution for the GMFP on graph $G = (V,E)$ (Section 7.3.1), to distinguish between these two solutions as the minimum flow in both these cases will be two. However, it is now shown that the version of the MMP where a single mule is required to collect the entire data from a single sensor can be solved by constructing a new graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, and solving a modified version of the GMFP on this graph. The construction process of graph $\mathcal{G}$ is described next.

As discussed earlier, in the graph shown in Figure 14, corresponding to each sensor $a_k, 1 \leq k \leq n$, there exists a set of edges $E_k = (location,\ time)$ pairs of the form $(X_{k,i}, T_{k,j})$, such that a mule is able read μ units of data from sensor a_k if it is present at location $X_{k,i}$ at time $T_{k,j}$. In Figure 14, all such $(location,\ time)$ pairs are enclosed within a black rectangle, which is now referred to as a *layer*. To accommodate the requirement that a single mule collects all the data from a given sensor, graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$ is constructed by replicating the sole layer of G, n times in $\mathcal{G}$. The structure of the edges and nodes in each layer is kept as is, but the nodes and edges of different layers are distinguished by associating them with the

layers they belong to. For example, location and time nodes of the form $X_{k,i}$ and $T_{k,j}$ in V, is respectively represented as $X_{k,i,m}$ and $T_{k,j,m}$ for each layer $m, 1 \leq m \leq n$ in $\mathcal{V}$. Similarly $(location,\ time)$ edges of the form $(X_{k,i}, T_{k,j})$ in E, is represented as $(X_{k,i,m}, T_{k,j,m})$ for each layer $m, 1 \leq m \leq n$ in $\mathcal{E}$. The set of edges of the form $(X_{k,i,m}, T_{k,j,m})$ that appear in layer $m, 1 \leq m \leq n$ is referred to as *Edges of Layer m* and is denoted by $\mathcal{EL}_m$.

Corresponding to each sensor a_k, in each layer $m, 1 \leq k, m \leq n$, there exists a set of edges $\mathcal{E}_{k,m} = (location,\ time)$ pairs of the form $(X_{k,i,m}, T_{k,j,m})$ signifying a location $X_{k,i,m}$ where a mule can be present at time $T_{k,j,m}$ to read μ units of data from sensor a_k. *Edges Across Layers for Sensor k* or $\mathcal{EALS}_k$, is defined to be the set of all edges across all layers that signify this $(location,\ time)$ pair where a mule can be present to collect μ units of data from sensor a_k. That is:

$$\mathcal{EALS}_k = \bigcup_{m=1}^{n} \mathcal{E}_{k,m}, \qquad \forall k = 1, \ldots, n$$

In addition to introducing the nodes and edges discussed above, for each layer $m, 1 \leq m \leq n$, additional nodes u_m, v_m, and an edge $u_m \rightarrow v_m$ is introduced in $\mathcal{G}$ as shown in Figure 15. A source node S is also introduced in $\mathcal{G}$ and is connected to all $u_m, 1 \leq m \leq n$ nodes. For each layer $m, 1 \leq m \leq n$ node v_m is connected to all $location$ nodes in layer m, that is $v_m \rightarrow X_{k,i,m}$, $\forall X_{k,i,m} \in \mathcal{EL}_m$. Lastly, a sink node D is introduced and all $time$ nodes of the form $T_{k,j,m} \in \mathcal{E}$ is connected to D as shown in Figure 15. Note that for clarity, all nodes and edges are not shown in Figure 15.

The MMP for mules without intelligence can be solved by solving a generalized version of the MFP, although it may be noted that this generalization is different from the version of the MFP for mules with intelligence (Section 7.3.1). It may be recalled that in the solution to the GMFP discussed in Section 7.3.1, the lower bound on the flow requirement l_k was associated with a set of edges $E_k \subseteq E$ of the graph $G = (V, E)$. In this version of the GMFP

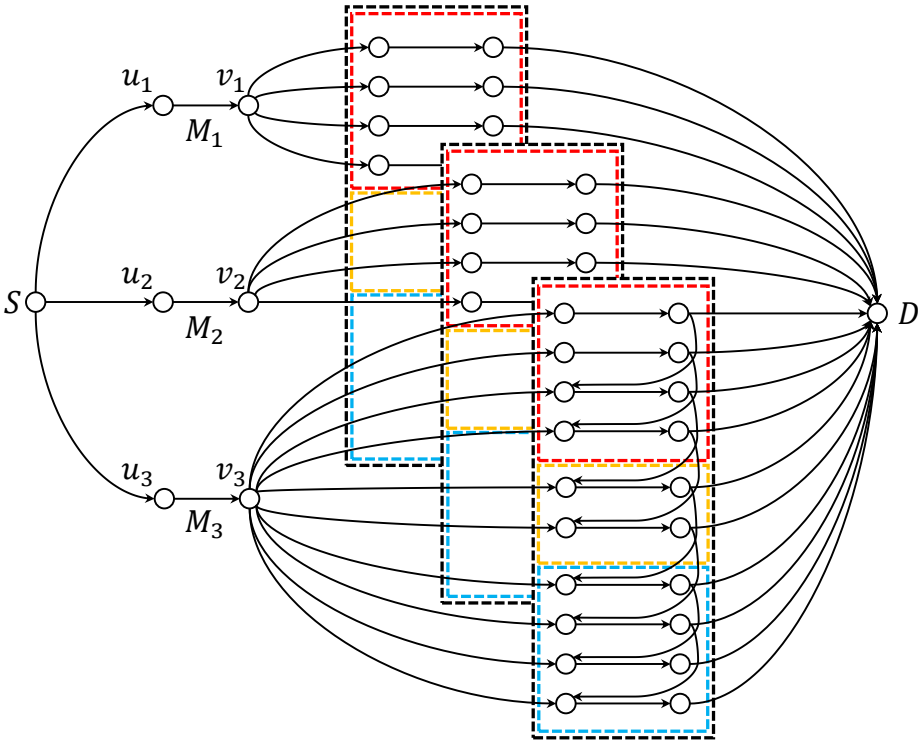


Figure 15: Mules without intelligence – Modified MMP graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ constructed from the problem instance of Fig. 13

titled *New Generalized Minimum Flow Problem* (NGMFP), the lower bound on the flow requirement l_k is no longer associated with a set of edges, but instead with a set of set of edges $\mathcal{EALS}_k \subseteq \mathcal{E}$.

The lower bound requirement of the NGMFP states that *there should be at least l_k units of flow through the edges of at least one set of edges $\mathcal{E}_{k,m}, 1 \leq m \leq n$* for all $k, 1 \leq k \leq n$. Because of the structure of the graph $\mathcal{G}$, this lower bound requirement, together with the constraint that the upper bound of capacity of each edge set to one, the solution of the NGMFP on $\mathcal{G}$ results in the solution of the MMP for mules without intelligence when l_k is set to $l_k = d'_k$, where d'_k is the number of time units required to collect d_k data units from sensor a_k, at μ units of data per δ unit of time. The NGMFP can be solved by using Integer Linear Programming (ILP), and is formulated using the following inputs:

Given (i) a set of sensors $\mathcal{A} = \{a_1, \ldots, a_n\}$, and a weight d'_k, representing the time units needed to collect d_k data units from sensor a_k at μ units of data per δ unit of time, (ii) a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with subsets of edges associated with each layer $\mathcal{EL}_m, 1 \leq m \leq n$, and a subset of edges associated with each sensor $\mathcal{EALS}_k, 1 \leq k \leq n$, and (iii) capacity of all edges set to one. The variable used for the ILP is first outlined:

For each sensor a_k, $\mathcal{EL}_m$, and directed edge (i, j):

$$y_{k,m} = \begin{cases} 1, & \text{if } \sum_{(i,j)\in(\mathcal{EALS}_k \cap \mathcal{EL}_m)} f(i,j) \geq d'_k \\ 0, & \text{otherwise} \end{cases}$$

The objective of the ILP is as follows:

$$\textit{minimize } F$$

subject to,

$$\sum_{j\in\mathcal{V}} f(i,j) - \sum_{j\in\mathcal{V}} f(j,i) = \begin{cases} F, & i = S \\ 0, & i \neq S, D \\ -F, & i = D \end{cases}$$

$$\forall k, m, 1 \leq k, m \leq n, \text{ if the edge } (i,j) \in (\mathcal{EALS}_k \cap \mathcal{EL}_m)$$

$$\sum f(i,j) \geq d'_k \times y_{k,m}$$

$$\sum_{m=1}^{n} y_{k,m} \geq 1, \forall k = 1, \ldots, n$$

$$\forall (i,j) \in \mathcal{E}, f(i,j) \leq c(i,j)$$

$$\forall f(i,j) = 0/1$$

$$\forall y_{k,m} = 0/1$$

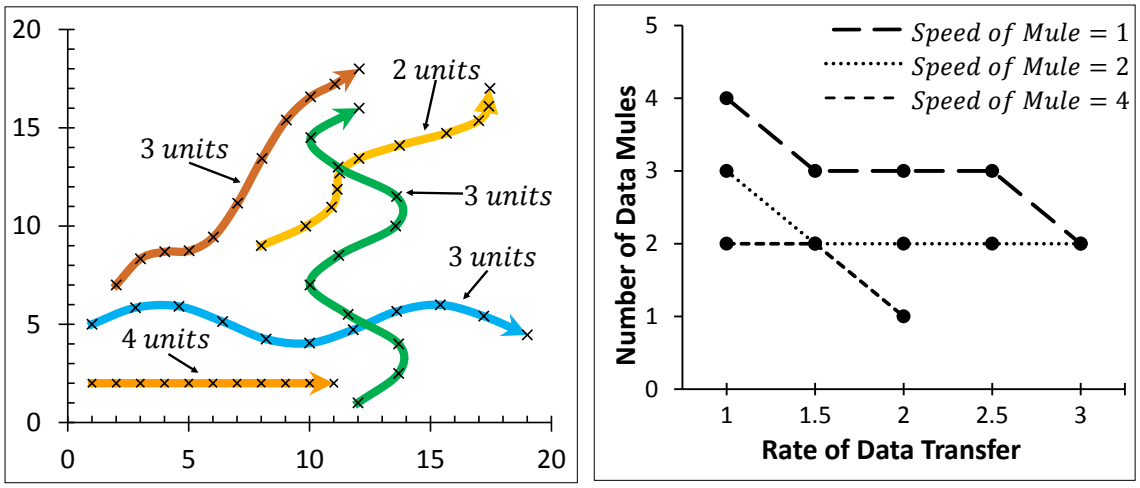
Using a similar reasoning from Theorem 14, it can be shown that a valid flow for the NGMFP provides the minimum number of mules (and their trajectories), required to collect data from all mobile sensors within the specified data collection time $\mathcal{T}$, such that data collected from a single sensor is not fragmented across multiple mules.

7.3.3 Extension to Higher Dimensions

In the preceding sections, the MMP was solved by constructing a graph $G = (V, E)$ ($\mathcal{G} = (\mathcal{V}, \mathcal{E})$) from an instance of the MMP problem and solving the GMFP (NGMFP) on it. An explanation for the graph construction process was provided through an example where the movements of sensors and mules were restricted to a one dimensional space. However, the proposed solution technique for the MMP is not restricted to only one dimensional movement of the sensors and mules. A critical component of the graph is the directed edges of the form *location* $\rightarrow$ *time* node pairs. If the locations of the sensors are restricted to one dimension, *location* $\rightarrow$ *time* node pair takes the form *(x)* $\rightarrow t$, where x is the location and t is the time. If the locations of the sensors are expanded to two or three dimensions the *location* $\rightarrow$ *time* node pairs will take the form *(x, y)* $\rightarrow t$ or *(x, y, z)*$\rightarrow t$, respectively to capture the two or three dimensional coordinates. However, such a representation will not in any way affect the generalized minimum flow based approach to the solution of the MMP that was presented in this dissertation.

7.4 Experimental Results

Experimental results for the solution techniques for the MMP is presented in this section. For these experiments, 5 mobile sensors in a 2-dimensional deployment area over the



(a) Sensor Trajectories

(b) MMP Results

Figure 16: (a) Trajectories and available data units of 5 sensors in the time interval [0-10], (b) Number of mules vs. rate of data transfer, varying mule speeds in time interval [0-8], $\varepsilon = 1.0, \delta = 1.0, r = 1$

time interval [0-10] was considered. The sensor trajectories and their available data units considered for the experiments are shown in Figure 16(a). The sensor trajectories were specified by unique parametric equations and thus not all speeds considered were uniform and constant.

IBM CPLEX Optimizer 12.5 was used to solve the ILPs to compute solutions for the MMP of Figure 16(a) under both intelligent and unintelligent mule settings. To discretize the deployment area, δ and ε were set to $\delta = 1$ and $\varepsilon = 1$. The impact of different sensor and mule parameters were also investigated, namely, the data transfer rate μ per δ unit of time, and the speed of the mule, on the total number of mules needed to gather data from all sensors in $\mathcal{T} = 8$ time. Figure 16(b) shows the required number of mules at a given speed, as the available rate of data transfer (μ) is varied. For this specific problem instance and the given mule parameters, the number of mules required to read all sensor data within $\mathcal{T} = 8$ time does not vary under the intelligent and unintelligent mule settings. However,

the computed mule trajectories were observed to be different in these two settings. The experiments confirmed that for a given mule speed, increasing the data transfer rate lowers the number of mules required.

In these experiments, the variables ε and δ that were used to discretize time and space were also varied. The observations indicate that smaller values of ε and δ allow the proposed solution technique to be closer to the optimal solution in a continuous setting (when space and time are not discretized). This is based on the fact that smaller values of ε and δ increases the granularity of the solution space (by increasing the number of $(location, time)$ pairs), that is considered by the solution. Though this may result in solutions closer to the optimal, smaller values of ε and δ considerably increases the cost of computation.

Chapter 8

RAPTOR: A NETWORK TOOL FOR MITIGATING THE IMPACT OF SPATIALLY CORRELATED FAILURES IN INFRASTRUCTURE NETWORKS

8.1 Introduction

It is extremely important that planners for large wide area networks have the right tools to design robust and resilient networks that can effectively withstand large scale *geographically correlated failures* in their networks. Such failures can be triggered by nature (hurricane or earthquake), or by humans (nuclear attack or conventional weapon attack over a large geographical area). The characteristic of such *spatially correlated*, or *region based* faults is that they are *massive* but *localized* faults. As noted by the authors in [37], using *network connectivity* [38] as a metric for evaluating the *fault-tolerance capability* of a network is inadequate for such failures as it does not capture the characteristics of spatially correlated failures. For instance, network connectivity as a metric ignores the *locality* of the fault, i.e., the faulty nodes/edges may be close or far away from each other. Also, connectivity as a metric does not capture important structural properties of the network such as the *number*, or *size* of the *connected components* [38] into which a network disintegrates when the number of failed nodes/edges *exceeds* the node/edge connectivity of the network.

With research support from the U.S. Defense Threat Reduction Agency (an agency whose mission is to protect the U.S. against *Weapons of Mass Destruction*, such as nuclear, biological or chemical attacks), over the last six years the Network Science Lab at Arizona State University, has worked towards developing a wide ranging set of concepts and techniques for enhancing network robustness against *spatially correlated*, or *region based*

faults. These concepts and techniques have recently been incorporated into RAPTOR, an advanced network planning and management tool [39], for the benefit of network designers, planners and operators. This chapter first describes the novel concepts developed to design networks that are robust against region based faults, and then outlines how these concepts have been incorporated into the tool. The goal of this chapter is to bring to the attention of the networking research community the existence of RAPTOR as a tool that consolidates a large body of work on spatially correlated failures, and as a tool that can be used by the community to meet the needs for robust network design against region based faults. To this effect, this chapters' contribution does not lie in new analytical findings, but in service to the networking research community.

The tool described in this chapter is intended to support design and analysis of single layered and multi-layered interdependent heterogeneous networks. In this context, RAPTOR is particularly suitable for planning and design of critical infrastructures. For example, from the single network layer perspective, RAPTOR can enable backbone communication network providers, such as AT&T, Sprint, Qwest and Level 3 Communications, to (i) identify the most vulnerable parts of their network against region based faults, and (ii) reinforce the network with least cost to reduce, or eliminate the threat of network disruption due to a region based fault. From a multi-layer perspective, RAPTOR can be used for design and analysis of smart cities, where heterogeneous networks such as power, communication, water, and gas distribution networks form a complex interdependent ecosystem where failures in one network may impact another. For instance, a leak in the water distribution network, may deteriorate other nearby (spatially correlated) infrastructures such as gas or electricity whose pipes and cables may get affected due to the leak. In this context, a tool like RAPTOR can be used by utility companies and city planners to quickly perform (i) root

cause analysis of failure, and (ii) forecast fault evolution, to direct repairs and maintenance towards specific network components and restrict fault propagation.

Several studies in the network research community have focused on different aspects of spatially correlated or region-based faults in networks [40]–[46], however, there does not exist an executable platform that consolidates the findings and techniques of these studies into a readily usable tool. The tool RAPTOR is intended to fill that gap and be such a platform that can incorporate the outcomes developed in studies such as [40]–[46] into executable modules that can be integrated into RAPTOR. This will allow network designers, planners and operators to use the results of these studies in their real world operational networks.

8.2 Concepts, Metrics, and Solution Techniques

In this section a brief overview is provided of the underlying concepts, metrics and solution techniques that the tool RAPTOR utilizes to carry out its functional operations. The tool is built as a modular execution engine that can execute smaller reusable modules to perform desired operations on a network topology. In this respect, the current version of the tool comprises of different modules that deal with both static and dynamic aspects of robust and resilient network design. The modular approach allows design, development and testing of these modules to be done independently and defers the integration into the tool until a module meets its functional requirements. In the following sub-sections a brief overview of the analytical foundations of these modules is presented. It may be noted that, as of writing this chapter not all modules have been implemented and integrated into the tool. Accordingly, the ongoing work is highlighted in the discussion below.

8.2.1 Region-Based Fault Metrics Computation Module

As outlined in Section 8.1, connectivity as a metric fails to capture several characteristics of the network in presence of spatially correlated failures. For instance, the number or size of the connected components into which a network disintegrates in the presence of a spatially correlated fault is not captured by the traditional connectivity metric. In order to overcome these gaps and capture such network state characteristics, several metrics and their computation techniques have been proposed by the research community. For a given network topology, RAPTOR can analyze the network and compute metrics pertinent to network state in the presence of spatially correlated faults. The following metrics are supported by the tool:

8.2.1.1 Region-Based Connectivity Metric Computation

Region based connectivity can be considered under two fault models – (i) Single Region Fault Model (sRFM) where faults are confined to a single region [37], and (ii) Multiple Region Fault Model (mRFM) where faults are confined to k regions for some specified k [47].

Formally, in sRFM, the *single-region-based (node) connectivity* of graph G with a specified definition of region R, $s\kappa_R(G)$, is defined as follows: Suppose that $\{R_1, \ldots, R_k\}$ is the set of all possible regions of the graph G. Consider a k-dimensional vector T whose i-th entry, $T[i]$, indicates the number of nodes in region R_i whose failure will disconnect the graph G. If the graph G remains connected even after the failure of all nodes of the region R_i then $T[i]$ is set equal to ∞. The *region-based connectivity* of a graph G with region R, is

then computed as follows:

$$s\kappa_R(G) = \min_{1 \leq i \leq k} T[i]$$

In mRFM, the *multi-region-based (node) connectivity* of graph G with a specified definition of region R, $m\kappa_R(G)$, is defined as the minimum number of regions whose removal (i.e., removal of all nodes in the regions and edges incident on them) will disconnect the graph.

Polynomial time algorithms to compute region-based connectivity in sRFM was presented in [37]. RAPTOR provides an implementation of this algorithm that can be used to compute the Region-based Connectivity for a given network topology.

8.2.1.2 Region-Based Component Decomposition Number Metric Computation

Proposed by the authors of [48], the Region-Based Component Decomposition Number, or RBCDN of graph $G = (V, E)$ with a specified definition of region R is defined as follows: Suppose that $\{R_1, \ldots, R_k\}$ is the set of all possible regions of the graph G. Consider a k-dimensional vector C whose i-th entry, $C[i]$, indicates the number of connected components in which G decomposes when all entities in R_i fails. RBCDN of a graph G with region R is computed as follows:

$$\delta_R(G) = \max_{1 \leq i \leq k} C[i]$$

RBCDN as a metric provides an insight into the worst case scenario on how fragmented a network can become in the presence of a spatially correlated fault. In [48] the authors propose techniques to compute the RBCDN and RAPTOR provides an implementation of this algorithm that can be used on user selected network topologies.

8.2.1.3 Region-Based Smallest/Largest Component Size Metric Computation

The Region-Based Smallest (Largest) Component Size, or RBSCS/RBLCS was proposed in [49], and is defined for a graph $G = (V, E)$ with a specified definition of region R, as follows: Suppose that $\{R_1, \ldots, R_k\}$ is the set of all possible regions of the graph G. Consider a k-dimensional vector C_S (C_L) whose i-th entry, $C_S[i]$ ($C_L[i]$), indicates the size of the smallest (largest) connected component in which G decomposes when all nodes in R_i fails. The RBSCS $\alpha_R(G)$ and RBLCS $\beta_R(G)$ of graph G with region R is defined as:

$$\alpha_R(G) = \min_{1 \leq i \leq k} C_S[i] \text{ and } \beta_R(G) = \min_{1 \leq i \leq k} C_L[i]$$

The RBLCS and RBSCS metrics provide insights on how well a network's performance degrades in the presence of region-based faults. Depending on the needs of graceful performance degradation, network designers may choose to design networks that have a small value of RBCDN ($\delta_R(G)$) and a high value of either RBLCS ($\alpha_R(G)$) or RBSCS ($\beta_R(G)$). RAPTOR allows the user to compute the RBLCS and RBSCS metrics for a chosen network topology.

8.2.2 Distinct Regions Computation Module

It may be noted that all the previously defined metrics operate on a given graph and a set of regions. Thus, there is a need for techniques that compute the set of regions, given a network and some fault specification. In [49], given a graph G's layout on a two-dimensional plane and a fault radius r, the authors provide a polynomial time algorithm to compute all *distinguishable* or *distinct* circular regions with radius r. Two fault regions are considered *indistinguishable* if they contain the same set of links and nodes. The authors considered both wired networks, where nodes and edges can be part of a failure region, and

wireless networks, where only nodes can be part of a failure region. It was shown in [49] that the number of distinct regions in wireless and wired networks are $O(n^2)$ and $O(n^4)$ respectively, and that all distinct regions can be computed in $O(n^6)$ time, where n is the number of nodes in the network.

RAPTOR comes bundled with an implementation of the technique outlined in [49]. Given a network topology and a fault radius, RAPTOR can compute all distinct regions of the network which can then be used by other modules of the tool, such as the Metric Computation Module and the Region-disjoint Path Computation Module (discussed next).

8.2.3 Region-Disjoint Paths Computation Module

For a graph $G = (V,E)$, a set of region-disjoint paths $\mathcal{P}$ between a source node s and destination node d with a specified definition of region R, is defined as follows: Suppose that $\{R_1,\ldots,R_k\}$ is the set of all possible regions of graph G and path $\mathcal{P}_u \in \mathcal{P}$ contains a set of nodes and edges from G such that $\mathcal{P}_u$ forms a path from s to d, $\{s,d\} \in V$. Then, for every pair of paths $\{\mathcal{P}_u,\mathcal{P}_v\} \in \mathcal{P}, u \neq v$, $\mathcal{P}_u$ and $\mathcal{P}_v$ are region-disjoint, i.e. there is no region in R that both the paths traverse. Formally, region-disjoint paths are defined as follows, for all $i = 1,\ldots,k$:

$$|(\mathcal{P}_u \cap R_i) \cap (\mathcal{P}_v \cap R_i)| = 0, \forall \{\mathcal{P}_u,\mathcal{P}_v\} \in \mathcal{P}, u \neq v$$

Although region-disjoint path computation has been addressed in [43], the authors consider a model where faults do not cause edges to fail unless a failed edge is associated with a failed node. This assumption is considerably restrictive and possibly unusable for designers of larger networks where spatially correlated faults can affect nodes and edges independently. In order to overcome this limitation the RAPTOR tool supports computation of region-disjoint paths in the presence of circular faults using an Integer Linear Program

(ILP) that doesn't presuppose any such restrictions. The tool is capable of computing two region-disjoint paths from given source and destination nodes such that the sum of lengths of the two paths is minimum. Also, as the source (destination) node is part of a region that is traversed by both paths (as both paths have the same starting and ending points), no region disjoint path may exist. To accommodate this situation the tool accommodates the use of *no-fault zones* – a circular area around the source and destination nodes that is immune to faults. Future extensions of this module include computing more than two paths, and including other selection criteria such as minimizing the maximum path length.

8.2.4 Region-Based Fault Tolerant Distributed File Storage Module

In the preceding discussions the importance of a node in keeping the network connected is emphasized, however, individual nodes can also act as data stores of the network and the removal of a node from a network (due to a region-based fault), may not only cause connectivity losses, but also data losses. To address such data loss risks, distributed storage techniques are often employed that enhances data survivability in the presence of faults. One such technique is redundancy, such as by (i) storing multiple copies of the entire file, or (ii) storing different fragments of the same file at different nodes in the network. In the popular (N,K), $N \geq K$ file distribution scheme, from a file F of size $|F|$, N segments of size $|F|/K$ are created in such a way that it is possible to reconstruct the entire file by accessing any K segments. For such a reconstruction scheme to work, it is essential that the K segments of the file are stored in nodes that are connected to each other in the network. However, in the event of failures, the network may become disconnected (i.e., split into several connected components) and K segments may not be accessible in the residual network to reconstruct the file F.

From the context of data survivability in the presence of spatially correlated faults in networks, RAPTOR supports a "Region-based Distributed File Storage Module" that implements an algorithm proposed in [46] that ensures: (i) even when the network is fractured into disconnected components due to a region-based fault, at least one of the largest components will have access to at least K distinct file segments with which to reconstruct the entire file, and (ii) the total storage requirement is minimized. As of writing this dissertation, this module is currently under development and will be part of the tool upon its completion.

8.2.5 Robust Multi-Layer Interdependent Network Design Module

In today's world, a multitude of heterogeneous interconnected networks form a symbiotic ecosystem that supports all of the economic, political and social aspects of human life. For example, the critical infrastructures of the nation such as the power grid and the communication network are highly interdependent on each other, and any adverse effects on one network can affect the other network. Thus, isolated network analysis is no longer sufficient to design and operate such interconnected and interdependent network systems.

As noted in Chapter 3, recognizing this need for a deeper understanding of the interdependency in such multi-layered network systems, significant efforts have been made by the research community in the last few years, and accordingly, a number of analytical models have been proposed to analyze such interdependencies [2], [3], [7]. However, most of these models are simplistic and fail to capture the complex interdependencies that may exist between entities of the power grid and communication networks. To overcome the limitations of existing models, the *Implicative Interdependency Model* (presented in Chapter 3), is able to capture such complex interdependencies. Utilizing this model, several problems

on multi-layer interdependent networks have been studied, such as (i) identification of the $\mathcal{K}$ most vulnerable nodes (Chapter 4), (ii) root cause analysis of failures [50], (iii) the entity hardening problem [51], (iv) the smallest pseudo-target set identification problem (Chapter 5), and (v) the robustness analysis problem [52].

This module will support multi-layer network interdependency modeling using the *Implicative Interdependency Model*, and analysis of multi-layer networks using the techniques proposed in [50]–[54]. The module is currently under development and will be part of RAPTOR upon its completion.

8.2.6 Module for Progressive Recovery from Region-Based Failures

With this module, the tool addresses post-fault recovery techniques in the aftermath of region-based faults on multi-layer interdependent networks. To restore an interdependent network system from a post-fault scenario to its pre-failure state, all the faulty network entities (nodes/edges) have to be repaired or replaced. However, resource limitations may prevent simultaneous restoration of all failed units of the network. Accordingly, the failed units have to be restored in a *sequenced* manner. As each network entity in its *operational* state adds some *utility value* to the interdependent network system, when a unit recovers from a *failed state* to an *operational* state, the unit starts providing some "benefit" to the system. Since different units have different utility values to the system, the sequence in which the failed units are restored is important as the recovery sequence determines the cumulative system utility during the recovery process.

As discussed in Chapter 3, the *Implicative Interdependency Model* provides a powerful technique for modeling dependencies in multi-layer interdependent networks. Using this model, the authors of [55] have studied the progressive recovery problem in interdepen-

dent networks with the objective of maximizing system utility during the system recovery process. This RAPTOR module will implement the progressive recovery algorithm of [55], which can then be used to sequence recovery of network entities from a post-fault to a pre-fault network state that maximizes system utility during the recovery process. This module is currently under development and will be part of RAPTOR upon its completion.

8.3 Architecture and System Capabilities

In this section the system architecture is first outlined, and then the different capabilities of RAPTOR is discussed.

8.3.1 System Architecture

RAPTOR is implemented as a web-application that allows the user to remotely connect and operate the tool from a browser. The web-application follows the standard three-tier architecture and has a client tier, application tier, and database tier. The tool has been developed following the Model-View-Controller (MVC) design pattern. Figure 17 outlines the high level architecture and some of the components of the tool.

The tool is currently accessible from Arizona State University's WAN, and runs from a testbed server. The tool's web-application is deployed on an Apache Tomcat 7 instance, and the repository used is MySQL. The application tier business logic for operations on network topologies, such as Region-Based Fault Analysis and Region Disjoint Path Analysis, are implemented in Java. Additional packages and libraries, such as IBM ILOG CPLEX Optimization Studio libraries (required for solving Integer Linear Programs), are setup and

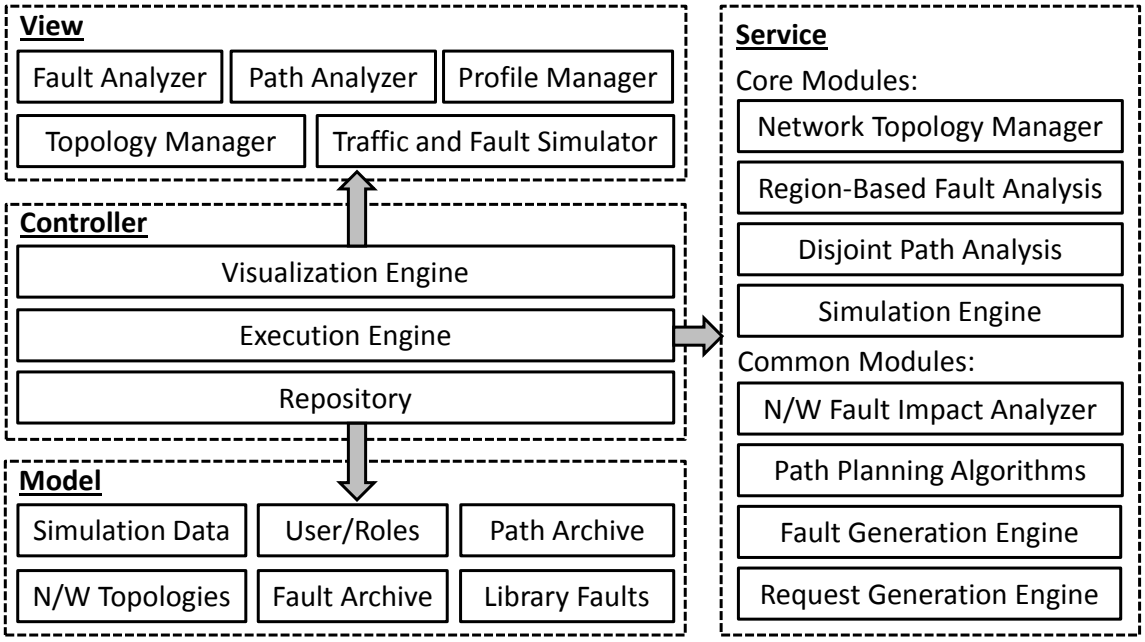


Figure 17: High-Level Architecture of the tool RAPTOR

made available on the testbed server. The testbed server is a 64-bit Intel Core 2 Quad Core (2.66 GHz) system with 8 GB of RAM running Ubuntu 14.04.

8.3.2 System Capabilities

The tool RAPTOR is designed to be used by following a three step workflow comprising of (i) Network Creation, (ii) Network Analysis, and (iii) Network Simulation. Accordingly, the individual features and the executable modules of the tool are bundled around these three workflows. The following list enumerates the current high-level features of the tool and the corresponding workflows that each feature emulates:

1. Topology Management (Network Creation)
2. Fault Analysis (Network Analysis)

3. Path Analysis (Network Analysis)
4. Traffic and Fault Impact Simulation (Network Simulation)

Each of the above features are accessible from a tabbed interface and can be navigated to from any part of the application. In the following subsections each of the features of the tool is described along with a brief functional overview of the features.

8.3.2.1 Topology Management

Network Creation is the first step of RAPTOR's workflow and the *Topology Manager* interface allows users to create, edit, save and delete network topologies. The Topology Manager presents the user with a geographical map interface that she can interact with to manage network topologies. The displayed map tiles are rendered from OpenStreetMap [56]. RAPTOR uses the OpenLayers API to support an user interactive map interface.

To create the topology and place nodes and edges on the map, the user can either point-and-click on the map itself, or can type in specific latitude and longitude coordinates and then proceed to add the network entity. Capacities for each edge (in Gigabits per second), can also be specified during the edge creation process. Once a network topology is created, the topology must be saved to be used for Network Analysis and Network Simulation. The topologies are saved on the RAPTOR server and can be loaded back into the Topology Manager to edit entities or attributes of the network.

Figure 18 shows a screen grab of the Topology Manager. As seen in the figure, the map interface is on the right and the user interact-able menu is on the left. The user can click on the map to add nodes and edges, or can alternatively type in the latitude and longitude coordinates in the input fields available on the menu. The menu also lists the nodes and

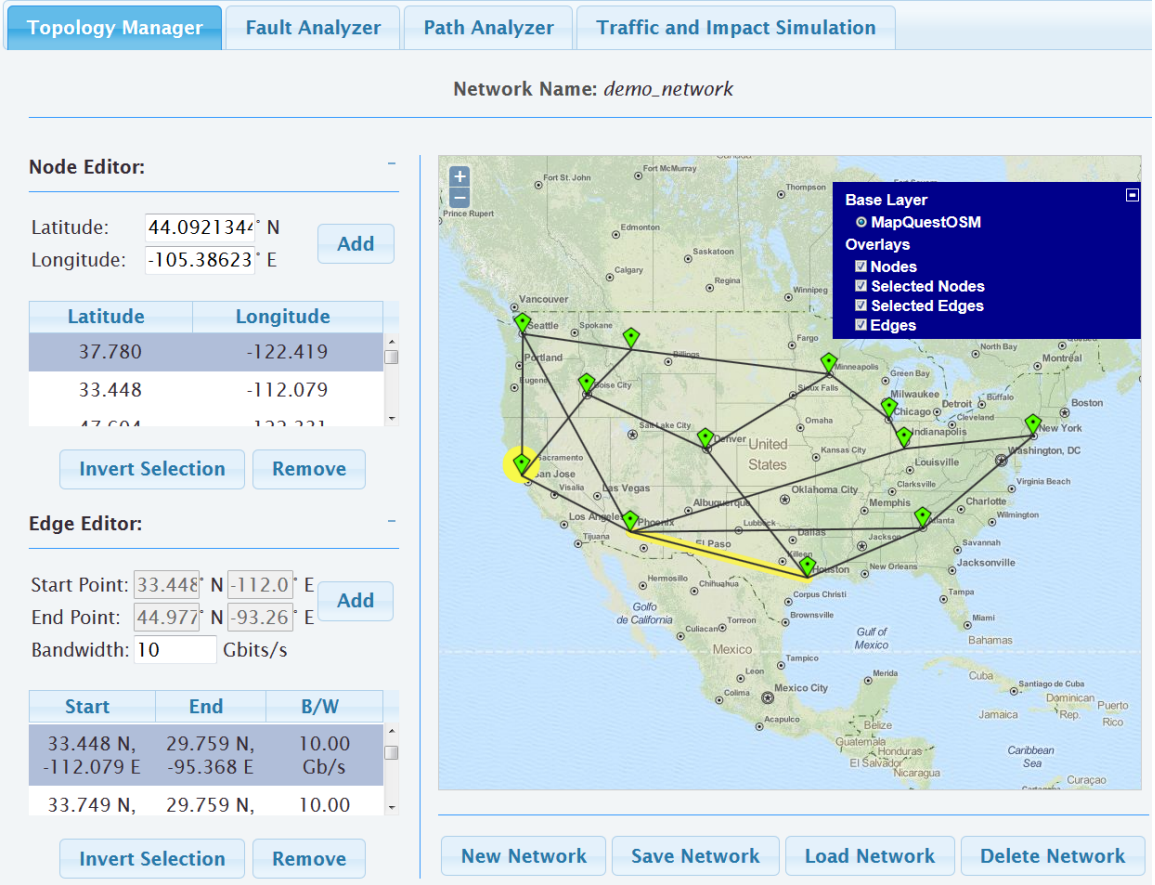


Figure 18: Topology Manager – create, edit and manage network topologies

edges that are part of the topology. Selecting an edge or node from these lists highlights the network entity on the map (in yellow), and the user can then proceed to remove the entity from the network if necessary. The displayed map overlays can be toggled from a dropdown menu available on the map (in blue in Figure 18). Finally, as seen in Figure 18, options for saving, loading, and deleting topologies are available to the user directly below the displayed map's dimensions.

8.3.2.2 Fault Analysis

Once network topologies are created from the Topology Manager, the *Fault Analyzer* can be used to analyze the created networks for their *resilience* in the presence of spatially correlated faults. In RAPTOR, network resiliency is measured by how well the network performs when benchmarked against the metrics outlined in Section 8.2.1. It may be noted that the metrics of Section 8.2.1 emphasize resilience from the aspect of connectivity in the presence of a spatially correlated fault. For example, the more number of disconnected components a network has due to a fault, the worse is the network's resilience (as captured by the metric RBCDN). It may also be noted that, for the purpose of this analysis the tool assumes that any network entity (nodes/edges), that fall within the fault area are all rendered inoperable, i.e. the fault model is deterministic, and if a network entity falls within the fault region, it necessarily fails. To carry out this analysis, the user first selects a network topology and can then choose to either perform a *generic fault* analysis, or a *specified fault* analysis. These analyses are described next.

Generic Fault Analysis: In the generic fault analysis, for a selected network topology, the user specifies a fault feature and the tool computes the values of the individual metrics listed in Section 8.2.1. RAPTOR can generically analyze circular faults, and the supported fault feature is the fault radius r.

As shown in Figure 19, the user can specify the fault radius r from the left menu. The tool then performs the generic fault analysis by (i) computing all the *distinct* regions with radius r using the techniques implemented in the module "Distinct Regions Computation Module" (Section 8.2.2), and (ii) computes the individual metrics using the techniques implemented in the module "Region-Based Fault Metrics Computation Module" (Section

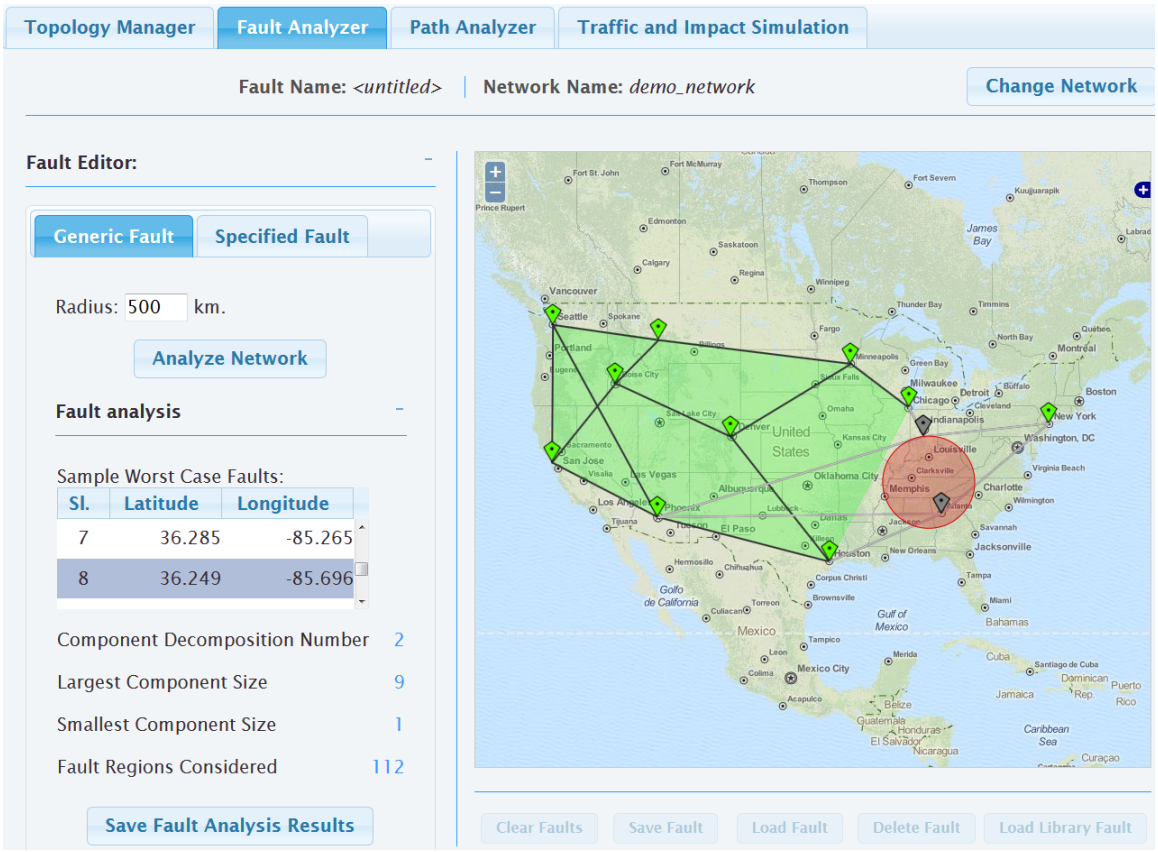


Figure 19: Fault Analyzer – generic fault analysis, metric computations

8.2.1). The results are subsequently reported back to the user. For the network selected in Figure 19 and radius $r = 500$ km., the computed Region-based Component Decomposition Number (RBCDN) is 2, the Region-based Largest Component Size (RBLCS) is 9 and the Region-based Smallest Component Size (RBSCS) is 1. Finally, the number of distinct regions computed is 112.

As shown in Figure 19, the user is also presented with sample worst case fault scenarios where a distinct fault causes the network to fragment into the same number of components as the RBCDN. Selecting one of the listed faults updates the displayed network with the fault's impact. In Figure 19 the fault centered at $36.249°N$, $-85.696°E$ is selected. The nodes and edges rendered inoperable by the fault are grayed out, while the surviving nodes

and edges are shown in green and black respectively. The connected components in the fragmented network are highlighted by a light-green region. In this example, the loss of the two grayed out nodes causes the network to fragment into two disconnected components: one with 9 components, and the other with 1 component. Options for saving the analysis results are available from the menu.

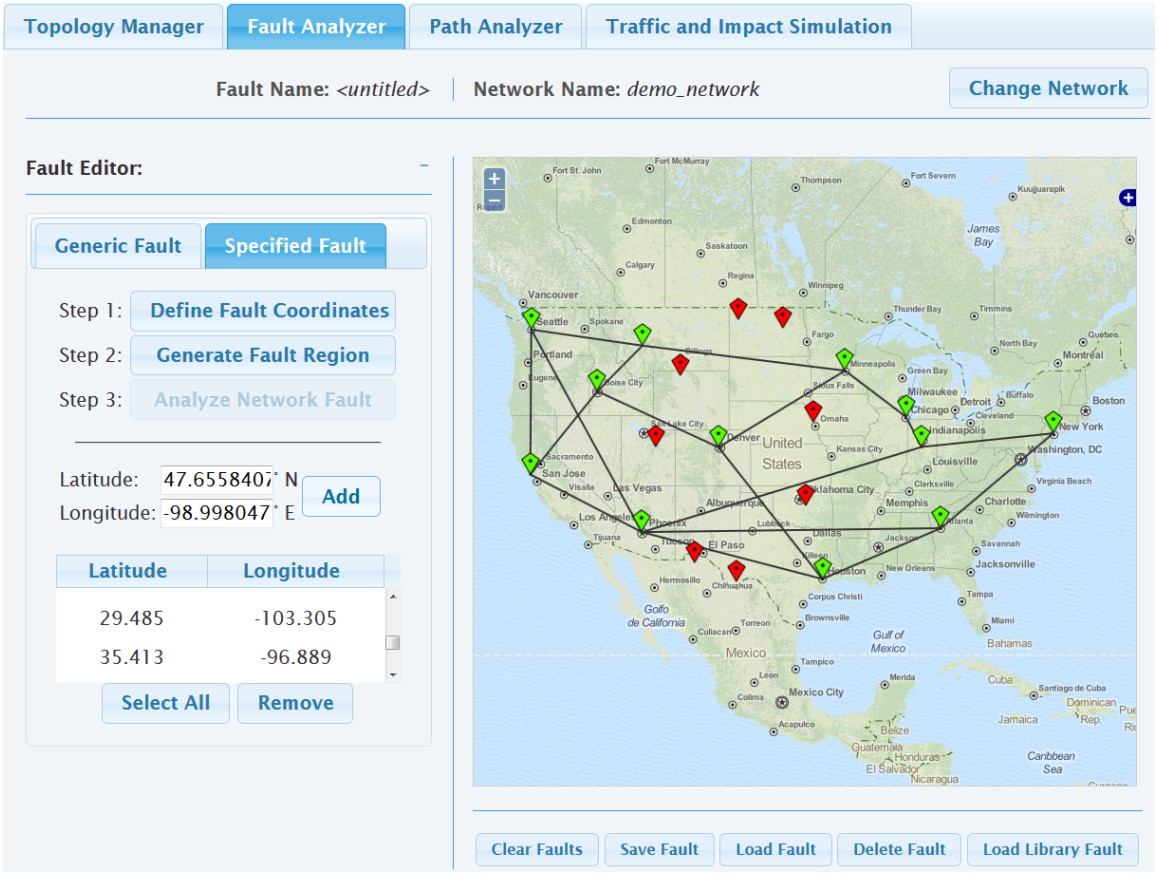


Figure 20: Fault Analyzer – Specified Fault Analysis with user specified fault coordinates

Specified Fault Analysis: In the specified fault analysis, the user can provide the exact coordinates of one or more faults and visualize the impact of these faults on the selected network. The user has the option to save and load faults to visualize the impact of a fault

on different networks. The tool also comes bundled with a set of *library faults* that the user can choose from to simulate fault impact on a network. The current set of library faults consist of the coordinates of the 50 states of the USA. The inclusion of a fault library in the tool is to provide the user with pre-defined fault scenarios based on known fault patterns, faults centered at a target of interest, or recorded faults, such as recorded fault impact zones of Level 4 hurricanes such as hurricane Katrina or hurricane Sandy.

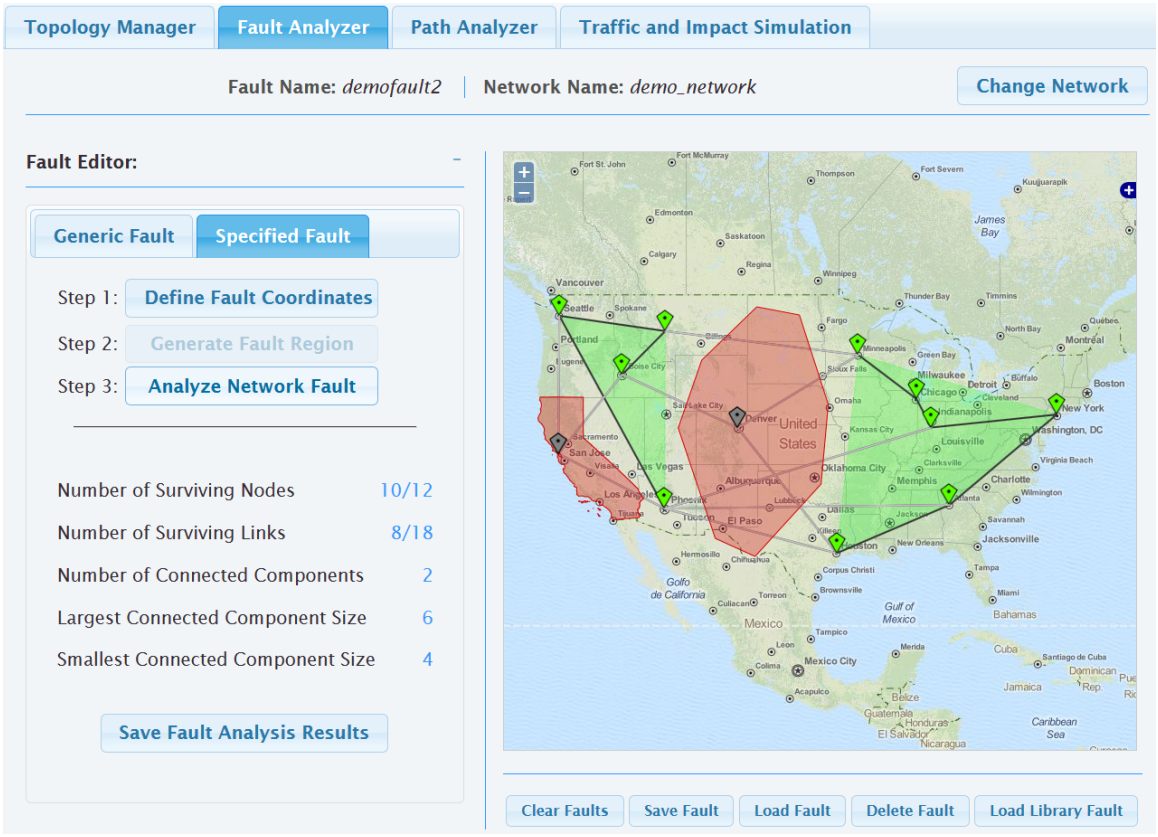


Figure 21: Fault Analyzer – Fault impact of the user specified fault and an imported library fault (coordinates for the state of California, USA)

As shown in Figure 20, to specify the exact coordinates of the fault region the user can either type in the exact coordinates of the fault region coordinates, or can click on the map to add such coordinates. The user also has the option for importing library faults. Once all

the fault regions are defined, RAPTOR can simulate the impact of the fault on the selected network.

In Figure 21, in addition to the user specified fault region of Figure 20, the boundary of the state of California has been imported from the fault library and the selected network has been analyzed for these two fault regions. The updated map shows the impacted nodes and edges in gray, while the operable nodes and edges are shown in green and black respectively. The connected components are shown with a green region. As seen in Figure 21 the left menu displays impact statistics such as, the number of surviving nodes/edges and the number of connected components. The user is provided with the option to save the analysis results for later reference, and also the option to save the defined fault regions for later use.

8.3.2.3 Path Analyzer

The *Path Analyzer* allows users to analyze a network by computing paths between source and destination nodes that provide protection against spatially correlated faults. The Path Analyzer allows users to specify a fault feature, and the tool then computes paths between the given source and destination nodes such that (i) at least one of the paths survive in the presence of one or more spatially correlated faults, and (ii) satisfy other network resource constraints.

In the current version of the tool the faults considered are circular faults and the supported fault feature that can be specified by the user is the fault radius r. The number of spatially correlated faults considered for path analysis is one, and the number of paths computed is two, i.e. RAPTOR computes two paths such that if a single circular fault with radius r occurs anywhere in the network, at least one of the paths computed will not be

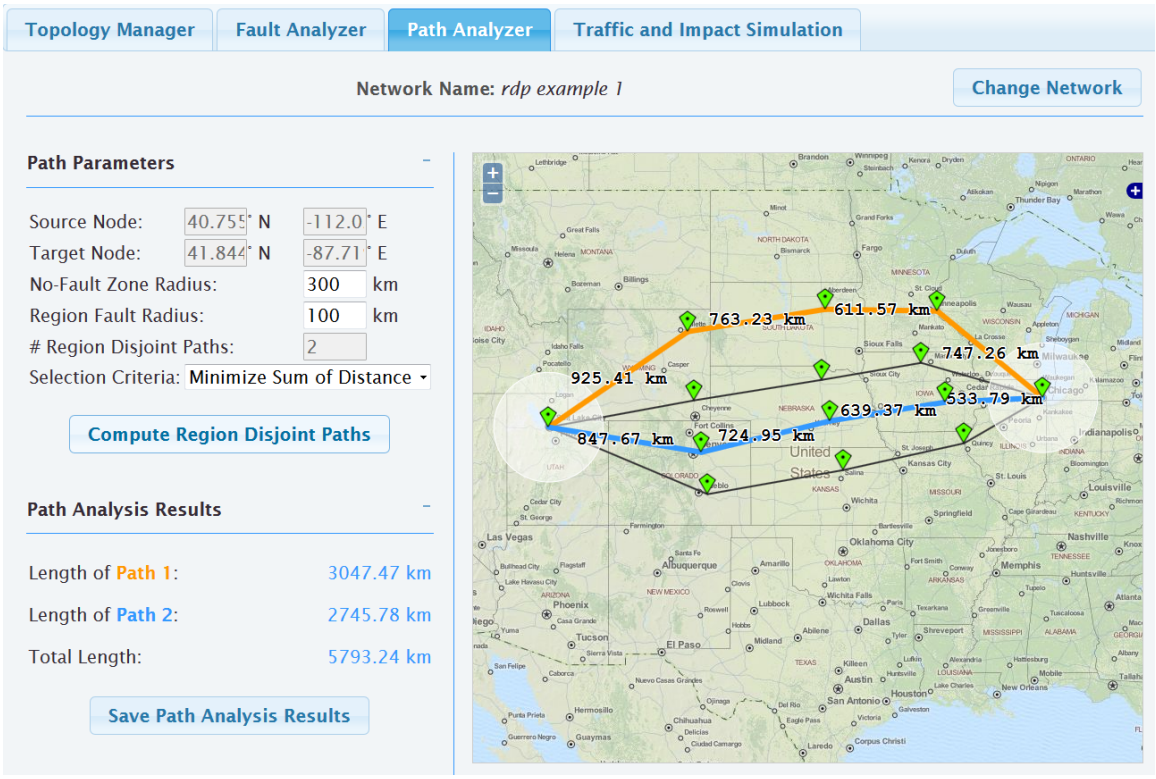


Figure 22: Region disjoint paths between a source and destination nodes for given fault radius $r = 100\ km.$ and no-fault zone radius $nf_r = 300\ km.$

affected by the fault. The network resource constraint supported is that the sum of lengths of the paths must be minimum.

It may be noted that a single fault can also render inoperable either the source node, or the destination node, or both, and thus there always exists a fault region such that no region disjoint paths may exist. To accommodate this situation when the source and/or destination nodes themselves are part of the fault region, the tool supports a *"No-Fault Zone"* parameter. The user can specify a no-fault zone radius nf_r for the source (destination) node that reserves a circular areas with radius nf_r centered at the source (destination) node such that network entities, or parts of a network entity (such as an edge segment), that fall within this no-fault zone are immune to faults.

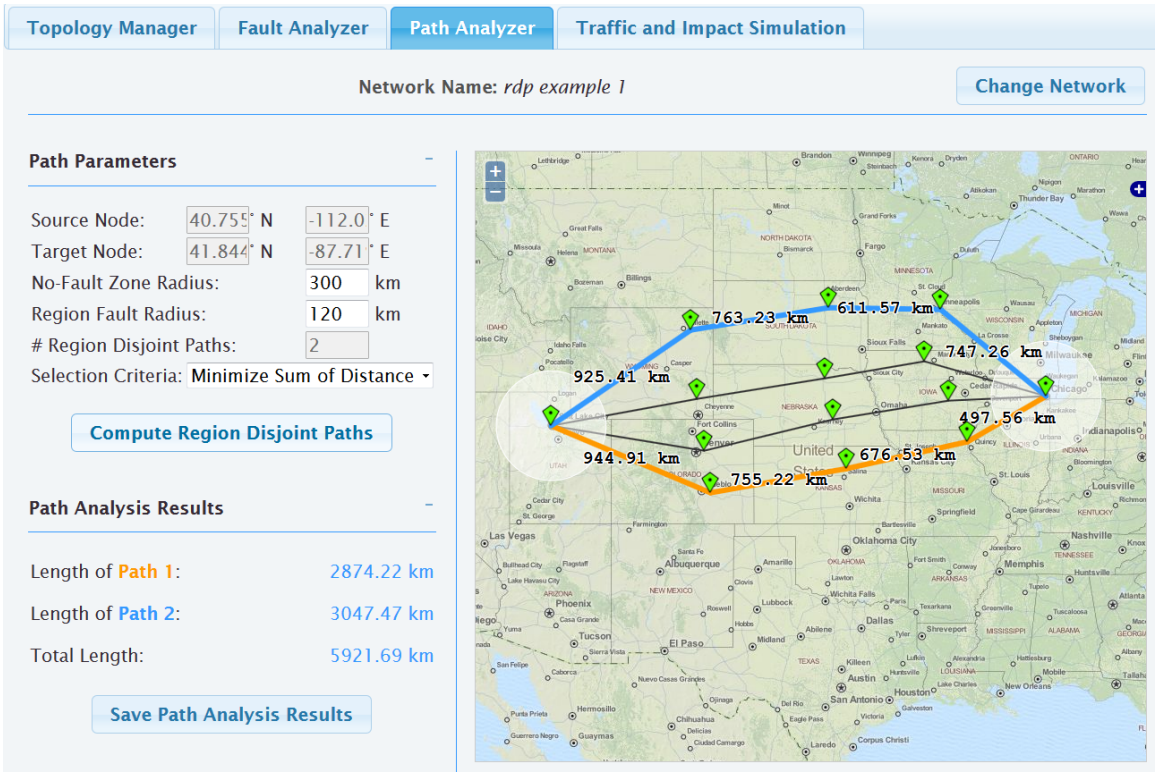


Figure 23: Region disjoint paths between a source and destination nodes for given fault radius $r = 120\ km.$ and no-fault zone radius $nf_r = 300\ km.$

Figures 22 and 23 show screen grabs of the path analyzer computation for different input values of fault radius (r). The no-fault zone is set to a radius of $nf_r = 300$ km. and is shown as a white circular region centered at the source and destination nodes. The computed paths are shown in orange and blue, and the lengths of each of these two paths are reported in the left menu. The effect of the path selection criteria, i.e. the sum of the lengths of the two paths must be minimum, is also visible in Figures 22 and 23. In Figure 22 when $r = 100$ km., the sum of lengths of the two paths is 5793.24 km., however in Figure 23 increasing r to 120 km., the previously computed paths are no longer feasible as a region fault exists that can impact both these paths. Hence, new paths are computed and the sum of the new lengths is 5921.69 km.

8.3.2.4 Traffic and Fault Impact Simulation

For a selected network, the *Traffic and Impact Simulator* allows users to generate traffic and faults to analyze the impact of faults on a load bearing network. To perform this analysis, a simulation schedule consisting of bandwidth requests and faults is generated by the tool using user provided simulation parameters. Parameters such as total number of time steps in the schedule, total number of requests in the schedule, minimum/maximum request bandwidth and minimum/maximum request hold times can be specified by the user. The source and destination nodes for each request can be generated randomly, or can be user specified. For introducing faults in the schedule, the user can specify the number of faults to introduce and can either specify the fault coordinates, or introduce random circular faults from the set of all possible *distinct* circular faults for a specified fault radius. Time intervals of the faults can be user specified, or can be randomly generated by the tool. Using these settings, the tool then generates a time stepped simulation schedule of requests and faults. The user can then select the routing algorithm to be used and proceed to run the simulation.

As shown in the screen grabs of Figures 24 and 25, the left menu of the Traffic and Impact Simulator contains the fault and simulation parameter fields that can be used to generate the schedule and run the simulation. The tables below the map's dimensions allow the user fine grained control over the requests and faults that will be simulated. Once the simulation is complete, for each time interval, the network state can be visualized from the "Event Simulation Results" table. The user can click on a row of this table to visualize the network state on the map for that specific time interval. The user can also "play" the simulation results and the tool will iterate over all the time steps and update the map with the network state at each step. In Figures 24 and 25 the impact of a fault and the corresponding

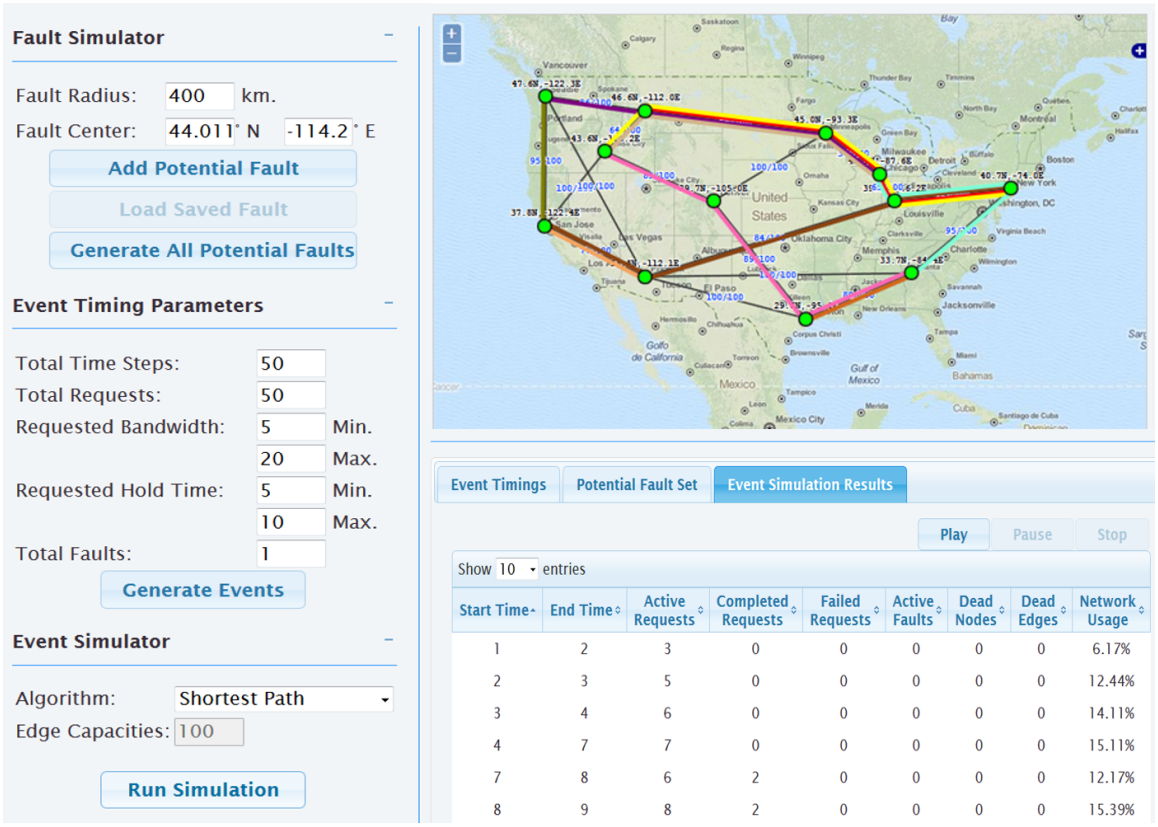


Figure 24: Traffic and Fault Impact Simulator – Pre-Fault network state

response of the network is shown. In Figure 24 the network is fault free, but in Figure 25 a fault is introduced and an edge is rendered inoperable. It can be seen that the red and yellow flows of Figure 24 are impacted by the fault, however, as bandwidth is available, in Figure 25 the flows are rerouted in response to this fault.

8.3.3 Performance Analysis

As a preliminary performance analysis of the tool, a 40 node test network was used where each node had a degree of at least 3. Using two users, the Path Analysis, Fault

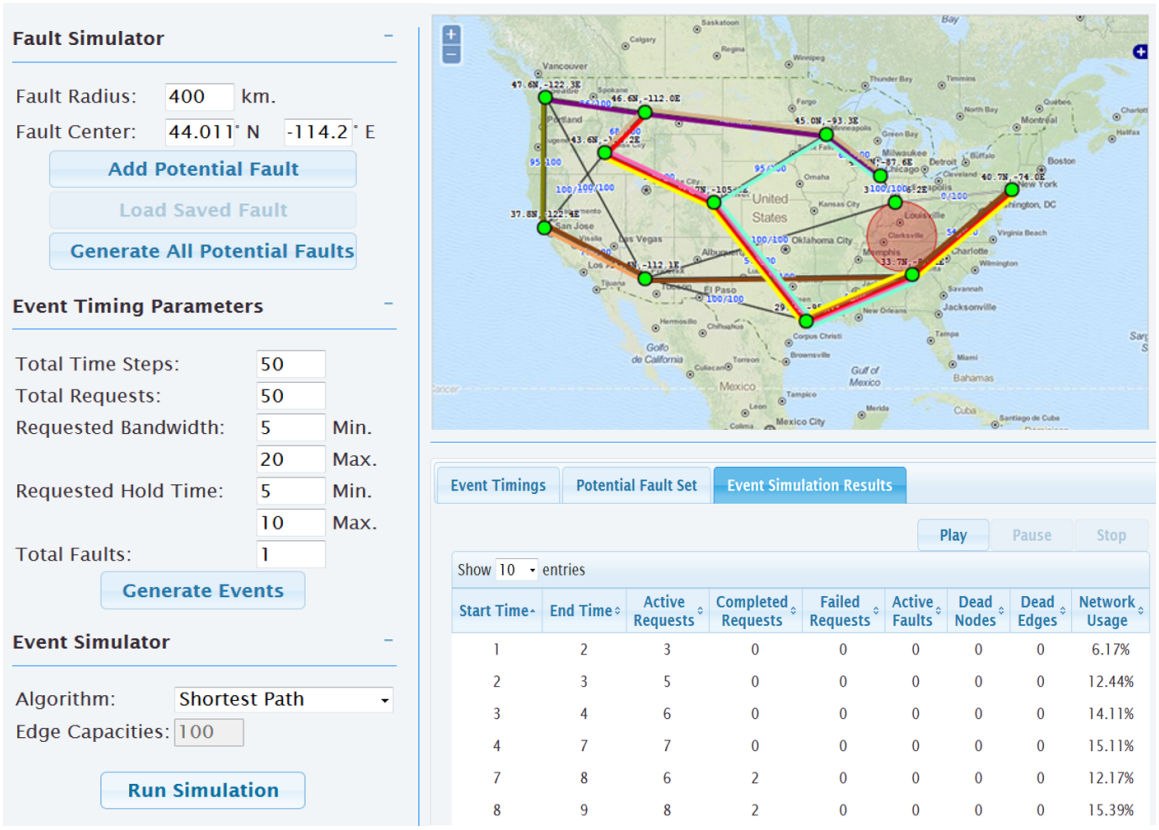


Figure 25: Traffic and Fault Impact Simulator – Post-Fault network state, rerouted red and yellow flows

Analysis and Traffic and Impact Simulator features were executed concurrently on the test network using different input parameters. In these tests it was observed that for both users, computations were completed within a minute of the user request. The Path Analysis module, however, required variable computation time depending on the source and destination nodes chosen. This is due to the fact that the Path Analysis computation is performed as an ILP, however, in all the tests, the path computations took at most 3 minutes.

Chapter 9

CONCLUSION AND FUTURE WORK

In this dissertation, an overview of the existing literature on interdependent network models, and the limitations of the existing models were presented. This dissertation also presented the *Implicative Interdependency Model* (IIM), a model that overcomes the limitations of existing graph based interdependent network models. The dissertation highlighted the need for studying the failure cascade propagation in critical interdependent networks as such failures can pose credible threats to the nation. Motivated by this need, in the IIM setting, this dissertation addressed the $\mathcal{K}$ *Most Vulnerable Nodes* ($\mathcal{K}$MVN) problem and the *Smallest Pseudo Target Set Identification Problem* (SPTSIP) in interdependent networks. The computational complexities of the problems were analyzed and solution techniques were proposed. The efficacy of the proposed solutions was evaluated using the power and communication network infrastructure data of Maricopa County, Arizona.

Apart from analysis of critical infrastructure networks, this dissertation also studied resource allocation problems in cyber-physical systems. Specifically, the *Reader Minimization Problem* (RMP) in a RFID system and the *Mule Minimization Problem* (MMP) in sensor networks were formulated in this dissertation. Both the RMP and MMP problems were shown to be NP-Complete, and flow based solution techniques were proposed by generalizing the well known minimum flow problem, and then solving the generalized minimum flow problem optimally using Integer Linear Programming.

This dissertation also presented a summary of the work done towards developing RAPTOR, a network planning and management tool intended to support design and analysis of single layer and multi-layer networks in the presence of spatially correlated faults. It

was noted that RAPTOR is particularly suitable for planning and design of critical infrastructures. To present a background of the tool, the underlying novel concepts that have been developed to enhance robustness of networks in presence of region based faults were first discussed, and how those concepts have been incorporated into the tool were then described. The goal of this dissertation was to bring to the attention of the networking research community about the existence of RAPTOR as a tool that consolidates a large body of work on spatially correlated faults. As of writing this dissertation, no such tool is available today that supports planning and designing of single layer and multi-layer networks in the presence of spatially correlated faults.

Apart from the work performed towards this dissertation, some of the possible future extensions of this work are outlined below:

An Attacker-Defender Game for Interdependent Power-Communication Networks

The $\mathcal{K}$-Most Vulnerable ($\mathcal{K}$MVN) problem addressed in Chapter 4 can be viewed from an attackers perspective where an attacker can jeopardize up to $\mathcal{K}$ nodes with the intention of causing the maximum damage to the network. This scenario can be further extended to include a defender of the network, such as the utility company or the Internet Service Provider (ISP), who can protect or "*harden*" up to $\mathcal{B}$ nodes of the network and intends to minimize the impact of an attack. This modified "attacker-defender" scenario can be modeled in a game theoretic setting where the attacker and defender are aware of each others existence, their available budgets, and also the underlying interdependent infrastructure. It will be interesting to study this problem in a game theoretic setting where both the players attempt to maximize their impact on the network with their available resources of $\mathcal{K}$ and $\mathcal{B}$ nodes respectively.

Incomplete or Incorrect Information in the IIM

As discussed in Chapter 3, in the *Implicative Interdependency Model* (IIM), the dependent

relationships between the network entities are represented using Boolean Logic and are termed as *Implicative Interdependency Relations* (IDRs). In the studies presented in this dissertation, it was assumed that for a given set of network entities, the set of *given* IDRs that define a system is completely accurate, i.e., the set of equations can correctly model the dependencies between the entities such that in the event of a failure at a time step (say $t = 0$), it is (accurately) known what entities will fail at the next time step ($t = 1$). However, such accurate information may not always be available. Thus it will be of importance to know the implications of working with incomplete or incorrect IDRs for the $\mathcal{K}$ most vulnerable nodes identification problem (Chapter 4), and the smallest pseudo target set identification problem (Chapter 5).

Analyzing the Impact of Discretization

In Chapter 6 and 7 the Reader Minimization Problem (RMP) and the Mule Minimization Problem (MMP) were respectively presented. In the proposed solution techniques for both problems time and space were discretized into equal intervals of δ and ε respectively. It was argued that the δ and ε parameters allowed the technique some control over the cost vs. quality of the computed solution. It was also experimentally shown that lower values of δ and ε increased computation time, but had the potential to improve the computed solution (by minimizing the number of required readers/mules). However, as of writing this dissertation no analytical analysis exists that establishes the relationship between the selection of the δ and ε parameters and the computed solution. It will be useful to analytically establish ranges of the δ and ε parameters that can ensure the computed solution is bounded within a given factor of the optimal solution in a continuous setting.

Region Disjoint Path Computation

In Chapter 8 it was outlined that the Region Disjoint Path computation implemented for the tool was carried out with the help of an Integer Linear Program (Section 8.2.3). As Integer

Linear Programs have the propensity to take an exponential amount of time depending upon the input, a better technique would involve computing region disjoint paths between a pair of source and destination nodes in polynomial time. As of writing this dissertation, no such techniques are known that compute region disjoint paths in wired networks, where edges can fail independently without node failure, in polynomial time. It will be useful to analyze the computational complexity of this problem and provide polynomial time algorithms to compute two (or more) region disjoint paths between a pair of source and destination nodes.

REFERENCES

[1] R. Smith, *U.S. risks national blackout from small-scale attack*, http://online.wsj.com/news/articles/SB10001424052702304020104579433670284061220, Mar. 2014.

[2] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, no. 7291, pp. 1025–1028, 2010.

[3] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin, "Networks formed from interdependent networks," *Nature Physics*, vol. 8, no. 1, 2011.

[4] J. Shao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, "Cascade of failures in coupled network systems with multiple support-dependence relations," *Physical Review E*, vol. 83, no. 3, p. 036 116, 2011.

[5] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, and R. Setola, "Modelling interdependent infrastructures using interacting dynamical models," *International Journal of Critical Infrastructures*, vol. 4, no. 1, pp. 63–79, 2008.

[6] P. Zhang, S. Peeta, and T. Friesz, "Dynamic game theoretic model of multi-layer infrastructure networks," *Networks and Spatial Economics*, vol. 5, no. 2, pp. 147–178, 2005.

[7] J.-F. Castet and J. H. Saleh, "Interdependent multi-layer networks: Modeling and survivability analysis with applications to space-based networks," *PloS one*, vol. 8, no. 4, e60402, 2013.

[8] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," *ArXiv preprint arXiv:1304.0356*, 2013.

[9] D. T. Nguyen, Y. Shen, and M. T. Thai, "Detecting critical nodes in interdependent power networks for vulnerability assessment," *Smart Grid, IEEE Transactions on*, vol. 4, no. 1, pp. 151–159, 2013.

[10] A. Bernstein, D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman, "Power grid vulnerability to geographically correlated failuresanalysis and control implications," in *INFOCOM, 2014 Proceedings IEEE*, IEEE, 2014, pp. 2634–2642.

[11] A. J. Wood and B. F. Wollenberg, *Power generation, operation, and control*. John Wiley & Sons, 2012.

[12] A. Fudenberg and J. Tirole, *Game Theory*. Ane Books, 2010.

[13] M. R. Garey and D. S. Johnson, "Computer and intractability," *A Guide to the NP-Completeness. New York, NY: WH Freeman and Co.*, 1979.

[14] J. Kleinberg and É. Tardos, *Algorithm design*. Pearson Education, 2006.

[15] N. Sherman, *Amazon showcases robots at massive new warehouse*, 2015. [Online]. Available: http://www.baltimoresun.com/business/bs-bz-amazon-grand-opening-20150928-story.html.

[16] E. Ackerman, *Flying inventory assistants are a good use for drones*, 2014. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/aerial-robots/flying-inventory-assistants-are-a-good-use-for-drones.

[17] J.-B. Eom and T.-J. Lee, "RFID reader anti-collision algorithm using a server and mobile readers based on conflict-free multiple access," in *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International*, IEEE, 2008, pp. 395–399.

[18] E. Hamouda, N. Mitton, and D. Simplot-Ryl, "Reader anti-collision in dense rfid networks with mobile tags," in *IEEE International Conference on RFID-Technologies and Applications (RFID-TA 2011)*, Barcelona, Spain, Sep. 2011.

[19] E. DiGiampaolo and F. Martinelli, "Mobile robot localization using the phase of passive uhf rfid signals," *Industrial Electronics, IEEE Transactions on*, vol. 61, no. 1, pp. 365–376, 2014.

[20] L. Shangguan, Z. Li, Z. Yang, M. Li, Y. Liu, and J. Han, "Otrack: Towards order tracking for tags in mobile rfid systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 8, pp. 2114–2125, 2014.

[21] K.-i. Hwang, S.-S. Yeo, and J. H. Park, "Distributed tag access with collision-avoidance among mobile rfid readers," in *Computational Science and Engineering, 2009. CSE'09. International Conference on*, IEEE, vol. 2, 2009, pp. 621–626.

[22] W. Zhu, J. Cao, H. Chan, X. Liu, and V. Raychoudhury, "Mobile rfid with a high identification rate," *Computers, IEEE Transactions on*, vol. 63, no. 7, pp. 1778–1792, 2014.

[23] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto, "Optimal packing and covering in the plane are np-complete," *Information processing letters*, vol. 12, no. 3, pp. 133–137, 1981.

[24] S. Even, *Graph algorithms*. Cambridge University Press, 2011.

[25] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network," in *Proceedings of the 3rd international conference on Embedded networked sensor systems*, ACM, 2005, pp. 154–165.

[26] D. Mascareñas, E. Flynn, C. Farrar, G. Park, and M. Todd, "A mobile host approach for wireless powering and interrogation of structural health monitoring sensor networks," *Sensors Journal, IEEE*, vol. 9, no. 12, pp. 1719–1726, 2009.

[27] R. Sugihara and R. K. Gupta, "Path planning of data mules in sensor networks," *ACM Trans. Sen. Netw.*, vol. 8, no. 1, 1:1–1:27, Aug. 2011, ISSN: 1550-4859. DOI: 10.1145/1993042.1993043. [Online]. Available: http://doi.acm.org/10.1145/1993042.1993043.

[28] A. Somasundara, A. Ramamoorthy, M. B. Srivastava, *et al.*, "Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines," in *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, IEEE, 2004, pp. 296–305.

[29] Y. Gu, D. Bozda, R. W. Brewer, and E. Ekici, "Data harvesting with mobile elements in wireless sensor networks," *Computer Networks*, vol. 50, no. 17, pp. 3449–3465, 2006, ISSN: 1389-1286. DOI: http://dx.doi.org/10.1016/j.comnet.2006.01.008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138912860600020X.

[30] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava, "Mobile element scheduling with dynamic deadlines," *IEEE Trans. Mob. Comput.*, vol. 6, no. 4, pp. 395–410, 2007. DOI: 10.1109/TMC.2007.57. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TMC.2007.57.

[31] A. Meliou, D. Chu, J. Hellerstein, C. Guestrin, and W. Hong, "Data gathering tours in sensor networks," in *Proceedings of the 5th international conference on Information processing in sensor networks*, ACM, 2006, pp. 43–50.

[32] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, ACM, 2004, pp. 187–198.

[33] M. M. Bin Tariq, M. Ammar, and E. Zegura, "Message ferry route design for sparse ad hoc networks with mobile nodes," in *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '06, Flo-

rence, Italy: ACM, 2006, pp. 37–48, ISBN: 1-59593-368-9. DOI: 10.1145/1132905.1132910. [Online]. Available: http://doi.acm.org/10.1145/1132905.1132910.

[34] M. Ma and Y. Yang, "Sencar: An energy-efficient data gathering mechanism for large-scale multihop sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 10, pp. 1476–1488, 2007, ISSN: 1045-9219. DOI: http://doi.ieeecomputersociety.org/10.1109/TPDS.2007.1070.

[35] G. Fenu and M. Nitti, "Strategies to carry and forward packets in VANET," in *Digital Information and Communication Technology and Its Applications - International Conference, DICTAP 2011, Dijon, France, June 21-23, 2011. Proceedings, Part I*, 2011, pp. 662–674. DOI: 10.1007/978-3-642-21984-9_54. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21984-9_54.

[36] T. Razafindralambo, N. Mitton, A. C. Viana, M. Dias De Amorim, and K. Obraczka, "Adaptive deployment for pervasive data gathering in connectivity-challenged environments," in *Eighth Annual IEEE International Conference on Pervasive Computing and Communications (PERCOM)*, France, Mar. 2010, p. 000. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00472656.

[37] A. Sen, B. H. Shen, L. Zhou, and B. Hao, "Fault-tolerance in Sensor Networks: A New Evaluation Metric," in *Proceedings of IEEE Infocom*, Barcelona, Spain, Apr. 2006, pp. 1–12.

[38] R. Diestel, *Graph Theory*. Springer, 2005.

[39] NetXT Lab, Arizona State University. (2016). Raptor: The Network Planning and Management Tool, [Online]. Available: http://netsci.asu.edu/networktool/.

[40] S. Neumayer and E. Modiano, "Network reliability with geographically correlated failures," in *Proceedings of IEEE INFOCOM, 2010*, 2010.

[41] Y. Cheng, M. T. Gardner, J. Li, R. May, D. Medhi, and J. P. Sterbenz, "Optimised heuristics for a geodiverse routing protocol," in *10th International Conference on the Design of Reliable Communication Networks (DRCN), 2014*, 2014, pp. 1–9.

[42] P. Agarwal, A. Efrat, S. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman, "The resilience of wdm networks to probabilistic geographical failures," in *Proceedings of IEEE INFOCOM*, 2011.

[43] S. Trajanovski, F. Kuipers, A. Ilic, J. Crowcroft, and P. Van Mieghem, "Finding critical regions and region-disjoint paths in a network," *IEEE/ACM Transactions on Networking*, vol. 23, no. 3, pp. 908–921, 2015.

[44] S. Banerjee, A. Das, A. Mazumder, Z. Derakhshandeh, and A. Sen, "On the impact of coding parameters on storage requirement of region-based fault tolerant distributed file system design," in *Computing, Networking and Communications (ICNC), International Conference on*, IEEE, 2014, pp. 78–82.

[45] A. Mazumder, A. Das, C. Zhou, and A. Sen, "Region based fault-tolerant distributed file storage system design under budget constraint," in *Reliable Networks Design and Modeling (RNDM), 2014 6th International Workshop on*, IEEE, 2014, pp. 61–68.

[46] A. Sen, A. Mazumder, S. Banerjee, A. Das, C. Zhou, and S. Shirazipourazad, "Region-based fault-tolerant distributed file storage system design in networks," *Networks*, vol. 66, no. 4, pp. 380–395, 2015.

[47] A. Sen, S. Murthy, and S. Banerjee, "Region-based connectivity-a new paradigm for design of fault-tolerant networks," in *High Performance Switching and Routing, 2009. HPSR 2009. International Conference on*, IEEE, 2009, pp. 1–7.

[48] S. Banerjee, S. Shirazipourazad, P. Ghosh, and A. Sen, "Beyond connectivity-new metrics to evaluate robustness of networks," in *High Performance Switching and Routing (HPSR), 2011 IEEE 12th International Conference on*, IEEE, 2011, pp. 171–177.

[49] S. Banerjee, S. Shirazipourazad, and A. Sen, "Design and analysis of networks with large components in presence of region-based faults," in *International Conference on Communications (ICC)*, IEEE, 2011, pp. 1–6.

[50] A. Das, J. Banerjee, and A. Sen, "Root cause analysis of failures in interdependent power-communication networks," in *Military Communications Conference (MILCOM), 2014 IEEE*, IEEE, 2014, pp. 910–915.

[51] J. Banerjee, A. Das, C. Zhou, A. Mazumder, and A. Sen, "On the entity hardening problem in multi-layered interdependent networks," in *WIDN Workshop (INFOCOM WKSHPS), 2015 IEEE Conference on Computer Communications*, IEEE, 2015, pp. 648–653.

[52] J. Banerjee, C. Zhou, A. Das, and A. Sen, "On robustness in multilayer interdependent networks," in *International Conference on Critical Information Infrastructures Security*, Springer, 2015, pp. 247–250.

[53] A. Sen, A. Mazumder, J. Banerjee, A. Das, and R. Compton, "Identification of k most vulnerable nodes in multi-layered network using a new model of interdependency," in *Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 2014, pp. 831–836.

[54] A. Das, C. Zhou, J. Banerjee, A. Sen, and L. Greenwald, "On the smallest pseudo target set identification problem for targeted attack on interdependent power-communication networks," in *Military Communications Conference, MILCOM 2015-2015 IEEE*, IEEE, 2015, pp. 1015–1020.

[55] A. Mazumder, C. Zhou, A. Das, and A. Sen, "Progressive recovery from failure in multi-layered interdependent network using a new model of interdependency," in *Conference on Critical Information Infrastructures Security (CRITIS), 2014*, Springer, 2014.

[56] OpenStreetMap Contributors. (2016). OpenStreetMap, [Online]. Available: www.openstreetmap.org.