2-28-2017

Oblivious Network Optimization and Security Modeling in Sustainable Smart Grids and Cities

Kianoosh G. Boroojeni
Florida International University, kghol002@fiu.edu

DOI: 10.25148/etd.FIDC001808

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

OBLIVIOUS NETWORK OPTIMIZATION AND SECURITY MODELING IN SUSTAINABLE SMART GRIDS AND CITIES

A dissertation submitted in partial fulfillment of the

requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Kianoush Gholamiboroujeni

2017

To: Interim Dean Ranu Jung
College of Engineering and Computing

This dissertation, written by Kianoush Gholamiboroujeni, and entitled Oblivious Network Optimization and Security Modeling in Sustainable Smart Grids and Cities, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this dissertation and recommend that it be approved.

_____
Leonardo Bobadilla

_____
Ning Xie

_____
Niki Pissinou

_____
Arif Sarwat

_____
S. S. Iyengar, Major Professor

Date of Defense: March 27, 2017

The dissertation of Kianoush Gholamiboroujeni is approved.

_____
Interim Dean Ranu Jung
College of Engineering and Computing

_____
Andrés G. Gil
Vice President for Research and Economic Development
and Dean of the University Graduate School

Florida International University, 2017

DEDICATION

I dedicate this work to my lovely parents, Fariba and Mehrab, who have supported me throughout my PhD years.

ACKNOWLEDGMENTS

I would like to thank NSF, NIH and other research funding institutes for supporting this work which is evolved from my research on oblivious network optimization and security in smart grids. I would like to express my sincere gratitude to my PhD advisor, Professor S. S. Iyengar, and other members of my PhD dissertation committee. Also, I am really grateful to my colleague, M. Hadi Amini, for his unbounded support and my friend, Mehrnoosh Asadi, for her continuous encouragement in my PhD years. Finally, I would like to thank my family for their unconditional support and encouragement throughout my life.

ABSTRACT OF THE DISSERTATION

OBLIVIOUS NETWORK OPTIMIZATION AND SECURITY MODELING IN SUSTAINABLE SMART GRIDS AND CITIES

by

Kianoush Gholamiboroujeni

Florida International University, 2017

Miami, Florida

Professor S. S. Iyengar, Major Professor

Today's interconnected world requires an inexpensive, fast, and reliable way of transferring information. There exists an increasingly important need for intelligent and adaptable routing of network flows. In the last few years, many researchers have worked toward developing versatile solutions to the problem of routing network flows in unpredictable circumstances. These attempts have evolved into a rich literature in the area of "oblivious network design" which typically route the network flows via a routing scheme that makes use of a spanning tree or a set of trees of the graph representation of the network.

In the first chapter, we provide an introduction to network design. This introductory chapter has been designed to clarify the importance and position of oblivious routing problems in the context of network design as well as its containing field of research. Part I of this dissertation discusses the fundamental role of linked hierarchical data structures in providing the mathematical tools needed to construct rigorous versatile routing schemes and applies hierarchical routing tools to the process of constructing versatile routing schemes. Part II of this dissertation applies the routing tools generated in Part I to address real-world network optimization problems in the area of electrical power networks, clusters of microgrids, and content-centric networks. There is an increasing concern regarding the secu-

rity and privacy of both physical and communication layers of smart interactive customer-driven power networks, better known as smart grids. Part III of this dissertation utilizes an advanced interdisciplinary approach to address existing security and privacy issues, proposing legitimate countermeasures for each of them from the standpoint of both computing and electrical engineering. The proposed methods are theoretically proven by mathematical tools and illustrated by real-world examples.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Network is a common concept that is used to describe a group of interconnected things. These connections allow for individuals to communicate and cooperate by passing information/contents *flows* through the various paths within a given network. In fact, we are all surrounded by networks. From scientific discoveries to natural phenomena, the concept of a network exists in biological, physical, and chemical circumstances, various kinds of transportation, telecommunication, energy distribution, etc. For instance, a computer network like the Internet has a vast number of cyber devices that transfer flows of data between computer hosts all around the globe.

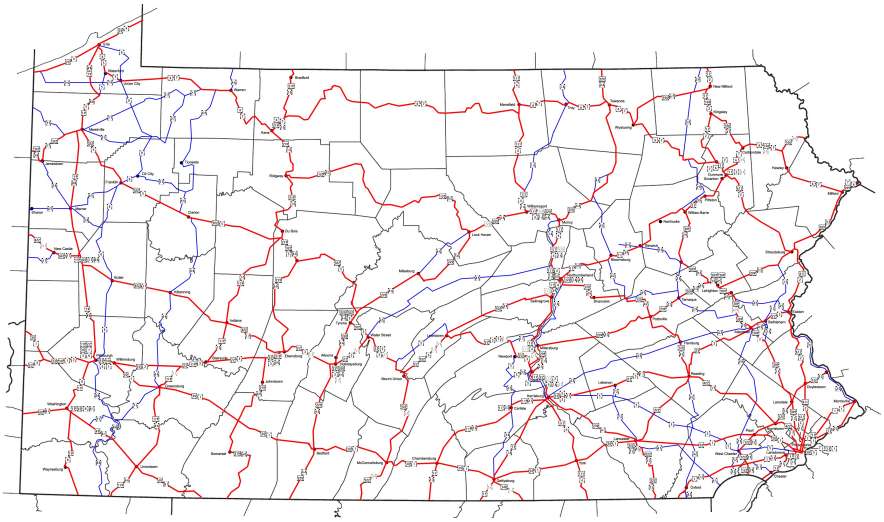


Figure 1.1: Pennsylvania's US Highways in 1928. Map by Timothy Reichard

Figure 1.1 exemplifies yet another familiar network representation, an outline of U.S. highways throughout Pennsylvania in the late 1920s. In this ground transportation network, cities are interconnected by roads through which vehicles pass in the form of network flows. Building an efficient ground transportation network and optimizing flow routes is very critical and leads to substantial savings in time and

flow costs, especially in the long run. Similar concerns exist in telecommunication networks as optimized routing algorithms in Internet routers reduce electrical costs and create a considerable increase in network bandwidth.

Optimizing the flow costs in various networks are usually modeled by Minimum-Costs Flow Problems (MCFPs). In contrast to the traditional MCFPs that are defined with a well-defined set of commodities (with specific size and source/destination nodes) and given flow cost function for every edge, oblivious network routing aims to solve an MCFP in which either the flow cost function is not specified (flow cost is oblivious), or the commodities size, source, and destination are not specified (commodities are oblivious).

Oblivious network routing approach solves oblivious MCFPs by employing a practical method not guaranteed to be perfect, but sufficient for the immediate goal which is obtaining an acceptable approximation of the optimal solution. In this regard, oblivious network routing is considered to be a heuristic method since it can be used to speed up the process of finding a satisfactory solution (while computing the optimal solution is impractical).

Research Questions

The focus of the dissertation is to address the following research questions:

Question 1 Can an oblivious routing algorithm be utilized in order to construct an overlay network routing scheme optimizing power generation cost in power networks?

Question 2 Can an oblivious routing algorithm be utilized in order to construct an overlay network routing scheme reducing the cost of power routing in clusters of microgrids?

Question 3 Can an oblivious routing algorithm be utilized in order to construct an overlay network routing scheme mitigating the risk of power congestion in power networks?

Question 4 Can an oblivious routing algorithm be utilized in order to create an overlay network routing scheme in order to mitigate the zombie DDoS attack in communicating networks?

1.1 Related Work

There has been a lot of research works to deal with network optimization problems, especially minimum-cost flow problems, including "Single-Sink Buy-at-Bulk" (SSBB) and "Multi-Sink Buy-at-Bulk" (MSBB). In order to have a literature review on these works, we classify them into classic and oblivious approaches.

Non-Oblivious Network Design

The non-oblivious, classic network design problems have been primarily considered in both Operations Research and the Computer Science literatures under the context of flows with concave costs. The single-sink variant of the problem was first introduced by Salman et al. [1]. They presented an $O(\log\ n)$-approximation for SSBB in Euclidean graphs by applying the method of Mansour and Peleg [2]. Further, Bartal's tree embeddings [3] can be used to improve their ratio to $O(\log\ n\ \log\ \log\ n)$. An $O(\log^2 n)$-approximation was given by Awerbuch et al. [4] for graphs with general metric spaces. Bartal et al. [5] further improved this result to $O(\log\ n)$. Moreover, Guha [6] provided the first constant-factor approximation to the problem, whose ratio was estimated to be around 9000 by Talwar [7]. This constant has been further improved by Grandoni and Rothvoss [8].

Oblivious Network Design

Goel et al. [9] built an overlay tree on a graph that satisfies the triangle-inequality. Their technique is based on the maximum matching algorithm that guarantees $(1 + \log k)$-approximation, where k is the number of sources. Their solution is oblivious with respect to the fusion cost function f. In a related paper [10], Goel et al. construct (in polynomial time) a set of overlay trees from a given general graph such that the expected cost of a tree for any f is within an $O(1)$-factor of the optimum cost for that f. A recent improvement by Goel [11] provides the first constant guarantee on the simultaneous ratio of $O(1)$.

Jia et al. [12] built a Group Independent Spanning Tree Algorithm (GIST) that constructs an overlay tree for randomly deployed nodes in a Euclidean 2 dimensional plane. The tree (that is oblivious to the number of data sources) simultaneously achieves $O(\log\ n)$-approximate fusion cost and $O(1)$-approximate delay. However, their solution assumes a constant fusion cost function.

Lujun Jia et al. [13] provided approximation algorithms for the Travelling Salesman Problem (TSP), Steiner tree and the set cover problems. They presented a polynomial-time $(O(\log(n)), O(\log(n)))$-partition scheme for general metric spaces. An improved partition scheme for doubling metric spaces is also presented that incorporates constant dimensional Euclidean spaces and growth-restricted metric spaces. The authors present a polynomial-time algorithm for Universal Steiner Tree (UST) problem that achieves the polylogarithmic stretch with an approximation guarantee of $O(\log^4 n/\log\log(n))$ for arbitrary metrics and derive a logarithmic stretch, $O(\log(n))$ for any doubling, Euclidean, or growth-restricted metric space over n vertices. Furthermore, they provided a lower bound of $\Omega(\log\ n/\log\log n)$ for UST that holds even when all the vertices are on a plane.

Gupta et al. [14] developed a framework to model the oblivious network design problems (MSBB) and give algorithms with poly-logarithmic approximation ratio. They developed oblivious algorithms that approximately minimize the total cost of routing with the knowledge of aggregation function, the class of load on each edge and nothing else about the state of the network. Their results show that if the aggregation function is summation, their algorithm provides a $O(\log^2 n)$ approximation ratio and when the aggregation function is max, the approximation ratio is $O(\log^2 n \log\log n)$. The authors claimed to provide a deterministic solution by de-randomizing their approach, although the complexity of this de-randomizing process is unclear.

1.2 Methods and Techniques

Oblivious network routing approach solves oblivious MCFPs by employing a practical method not guaranteed to be perfect, but sufficient for the immediate goal which is obtaining an acceptable approximation of the optimal solution. In this regard, oblivious network routing is considered to be a heuristic method since it can be used to speed up the process of finding a satisfactory solution (while computing the optimal solution is impractical).

Oblivious network routing is different with Dynamic (adaptive) Routing (DR) since DR proposes a different solution in response to any change of the MCFP oblivious components; while, oblivious routing deploys a single routing scheme for an oblivious MCFP with the aim of approximating the optimal solution of the problem with oblivious parameters.

The common characteristics of the oblivious routing schemes includes being pretty flexible to the obliviousness of the environment in which the commodities

are flowing and making the traffic flows distributed over the network and preventing the flow-congestion in some specific nodes or edges. Additionally, these types of routing schemes provide a low-cost flow routing in long term even if a wide range of unpredictable events occur in the network like bursty flow derived from a specific node or failure of some node in forwarding the flow through the network. In fact, the versatile routing schemes best fit to those networks that we have little/no knowledge regarding their current and future states.

1.3 Structure of the Dissertation

The dissertation is organized as follows. Part I discusses the fundamental role of linked hierarchical data structures in providing the mathematical tools needed to construct rigorous versatile routing schemes and applies hierarchical routing tools to the process of constructing oblivious routing schemes.

Part II of this dissertation applies the routing tools generated in Part I to address real-world network optimization problems in the area of electrical power networks, clusters of microgrids, and content-centric networks.

Part III of this dissertation utilizes an advanced interdisciplinary approach to address existing security and privacy issues, proposing legitimate countermeasures for each of them from the standpoint of both computing and electrical engineering. The proposed methods are theoretically proven by mathematical tools and illustrated by real-world examples.

Bibliography

[1] Salman, F. S., Cheriyan, J., Ravi, R., & Subramanian, S. (2000). Approximating the single-sink link-installation problem in network design. SIAM Journal on Optimization, 11(3), 595–610.

[2] Mansour, Y., & Peleg, D. (1994). An approximation algorithm for minimum-cost network design (Tech. Rep.). Jerusalem, Israel.

[3] Bartal, Y. (1994). Competitive analysis of distributed online problems - distributed paging. Unpublished doctoral dissertation.

[4] Awerbuch, B., & Azar, Y. (1997). Buy-at-bulk network design. In Focs '97 (p. 542). Washington, DC, USA: IEEE Computer Society.

[5] Bartal, Y. (1998). On approximating arbitrary metrices by tree metrics. In Stoc '98 (pp. 161–168). New York, NY, USA: ACM.

[6] Guha, S., Meyerson, A., & Munagala, K. (2001). A constant factor approximation for the single sink edge installation problems. In Stoc '01 (pp. 383–388). New York, NY, USA: ACM.

[7] Talwar, K. (2002). The single-sink buy-at-bulk lp has constant integrality gap. In Proceedings of the 9th international ipco conference on integer programming and combinatorial optimization (pp. 475–486). London, UK: Springer-Verlag.

[8] Grandoni, F., & Rothvoss, T. (2010). Network design via core detouring for problems without a core. In Icalp (pp. 490–502).

[9] Goel, A., & Estrin, D. (2003). Simultaneous optimization for concave costs: single sink aggregation or single source buy-at-bulk. In Soda '03 (pp. 499–505). Philadelphia, PA, USA: SIAM.

[10] Goel, A., & Post, I. (2009). An oblivious $O(1)$-approximation for single source buy-at-bulk. Foundations of Computer Science, Annual IEEE Symposium on, 0, 442–450.

[11] Goel, A., & Post, I. (2010). One tree suffices: A simultaneous $O(1)$-approximation for single-sink buy-at-bulk. Foundations of Computer Science, Annual IEEE Symposium on.

[12] Jia, L., Noubir, G., Rajaraman, R., & Sundaram, R. (2006). Gist: Groupindependent spanning tree for data aggregation in dense sensor networks. In Dcoss (pp. 282–304).

[13] Jia, L., Lin, G., Noubir, G., Rajaraman, R., & Sundaram, R. (2005). Universal approximations for tsp, steiner tree, and set cover. In Stoc '05: Proceedings of

the thirty-seventh annual acm symposium on theory of computing (pp. 386–395). New York, NY, USA: ACM.

[14] Gupta, A., Hajiaghayi, M. T., & Racke, H. (2006). Oblivious network design. In Soda '06: Proceedings of the seventeenth annual acm-siam symposium on discrete algorithm (pp. 970–979). New York, NY, USA: ACM.

PART I

OBLIVIOUS ROUTING ALGORITHMS

In contrast to the traditional Minimum-Cost Flow Problems (MCFPs) that are defined with a well-defined set of commodities (with specific size and source/destination nodes) and given flow cost function for every edge, oblivious network routing aims to solve an MCFP in which either the flow cost function is not specified (flow cost is oblivious), or the commodities size, source, and destination are not specified (commodities are oblivious).

Oblivious network routing approach solves oblivious MCFPs by employing a practical method not guaranteed to be perfect, but sufficient for the immediate goal which is obtaining an acceptable approximation of the optimal solution. In this regard, oblivious network routing is considered to be a heuristic method since it can be used to speed up the process of finding a satisfactory solution (while computing the optimal solution is impractical).

Oblivious network routing is different with Dynamic (adaptive) Routing (DR) since DR proposes a different solution in response to any change of the MCFP oblivious components; while, oblivious routing deploys a single routing scheme for an oblivious MCFP with the aim of approximating the optimal solution of the problem with oblivious parameters.

The common characteristics of the oblivious routing schemes includes being pretty flexible to the obliviousness of the environment in which the commodities are flowing and making the traffic flows distributed over the network and preventing the flow-congestion in some specific nodes or edges. Additionally, these types of routing schemes provide a low-cost flow routing in long term even if a wide range of unpredictable events occur in the network like bursty flow derived from a specific node or failure of some node in forwarding the flow through the network. In fact, the versatile routing schemes best fit to those networks that we have little/no knowledge regarding their current and future states.

CHAPTER 2

TOP-DOWN ROUTING SCHEMES IN OBLIVIOUS NETWORK DESIGN

The proliferation of network routing schemes has promoted massive network design to achieve low-latency and high-reliability with optimal cost. This chapter introduces an algorithm to construct an oblivious routing scheme to flexibly solve large-sized oblivious routing problems. More specifically, we construct an oblivious routing scheme to solve oblivious minimum-cost flow problems of arbitrary networks efficiently. This scheme is based on the top-down hierarchical routing tree mentioned in Appendix B. We also analyze the routing cost incurred by this routing scheme utilizing a quantifier called "competitiveness ratio" (see Appendix A).

2.1 Introduction

In this chapter, the oblivious routing scheme presented by Gupta, Hajiaghayi, and Racke (2006) [2] is described and analyzed thoroughly. This scheme gets benefits from the use of the top-down hierarchical decomposition tree described in Appendix B.

In all of the discussions made in this section regarding weighted connected graph $G = (V, E, w)$, we assume (w.l.o.g) that the minimum distance between any two distinct vertices of a graph is greater than one; i.e.

$$w(e) > 1 \qquad \forall e \in E \tag{2.1}$$

Additionally, the graph diameter is in the following form:

$$diam(G) = 2^h \qquad \text{for some } h \in \mathbb{N} \tag{2.2}$$

[0]Part of this chapter has been reprinted with permission from S. S. Iyengar and Kianoosh G. Boroojeni, "Oblivious Network Routing: Algorithms and Applications," MIT Press, 2015 [1].

The following lemma shows that these two assumptions don't restrict our discussion about weighted connected graphs to a special case.

Lemma 2.1.1. *For any connected graph $G = (V, E, w)$, there is a connected graph $G_c = (V, E, w_c)$ such that for every edge $e \in E$, $w_c(e) = c \cdot w(e)$ ($c \in \mathbb{R}^+$), $diam(G_c) = 2^h$ (for some $h \in \mathbb{Z}^+$), and for any two different vertices $u, v \in V$, $d_{G_c}(u, v) > 1$.*

Proof. Consider graph $G' = (V, E, w')$ where:

$$w'(e) = \frac{1+\varepsilon}{\min\limits_{e \in E}\{w(e)\}} \cdot w(e) \qquad \forall e \in E$$

such that $\varepsilon \in \mathbb{R}^+$ is some small positive number. It is easy to prove that $\forall u, v \in V$, if $u \neq v$, $d_{G'}(u, v) > 1$. Now, we define graph $G'' = (V, E, w'')$ such that:

$$w''(e) = \frac{2^{\lceil \log_2(diam(G')) \rceil}}{diam(G')} \cdot w'(e) \qquad \forall e \in E$$

You see that concerning the above definition of w'', $diam(G'')$ is 2^h, for $h = \lceil \log_2(diam(G')) \rceil$. Moreover, since

$$\frac{2^{\lceil \log_2(diam(G')) \rceil}}{diam(G')} \geq 1$$

$w''(e) \geq w'(e)$ for every $e \in E$. As the result, for the following value of c, graph $G_c = (V, E, w_c)$ satisfies the aforementioned conditions.

$$c = \frac{1+\varepsilon}{\min\limits_{e \in E}\{w(e)\}} \cdot \frac{2^{\lceil \log_2(diam(G')) \rceil}}{diam(G')}$$

□

In the remainder of this chapter, we first specify the most general case of the oblivious routing problem which can be solved using Gupta's routing scheme. Then, we introduce a special class of the hierarchical decomposition sequence known as

padded HDS. Additionally, a randomized algorithm is presented to generate a padded HDS. Finally, using the padded HDS of a graph, an oblivious routing scheme will be introduced and its competitiveness ratio will be calculated.

2.2 Oblivious Routing Problem Specification

In Appendix A, we introduced a specific type of the general routing problem in which the routing cost environment is oblivious. In other words, in such problems, instead of specifying one routing cost environment, we have a set of possible routing cost environments ($\mathbb{E}$) which makes the routing cost of the problem non-deterministic.

This chapter focuses on the oblivious routing problem of graph $G = (V, E, w)$ which satisfies Conditions 2.1 and 2.2. Moreover, the set of possible routing cost environments of the problem is in the following form:

$$\mathbb{E} = \left\{ (cost_e, \sum, \bar{K}) \middle| cost_e \in F_{\text{sub-additive}} \wedge \bar{K} \in \mathcal{K} \right\} \tag{2.3}$$

such that $F_{\text{sub-additive}}$ denotes the set of all the sub-additive[1] functions in the following form:

$$cost_e(f_1, f_2, \ldots, f_k) = w(e) \cdot rrc(f_1, f_2, \ldots, f_k) \qquad \text{where} \qquad rrc : \mathbb{R}^k_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$$

and $\mathcal{K}$ represents the set of all the possible sequences of k commodities[2] in graph G ($\forall k \in \mathbb{N}$).

[1] Assuming that A and B are two subsets of real numbers set $\mathbb{R}$, function $f : A^k \mapsto B$ is sub-additive if for every $x_1, y_1, x_2, y_2 \ldots, x_k, y_k \in A$, this is the case that: $f(x_1 + y_1, x_2 + y_2, \ldots, x_k + y_k) \leq f(x_1, x_2, \ldots, x_k) + f(y_1, y_2, \ldots, y_k)$.

[2] Regarding the definition of "commodity" in Appendix A, every commodity is a triple of source vertex, target vertex, and commodity value.

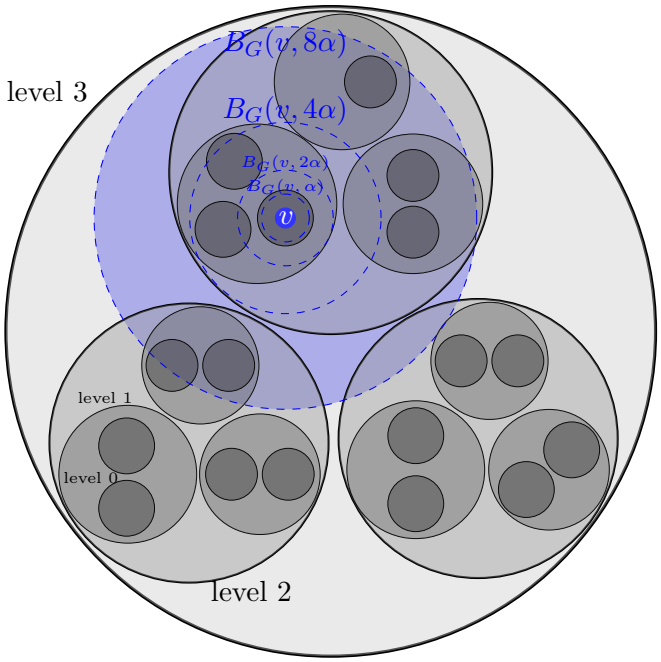


Figure 2.1: Schematic view of a padded node (v) in HDS $\bar{H} = (H_0, H_1, \ldots, H_3)$ of a weighted graph with diameter 8. Blue dashed circles denote the balls of center v and different radii. The gray circles specify the subsets which belong to partitions in different levels (darker circles belong to lower-level partitions).

2.3 Padded Hierarchical Decomposition Sequence

In Appendix B, the hierarchical decomposition sequence of a connected graph was introduced. Now, we define a special case of the HDS's in which each vertex is far away of the boundaries of its containing cluster in every level of partitioning. This padding property is important for the routing cost analysis of the oblivious routing scheme presented later in this chapter.

Definition 2.3.1. *Consider positive number $\alpha \leq 1$ and connected graph $G = (V, E, w)$ of diameter 2^h. For hierarchical decomposition sequence $\bar{H} = (H_0, H_1, \ldots, H_h)$ of graph G, vertex v is α-padded in $\bar{H}$ if:*

$$\forall i \in [0, h] : \exists C \in H_i \text{ such that } B_G(v, \alpha \cdot 2^i) \subseteq C \tag{2.4}$$

To illustrate the above definition, see Figure 2.1 which schematically depicts a padded vertex in an HDS of a graph of diameter 8 ($h = 3$). In this figure, every cluster is represented by a blue circle such that the shown parameter (D) denotes the maximum distance between the cluster members. Note that partition H_3 has only one cluster which has been represented by the largest circle. The only cluster is then partitioned into three smaller clusters which are included in H_2. The upper cluster of H_2 is also partitioned into three smaller ones which are members of H_1. Finally, the basic cluster containing vertex v has been figured (according to our assumption that the distance between any two distinct vertices is greater than one, every basic cluster contains *one and only one* vertex). We see that Proposition 2.4 is true for the shown vertex (v); henceforth, vertex v is α-padded in HDS $\bar{H}$.

Fakcharoenphol's Algorithm

Algorithm 1 RandomizedHdsGenerator

input: Connected graph $G = (V, E, w)$ of diameter 2^h
output: HDS $\bar{H} = (H_0, H_1, \ldots, H_h)$ of graph G
$\pi \leftarrow$ A uniformly random permutation on members of V
$\mathcal{U} \leftarrow \text{unif}[1/2, 1)^a$
$H_h \leftarrow \{V\}$
for $i \leftarrow h-1$ **to** 0 **do**
  $H_i \leftarrow \emptyset$
  for $C \in H_{i+1}$ **do**
    for $v \in C$ **do**
      $v.cluster \leftarrow \emptyset$
      $v.flag \leftarrow$ **true**
      $v.rep \leftarrow \pi\big(\min_j\{\pi(j) \in C \cap B_G(v, \mathcal{U} \cdot 2^{i-1})\}\big)$
    end for
    for $v \in C$ **do**
      for $u \in C$ **and** $u.flag =$ **true do**
        if $u.rep = v$ **then**
          $u.flag \leftarrow$ **false**
          $v.cluster \leftarrow v.cluster \cup \{u\}$
        end if
      end for
    end for
    for $v \in C$ **do**
      if $v.cluster \neq \emptyset$ **then**
        $H_i \leftarrow H_i \cup \{v.cluster\}$
      end if
    end for
  end for
end for

[a] Uniform random variable in interval $[1/2, 1)$

In 2003, Fakcharoenphol et al. [3] presented a randomized algorithm to generate a random HDS in which a quotient of vertices are α-padded with high probability ($\alpha \leq 1/8$). The main idea of this algorithm is to make a partition in the i^{th} level by putting some portion of the vertices of each upper-level cluster into a new smaller one such that the members of the new cluster would not be further than X_i from a special vertex called the *representative* vertex of the cluster (X_i is a uniformly

distributed random variable in interval $[2^i/4, 2^i/2)$). You can see more detailed description in Algorithm 1.

In this algorithm, a connected weighted graph is given as the input of the algorithm and a random HDS of the graph will result as the output. Additionally, every vertex v has three attributes: "*rep*" which is the *representative* vertex of v, "*cluster*" that is a set of vertices which have the single representative v, and the boolean "*flag*".

In the first two lines of this algorithm, we generate the uniformly random permutation π on vertex set V and random number $\mathcal{U}$ which is uniformly distributed over interval $[1/2, 1)$. Then, we start making the sequence of partitions by assigning $\{V\}$ to H_h. Consider the loop expanded from line 4 to 25 which makes other partitions. To obtain partition H_i from H_{i+1}, we first specify a representative for each vertex $v \in V$ in the following way: assuming $C \in H_{i+1}$ as the only H_{i+1} member which contains v, the representative of v is the first vertex in permutation π such that $v \in C \cap B_G(v, \mathcal{U} \cdot 2^{i-1})$ (see line 9). After specifying the representatives of all the vertices, attribute *cluster* of vertex v is developed as the set of vertices whose representatives are equal to v (see lines 11 to 18). Finally, partition H_i is obtained by gathering all of the nonempty developed clusters together into a set (lines 19-23).

Algorithm 1 constructs random HDS $\bar{H}$ which satisfies the following condition (for $\alpha \leq 1/8$).

$$\mathbf{Pr}\big[v \text{ is } \textit{not } \alpha\text{-padded in } \bar{H}\big] \leq O(\alpha \log(|V|)) \qquad \forall v \in V \tag{2.5}$$

This inequality guarantees that each graph vertex is α-padded with high-probability in the random HDS generated by Algorithm 1. To prove this claim, we first show that the resulting partition sequence of Algorithm 1 is an HDS; then, we prove Inequality 2.5.

Lemma 2.3.2. *Algorithm 1 constructs an HDS in the input graph.*

Proof. Regarding the third line of the algorithm, $H_h = \{V\}$. Consider partition H_i for every $i \leq h-1$. Since all the members of a cluster in H_i have the same representative vertex, for every two different vertices v_1 and v_2 in a cluster, $u = v_1.rep$ and $u = v_2.rep$. Henceforth, concerning line 9 of Algorithm 1, we obtain the following relations:

$$u \in B_G(v_1, \mathcal{U} \cdot 2^{i-1}) \rightarrow d_G(v_1, u) \leq \mathcal{U} \cdot 2^{i-1}$$

and

$$u \in B_G(v_2, \mathcal{U} \cdot 2^{i-1}) \rightarrow d_G(v_2, u) \leq \mathcal{U} \cdot 2^{i-1}$$

Subsequently, regarding the triangle inequality, this is the case that ($\mathcal{U} < 1$):

$$\begin{aligned} d_G(v_1, v_2) &\leq d_G(v_1, u) + d_G(v_2, u) \\ &\leq 2^{i-1} + 2^{i-1} \\ &\leq 2^i \end{aligned}$$

Up to here, we have shown that for every $i < h$, H_i is a 2^i-partition of the input graph. Additionally, we need to prove that for every $i < h$, assuming that cluster C belongs to H_i, there exists cluster $C' \in H_{i+1}$ such that $C \subseteq C'$. Since every iteration of the for statement expanded from line 6 to 24 uses one distinct member of H_{i+1} to construct a cluster of H_i, the mentioned condition holds for every $i < h$.

□

In order to show Inequality 2.5, first consider the following lemma.

Lemma 2.3.3. *Let $G = (V, E, w)$ denote a connected graph of diameter 2^h. For the HDS $\bar{H} = (H_0, H_1, \ldots, H_h)$ obtained by Algorithm 1 and the arbitrary vertex $v \in V$, if v is not α-padded in $\bar{H}$, there exists some $i \leq h-1$ such that:*

$$d_G(v, v.rep_i) > \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i \wedge B_G(v, \alpha \cdot 2^{i+1}) \subseteq C_v^{(i+1)}$$

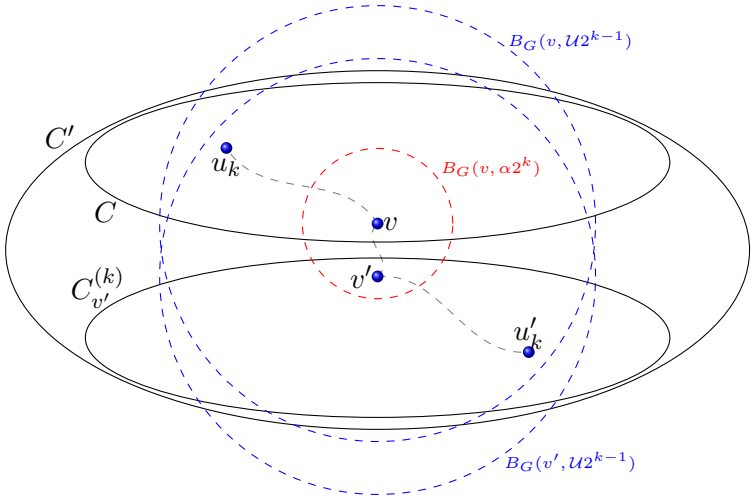
where $C_v^{(i+1)}$ *is the only* $(i+1)^{th}$*-level cluster containing* v*,* $v.rep_i$ *denotes the representative of vertex* v *at level* i *in Algorithm 1* ($\forall i < h$)*,* $\alpha \leq 1/8$*, and* $\mathcal{U}$ *is the random variable computed in line 2 of Algorithm 1.*

Proof. By contradiction, assume that if v is not an α-padded vertex in $\bar{H}$, this is the case that:

$$\forall i \leq h-1 : d_G(u_i, v) \leq \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i \vee B_G(v, \alpha \cdot 2^{i+1}) \not\subseteq C_v^{(i+1)} \tag{2.6}$$

where $u_i = v.rep_i$. Since v is not α-padded in $\bar{H}$, for some $k < h$, there exists cluster $C \in H_k$ such that $v \in C$ and $B_G(v, \alpha \cdot 2^k) \not\subseteq C$. In addition, let H_k be the highest level partition which contains such cluster, i.e.

$$\forall i \in [k+1, h] : \forall C_i \in H_i : B_G(v, \alpha \cdot 2^i) \subseteq C_i \vee B_G(v, \alpha \cdot 2^i) \cap C_i = \emptyset \tag{2.7}$$



Since $B_G(v, \alpha \cdot 2^k) \not\subseteq C$, there is vertex $v' \in B_G(v, \alpha \cdot 2^k)$ such that $v' \notin C$. Let $u'_k \in V$ denote the representative of vertex v' at level k. As far as vertex v' doesn't belong to C, u_k and u'_k are two distinct vertices of G. Assuming cluster $C' = C_v^{(k+1)} \in H_{k+1}$ as the only cluster of level $(k+1)$ that $C \subseteq C'$ (and subsequently $v \in$

C'), we will show the condition mentioned in 2.8 which contradicts the assumption that $u_k = v.rep_k$ and $u'_k = v'.rep_k$ (if u_k appears earlier than u'_k in permutation π, concerning Relation 2.8, $u_k = v'.rep_k$; otherwise, $u'_k = v.rep_k$).

$$u'_k \in C' \cap B_G(v, \mathcal{U} \cdot 2^{k-1}) \wedge u_k \in C' \cap B_G(v', \mathcal{U} \cdot 2^{k-1}) \tag{2.8}$$

Henceforth, to complete the proof, we only need to prove Relation 2.8.

By setting $i = k + 1$ in Proposition 2.7, we obtain the following relation:

$$B_G(v, \alpha \cdot 2^{k+1}) \subseteq C' \tag{2.9}$$

As the result, since $v' \in B_G(v, \alpha \cdot 2^k)$, vertex v' is also a member of set C'. Henceforth, concerning line 9 of Algorithm 1, $u'_k \in C' \cap B_G(v', \mathcal{U} \cdot 2^{k-1})$, and $u_k \in C' \cap B_G(v, \mathcal{U} \cdot 2^{k-1})$. According to Proposition 2.6 and Relation 2.9, we obtain the following inequality:

$$d_G(u_k, v) \leq \mathcal{U} \cdot 2^{k-1} - \alpha \cdot 2^k$$

Since v' is in $B_G(v, \alpha \cdot 2^k)$, this is the case that:

$$\begin{aligned} d_G(v', u_k) &\leq d_G(v', v) + d_G(v, u_k) \\ &\leq \alpha \cdot 2^k + (\mathcal{U} \cdot 2^{k-1} - \alpha \cdot 2^k) \\ &\leq \mathcal{U} \cdot 2^{k-1} \end{aligned}$$

In addition, since v and v' are not in the same k^{th}-level cluster and $v \in B_G(v', \alpha \cdot 2^k)$, $B_G(v', \alpha \cdot 2^k)$ is not a subset of the k^{th}-level cluster containing v'. This implies that v' is also not an α-padded vertex in $\bar{H}$. As the result, Proposition 2.6 is also true for v' and u'_k; i.e.

$$d_G(u'_k, v') \leq \mathcal{U} \cdot 2^{k-1} - \alpha \cdot 2^k$$

As long as vertex v' belongs to $B_G(v, \alpha \cdot 2^k)$, v also belongs to $B_G(v', \alpha \cdot 2^k)$. Subsequently,

$$\begin{aligned} d_G(v, u'_k) &\leq d_G(v, v') + d_G(v', u'_k) \\ &\leq \alpha \cdot 2^k + (\mathcal{U} \cdot 2^{k-1} - \alpha \cdot 2^k) \\ &\leq \mathcal{U} \cdot 2^{k-1} \end{aligned}$$

□

Concerning the above lemma, this is the case that:

$$\mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big] \leq \mathbf{Pr}\big[\exists i \leq h-1 : d_G(v.rep_i, v) > r_i\big]$$

Or equivalently,

$$\mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big] \leq \mathbf{Pr}\big[\exists i \leq h-1 : d_G(u, v) > \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i \wedge u = v.rep_i\big] \tag{2.10}$$

Note that in Inequality 2.10, for every $i < h$, this is the case that:

$$\begin{aligned} & d_G(v, v.rep_i) > \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i \\ \xrightarrow{(\mathcal{U} \geq \frac{1}{2})} & d_G(v, v.rep_i) > 2^{i-2} - \alpha \cdot 2^i \\ \xrightarrow{(\alpha \leq \frac{1}{8})} & d_G(v, v.rep_i) > 2^{i-2} - 2^{i-3} \\ \rightarrow & d_G(v, v.rep_i) > 0 \end{aligned}$$

which implies that $v \neq v.rep_i$ for every $i < h$.

Now, consider the following partition of $V - \{v\}$.

$$V - \{v\} = \bigcup_{j=1}^{h} (B_G(v, 2^j) - B_G(v, 2^{j-1}))$$

Using this partition, we rewrite Inequality 2.10 in the following form:

$$\begin{aligned} \mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big] \leq \mathbf{Pr}\Big[& \bigvee_{j=1}^{h} \big(u \in (B_G(v, 2^j) - B_G(v, 2^{j-1})\big) \\ & \wedge \big(\exists i \leq h-1 : d_G(u, v) > \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i \wedge u = v.rep_i\big)\Big] \end{aligned}$$

According to Algorithm 1, since $d_G(v, v.rep) \leq \mathcal{U} \cdot 2^{i-1} \leq 2^{i-1}$ $(\forall i \leq h-1)$, we obtain the following inequality ($r_i = \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i$):

$$
\begin{aligned}
&\mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big] \\
&\quad \leq \mathbf{Pr}\Big[\bigvee_{j=2}^{h-1} \big(u \in (B_G(v, 2^{j-1}) - B_G(v, 2^{j-2})\big) \\
&\quad \wedge \big(\exists i \leq h-1 : d_G(u,v) > r_i \wedge u = v.rep_i)\big)\Big] \\
&\quad \leq \sum_{j=2}^{h-1} \mathbf{Pr}\Big[u \in \big(B_G(v, 2^{j-1}) - B_G(v, 2^{j-2})\big) \\
&\quad \wedge \big(\exists i \leq h-1 : d_G(u,v) > r_i \wedge u = v.rep_i\big)\Big] \\
&\quad \leq \sum_{j=2}^{h-1} \mathbf{Pr}\Big[\exists i \leq h-1 : d_G(u,v) > r_i \wedge u = v.rep_i \;\Big|\; u \in B_G(v, 2^{j-1}) - B_G(v, 2^{j-2})\Big]
\end{aligned}
$$

In the above inequalities, we will find a relation between integers i and j. As $u = v.rep_i$, u is a member of $B_G(v, \mathcal{U} \cdot 2^{i-1})$. This implies that:

$$
\begin{aligned}
& \mathcal{U} \cdot 2^{i-1} - \alpha \cdot 2^i < d_G(u,v) \leq \mathcal{U} \cdot 2^{i-1} \\
\xrightarrow{1/2 \leq \mathcal{U} < 1} \quad & (1-4\alpha)2^{i-2} < d_G(u,v) < 2^{i-1}
\end{aligned}
$$

Moreover, since u is also in $B_G(v, 2^{j-1}) - B_G(v, 2^{j-2})$, we obtain the following relation between i and j.

$$
\begin{aligned}
2^{j-2} < d_G(u,v) \leq 2^{j-1} \rightarrow & \begin{cases} (1-4\alpha)2^{i-2} < 2^{j-1} & \text{if } i \geq j \\ 2^{j-2} < 2^{i-1} & \text{otherwise} \end{cases} \\
\rightarrow & \begin{cases} i < j + \log_2(\frac{2}{1-4\alpha}) & \text{if } i \geq j \\ i > j - 1 & \text{otherwise} \end{cases}
\end{aligned}
$$

Regarding the assumption that $\alpha \leq 1/8$, $j \leq i \leq j+2$. Subsequently, we obtain the following upper-bound for $\mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big]$:

$$
\begin{aligned}
&\mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big] \\
&\leq \sum_{j=2}^{h-1} \mathbf{Pr}\Big[\bigvee_{\substack{i<h \\ j\leq i\leq j+2}} \big(d_G(u,v) > \mathcal{U}\cdot 2^{i-1} - \alpha\cdot 2^i \wedge u = v.rep_i\big) \;\Big|\; \\
&u \in B_G(v,2^{j-1}) - B_G(v,2^{j-2})\Big] \\
&\leq \sum_{j=2}^{h-1} \sum_{\substack{i<h \\ j\leq i\leq j+2}} \mathbf{Pr}\Big[d_G(u,v) > \mathcal{U}\cdot 2^{i-1} - \alpha\cdot 2^i \wedge u = v.rep_i \;\Big|\; \\
&u \in B_G(v,2^{j-1}) - B_G(v,2^{j-2})\Big]
\end{aligned}
$$

Assuming C_v^{i+1} as the only cluster in H_{i+1} that contains v, the value of the following conditional probability

$$
\mathbf{Pr}\Big[d_G(u,v) > \mathcal{U}\cdot 2^{i-1} - \alpha\cdot 2^i \wedge u = v.rep_i \;\Big|\; u \in B_G(v,2^{j-1}) - B_G(v,2^{j-2})\Big]
$$

is equal to the probability of the following event considering the assumption that $u \in B_G(v,2^{j-1}) - B_G(v,2^{j-2})$ (see Algorithm 1).

vertex u is the first vertex (concerning permutation π) which belongs to set $B_G(v,\mathcal{U}\cdot 2^{i-1}) \cap C_v^{i+1}$ and also satisfies the following equation:

$$
\mathcal{U}\cdot 2^{i-1} - \alpha\cdot 2^i < d_G(u,v) \leq \mathcal{U}\cdot 2^{i-1} \tag{2.11}
$$

Note that regarding Lemma 2.3.3, $B_G(v,2^{i-2}) \subseteq C_v^{i+1}$ which implies that set $B_G(v,\mathcal{U}\cdot 2^{i-1}) \cap C_v^{i+1}$ equals $B_G(v,\mathcal{U}\cdot 2^{i-1})$.

Inequality 2.11 implies that:

$$
\frac{d_G(u,v)}{2^{i-1}} \leq \mathcal{U} < \frac{d_G(u,v)}{2^{i-1}} + 2\alpha
$$

Additionally, as $\mathcal{U}$ is uniformly distributed over interval $[1/2,1)$, this is the case that:

$$
\mathbf{Pr}\big[\mathcal{U} = x\big] \leq 2\mathrm{d}x \qquad\qquad \forall x \in \mathbb{R}
$$

Moreover, the probability that a vertex appears earlier than n other vertices in the uniformly random permutation π is equal to $1/(n+1)$. Consequently,

$$\begin{aligned}
&\mathbf{Pr}\Big[d_G(u,v) > \mathcal{U}\cdot 2^{i-1} - \alpha\cdot 2^i \wedge u = v.rep_i \;\Big|\; u \in B_G(v,2^{j-1}) - B_G(v,2^{j-2})\Big] \\
&\qquad = \int_{\frac{d_G(u,v)}{2^{i-1}}}^{\frac{d_G(u,v)}{2^{i-1}}+2\alpha} \frac{1}{|B_G(v, x\cdot 2^{i-1}) \cap C_v^{i+1}|} \cdot \mathbf{Pr}\big[\mathcal{U} = x\big] \\
&\qquad = \int_{\frac{d_G(u,v)}{2^{i-1}}}^{\frac{d_G(u,v)}{2^{i-1}}+2\alpha} \frac{1}{|B_G(v, x\cdot 2^{i-1})|} \cdot \mathbf{Pr}[\mathcal{U} = x] \\
&\qquad \le \int_{\frac{d_G(u,v)}{2^{i-1}}}^{\frac{d_G(u,v)}{2^{i-1}}+2\alpha} \frac{1}{|B_G(v, x\cdot 2^{i-1})|} \cdot 2\mathrm{d}x \\
&\qquad \le \frac{1}{|B_G(v, \frac{d_G(u,v)}{2^{i-1}}\cdot 2^{i-1})|} \int_{\frac{d_G(u,v)}{2^{i-1}}}^{\frac{d_G(u,v)}{2^{i-1}}+2\alpha} 2\mathrm{d}x \\
&\qquad = \frac{4\alpha}{|B_G(v, d_G(u,v))|} \\
&\qquad \le \frac{4\alpha}{|B_G(v, 2^{j-2})|}
\end{aligned}$$

The recent inequality leads to the following inequalities:

$$\begin{aligned}
\mathbf{Pr}\big[v \text{ is not } \alpha\text{-padded in } \bar{H}\big] &\le \sum_{j=2}^{h-1} \sum_{\substack{i\le h \\ j\le i\le j+2}} \frac{4\alpha}{|B_G(v,2^{j-2})|} \\
&\le \sum_{j=2}^{h-1} \frac{12\alpha}{|B_G(v,2^{j-2})|} \\
&\le 12\alpha \sum_{n=1}^{|V|} \frac{1}{n} \\
&\le 12\alpha \int_1^{|V|+1} \frac{\mathrm{d}x}{x} \\
&\le (12\ln 2)\cdot \alpha \log(|V|+1)
\end{aligned}$$

Or equivalently, we obtain Inequality 2.5 for every $v \in V$.

2.4 The Oblivious Routing Scheme Construction

Now, we are ready to address how to construct an oblivious routing scheme for the aforementioned routing problem. First, we present some preliminary definitions.

Assume that p_1 denote a walk from v_1 to v_2 and p_2 denote a walk from v_2 to v_3 in graph $G = (V, E, w)$. The merge operator $\oplus$ is defined on p_1 and p_2 and results in another walk represented by $p_1 \oplus p_2$ in graph G. Walk $p_1 \oplus p_2$ is a walk in G and obtained by moving from v_1 to v_2 using p_1 and continuing the way to v_3 on walk p_2.

Definition 2.4.1. *Consider the HDS $\bar{H}$ of graph $G = (V, E, w)$ and its corresponding HDT $T = (V_T, E_T)$. The representative of tree vertex (C, i) is defined as an arbitrary graph vertex belonging to cluster C $(\forall (C, i) \in V_T)$.*

According to Definition 2.4.1, every tree vertex of an HDT has some representative in the associated graph. More specifically, for each leaf of an HDT in the form $(C, 0)$, its representative is defined as the only member of basic cluster C (note that since graph G satisfies Condition 2.1, each basic cluster contains only one vertex.)

Definition 2.4.2. *Consider the HDT T of graph $G = (V, E, w)$ and its corresponding HDS $\bar{H} = (H_1, H_2, \ldots, H_k)$. For every path p of tree T, the projection of path p on graph G is defined as the following path in graph G:*

$$projection(p) = \bigoplus_{\{u,v\} \in p} SP_G(u', v')$$

such that $SP_G(u', v')$ represents the shortest path between vertices u' and v' in graph G; and also, $u', v' \in V$ denote the representatives of tree vertices u and v respectively.

Here is the algorithm which computes the projection of any path of an HDT on its corresponding graph (note that in this algorithm, for every vertex v of HDT T, $v.rep$ specifies its representative vertex in the associated graph).

Algorithm 2 PROJECTION

input: Path p which is between two leaves of HDT T & the graph in which tree T is defined
output: The projection of tree path p on graph G
$p_G \leftarrow \emptyset$
for each tree edge $\{u, v\}$ **in** p **do**
  $p' \leftarrow$ the shortest path in G from vertex $u.rep$ to vertex $v.rep$
  $p_G \leftarrow p_G \oplus p'$
end for
return p_G

Fractional Scheme

The manner in which we construct a fractional oblivious routing scheme for the aforementioned problem is as follows: first, we use Algorithm 1 to generate $c \cdots \log |V|$ HDS's for the given connected graph $G = (V, E, w)$ (the value of c is discussed later). Then, for every generated HDS, we compute its corresponding HDT. For every pair of vertices $u, v \in V$, we mark $\log |V|$ HDTs (among $c \cdot \log |V|$ generated HDTs) in which u and v are both α-padded (for some $\alpha \leq \alpha_0 = \Omega(\frac{1}{\log |V|})$). Assuming that sequence $T_1, T_2, \ldots, T_{\log |V|}$ represents the marked HDTs, the suggested fractional flow between any pair of vertices u and v is obtained by the following equation:

$$\mathbb{S}(s,t) = \big\{(q_i, \frac{1}{\log |V|}) \big| q_i = projection(p_{s,t}^{(i)}) \wedge i \in [1, \log |V|]\big\}$$

where $p_{s,t}^{(i)}$ denotes the path between leaves $(\{s\}, 0)$ and $(\{t\}, 0)$ in HDT T_i (for every $i \in [1, \log |V|]$). Note that in the above equation, we equally divide the flow into $\log |V|$ paths between s and t.

Algorithm 3 describes how to generate a fractional routing scheme in detail. Note that since the input graph of the algorithm satisfies Condition 2.1, every leaf of an HDT tree of the graph is in form $(C, 0)$ such that $|C| = 1$.

Additionally, in Algorithm 3, it has been implicitly assumed that for every two graph vertices, there are $\log|V|$ HDS's in which the both of vertices are α-padded. We will address this claim thoroughly at the rest of this section.

Previously, we introduced a randomized algorithm (originally presented by Fakcharoenphol et al.) which receives a weighted connected graph as input and generates a random HDS in which any graph vertex is α-padded with high-probability ($\alpha \leq 1/8$). Now, consider the following theorem regarding this algorithm:

Theorem 2.4.3. *Let $G = (V, E, w)$ denote a connected graph. Additionally, assume that running Algorithm 1 for $n = c \cdot \log|V|$ times outputs these HDS's: $\bar{H}^{(1)}, \bar{H}^{(2)}, \ldots, \bar{H}^{(n)}$ ($c \geq 2$). There is real number $\alpha_0 = \Omega(\frac{1}{\log|V|})$ such that:*

For every pair of vertices $u, v \in V$, with the following probability, there exists at-least $\log|V|$ HDS's (among the n HDS's) in which u and v are both α-padded for every $\alpha \leq \min\{\alpha_0, 1/8\}$.

$$\boldsymbol{Pr}\big[\textit{there exist at-least } \log|V| \textit{ HDS's in which } u \,\&\, v \textit{ are } \alpha\textit{-padded}\big] \geq 1 - e^{-\frac{(c-2)^2}{2c}\log|V|}$$

Algorithm 3 Top-DownRoutingSchemeGenerator (fractional)

input: Connected graph $G = (V, E, w)$ of diameter 2^h
output: Oblivious routing scheme $\mathbb{S}$
for $i \leftarrow 1$ **to** $c \log |V|$ **do**
  $\bar{H}^{(i)} \leftarrow$ RandomizedHdsGenerator(G)
  $T^{(i)} \leftarrow$ the HDT corresponding to $\bar{H}^{(i)}$
end for
$n \leftarrow \log |V|$
for s **in** V **do**
  for t **in** $V - \{s\}$ **do**
    for $i \leftarrow 1$ **to** $c \cdot \log |V|$ **do**
      if $n = 0$ **then**
        break
      end if
      if s and t are α-padded in $\bar{H}^{(i)}$ **then**
        mark tree $T^{(i)}$ as a "padding tree"
        $n \leftarrow n - 1$
      end if
    end for
    $\mathbb{S}(s, t) \leftarrow \emptyset$
    for each padding tree T **do**
      $p \leftarrow$ the only path existed between $(\{s\}, 0)$ and $(\{t\}, 0)$ in tree T
      $p_G \leftarrow$ Projection(p, G)
      $\mathbb{S}(s, t) \leftarrow \mathbb{S}(s, t) \oplus (p_G, \frac{1}{\log |V|})$ Unmark T
    end for
  end for
end for
return $\mathbb{S}$

Proof. Let U_i and V_i respectively denote the following events ($\forall i \in [1, n]$):

$$U_i : \text{ "Vertex } u \text{ is } \alpha\text{-padded in } \bar{H}^{(i)}\text{"}$$

$$V_i : \text{ "Vertex } v \text{ is } \alpha\text{-padded in } \bar{H}^{(i)}\text{"}$$

Concerning Inequality 2.5, there exists some $\alpha_0 = \Omega(\frac{1}{\log |V|})$ such that for every $\alpha \leq \min\{\alpha_0, 1/8\}$ this is the case that:

$$\mathbf{Pr}\left[\overline{U_i}\right] \leq p \qquad \forall i \in [1, n]$$

and $\mathbf{Pr}\big[\overline{V_i}\big] \leq p$ for every $i = 1, 2, \ldots, n$ such that p is a real number in interval $(0, 1/2)$. This implies that:

$$\begin{aligned}
\mathbf{Pr}\big[U_i \wedge V_i\big] &= \mathbf{Pr}\big[\overline{\overline{U_i} \vee \overline{V_i}}\big] \\
&= 1 - \mathbf{Pr}\big[\overline{U_i} \vee \overline{V_i}\big] \\
&\geq 1 - \mathbf{Pr}\big[\overline{U_i}\big] - \mathbf{Pr}\big[\overline{V_i}\big] \\
&\geq 1 - 2p
\end{aligned}$$

Henceforth, assuming that $\mathbf{Pr}\big[U_i \wedge V_i\big] = q$, we obtain the following inequality:

$$q \geq 1 - 2p \tag{2.12}$$

Now, considering $(U_1 \wedge V_1), (U_2 \wedge V_2), \ldots, (U_n \wedge V_n)$ as a sequence of n independent results of a Bernoulli trial and X as a random variable which counts the number of *true* results, this is the case that $X \sim B^3(n, q)$. As the result,

$$\begin{aligned}
\mathbf{Pr}\big[&\text{there exist at-least } \log|V| \text{ HDS's in which} \\
&u \;\&\; v \text{ are } \alpha\text{-padded}\big] \geq 1 - F_X(\log|V| - 1)
\end{aligned}$$

such that F_X is the cumulative distribution function of variable X. Here, we use Inequality 2.12 and the Hoeffding's inequality to find an upper-bound for the value of $F_X(\log|V| - 1)$:

$$\begin{aligned}
F_X(\log|V| - 1) &\leq F_X(\log|V|) \\
&\leq e^{-2\frac{(qc-1)^2}{c}\log|V|} \\
&\leq e^{-2\frac{(c-2pc-1)^2}{c}\log|V|}
\end{aligned}$$

Consequently, by letting $p = 1/4$, we obtain the following inequality:

$$\begin{aligned}
\mathbf{Pr}\big[&\text{there exist at-least } \log|V| \text{ HDS's in which} \\
&u \;\&\; v \text{ are } \alpha\text{-padded}\big] \geq 1 - e^{-\frac{(c-2)^2}{2c}\log|V|}
\end{aligned}$$

[3]Binomial distribution with parameters n and q.

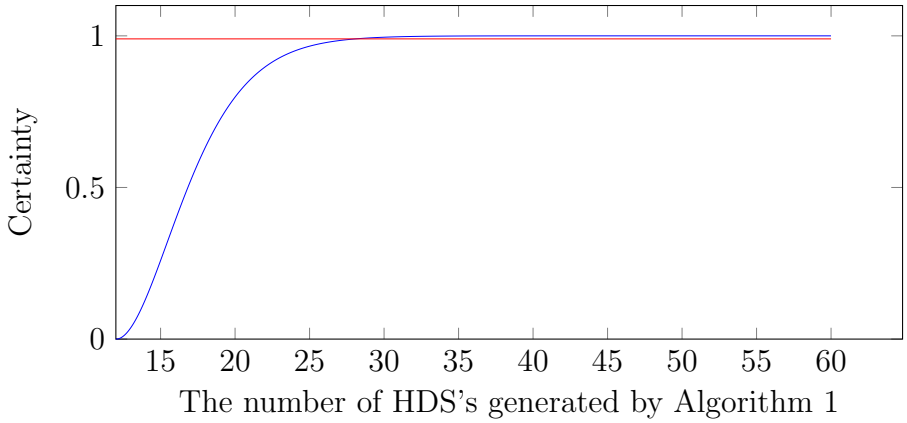


Figure 2.2: The plot of the probability low-threshold (see Inequality 2.13). In this plot, we assume that $\log|V| = 6$; i.e. $n = 6c$. As you see, if we generate more than 30 HDS's using Algorithm 1, any pair of vertices will be α-padded in 6 of them with certainty of more than 99%.

Or equivalently ($n = c\log|V|$),

$$\mathbf{Pr}\big[\text{there exist at-least } \log|V| \text{ HDS's in which } u \,\&\, v \text{ are } \alpha\text{-padded}\big] \geq 1 - e^{-\frac{(n-2\log|V|)^2}{2n}} \tag{2.13}$$

□

In Theorem 2.4.3, by increasing the value of c, we can reach to larger probability low-threshold; i.e. by generating more HDS's ($n = c\log|V|$), the vertices are α-padded with more certainty. Figure 2.2 shows the scatter plot of the low-threshold of padding probability versus the number of HDS's n.

Integral Scheme

Now, we make a similar algorithm to generate an integral versatile solution for the aforementioned oblivious routing problem. As you see in Algorithm 4, at first, we make n HDS's using Algorithm 1; then, we compute the associated n HDTs of the generated HDS's. For every pair of vertices s and t, we will find an HDT in which

the vertices are both α-padded (similar to the fractional version, we can make the value of n large enough to make sure that there is such α-padding HDT). Finally, the suggested path between vertices s and t in the input graph is the projection of the only path between leaves $(\{s\}, 0)$ and $(\{t\}, 0)$ in the α-padding HDT.

Algorithm 4 Top-DownRoutingSchemeGenerator (integral)

input: Connected graph $G = (V, E, w)$ of diameter 2^h
output: Oblivious routing scheme $\mathbb{S}$
for $i \leftarrow 1$ **to** n **do**
  $\bar{H}^{(i)} \leftarrow$ RandomizedHdsGenerator(G)
  $T^{(i)} \leftarrow$ the HDT corresponding to $\bar{H}^{(i)}$
end for
for s **in** V **do**
  for t **in** $V - \{s\}$ **do**
    for $i \leftarrow 1$ **to** $c \cdot \log |V|$ **do**
      if s and t are α-padded in $\bar{H}^{(i)}$ **then**
        $T \leftarrow T^{(i)}$
        break
      end if
    end for
    $p \leftarrow$ the only path existed between $(\{s\}, 0)$ and $(\{t\}, 0)$ in tree T
    $\mathbb{S}(s, t) \leftarrow$ Projection(p, G)
  end for
end for
return $\mathbb{S}$

Figure 2.3 represents a path suggested by Algorithm 4 through a weighted grid graph $G_{5\times5} = (V, E, w)$ such that $w(e) = 2$ for every $e \in E$. As it is shown, an HDT of the graph has been depicted schematically. In this figure, each tree vertex is labeled by the set of vertices belonging to its corresponding cluster. The underlined numbers specify the representatives of each tree vertex in the grid graph. Additionally, the way of projecting a tree path on the graph has been specified.

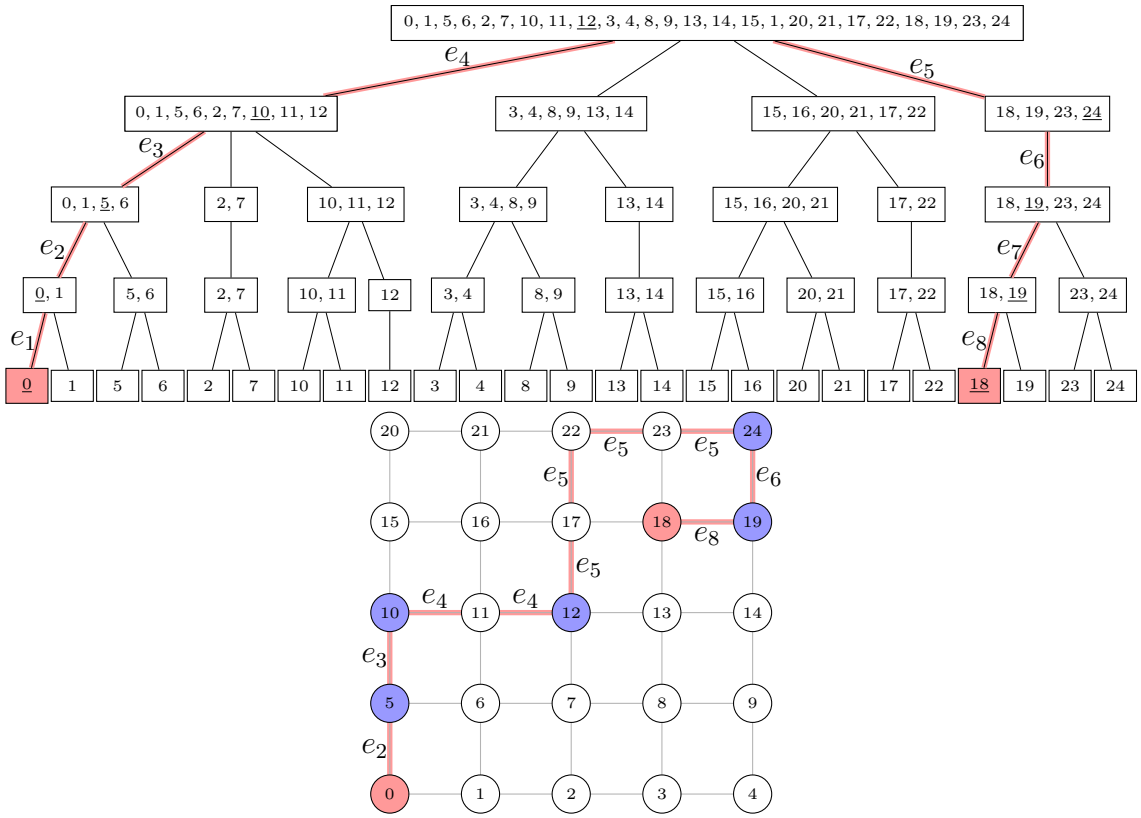


Figure 2.3: An example of using Algorithm 4 for routing a commodity of source 0 and target 18 through the shown grid graph ($G_{5\times5}$). Note that weight of each edge in the grid is assumed to be 2.

2.5 Routing Cost Analysis

As mentioned in Appendix A, in order to analyze the cost efficiency of a versatile routing scheme, we use the competitiveness ratio which is defined as the the maximum cost incurred by the oblivious routing scheme divided by the routing cost of the optimal solution. We first find a lower-bound for the cost of the optimal solution (whether it is fractional or integral). Then, we compute a high-threshold for the routing cost of the solution suggested by the schemes described previously. Finally, we find the competitiveness ratios of both the fractional and integral routing schemes.

Let's introduce the concepts of α-padding cover and cutting a commodity in a weighted graph.

Definition 2.5.1. *Let $G = (V, E, w)$ denote a connected graph.*

Assume that $\mathcal{H}$ represents a set of $\log|V|$ HDS's of graph G and $\mathcal{E}$ denotes a routing cost environment in graph G. Set $\mathcal{H}$ is called an α-padding cover in environment $\mathcal{E}$ if this is the case that:

$\forall i \in [1, k] : \exists \bar{H} \in \mathcal{H}$ *such that vertices $\Pi_1(K)$ and $\Pi_2(K)$ are α-padded in HDS $\bar{H}$*

Furthermore, HDS $\bar{H}$ cuts commodity K in the i^{th} level if ($i \leq h - 1$):

$$\exists C \in \Pi_i\big(\bar{H}\big) : \Pi_1\big(K\big) \in C \wedge \Pi_2\big(K\big) \notin C$$

Additionally, a *closed cut set* of a commodity is defined in the following way:

Definition 2.5.2. *Consider a general routing problem of weighted graph $G = (V, E, w)$ and routing cost environment $\mathcal{E}$. Set $X \subseteq E$ is called a closed cut set of the i^{th} commodity in environment $\mathcal{E}$, if:*

(i) there exists an integral solution solution like $\bar{p}$ such that:

$$\sum_{e \in X} f_{\bar{p}}^{(i)}(e) = \Pi_3(K_i)$$

(ii) and there is no path between vertices $\Pi_1(K_i)$ and $\Pi_2(K_i)$ in graph $G^X = (V, E - X)$.

where $(K_1, K_2, \ldots, K_k)$ denotes the sequence of commodities in environment $\mathcal{E}$.

Now, consider the following theorem which presents a lower-bound for the solution cost of the mentioned routing problem.

Theorem 2.5.3. *Consider the oblivious routing problem of graph $G = (V, E, w)$ of diameter 2^h and set of possible routing cost environments $\mathbb{E}$ which was defined in Equation 2.3. If function S denotes an arbitrary solution of the problem (either integral or fractional) and set $\mathcal{H}_{\mathcal{E}}$ represents an α-padding cover in environment $\mathcal{E}$ (for every $\mathcal{E} \in \mathbb{E}$), this is the case that:*

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) \geq \Theta\Big(\frac{\alpha}{4|\mathcal{H}_{\mathcal{E}}|}\sum_{i=0}^{h-1}\sum_{\bar{H}\in\mathcal{H}_{\mathcal{E}}} 2^i \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big)\Big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

where $\alpha = \min\{1/4, \alpha_0\}$ for some $\alpha_0 = \Omega(\log|V|)$ and $b_{n,i}(\bar{H})$ is defined in the following way:

$$b_{n,i}(\bar{H}) = \begin{cases} \Pi_3(K_n) & \text{if } \Pi_1(K_n) \text{ and } \Pi_2(K_n) \text{ are } \alpha\text{-padded in } \bar{H} \\ & \text{and } \bar{H} \text{ cuts } K_n \text{ in the } i^{th} \text{ level} \\ 0 & \text{otherwise} \end{cases}$$

and $(K_1, K_2, \ldots, K_k)$ denotes the sequence of commodities in environment $\mathcal{E}$.

Proof. Since S is the solution function of the oblivious routing problem, the network routing cost is computed by the following equation:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) = nrc_{\pi}\big(cost_{\pi_1}(f_{11}, f_{12}, \ldots, f_{1k}), cost_{\pi_2}(f_{21}, f_{22}, \ldots, f_{2k}), \ldots \\ , cost_{\pi_{|E|}}(f_{|E|1}, f_{|E|2}, \ldots, f_{|E|k})\big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

where $f_{i,j}$ denotes the traffic flow value of some specific commodity in the i^{th} edge (concerning permutation π):

$$f_{i,j} = f^{(j)}_{S(\mathcal{E})}(\pi_i) \qquad \forall i \in [1, |E|], j \in [1, k]$$

Concerning Equation 2.3, we can simplify the cost function in the following way:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) = \sum_{e\in E} w(e) \cdot rrc\big(f^{(1)}_{S(\mathcal{E})}(e), f^{(2)}_{S(\mathcal{E})}(e), \ldots, f^{(k)}_{S(\mathcal{E})}(e)\big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

Additionally, since $\mathcal{H}_{\mathcal{E}}$ denotes an α-padding cover in environment $\mathcal{E} \in \mathbb{E}$, this is the case that:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) = \frac{1}{|\mathcal{H}_{\mathcal{E}}|} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{e \in E} w(e) \cdot rrc\big(f^{(1)}_{S(\mathcal{E})}(e), f^{(2)}_{S(\mathcal{E})}(e), \ldots, f^{(k)}_{S(\mathcal{E})}(e)\big)$$
$$\geq \frac{1}{|\mathcal{H}_{\mathcal{E}}|} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{e \in E} w(e) \cdot rrc\big(g_1(e, \bar{H}), g_2(e, \bar{H}), \ldots, g_k(e, \bar{H})\big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

where:

$$g_n(e, \bar{H}) = \begin{cases} f^{(n)}_{S(\mathcal{E})}(e) & \text{if } \Pi_1(K_n) \ \& \ \Pi_2(K_n) \text{ are } \alpha\text{-padded in } \bar{H} \\ 0 & \text{otherwise} \end{cases} \qquad \forall n \in [1, k] \tag{2.14}$$

and $(K_1, K_2, \ldots, K_k)$ represents the sequence of commodities in environment $\mathcal{E}$.

Now, we consider the following subset of edge set E for every $\bar{H} \in \mathcal{H}$:

$$\bigcup_{i=0}^{h-1} \bigcup_{C \in \Pi_i(\bar{H})} E_{i,C} \subseteq E \tag{2.15}$$

such that $E_{i,C} = \emptyset$ if C doesn't contain a commodity terminal (source and target) or C contains both terminals of a single commodity. On the other hand, if there exists some commodity terminal in C and $\bar{H}$ cuts it in the i^{th} level, $E_{i,C}$ is obtained by the following equation:

$$E_{i,C} = \big\{\{u, v\} \in E \big| \{u, v\} \subseteq C \wedge \max\{m_u, m_v\} \in (\alpha \cdot 2^{i-1}, \alpha \cdot 2^i]\big\}$$

where m_u denotes the minimum graph distance between u and any commodity terminal which belongs to C (similar definition for m_v). See Figure 2.4 for illustration.

According to Relation 2.15, we obtain the following lower-bound for the solution routing cost:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) \geq \frac{1}{|\mathcal{H}_{\mathcal{E}}|} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{i=0}^{h-1} \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in E_{i,C}} w(e) \times$$
$$rrc\big(g_1(e, \bar{H}), g_2(e, \bar{H}), \ldots, g_k(e, \bar{H})\big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

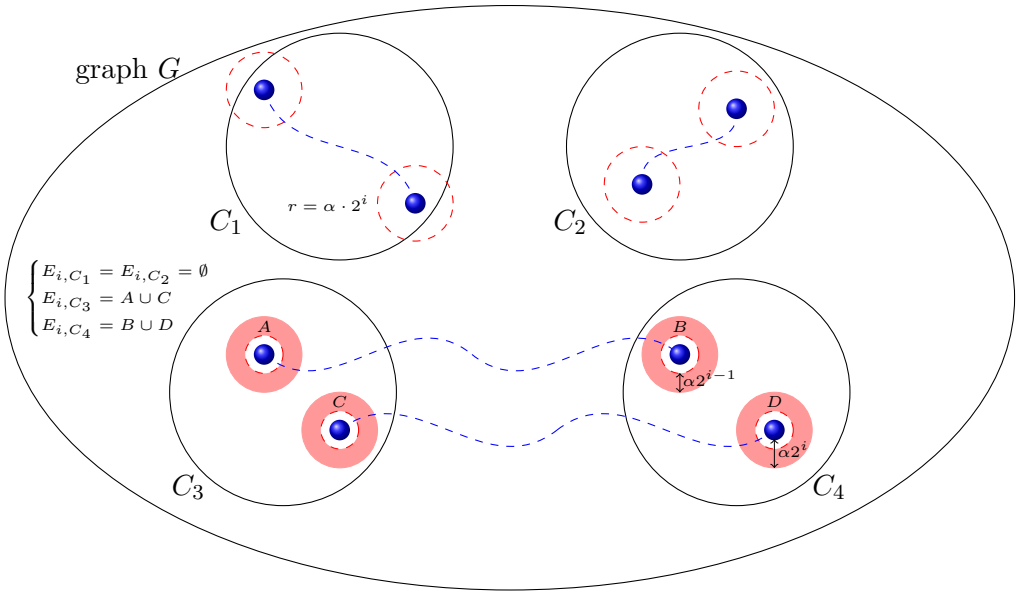


Figure 2.4: A schematic view of graph G and an HDS at level i. As you see, cluster C_1 doesn't contain any α-padded commodity terminal ($E_{i,C_1} = \emptyset$). Additionally, the pair of α-padded vertices in C_2 are terminals of one commodity (subsequently, $E_{i,C_2} = \emptyset$). However, there are α-padded terminals in clusters C_3 and C_4 such that the shown HDS has cut their corresponding commodities in the i^{th} level.

Now, we construct graph $G' = (V', E', w')$ using graph $G = (V, E, w)$ in the following algorithm:

"At first, assign sets $\emptyset$ and V to E' and V' respectively. Then, for every edge $e \in E$, add the following $(\lfloor w(e) \rfloor - 1)$ vertices to set V': $x_{1e}, x_{2e}, \ldots, x_{(\lfloor w(e) \rfloor - 1)e}$. Additionally, add the following $\lfloor w(e) \rfloor$ edges to set E':

$$\{u, x_{1e}\}, \{x_{1e}, x_{2e}\}, \ldots, \{x_{(\lfloor w(e) \rfloor - 1)e}, v\} \tag{2.16}$$

where $e = \{u, v\}$. For every $e' \in E'$, assign $w'(e') = 1$."

Note that in this algorithm, the maximum error of distance function $d_{G'}$ for any pair of adjacent vertices in V is 1 (in comparison with function d_G). We can scale up the weight function w so that this error becomes zero; i.e.

$$d_G(u, v) = d_{G'}(u, v) \qquad \forall u, v \in V$$

Every solution of the oblivious routing problem in graph G can be mapped to G' by mapping the flow of any edge $e = \{u, v\}$ of graph G to its corresponding sequence of edges (mentioned in 2.16). In the exercises, you will be asked to prove that the routing cost of the mapped solution in G' is lower than its associated cost in G. As the result:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) \geq \frac{1}{|\mathcal{H}_{\mathcal{E}}|} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{i=0}^{h-1} \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in E'_{i,C}} rrc\big(g_1(e, \bar{H}), g_2(e, \bar{H}), \ldots, g_k(e, \bar{H})\big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

such that $E'_{i,C} \subseteq E'$ and is defined the same as $E_{i,C}$. Consider the following claim regarding set $E'_{i,C}$:

Claim 2.5.4. *For every environment $\mathcal{E} \in \mathbb{E}$, HDS $\bar{H} \in \mathcal{H}_{\mathcal{E}}$, level $i \leq h-1$, and cluster $C \in \Pi_i(\bar{H})$, the following proposition holds:*
assuming K as a commodity of environment $\mathcal{E}$ and set $E'_{i,C} \neq \emptyset$, if K has a terminal in C and is cut by HDS $\bar{H}$ in the i^{th} level, there will exist $\lfloor \alpha \cdot 2^i \rfloor - \lfloor \alpha \cdot 2^{i-1} \rfloor$ closed cut sets of K in graph G' such that they are disjoint subsets of $E'_{i,C}$.

In exercises, you will be asked to prove Claim 2.5.4. If $T_{i,C}$ denotes one of the mentioned closed cut sets, we obtain this inequality:

$$\begin{aligned} C_{\mathcal{E}}\big(S(\mathcal{E})\big) &\geq \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{i=0}^{h-1} \sum_{C \in \Pi_i(\bar{H})} \frac{\alpha \cdot 2^{i-1} - 1}{|\mathcal{H}_{\mathcal{E}}|} \sum_{e \in T_{i,C}} rrc\big(g_1(e, \bar{H}), g_2(e, \bar{H}), \ldots, g_k(e, \bar{H})\big) \\ &\geq \Theta\Big(\sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{i=0}^{h-1} \frac{\alpha \cdot 2^{i-1}}{|\mathcal{H}_{\mathcal{E}}|} \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in T_{i,C}} rrc\big(g_1(e, \bar{H}), g_2(e, \bar{H}), \ldots, g_k(e, \bar{H})\big)\Big) \end{aligned}$$

In addition, regarding the assumption that function rrc is sub-additive, we conclude the following inequality:

$$\begin{aligned} C_{\mathcal{E}}\big(S(\mathcal{E})\big) \geq \Theta\Big(\frac{1}{|\mathcal{H}_{\mathcal{E}}|} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} \sum_{i=0}^{h-1} \alpha 2^{i-1} rrc\big(& \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in T_{i,C}} g_1(e, \bar{H}), \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in T_{i,C}} g_2(e, \bar{H}) \\ & \ldots, \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in T_{i,C}} g_k(e, \bar{H})\big)\Big) \end{aligned}$$

Since $T_{i,C}$ makes a cut set for every padded commodity terminal inside C, Equation 2.17 is true for every $i = 0, 1, \ldots, h-1$.

$$b_{n,i}(\bar{H}) = \frac{1}{2} \sum_{C \in \Pi_i(\bar{H})} \sum_{e \in T_{i,C}} g_n(e, \bar{H}) \qquad \forall \bar{H} \in \mathcal{H}_{\mathcal{E}}, \forall n \in [1, k] \tag{2.17}$$

Note that the RHS of the above equation has been divided by two because of the fact that each commodity has two terminals. Finally, Equation 2.17 implies that:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) \geq \Theta\Big(\frac{\alpha}{4|\mathcal{H}_{\mathcal{E}}|} \sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^i \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big)\Big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

□

Assume that we focus on the oblivious routing problem of the following set of possible routing cost environments:

$$\mathbb{E}' = \big\{(cost_e, \sum, \bar{K}) \in \mathbb{E} \big| cost_e \in F_{\text{commodity-ind}}\big\} \tag{2.18}$$

where set $\mathbb{E}$ is defined in Equation 2.3, and $F_{\text{commodity-ind}}$ denotes the set of all the functions which hold the following condition:

$$cost_e(F) = cost_e(F_1) + cost_e(F_2) \qquad \forall e \in E$$

such that F, F_1, and F_2 are sequences of k real positive numbers such that:

$$\Pi_i(F_1) = \begin{cases} \Pi_i(F) & \text{if } \Pi_i(F_2) = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in [1, k]$$

If an edge routing cost function has this property, it is called to be *commodity-independent*; i.e. the routing cost of any commodity doesn't affect the amount of cost incurred by other commodities. There are many real-world examples which have commodity-independent edge routing cost functions.

By restricting the set of possible routing cost environments to $\mathbb{E}'$, regarding the proof of Theorem 2.5.3, we obtain the following equation for every $\mathcal{E} \in \mathbb{E}'$

$$rrc\big(f^{(1)}_{S(\mathcal{E})}(e), f^{(2)}_{S(\mathcal{E})}(e), \dots, f^{(k)}_{S(\mathcal{E})}(e)\big) = \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} rrc\big(g_1(e, \bar{H}), g_2(e, \bar{H}), \dots, g_k(e, \bar{H})\big)$$

such that e is an arbitrary graph edge and $g_n(e, \bar{H})$ is the function defined in Equation 2.14. With some discussion similar to what made in the proof of Theorem 2.5.3, we obtain the following lower-bound for the network routing cost of an arbitrary solution to the oblivious routing problem of set $\mathbb{E}'$ of possible routing cost environments:

$$C_{\mathcal{E}}\big(S(\mathcal{E})\big) \geq \Theta\Big(\frac{\alpha}{4} \sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^i \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \dots, b_{k,i}(\bar{H})\big)\Big) \qquad \forall \mathcal{E} \in \mathbb{E}' \tag{2.19}$$

Competitiveness Ratio of The Integral Scheme

Finally, in the following theorem, we will find an upper-bound for the competitiveness ratio of the integral routing scheme specified in Algorithm 4.

Theorem 2.5.5. *If $\mathbb{S}_{integral}$ denotes the integral versatile routing scheme specified in Algorithm 4, the competitiveness ratio of $\mathbb{S}_{integral}$ in set $\mathbb{E}$ of possible routing cost environments (defined in Equation 2.3) has the following upper-bound:*

$$CR(\mathbb{S}_{integral}, \mathbb{E}) \leq \Theta(\log^2 |V|)$$

Additionally, if $\mathbb{E}'$ denotes the set defined in Equation 2.18, we will get the following upper-bound for the competitiveness ratio of the scheme:

$$CR(\mathbb{S}_{integral}, \mathbb{E}') \leq \Theta(\log |V|)$$

Proof. At first, we prove the following upper-bound for the routing cost of the solution suggested by scheme $\mathbb{S}_{\text{integral}}$ in environment $\mathcal{E} \in \mathbb{E}$ of possible routing cost environments:

$$C_{\mathcal{E}}\big(S_{\text{int}}(\mathcal{E})\big) \leq \sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^i \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big) \tag{2.20}$$

such that S_{int} denotes the suggested solution by scheme $\mathbb{S}_{\text{integral}}$ and $b_{n,i}(\bar{H})$ is defined in the following way (for every $n \in [i, k]$):

$$b_{n,i}(\bar{H}) = \begin{cases} \Pi_3(K_n) & \text{if } \Pi_1(K_n) \text{ and } \Pi_2(K_n) \text{ are } \alpha\text{-padded in } \bar{H} \ \& \ \bar{H} \text{ cuts } K_n \text{ in the } i^{th} \text{ level} \\ 0 & \text{otherwise} \end{cases}$$

and $(K_1, K_2, \ldots, K_k)$ denotes the sequence of commodities in environment $\mathcal{E}$.

Now, we prove Inequality 2.20.

$$C_{\mathcal{E}}\big(S_{\text{int}}(\mathcal{E})\big) = \sum_{e \in E} w(e) \cdot rrc\big(f^{(1)}_{S(\mathcal{E})}(e), f^{(2)}_{S(\mathcal{E})}(e), \ldots, f^{(k)}_{S(\mathcal{E})}(e)\big) \qquad \forall \mathcal{E} \in \mathbb{E}$$

Let (s, t, val) denote the n^{th} commodity in environment $\mathcal{E} \in \mathbb{E}$. Consider path $p = \mathbb{S}_{\text{integral}}(s, t)$ in graph $G = (V, E, w)$. Regarding Algorithm 4, path p is the union of a number of paths which are projections of tree edges in different levels of HRT $T^{(m)}$ for some m (see line 3). Additionally, assume that path $q \subseteq p$ denotes the projection of edge $\{(C, i), (C', i+1)\}$ of tree $T^{(m)}$, $u = (C, i).rep$, and $u' = (C', i+1).rep$ for some $i < h$ ($h = \lfloor \log \text{diam}_G \rfloor$). Since u and u' belong to cluster C' and C' is at level $i+1$, this is the case that:

$$d_G(u, u') \leq 2^{i+1}$$

Henceforth, this is the case that:

$$\begin{aligned} C_{\mathcal{E}}\big(S_{\text{int}}(\mathcal{E})\big) &= \sum_{e \in E} w(e) \cdot rrc\big(f^{(1)}_{S(\mathcal{E})}(e), f^{(2)}_{S(\mathcal{E})}(e), \ldots, f^{(k)}_{S(\mathcal{E})}(e)\big) \\ &\leq \sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^{i+1} \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big) \end{aligned}$$

Competitiveness ratio is computed by dividing the upper-bound obtained by Inequality 2.20 to the lower-bound of routing cost mentioned in Theorem 2.5.3. Consequently, we find the following high-threshold for the competitiveness ratio of scheme $\mathbb{S}_{\text{integral}}$:

$$CR(\mathbb{S}_{\text{integral}}, \mathbb{E}) \leq \frac{\sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^{i+1} \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big)}{\Theta\Big(\frac{\alpha}{4|\mathcal{H}_{\mathcal{E}}|} \sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^{i} \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big)\Big)}$$
$$\leq \Theta\big(\frac{8|\mathcal{H}_{\mathcal{E}}|}{\alpha}\big)$$

Since $|\mathcal{H}_{\mathcal{E}}| = \Theta(\log |V|)$, by letting$\alpha = \alpha_0 = \Omega(\frac{1}{\log |V|})$, we obtain some high-threshold equal to $\Theta(\log^2 |V|)$.

Additionally, in order to find a high-threshold for the competitiveness ratio of the versatile scheme in set $\mathbb{E}'$ of possible routing cost environments, we need to use Inequality 2.19. Consequently, this is the case that:

$$CR(\mathbb{S}_{\text{integral}}, \mathbb{E}') \leq \frac{\sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^{i+1} \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big)}{\Theta\Big(\frac{\alpha}{4} \sum_{i=0}^{h-1} \sum_{\bar{H} \in \mathcal{H}_{\mathcal{E}}} 2^{i} \cdot rrc\big(b_{1,i}(\bar{H}), b_{2,i}(\bar{H}), \ldots, b_{k,i}(\bar{H})\big)\Big)}$$
$$\leq \Theta\big(\frac{8}{\alpha}\big)$$

This implies that there is an upper-bound of $\Theta(\log |V|)$ for the competitiveness ratio if the routing cost environment is restricted to the commodity-independent functions for computing the edge routing costs.

□

2.6 Summary and Outlook

In this chapter, we introduced an approach to construct a top-down routing scheme for the oblivious routing problems. In this approach, we made a hierarchical decom-

position tree and then computed a path which connects the successive levels of such hierarchical decomposition tree together. Then, We explained that how the scheme is implemented using the hierarchical decomposition tree mentioned in Appendix B. Additionally, we mathematically analyzed the competitiveness ratio of both the fractional and integral types of the scheme in some specific range of the routing cost environments.

Bibliography

[1] S. S. Iyengar and Kianoosh G. Boroojeni, "Oblivious Network Routing: Algorithms and Applications," MIT Press, 2015.

[2] Gupta, Anupam, Mohammad T. Hajiaghayi, and Harald Rcke. "Oblivious network design." Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. Society for Industrial and Applied Mathematics, 2006.

[3] Fakcharoenphol, Jittat, Satish Rao, and Kunal Talwar. "A tight bound on approximating arbitrary metrics by tree metrics." Proceedings of the thirty-fifth annual ACM symposium on Theory of computing. ACM, 2003.

CHAPTER 3

BOTTOM-UP ROUTING SCHEMES IN OBLIVIOUS NETWORK DESIGN

The proliferation of network routing schemes has promoted massive network design to achieve low-latency and high-reliability with optimal cost. This chapter introduces an algorithm to construct an oblivious routing scheme to flexibly solve large-sized oblivious routing problems. More specifically, we construct an oblivious routing scheme to solve oblivious minimum-cost flow problems of arbitrary networks efficiently. This scheme is based on the bottom-up hierarchical independence tree introduced in Appendix B. We also analyze the routing cost incurred by this routing scheme utilizing a quantifier called "competitiveness ratio" (see Appendix A).

3.1 Introduction

In this chapter, we present the bottom-up oblivious routing scheme presented by Srini, Busch, and Iyengar [2]. This routing scheme suggests a oblivious solution to the single-source oblivious routing problem.

In Appendix B, three different types of Hierarchical Independence Trees (HITs) were introduced and addressed in detail. Before starting our discussion on the routing scheme, we summarize some important properties of the HITs. We use the following contracted notations in this chapter:

Consider HIT $T^s = (V_{T^s}, E_{T^s})$ of graph G.

- Assuming level i vertex (v, i) and its parent $(u, i+1)$ in tree T^s, μ_i denotes *the upper-bound of the distance between their corresponding vertices in graph*

[0]Part of this chapter has been reprinted with permission from S. S. Iyengar and Kianoosh G. Boroojeni, "Oblivious Network Routing: Algorithms and Applications," MIT Press, 2015 [1].

G:

$$\mu_i = \sup\left\{d_G(u,v)\middle|\{(u,i+1),(v,i)\} \in E_{T^s}\right\} \qquad \forall i \leq h-1 \qquad (3.1)$$

- For every level i vertex (v,i) of tree T^s, *the upper-bound of the graph distance between v and the corresponding vertex of every leaf in subtree $T^{ss}_{(v,i)}$* is represented by δ_i:

$$\delta_i = \sup\left\{d_G(u,v)\middle|u \in \mathcal{L}_{T^s}(v,i) \wedge (v,i) \in V_{T^s}\right\} \qquad \forall i \leq h \qquad (3.2)$$

- If (v,i) denotes some vertex in T^s such that $v \neq s$, *the lower-bound of the graph distance between source vertex s and the corresponding vertex of every leaf in subtree $T^s_{(v,i)}$* is denoted by ϕ_i:

$$\phi_i = \inf\left\{d_G(s,u)\middle|u \in \mathcal{L}_{T^s}(v,i)\ \&\ (v,i) \in V_{T^s} \wedge v \neq s\right\} \qquad \forall i \leq h \quad (3.3)$$

Table B.1 compares the different types of hierarchical independence trees based on parameters μ_i, δ_i, and ϕ_i. As we see in Lemma B.3.2 and B.4.3, assuming that T^s is an HIT type-0 or type-2, this is the case that:

$$B_G(s, 2^i - 1) \subseteq \mathcal{L}_{T^s}(s,i) \qquad i \leq h$$

which implies that the graph distance between the source vertex and those which are not in set $\mathcal{L}_{T^s}(s,i)$ is at least 2^i. Additionally, for every vertex $u \notin \mathcal{L}_{T^s}(s,i)$ there is some vertex $v \neq s$ in the i^{th} level that $u \in \mathcal{L}_{T^s}(v,i)$. Hence, we conclude that $\phi_i = 2^i$, and also:

$$\phi_i - 1 = \sup\left\{d_G(s,v)|v \in \mathcal{L}_{T^s}(s,i)\right\} \qquad \forall i \leq h \qquad (3.4)$$

Moreover, Figure 3.1 depicts some example which verifies that for an HIT type-1, $\phi_i \leq 2^i$. It is easy to verify the other contents of Table B.1 by considering the HIT definitions in Appendix B, Lemma B.2.5, and Lemma B.3.2.

HIT	$\log_2(\mu_i)$	$\log_2(\delta_i)$	$\log_2(\phi_i)$	HIS type
type-0	$i+1$	$i+1$	i	source-oriented
type-1	$i+1$	$i+1$	$< i$	basic
type-2[a]	$i+2$	$i+2$	i	basic

[a]Presented by Srini, et al. in 2010 [2].

Table 3.1: Comparison of different types of HIT.

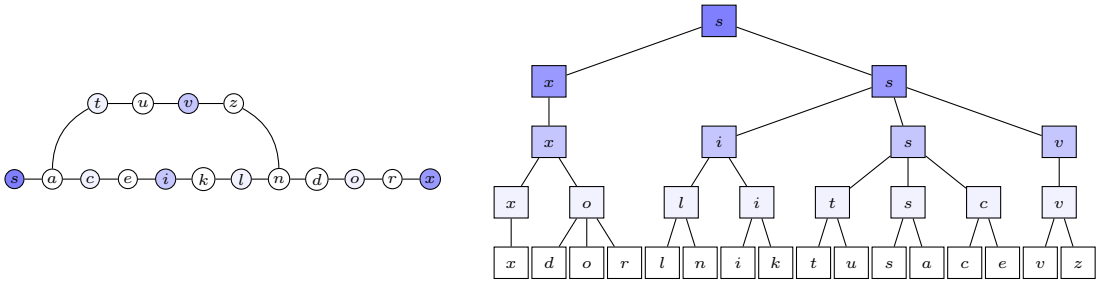


Figure 3.1: Schematic view of an HIS in some connected graph (left figure), and its corresponding HIT type-1 (right one). In this figure, $d \in \mathcal{L}_{T^s}(x, 3)$, but $d_G(s, d) < 2^3$. Moreover, $B_G(s, 2^3 - 1) \not\subseteq \mathcal{L}_{T^s}(s, 3)$.

3.2 Problem Specification

The routing problem is defined in the connected unweighted graph $G(V, E)$ (or, equivalently, weighted graph $(V, E, unit)$). Assuming that $s \in V$ and $D \subseteq V$ respectively denote the *source vertex* and the *set of target vertices*, the routing process is done in routing cost environment $\mathcal{E} \in \mathbb{E}$ such that $\mathbb{E}$ is defined in the following way:

$$\mathbb{E} = \left\{ (rrc, \sum, \bar{K}) \middle| rrc \in F_{\text{sub-additive}} \wedge \bar{K} \in \mathcal{K} \right\} \tag{3.5}$$

where $F_{\text{sub-additive}}$ is the set of all the sub-additive functions in the form $rrc : (\mathbb{Z}_{\geq 0})^{|D|} \mapsto \mathbb{R}_{\geq 0}$ and $\mathcal{K}$ is the set of all the commodity sequences of common source $(s \in V)$ and value (1); i.e.

$$\mathcal{K} = \left\{ (K_1, K_2, \ldots, K_{|D|}) \middle| K_i = (s, t_i, 1) \wedge t_i \in D \ \forall i \in [1, |D|] \right\} \tag{3.6}$$

Note that the network routing cost in this problem is the summation function. At the rest of this chapter, we represents this problem in form $P_{s,D}(G, \mathbb{E})$.

3.3 Construction of the Bottom-Up Oblivious Routing Scheme

After specifying the oblivious routing problem, here we use the HIT type-0 as an well-designed hierarchical routing tool to construct a competitive bottom-up routing scheme. Using this scheme, we will find a low-cost solution to the aforementioned problem. At first, we introduce the concept of projecting an HIT path on the corresponding graph of the HIT.

Definition 3.3.1. *Let $T^s = (V_{T^s}, E_{T^s})$ denote an HIT type-0 of height h in graph $G = (V, E)$. For every tree edge of level $i \leq h-1$ in the form $e = \{(u, i), (v, i+1)\}$, the projection of e on graph G is defined as the shortest path between vertices u and v in G and denoted by $proj(e, G)$. In addition, if p represents a path in tree T^s, the projection of p on graph G is defined in the following form:*

$$proj(p, G) = \bigoplus_{e \in p} proj(e, G) \tag{3.7}$$

For any routing problem in the form $P_{s,D}(G, \mathbb{E})$, the versatile routing scheme suggested by HIT type-0 T^s is defined as the following function:

$$\mathbb{S}_{T^s} : (\{s\} \times V) \mapsto (\text{the set of all the paths in } G)$$

such that for every target $t \in D$, this is the case that:

$$\mathbb{S}_{T^s}(s, t) = proj(p_{T^s}((t, 0), (s, h)), G)$$

such that $p_{T^s}((t, 0), (s, h))$ denotes the only path existed between the root of T^s and leaf $(t, 0)$. Note that $\mathbb{S}_{T^s}(s, t)$ is not necessarily a simple path in G (as it may cross itself).

3.4 Solution Cost Analysis

Assume that $D = \{t_1, t_2, \ldots, t_{|D|}\}$. Regarding Equation 3.5, cost environment $\mathcal{E} \in \mathbb{E}$ is in the following form:

$$\mathcal{E} = (c(\sum), \sum, (K_1, K_2, \ldots, K_{|D|}))$$

such that for every $i \in [1, |D|]$

$$K_i = (s, t_i, 1)$$

Scheme $\mathbb{S}_{T^s}$ provides a solution of the problem of cost environment $\mathcal{E}$ in the following form:

$$p = (p(t_1), p(t_2), \ldots, p(t_{|D|}))$$

where $p(t_j)$ is the projection of the unique tree path between $(t_j, 0)$ and (s, h) on graph G. Moreover, for every $j \in [1, |D|]$,

$$p(t_j) = \bigoplus_{i=0}^{h-1} p_i(t_j) \tag{3.8}$$

such that $p_i(t_j)$ is the projection of tree edge $\{parent_i((t_j, 0)), parent_{i+1}((t_j, 0))\}$ on G.

Concerning the definition of flow function in Appendix A, this is the case that:

$$f_p^{(j)}(e) = \begin{cases} 1 & e \in p(t_j) \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in [1, |D|]$$

In addition, assume that $f_p^{(j)}(e, i)$ is defined as the following:

$$f_p^{(j)}(e, i) = \begin{cases} 1 & e \in p_i(t_j) \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \leq h, j \in [1, |D|] \tag{3.9}$$

Using Equation 3.8, for every $j \in [1, |D|]$:

$$f_p^{(j)}(e) \leq \sum_{i=0}^{h-1} f_p^{(j)}(e, i) \tag{3.10}$$

Regarding the definition of the edge routing cost function in Appendix A, the routing cost of edge e in solution p is equal to:

$$C_p(e) = c(\sum_{j=1}^{|D|} f_p^{(j)}(e))$$

Since c is an increasing, concave function, regarding Inequality 3.10,

$$C_p(e) \leq c(\sum_{j=1}^{|D|}\sum_{i=0}^{h-1} f_p^{(j)}(e,i)) \leq \sum_{i=0}^{h-1} c(\sum_{j=1}^{|D|} f_p^{(j)}(e,i)) \tag{3.11}$$

According to Equation 3.9,

$$\sum_{j=1}^{|D|} f_p^{(j)}(e,i) = |F_p(e,i)|$$

where $F_p(e,i)$ is defined as below:

$$F_p(e,i) = \{v | e \in p_i(v) \wedge v \in D\}$$

In Definition B.2.6, we defined partition $\mathcal{V}_i$ on the vertex set of graph G:

$$\mathcal{V}_i = \{\mathcal{L}_{T^s}(u,i) | u \in I_i\} \qquad \forall i \in [0,h]$$

In the above equation, we assumed that T^s is an HIT type-1; however, the two other types also partition the vertex set in the same way. We use $\mathcal{V}_i$ to partition $F_p(e,i)$ in the following form:

$$F_p(e,i) = \bigcup_{u \in I_i} \{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}$$

Hence, this is the case that:

$$\sum_{j=1}^{|D|} f_p^{(j)}(e,i) = |F_p(e,i)| = \sum_{u \in I_i} |\{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}|$$

Using this equation and Inequality 3.11, we obtain:

$$\begin{aligned} C_p(e) &\leq \sum_{i=0}^{h-1} c(\sum_{u \in I_i} |\{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}|) \\ &+ - \leq \sum_{i=0}^{h-1}\sum_{u \in I_i} c(|\{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}|) \end{aligned}$$

Now we use the definition of the network routing cost function to compute the network routing cost of solution p in cost environment $\mathcal{E}$.

$$C_{\mathcal{E}}(p) = \sum_{e \in E} C_p(e) \leq \sum_{e \in E} \sum_{i=0}^{h-1} \sum_{u \in I_i} c(|\{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}|)$$

or equivalently,

$$C_{\mathcal{E}}(p) \leq \sum_{i=0}^{h-1} \sum_{u \in I_i} \sum_{e \in E} c(|\{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}|)$$

In addition,

$$\sum_{e \in E} c(|\{v | e \in p_i(v) \wedge v \in D \cap \mathcal{L}_{T^s}(u,i)\}|) = \sum_{e \in p_i(v)} c(|D \cap \mathcal{L}_{T^s}(u,i)|) + \sum_{e \notin p_i(v)} c(0)$$

which implies that ($c(0) = 0$):

$$C_{\mathcal{E}}(p) \leq \sum_{i=0}^{h-1} \sum_{u \in I_i} \sum_{e \in p_i(v)} c(|D \cap \mathcal{L}_{T^s}(u,i)|) = \sum_{i=0}^{h-1} \sum_{u \in I_i} |p_i(v)| \cdot c(|D \cap \mathcal{L}_{T^s}(u,i)|)$$

As the result, we find an upper-bound of cost $C_{\mathcal{E}}(p)$ for every environment $\mathcal{E} \in \mathbb{E}$ and suggested solution p of the presented scheme in this chapter:

$$C_{\mathcal{E}}(p) \leq \sum_{i=0}^{h-1} \sum_{u \in I_i} \mu_i \cdot c(|D \cap \mathcal{L}_{T^s}(u,i)|) \tag{3.12}$$

The following lemma provides a lower-bound of the minimum network routing cost in the same problem (cost of optimized solution):

Lemma 3.4.1. *If $\bar{p}^* = (p^*(t_1), p^*(t_2), \ldots, p^*(t_{|D|}))$ denotes an optimal solution of the problem of demand set $D = \{t_1, t_2, \ldots, t_{|D|}\}$ in cost environment $\mathcal{E}$,*

$$C^*_{\mathcal{E}}(\bar{p}^*) \geq \sum_{u \in J_i - \{s\}} \phi_i \cdot c(|D \cap \mathcal{L}_{T^s}(u,i)|) \qquad \forall i \in [0,h], \mathcal{E} \in \mathbb{E} \tag{3.13}$$

such that T^s is some HIT in G corresponding to $\bar{I}_s$, and for every $i \in [0,h]$, $J_i \subseteq I_i$ is a $(2\delta_i + 2\phi_i + 1)$-independent set.

Proof. Since $\mathcal{E} \in \mathbb{E}$, network routing cost of optimal solution is:

$$C^*_{\mathcal{E}}(\bar{p}^*) = \sum_{e \in E} c(f_{\bar{p}^*}(e))$$

where $f_{\bar{p}^*}(e) = \sum_{j=1}^{|D|} f^{(j)}_{\bar{p}^*}(e)$.

Moreover, if edge e doesn't belong to $p^*(t)$ for every $t \in D$, e will have no flow and its routing cost will be zero. Consequently, we obtain the following equation:

$$C^*_{\mathcal{E}}(\bar{p}^*) = \sum_{e \in \bigcup\limits_{t \in D} p^*(t)} c(f_{\bar{p}^*}(e))$$

Now, in order to find a lower-bound for $C^*_{\mathcal{E}}(\bar{p}^*)$, we only consider the cost of routing to those demand vertices that belong to $\mathcal{L}_{T^s}(u, i)$ for some $u \in J_i - \{s\}$. In fact, we only compute the routing cost of those paths in the form $p^*(v)$ such that:

$$v \in \bigcup_{u \in J_i - \{s\}} (D \cap \mathcal{L}_{T^s}(u, i)) \tag{3.14}$$

Regarding Relation 3.14, v also belongs to $\mathcal{L}_{T^s}(u, i)$ (for some $u \in J_i - \{s\}$) which implies that $|p^*(v)| \geq d_G(s, v) \geq \phi_i$ (see Equation 3.4). As the result, we can divide each $p^*(v)$ to two subpaths $p^*_1(v)$ and $p^*_2(v)$ such that $p^*_1(v)$ starts at vertex v, $p^*_2(v)$ ends at s, and $|p^*_1(v)| = \phi_i$. By ignoring routing cost of $p^*_2(v)$, we obtain:

$$C^*_{\mathcal{E}}(\bar{p}^*) \geq \sum_{e \in P} c(f_{\bar{p}^*}(e)) \tag{3.15}$$

such that

$$P = \bigcup_{u \in J_i - \{s\}} P_u$$

and

$$P_u = \bigcup_{v \in D \cap \mathcal{L}_{T^s}(u,i)} p^*_1(v)$$

Considering u_1 and u_2 as two different members of J_i, we will show that $P_{u_1} \cap P_{u_2} = \emptyset$. By contradiction, assume that there is some edge in both P_{u_1} and P_{u_2}.

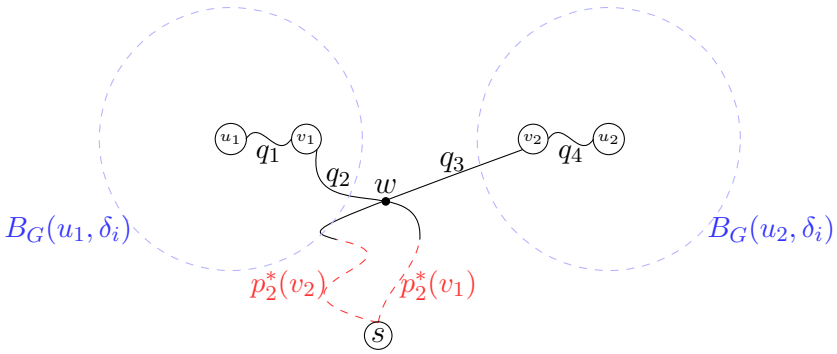


Figure 3.2: Solution paths $p^*(v_1)$ and $p^*(v_2)$ cross each other at w. v_1 and v_2 belong to $\mathcal{L}_{T^s}(u_1, i)$ and $\mathcal{L}_{T^s}(u_2, i)$, respectively.

As thre result, there are two vertices v_1 and v_2 such that $v_1 \in D \cap \mathcal{L}_{T^s}(u_1, i)$, $v_2 \in D \cap \mathcal{L}_{T^s}(u_2, i)$, and $p_1^*(v_1) \cap p_1^*(v_2) \neq \emptyset$. Let w denote the vertex in which $p_1^*(v_1)$ and $p_1^*(v_2)$ meet each other. Moreover, assume that q_1 is the shortest paths in G from u_1 to v_1 (q_4 is defined similarly for u_2 and v_2). See Figure 3.2 which specifies paths q_1, q_2, q_3, and q_4 (as you can see, q_2 represents the subpath of $p_1^*(v_1)$ connecting v_1 to w, and q_3 is between v_2 and w). Since v_1 is in some $\mathcal{L}_{T^s}(u, i)$ such that $u \neq s$, regarding Equation 3.2, $|q_1| = d_G(u_1, v_1) \leq \delta_i$. Because of the same reason, $|q_4| = d_G(u_2, v_2) \leq \delta_i$. Moreover, as $|p_1^*(v_1)|$ and $|p_1^*(v_2)|$ are not greater than ϕ_i, $|q_2| \leq \phi_i$ and $|q_3| \leq \phi_i$. Finally, we conclude that path $q_1 \cup q_2 \cup q_3 \cup q_4$ is a path of length not greater than $2\phi_i + 2\delta_i$ from u_1 to u_2 which contradicts the assumption that J_i is $(2\delta_i + 2\phi_i + 1)$-independent set.

Concerning that for two different vertices u_1 and u_2 in $J_i - \{s\}$, $P_{u_1} \cap P_{u_2} = \emptyset$, we rewrite Inequality 3.15 in the following form:

$$C_{\mathcal{E}}^*(\bar{p}^*) \geq \sum_{u \in J_i - \{s\}} (\sum_{e \in P_u} c(f_{\bar{p}^*}(e))) \tag{3.16}$$

Assume that for every $v \in D \cap \mathcal{L}_{T^s}(u, i)$, path $p^*(v)$ equals some specific path p such that $|p| = \phi_i$. In this case, as for every edge in p, $f_{\bar{p}^*}(e) = |D \cap \mathcal{L}_{T^s}(u, i)|$,

$$\sum_{e \in P_u} c(f_{\bar{p}^*}(e)) = \phi_i \cdot c(|D \cap \mathcal{L}_{T^s}(u, i)|).$$

Moreover, as c is a concave function, cost of this case is a lower-bound of incurred cost in other possible cases. Subsequently,

$$\sum_{e \in P_u} c(f_{\bar{p}^*}(e)) = \phi_i \cdot c(|D \cap \mathcal{L}_{T^s}(u,i)|) \qquad \forall u \in J_i - \{s\} \tag{3.17}$$

Inequalities 3.16 and 3.17 imply that the proof is complete.

□

Graph $G_i(I_i, E_i)$ is called the *level-i graph of* G if:

$$\forall u, v \in I_i : (\{u,v\} \in E_i) \leftrightarrow (d_G(u,v) \leq 2\delta_i + 2\phi_i)$$

Lemma 3.4.2. *If $\chi(G_i)$ denotes the chromatic number[1] of graph G_i, this is the case that:*

$$\exists J_i : \qquad \sum_{u \in I_i} c(|D \cap \mathcal{L}_{T^s}(u,i)|) \leq \chi(G_i) \cdot \sum_{u \in J_i - \{s\}} c(|D \cap \mathcal{L}_{T^s}(u,i)|) \tag{3.18}$$

such that T^s is some HIT in G corresponding to $\bar{I}_s$, and for every $i \in [0,h]$, $J_i \subseteq I_i$ is a $(2\delta_i + 2\phi_i + 1)$-independent set.

In the exercises, you are asked to prove Lemma 3.4.2. Finally, we obtain an upper-bound for the competitiveness ratio of routing scheme $\mathbb{S}_{T^s}$:

Theorem 3.4.3. *The competitiveness ratio of the versatile routing scheme $\mathbb{S}_{T^s}$ for the mentioned set of possible cost environments $\mathbb{E}$ has the following upper-bound:*

$$CR(\mathbb{S}_{T^s}, \mathbb{E}) \leq h \cdot \max_{i=0}^{h-1} \{ \frac{\chi(G_i) \cdot \mu_i}{\phi_i} \} \tag{3.19}$$

Proof. The theorem is directly obtained by Inequalities 3.12, 3.13, and 3.18.

□

[1]The smallest number of colors needed to color graph $G = (V, E)$ in a way that for every edge $\{u, v\} \in E$, u and v have different colors. Chromatic number of G is denoted by $\chi(G)$.

3.5 Summary and Outlook

In this chapter, we introduced an approach to construct a bottom-up routing scheme for the oblivious routing problems. In this approach, we made a hierarchical decomposition tree and then computed a path which connects the successive levels of such hierarchical decomposition tree together. First, a bottom-up oblivoius routing scheme for unweighted graphs was introduced. This routing scheme is based on the hierarchical independence trees defined in Appendix B. Moreover, we analytically addressed how the competitiveness ratio of the presented scheme can be bounded to a deterministic asymptotic upper-bound.

Bibliography

[1] S. S. Iyengar and Kianoosh G. Boroojeni, "Oblivious Network Routing: Algorithms and Applications," MIT Press, 2015.

[2] Srinivasagopalan, Srivathsan, Costas Busch, and S. S. Iyengar. "An oblivious spanning tree for single-sink buy-at-bulk in low doubling-dimension graphs." IEEE Transactions on Computers 61.5 (2012): 700-712.

PART II

APPLICATIONS OF OBLIVIOUS NETWORK OPTIMIZATION

CHAPTER 4

AN ECONOMIC DISPATCH ALGORITHM FOR CONGESTION MANAGEMENT OF SMART POWER NETWORKS: AN OBLIVIOUS ROUTING APPROACH

We present a novel oblivious routing economic dispatch (ORED) algorithm for power systems. The method is inspired by the oblivious network design which works perfectly for networks in which different sources (generators) send power flow toward their destinations (load points) while they are unaware of the current network state and other flows. Basically, our focus is on the economic dispatch while managing congestion and mitigating power losses. Furthermore, we study a loss-minimizing type of the economic dispatch which aims to minimize the emission by optimizing the total power generation rather than system cost. Compared to state-of-the-art economic dispatch methods, our algorithm is independent of network topology and works for both radial and non-radial networks. Our algorithm is thus suited for large-scale economic dispatch problems that will emerge in the future smart distribution grids with host of small, decentralized, and flexibly controllable prosumers; i.e., entities able to consume and produce electricity. The effectiveness of the proposed ORED is evaluated via the IEEE 57-bus standard test system. The simulation results verify the superior performance of the proposed method over the current methods in the literature in terms of congestion management and power loss minimization.

[0]Part of this chapter has been reprinted with permission from Kianoosh G. Boroojeni, et al., "An Economic Dispatch Algorithm for Congestion Management of Smart Power Networks: An Oblivious Routing Approach," Energy Systems, vol. 7, no. 27, 2016.

4.1 Introduction

There has been a growing trend in the electric power system from a few central bulk generation units to a large number of distributed generation units. The ever-increasing integration of the distributed generation, distributed storage, demand response, and communication technologies are major drivers for transition to the smart grid [20]. Evolving modern technologies, such as highly-efficient photovoltaics, are the game-changer in distributed renewable resources' utilization. According to a report of U.S. Department of Energy in 2008 [21], recent advances in wireless communications and smart grid designs have enabled the development of low cost power distribution systems. Smart distribution networks might have some specific functionalities such as self-healing, self-decision making, and resiliency [22, 23]. These requirement compels a new way of designing smart distribution grids for an economically-dispatched power distribution. As eonomic dispatch (ED) is one of the optimization problems related to the power systems operation, several techniques used in the literature to solve this problem, e.g., mixed integer linear programming (MILP), constraint relaxation, and network approach [24]. Due to large-scale nature of the optimization problems in power systems, we have to utilize numerically-efficient algorithms to overcome the complexity of this problems, specifically the ED problem [25].

4.1.1 Related Work

Economic dispatch: ED is one of the important optimization problems of the power systems operation that calculates the optimum scheduling of the committed generation units. However, the unit commitment (UC) problem is solved on a day-ahead basis; ED is solved on an hourly basis and uses the results of UC [42, 43]. Numerous

heuristic and exact solution strategies are proposed to solve ED. A novel particle swarm optimization is proposed in [44] to solve the nonconvex ED problems. In [45], a multi-agent systems-based load management is proposed for the smart distribution networks. In [46], the graph structure of power flow problem is explored comprehensively. Sanjari et al. in [12] combined day-ahead and hour-ahead optimizations to design an online controller which optimally dispatches the resources. According to [48, 49], current widely-used ED methods utilize MILP to solve the ED problem. Botterud *et al* proposed a demand dispatch considering large utilization of wind power [50]. According to [51], high penetration of energy storages makes ED problem highly non-convex and hard to solve. Li *et al.* proposed an exact relaxation approach to relax this non-convex problem to a convex form. According to [52], in order to obtain the communities cost model, *i.e.* distribution system feeders' cost models for a given network, the objective function is to minimize the cost of connections between load points and Fossil-Fuel Power Plants (FFPP). The objective function includes two terms, a fixed charge of connection as well as the variable cost per unit power flow. Khator *et al.* suggest a mixed 0-1 linear programming formulation to solve this optimization problem as the variable cost can be formulated as a non-convex function.

An electricity cost optimization model for a smart distribution grid with distributed generation and bidirectional power transactions was presented by Yi Liu *et al.* in 2013 [53]. Their work is based on a two-tier pricing model for trading electricity between the communities which are equipped with renewable power resources. They classify the power demand into deferrable and essential load and make some scheduling decisions to optimize the electricity trading costs between communities.

One of the most critical challenges of designing smarter grids is to make the power distribution facilities more flexible to the distributed nature of the genera-

tion units [54]. A survey on the existing distributed algorithms to solve the power dispatch problems, i.e., ED and optimal power flow, as well as a novel nodal-based distributed algorithm to solve the distributed energy management problems is presented in [20]. In addition, the customers have a time-varying usage pattern in the context of an electricity distribution system [13, 53]. In the smart distribution networks, due to the high penetration of the distributed generations, we might face line congestions. Hence, there is an exigent requirement for developing ED methods which prevent the congestion in the distribution network. Maknouninejad et al. proposed a practical method to deal with large scale utilization of distributed generations (DGs)by classifying them into some groups which have been dispatched in a microgrid [23]. According to [23], virtual leaders of these groups used utilization ratio of DGs which indicates the percentage of the available power each DG. The proposed ration was propagated to each group utilizing cooperative control. Eventually, they used a game theoretic approach to model the interaction of microgrids with the main grid and optimize the power flow.

Unit commitment (UC) and ED problems play a pivotal role in in managing and optimizing the operations of power systems [24]. In order to solve UC problem, a mixed integer programming is solved which involves the solution of ED as well. In [24], a mixed-integer state-space model was developed to solve these two problems considering the network constraints.

In [25], a multi-period linear optimization model was presented to provide the network model in terms of equations for power generation and transmission considering DC power flow model. Nolden et al. also described the application of AC power flow model for an in-depth representation of power systems at different voltage levels. Furthermore, they applied the proposed multi-period linear optimization model for analyzing the regional long-term expansion of German power network

while considering congestion management in the transmission grid. In our study we mainly focus on the congestion management while solving power flow problem.

A modeling approach for greenhouse gas emissions quotas is proposed in [26]. This approach was deployed into a stochastic dual dynamic programming algorithm which is commonly used for dealing with the hydro-thermal generation units scheduling problem. According to Rebennack et al. this approach is flexible and capable of considering the detailed model of emission and its corresponding constraints.

In [27], an optimal capacity expansion strategy for the power grid was developed to minimize the possibility of blackouts. This paper also considered the capacity-expansion decisions, i.e. the new transmission lines and the additional power flow capacity was taken into account in the expansion problem.

Oblivious Network Routing

Oblivious network routing is an approximation solution to the Minimum-Cost Flow Problem (MCFP) which is an optimization and decision problem. MCFP aims to find the cheapest possible way of sending a certain amount of flow through a network. MCFP is one of the most fundamental among all flow and circulation problems because most other such problems can be reduced to this problem. In many real-world applications of MCFP, we have little or no information regarding the flow cost function, size, source and destination of commodities flowing through the network. These types of obliviousness makes an MCFP an uncertain problem which cannot be solved by tranditional algorithms (e.g. Ford-Fulkerson [28]). This need motivates a plenty of research works which leads to the creation of many apporximation algorithms to solve oblivious MCFP. Most of them consider the offline problem with centralized control over the network. Among those, some research works like [29, 30, 31, 32, 33, 34, 35] have proposed approximating solutions for the

MCFP with general cost function; while others ([38, 39, 40, 41, 42, 43]) have solved the buy-at-bulk problem which only covers the MCFPs with concave/monotone sub-additive cost functions. For example, Goel and Estrin [41] gave an algorithm for off-line single-sink network flow problem where the flow cost shows concave behavior with respect to the size of commodities which flow through each edge of the network. Also, Rebennack et al. [36] proposed a continuous, bilinear formulation for the fixed charge network flow problem. They derived an exact solution based on their proposed formulation which converges in a finite number of steps[36]. On the other hand, there are a few works like [37, 20] which consider the online form of the MCFP and propose distributed algorithms to approximate the optimal solution.

Some works like [32, 34, 35, 41] aim to approximate the expected value of aggregated cost incurred by flowing commodities throughout the network; however, the algorithms proposed in [19, 20, 35] aim to asymptotically bound the flow cost of the solution to some coefficient of the optimal/minimum cost; this coefficient is called the *competitiveness ratio* of the algorithm (or its corresponding routing scheme).

A comprehensive survey on the studies related to optimal power flow is provided in [44]. Frank et al. considered OPF as one of the most important nonlinear optimization problems which has been widely studied. In [44], deterministic methods toward OPF were introduced. Furthermore, the state-of-the-art stochastic methods for solving OPF is examined in [45].

Approximated solutions of MCFP usually propose a routing scheme in the form of a spanning tree or a set of trees on the graph representing the network (See [32, 33, 34]). This kind of routing schemes (oblivious routing schemes) have been thoroughly studied in our recently published book [19].

Oblivious routing schemes have the common characteristics of being flexible to the time-varying network topology. They also handle dynamic source-sink flows by

dispatching the traffic flows in a distributed manner over the network and preventing the edge/node congestion [19].

4.1.2 Our Contribution

In this work, we present a novel algorithm to solve ED utilizing oblivious network routing approach which minimizes the line congestion through the arbitrary topology of distribution network. The proposed Oblivious Routing-based ED, denoted by $\mathcal{ORED}$, works perfectly for large-scale networks with time-varying network topology and dynamic source-sink flows (outline of the proposed solution is presented in Figure 5.3). The goal of this chapter is to implement an efficient congestion management approach and to minimize the power loss of the power distribution network. The proposed $\mathcal{ORED}$ takes advantage of an oblivious routing scheme to prevent the line power congestion. In other words, the pivotal idea of our method is to utilize the oblivious routing schemes which we proposed in [19] to achieve a congestion-free distribution network while satisfying load demands reliably. Furthermore, we propose an asymptotic high-threshold on the line power congestion. Our proposed algorithm is thus optimally suited for the very large-scale dispatch problems that significantly arises in future smart distribution grids with small, decentralized, and flexibly controllable renewable generation units. Theoretically, we show that the total cost of the energy distribution is not be more than $O(\log^2 n)$ times as the minimum possible cost where n is the number of distributed power generation units.

The rest of the chapter is organized as follows. Section 2 specifies the economic dispatch model in the form of a graph optimization problem. Section 3 is devoted to presenting a short introduction to oblivious network design and some preliminary

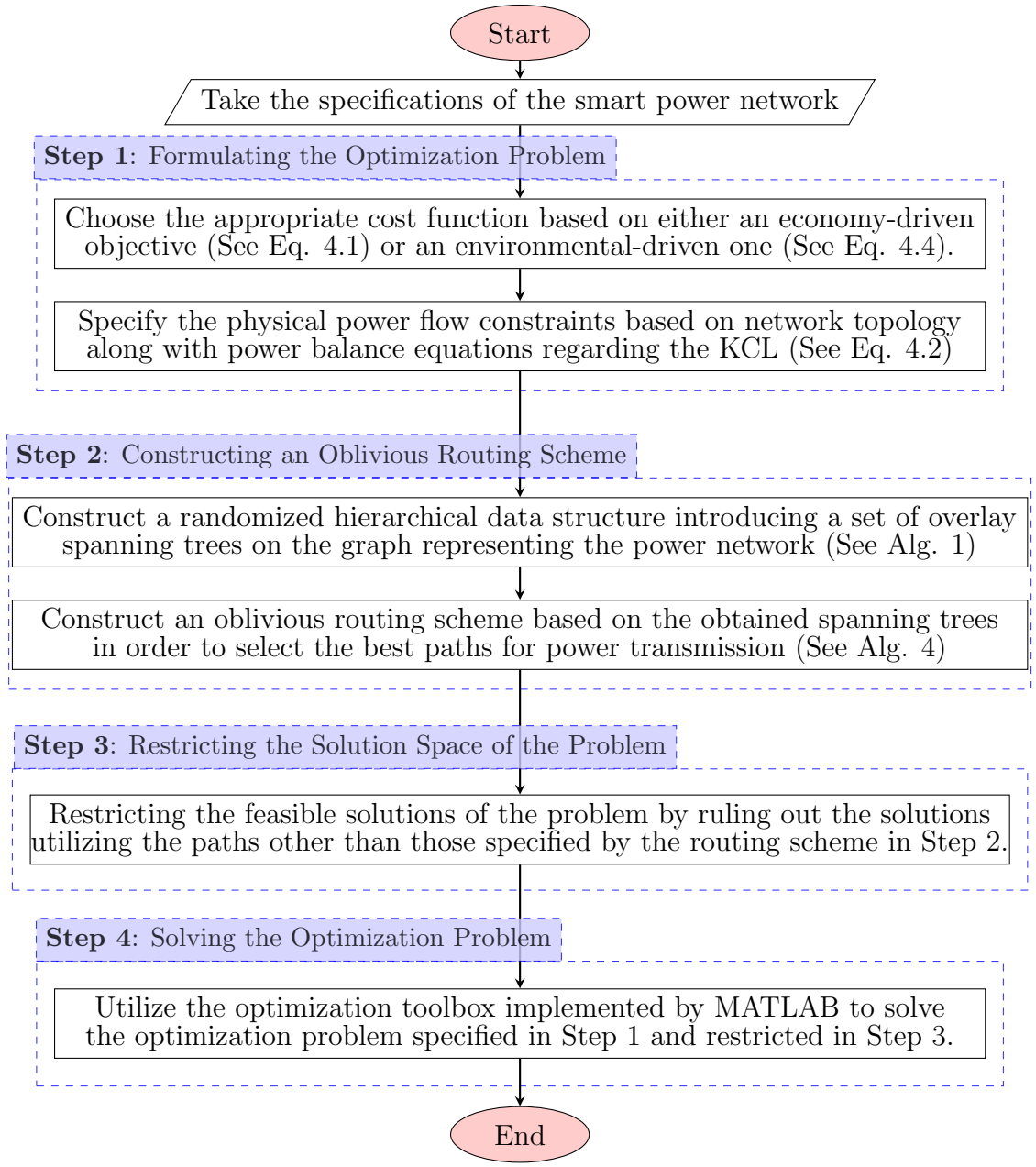


Figure 4.1: Flowchart of the proposed economic dispatch algorithm $\mathcal{ORED}$.

definitions. In section 4, we describe the proposed novel $\boldsymbol{\mathcal{ORED}}$ algorithm and its competence theoretically. In Section 5, a case study is presented and the superior performance of our novel dispatch algorithm is proved experimentally. Finally, In Section 6 summary and conclusions are given.

4.2 Preliminaries

In contrast to the traditional MCFPs that are defined with a well-defined set of commodities (with specific size and source/destination nodes) and given flow cost function for every edge, oblivious network routing aims to solve an MCFP in which either the flow cost function is not specified (flow cost is oblivious), or the commodities size, source, and destination are not specified (commodities are oblivious).

Oblivious network routing approaches to oblivious MCFPs by employing a practical method not guaranteed to be perfect, but sufficient for the immediate goals and an acceptable approximation of the optimal solution. In this regard, oblivious network routing is considered to be a heuristic method since it can be used to speed up the process of finding a satisfactory solution (while computing the optimal routing is impractical).

Oblivious network routing is different with Dynamic (adaptive) Routing (DR) since DR proposes a different solution in response to any change of the MCFP oblivious components; while, oblivious routing deploys a single routing scheme for an oblivious MCFP with the aim of approximating the optimal solution of the problem with oblivious parameters.

The common characteristics of the oblivious routing schemes includes being pretty flexible to the obliviousness of the environment in which the commodities are flowing and making the traffic flows distributed over the network and preventing the flow-congestion in some specific nodes or edges. Additionally, these types of routing schemes provide a low-cost flow routing in long term even if a wide range of unpredictable events occur in the network like bursty flow derived from a specific node or failure of some node in forwarding the flow through the network. In fact, the versatile routing schemes best

fit to those networks that we have little/no knowledge regarding their current and future states.

4.2.1 Preliminaries to Oblivious Network Routing

At first, we provide some basic definitions and preliminary data structures needed to explain the $\mathcal{ORED}$ algorithm. In the rest of the chapter, $\mathcal{G} = (V, E, w)$ is considered as a weighted connected graph of diameter[1] 2^h (for some integer h) where $w_e > 1$ for every $e \in E$. The mentioned conditions on the graph diameter and its weight function doesn't reduce its generality because any weighted connected graph can be converted to $\mathcal{G}$ by multiplying its weight function.

Assuming Δ as a positive real number, Δ-partition of $\mathcal{G}$ is defined as a partition[2] of V into a number of subsets S such that there exist no pair of nodes in S with distance[3] of more than Δ from each other.

Hierarchical decomposition sequence (HDS) $\bar{H}$ of graph $\mathcal{G}$ is the vector $\bar{H} = (H_0, H_1, \ldots, H_h)$ where $H_h = \{V\}$, H_i is a 2^i-partition, and if S belongs to H_i, there exists some $S' \in H_{i+1}$ such that $S \subseteq S'$ for every $i = 0, 1, \ldots, h-1$. Partition H_i is called the i^{th} level partition of $\bar{H}$. Node v is called α-padded in $\bar{H}$ $(0 < \alpha < 1)$ if for every i, the i^{th}-level partition H_i contains a set $S \subseteq V$ such that S completely

[1]Diameter of a graph is defined as the maximum distance between any pair of nodes in the graph.

[2]Partition of set A is a set of A subsets $\{A_i | i \in [1,k]\}$ such that $\bigcup_{i=1}^{k} A_i = A$, $\forall i \in [1,k] : A_i \neq \emptyset$, and $\forall i \neq j : A_i \cap A_j = \emptyset$.

[3]The distance $d_{\mathcal{G}}(u,v)$ between nodes u, v is defined as the length of shortest path between u and v in $\mathcal{G}$. The length of a path in a weighted graph is defined as the total weight of the edges participating in the path.

covers ball[4] $B_{\mathcal{G}}(v, \alpha \cdot 2^i)$; in other words, for every level i, there exists some $S \in H_i$ such that $B_{\mathcal{G}}(v, \alpha \cdot 2^i) \subseteq S$ [19]. See Figure 2.1 for illustration.

In 2003, Fakcharoenphol *et al.* [17] proposed a randomized algorithm to generate a random HDS in which a minimum quotient of nodes are asymptotically guaranteed to be α-padded with high probability ($\alpha \leq 1/8$). The main idea of this algorithm is to construct the i^{th}-level partitions by dividing each $(i+1)^{th}$-level set $S \in H_{i+1}$ into smaller subsets $S_1, S_2, \ldots$ such that there exists no node $r \in S_j$ such that r is further than δ_i from an S's special member called its *representative*. Value of δ_i is chosen randomly in the interval $[2^{i-2}, 2^{i-1})$. For more details, see Algorithm 1.

Finally, we define the Hierarchical Decomposition Tree (HDT) which is the main data structure utilized for the novel $\mathcal{ORED}$ algorithm and is constructed based on the HDS $\bar{H}$. HDT is an h-level tree where its i^{th} level nodes are the members of H_i (for every $i = 0, 1, \ldots, h$) and any i^{th} level tree node $S \in H_i$ is connected to its parent $S' \in H_{i+1}$ in upper-level $(i+1)$ if $S \subseteq S'$.

4.2.2 Constructing the Oblivious Routing Scheme

First, we show how Algorithm 4 constructs the oblivious routing scheme (mentioned in [19]) which plays the main role in simplification of the couple optimization problems mentioned in Section II by placing more restrictions on the problems, shrinking their feasible areas, and subsequently reducing the computational complexity of them so that their solutions become computationally feasible.

Algorithm 4 gets the weighted graph $\mathcal{G}$ as its input and returns function $\mathbb{S} : \{G_i\}_1^n \times \{F_i\}_1^q \mapsto \mathcal{P}(E)$ which represents the oblivious routing scheme and specifies

[4]Set $B_{\mathcal{G}}(v, r) \in V$ is called a ball of center $v \in V$ of radius r if $u \in B_{\mathcal{G}}(v, r)$ iff $d_{\mathcal{G}}(u, v) \leq r$.

a path [5] in graph $\mathcal{G}$ for every pair of source-sink nodes. The algorithm starts with generating $27 \log |V|$ random HDS's utilizing randomized Algorithm 1. The reason for generating multiple random HDS's is to assure the existence of at-least one HDS $\bar{H}$ for every pair of nodes $source \in G$ and $sink \in F$. In fact, Iyengar *et al.* in [19] proved that the probability of existing at-least one α-padding HDS among the $c \log |V|$ HDS's constructed by Algorithm 1 is more than $1 - e^{-\frac{(c-2)^2}{2c}}$ (for every $\alpha \leq 1/8$). By considering $c \geq 27$, the mentioned probability would be greater than $1 - 10^{-5}$ which provides a reasonable theoretical guarantee that Algorithm 4 finds atleast one HDS for every source-sink pair.

After creating $27 \log |V|$ random HDS's and their corresponding HDTs, Algorithm 4 runs its main loop in lines 5 to 11 for every pair of source-sink nodes. Assume T as the HDT corresponding to the α-padding HDS of an arbitrary pair of source-sink nodes. Tree T would have a pair of leaves (level-zero nodes) corresponding to the source and sink (as $\mathcal{G}$ is supposed to be a weighted graph of the weight function greater than one for every edge). Let p denote the only tree path connecting the mentioned leaves together. Finally, the resulted routing scheme is computed in line 11 where the path between source and sink nodes $\mathbb{S}(source, sink)$ is obtained by projecting p on graph $\mathcal{G}$. The projection of path p of HDT T on graph $\mathcal{G}$ is defined in the following way:

[5] In this chapter, path of a graph is considered as a simple path and represented by a subset of edge set E such that there exist a permutation of edges in a path where the first edge is incident to the start node of the path, each two consecutive edges are incident to a common node, and the last edge is incident to the end node of the path.

Consider γ_e as the shortest path between the incident nodes of arbitrary edge $e \in p$ in graph $\mathcal{G}$. The projection of tree path p on the graph is obtained by concatenating all of the shortest paths γ_es back to back. In the case that the concatenation result is not a simple path and has crossed some nodes more than once, the projected path is the shortest simple path corresponding to the concatenation result.

4.3 Economic Dispatch Modeling and Problem Formulation

In this study, we consider the AC optimal power flow problem which is comprehensively introduced in [46] from an operation research perspective. Generally, ED problem refers to the dispatch of generators based on their cost functions among the specified demand. Security-constrained ED (SCED) is defined by considering N-1 reliability criterion. In [46] the SCED problem is defined by adding the N-1 reliability criterion to the ED problem. However, in [47] the SCED is formulated as ED with additional constraints which is the main goal of our proposed oblivious routing-based ED as well. We refer to our proposed ED problem as congestion-constrained ED. Economic dispatch is the problem of specifying the generation level of some generators to satisfy load demand while minimizing the generation cost. In the congestion-constrained economic dispatch, the line flow limits are considered as constraints in the optimization problem.In this section, economic dispatch is modeled as a routing problem in a general weighted graph and formulated as an optimization problem of linear constraints.

Consider a smart distribution network including several feeders in a city. Each feeder has been designed to provide electricity for a specific so-called *community*. There are a number of electricity customers in the community. There are also a num-

ber of conventional power plants[6] (with certain dispatchable generation capacity) which are installed at the transmission level, i.e. out of the communities, and are connected to the distribution network via distribution substations in order to satisfy the communities load demand. Fossil-Fuel Power Plants (FFPPs) such as Coal and Gas-fired power plants generate the electric energy which can be injected to each community in the city through the distribution network. There is a bidirectional power transmission line between any two connected buses.

Let $S_{x,y}$ denote the complex power which flows through line $e = \{x, y\}$. In the AC power flow formulation, we have:

$$S_{x,y} = V_x Y^*_{x,y}(V_x - V_y e^{-j(\theta_x - \theta_y)}),$$

where Y represents the admittance matrix, Y^* is the notation for complex conjugate, $V_.$ and $\theta_.$ specify the voltage magnitude and voltage angle of an arbitrary bus respectively, and imaginary unit is denoted by j. Furthermore, $\Re\{S_{x,y}\}$ shows the real component of $S_{x,y}$. Our optimization problem uses $f_{x,y} = \Re\{S_{x,y}\}$ as the optimization variable directly. However, our model is extendable to the full AC power flow equation where the optimization variables are the voltage magnitudes and phase angle differences [48].

Generation Cost Function

The cost function of conventional generation units is considered to be quadratic, i.e., for generator b the cost of generating g_b units of power is $C_b(g_b) = \alpha_b g_b^2 + \beta_b g_b + \gamma_b$, where α_b, β_b, and γ_b are the constant coefficients which are determined based on the generator type [66].

[6] As this chapter focuses on the novel method for solving the economic dispatch problem, we assume the throughput of the generator to be deterministic. This way, the method can be illustrated more clearly.

Optimization Problem

We assume that the network topology is fixed and does not change during the routing process.Assume that V and E denote the set of buses (communities/FFPPs) and medium/low voltage transmission lines of a power distribution network respectively. The network is represented by a *general* weighted graph $\mathcal{G} = (V, E, w)$ where $w : V^2 \mapsto \mathbb{R}_{\geq 0}$ specifies the weight of each line/edge which is directly proportional to its impedance. Considering that F and G are the set of feeders and FFPPs respectively, $V = F \cup G$, every feeder $a \in F$ has a power demand of d_a, and every FFPP $b \in G$ can generate power of value $g_b \in \{0\} \cup [\min_b, \max_b]$ where $\min_b$ and $\max_b$ represent the minimum and maximum generation limits of FFPP b respectively. The corresponding hourly economic dispatch problem is formulated as the problem of minimizing the total generation cost of FFPPs $\sum_{b \in G} C_b(g_b)$ subject to:

$$\begin{cases} (i) & g_b \in \{0\} \cup [\min_b, \max_b],\ \forall b \in G \\ (ii) & \sum\limits_{x \neq b} f_{x,b} = -g_b,\ \forall b \in G \\ (iii) & \sum\limits_{x \neq a} f_{x,a} = d_a,\ \forall a \in F \\ (iv) & f_{x,y} + f_{y,x} + loss_e = 0,\ \forall e = \{x, y\} \in E \\ (v) & f_{x,y} = 0, \text{ if } \{x, y\} \notin E \\ (vi) & |f_{x,y}| \leq \mathcal{L}_{x,y}, \text{ if } \{x, y\} \in E \end{cases}$$

where $loss : E \mapsto \mathbb{R}_{\geq 0}$ specifies the power loss in each transmission line, $f_{x,y}$ denotes the amount of power flow from bus x to bus y for every $\{x, y\} \in E$ (if the power flows in the reverse direction, $f_{x,y}$ has a negative value).Note that the symbol $\mathcal{L}_{x,y}$ in the sixth condition specifies a high-threshold for the flow congestion of line connecting busses x and y. Obviously, $f_{x,y} = 0$ if there exists no line between buses x and y or the line between the buses is not utilized effectively ($\{x, y\} \notin E$). Moreover, the

value of $loss_e$ is considered to be directly proportional to the line impedance or w_e ($\forall e \in E$) and a function of power flow; i.e.

$$loss_e = w_e \times \max\{|f_{x,y}|, |f_{y,x}|\} \qquad \text{where } e = \{x, y\}$$

By replacing g_b with its equal expression in the second aforementioned constraint, the minimization problem is reformulated in the following form:

$$\min_{f_{\cdot,\cdot}} \Big\{ \sum_{b \in G} C_b\big(- \sum_{x \neq b} f_{x,b}\big) \Big\} \tag{4.1}$$

$$\text{s.t.} \begin{cases} \sum\limits_{x \neq b} f_{x,b} \in \{0\} \cup [-\max_b, -\min_b] & \forall b \in G \\ \sum\limits_{x \neq a} f_{x,a} = d_a & \forall a \in F \\ f_{x,y} + f_{y,x} + w_e \times \max\{|f_{x,y}|, |f_{y,x}|\} = 0 & \forall e = \{x, y\} \in E \\ f_{x,y} = 0 & \text{if } \{x, y\} \notin E \\ |f_{x,y}| \leq \mathcal{L}_{x,y} & \text{if } \{x, y\} \in E \end{cases} \tag{4.2}$$

The last condition mentioned in 4.2 enforces the solution not to exceed the high-thresholds of each line ($\mathcal{L}_{x,y}$). Also, later in Section IV, we prove that the flow of each line in the proposed $\mathcal{ORED}$ does not exceed an asymptotic upper-bound value.

Note that the specified objective function in (4.1) is aimed to minimize the total generation cost. However, in order to obtain a loss minimizing dispatch, we may compromise the generation cost to minimize the total distribution network loss. Subsequently, in order to have a loss-driven optimization problem rather than an economy-driven one, the objective function specified in 4.1 should be re-formulated in the following way:

$$\min \sum_{e \in E} loss_e = \min_{f_{\cdot,\cdot}} \Big\{ \sum_{\{x,y\} \in E} -(f_{x,y} + f_{y,x}) \Big\} \tag{4.3}$$

subject to the constraints mentioned in (4.2). Moreover, let's consider the fact that total power generation is equal to the overall demand added by the entire amount of power loss in the network. Consequently, since the overall power demand is the summation of some given constants (d_as), the problem of minimizing the entire power loss is equivalent to optimizing the total power generation:

$$\min \sum_{b \in G} g_b = \min_{f_{\cdot,\cdot}} \Big\{ - \sum_{b \in G} \sum_{x \neq b} f_{x,b} \Big\} \tag{4.4}$$

Section 4.4 presents two novel economic dispatchers ECO-$\mathcal{ORED}$ and LM-$\mathcal{ORED}$ to minimize the objective functions mentioned in (4.1) and (4.4) respectively (both minimization problems have the same set of constraints specified in (4.2)).

4.4 Oblivious Routing Economic Dispatch ($\mathcal{ORED}$) for Smart Power Networks

This section proposes the novel $\mathcal{ORED}$ method which utilizes a top-down oblivious routing scheme to solve the economic dispatch problem of the distribution network and find the paths through which the power can flow. Moreover, the method uses the optimization toolbox of MATLAB 2015a [18] made for minimizing the class of linearly-constrained optimization problems in order to compute the amount of each power flow passing through the distribution network.

In the Appendix, we showed how the Algorithm 4 constructs the oblivious routing scheme (mentioned in [19]) which plays the main role in simplification of the couple optimization problems mentioned in Section II by placing more restrictions on the problems, shrinking their feasible areas, and subsequently reducing the computational complexity of them so that their solutions become computationally feasible. In this section, two different variations of minimization problems which formulate

the economic dispatch problem is described. Finally, an asymptotic analysis on the performance of our economic dispatch problem is presented.

Hereafter, we focus on the graph-based solution of the economic dispatch problem. Regarding the schematic Figure 4.2, path p_{ij} connects the i^{th} source (generation unit) G_i and the j^{th} sink F_j (demand side) through the distribution network. Based on what has already mentioned in this section, Algorithm 4 creates the scheme $\mathbb{S}$ which specifies path $p_{ij} = \mathbb{S}(G_i, F_j)$ for every i and j. Consequently, the remaining subproblem that needs to be addressed is finding the efficient amount of power ϕ_{ij} which is dispatched from the i^{th} generation unit to the j^{th} sink through the path p_{ij} for every i and j. This means that utilization of oblivious routing scheme reduces the computational complexity of the aforementioned optimization problems with $|V|^2$ variables (in the form of $f_{.,.}$) defined in Section II. The reduced form of the problem (which is computationally feasible and has very lower computational complexity) has only $|F| \times |G|$ variables in the form of $\phi_{.,.}$.

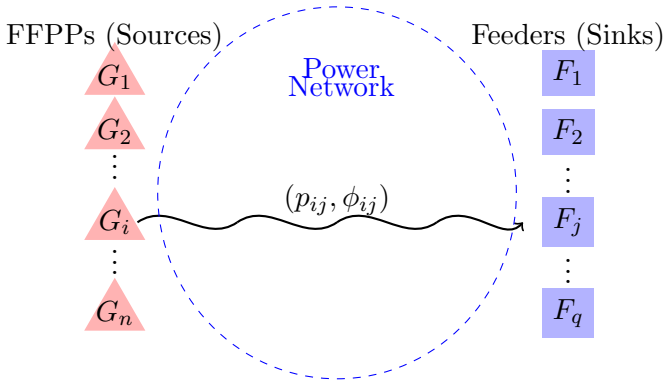


Figure 4.2: Schematic view of the sources and sinks in the economic dispatch problem and the variables of the minimization problem needed to be solved.

Economy-Driven Optimization Problem

As mentioned in Section II, the economic dispatch problem which aims to minimize the generation cost is formulated by the optimization problem with the objective

function specified in (4.1) subject to the conditions mentioned in (4.2). In order to adjust the problem formulation in Section II with the graph-based formulation specified in this section, we replace the flow variable $f_{x,y}$ with the expression in the following form (for every $x, y \in V$):

$$f_{x,y} = \sum_{i=1}^{n} \sum_{j=1}^{q} \phi'_{ij}(x, y) \tag{4.5}$$

where $\phi'_{ij} : V^2 \mapsto \mathbb{R}$ is a function specifying the amount of power dispatched from the i^{th} source G_i to the j^{th} sink D_j passing through the line $\{x, y\}$ and seen in the direction $\overrightarrow{xy}$. ϕ'_{ij} is formally defined in the following way:

$$\phi'_{ij}(x, y) = \begin{cases} \phi_{ij} - \mathcal{LOSS}_{ij}(y) & p_{ij} \text{ passes from } x \text{ to } y; \\ -\phi_{ij} + \mathcal{LOSS}_{ij}(y) & p_{ij} \text{ passes from } y \text{ to } x; \\ 0 & \{x, y\} \notin p_{ij} \end{cases}$$

where $\mathcal{LOSS}_{ij} : V \mapsto \mathbb{R}_{\geq 0}$ is the loss function that returns the amount of power loss which occurs while passing ϕ_{ij} units of power through the path p_{ij} from the start point G_i up to the input node.

Consequently, the new formulation of the optimization problem is obtained by replacing the variables in the form $f_{.,.}$ with their equivalent values specified in (4.5). The resulting problem has only $|F| \times |G|$ variables and can be solved by a strong optimization tool (in this chapter, we utilized *fmincon* function in the optimization toolbox of MATLAB 2015a [18]). We refer to the proposed graph-based solution of this problem as ECO-$\mathcal{ORED}$.

Loss Minimization Problem

Similar to what we did for reformulating the economy-driven optimization problem, the variables in the form $f_{.,.}$ should be replaced by their equivalent values specified

in (4.5). This way, we have a new optimization problem aimed to minimize the reformulated expression (4.4) which is in the form:

$$-\sum_{b\in G}\sum_{x\neq b}\sum_{i=1}^{n}\sum_{j=1}^{q}\phi'_{ij}(x,b)$$

subject to conditions (4.2) reformulated by replacing $f_{.,.}$ with $\sum_{i=1}^{n}\sum_{j=1}^{q}\phi'_{ij}(\cdot,\cdot)$. We refer to the proposed graph-based solution of this problem as LM-$\mathcal{ORED}$.

Asymptotic Analysis of the Proposed $\mathcal{ORED}$ Algorithms

Here, we claim that the dispatch/generation cost of the proposed algorithm is asymptotically competent compared with the optimal solution. Iyengar *et al.* in [19] have proved that the competitiveness ratio of this routing scheme is $O(\log^2|V|)$ if the dispatch cost incurred in every edge is a concave/sub-additive function of the flow passing through the edge. As Khator *et al.* in [52] formulated the cost function of economic dispatch problem as a concave function, the mentioned claim is true. Additionally, as [20] proposes, a $O(\log^2|V|\log\log|V|)$ competitiveness ratio for the maximum edge congestion of the oblivious routing scheme used in this chapter, the proposed algorithm asymptotically assures that any line in the distribution network has bounded maximum power congestion.

4.5 Case Study and Simulation Results

In order to verify the effectiveness of the proposed oblivious economic dispatch approach, we implement it on IEEE 57-bus test systems [69]. Furthermore, we utilize our method to find the power flow results for a 9-bus test system so that we can validate the results on a simple test system.

4.5.1 9-bus Test Network

First, we implement the proposed $\mathcal{ORED}$ algorithms on a 9-bus test network shown in Figure 4.3 to evaluate the results clearly utilizing the network topology.

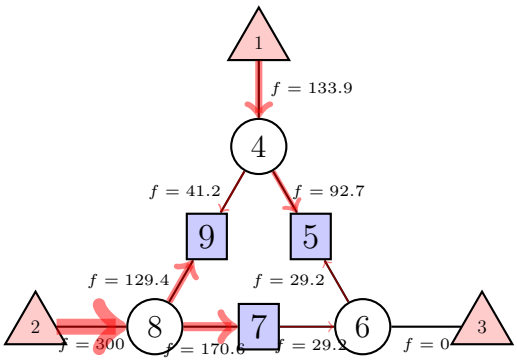
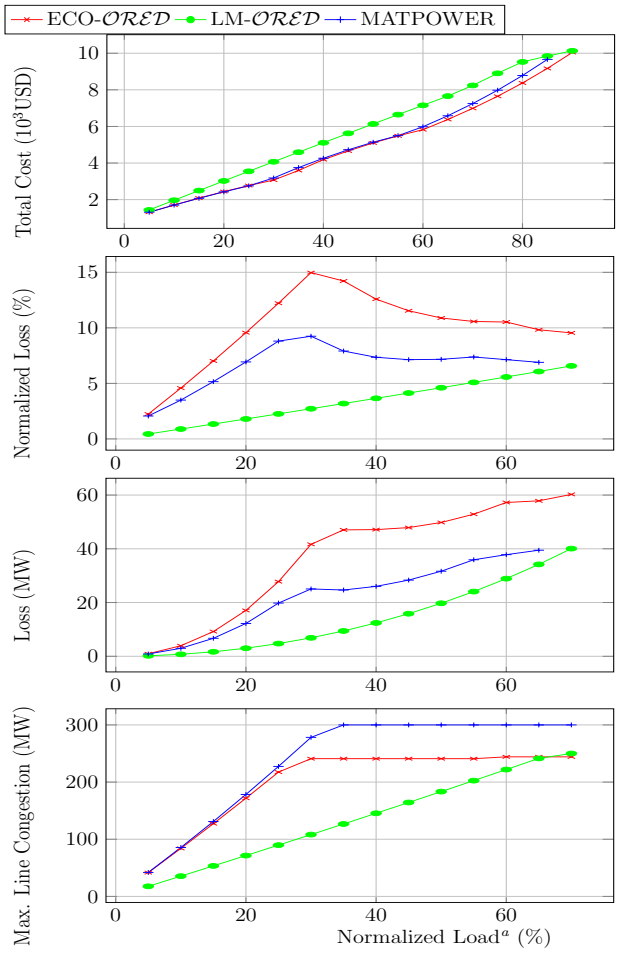


Figure 4.3: Schematic view of the 9-buses network which has three generators/sources (triangular buses) and three sinks (square buses). The obliviously-routed flow has been specified with red arrows on the network connection lines. Buses 5, 7, and 9 have received 121.9, 141.4, and 170.6 MW respectively. Buses 1 and 2 have generated 133.9 and 300 MW respectively; while the dispatched power generation at bus 3 is zero. For the sake of illustration, the energy loss is considered to be zero.

The grid has three generators which supply power for three other busses with different amount of power demands. Figure 4.3 is an example of power flow throughout a given network of communities which delivers the power supplied by the three FFPPs (triangular nodes) to buses 5, 7, and 9 which have 115.5, 122.3, and 152.2 MW power demand respectively. Additionally, the maximum dispatchable capacity of FFPPs are considered to be 250, 300, and 270 MW; while buses 4, 6, and 8 have been assumed to have no demand. Figure 5 depicts the result comparison of three different methods (ECO-$\mathcal{ORED}$, LM-$\mathcal{ORED}$, and MATPOWER [70]) for the aforementioned economic dispatch problem. According to the results of this figure, LM-$\mathcal{ORED}$ outperforms the other two approaches in terms of loss reduction and congestion management. However, this approach leads to higher cost comparing to



[a]Normalized load is a number in $[0, 1)$ and is defined as the total load divided by the total generation capacity. As a portion of generated power is lost in the distribution network, the total normalized load can't reach one. Also, due to network topology and line capacity, the total load can't exceed a certain level.

Figure 4.4: Comparison of the proposed algorithms (green and economic ones) with MATPOWER on the 9-bus test system.

the two others. On the other hand, the results of both ECO-$\mathcal{ORED}$ and MATPOWER are more cost-effective than LM-$\mathcal{ORED}$. This observation validates the effectiveness of our proposed methods where we claimed that both $\mathcal{ORED}$ methods have competent congestion management. Additionally, the plots can be explained by the difference between the objective functions in (4.1) and (4.4).

4.5.2 IEEE 57-Bus Test Network

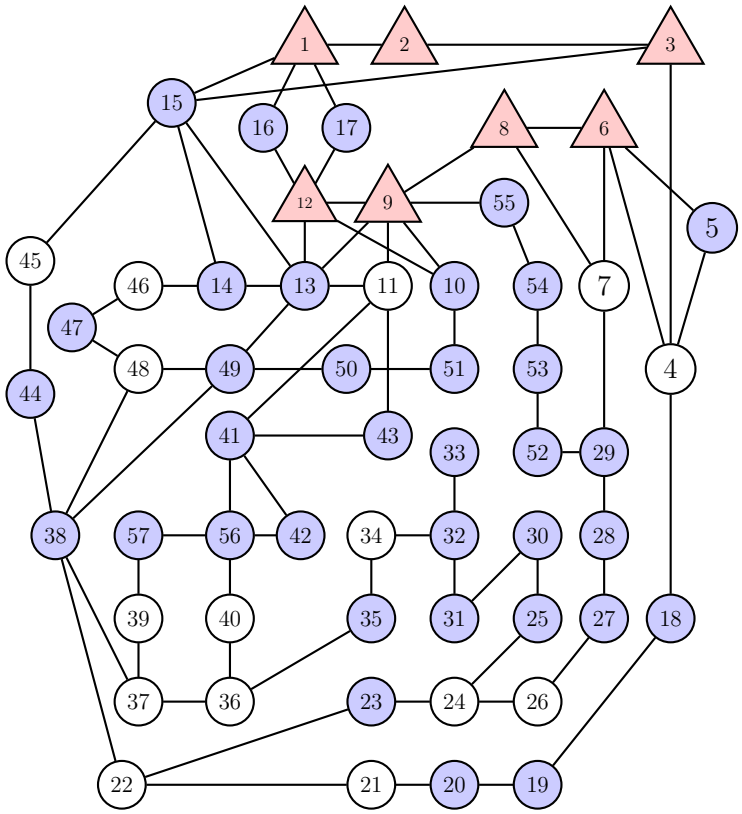


Figure 4.5: IEEE 57-Bus standard test network

Figure 9.3 depicts the graph representation of the IEEE 57-Bus standard test network which is a sample of non-radial electric grids. The colored circles in this figure represent the communities with non-zero demand while triangular nodes are symbols of the 7 generation units in this grid. The simulation procedure that we did in this test system is running the algorithms ECO-$\mathcal{ORED}$ and LM-$\mathcal{ORED}$ for some given demand/generation values and compare the results with the output of MATPOWER software for the same given input. This comparison has illustrated by four plots shown in Figure 4.6.

According the simulation results, the proposed $\mathcal{ORED}$ algorithm outperfomrs the MATPOWER solutions in terms of both minimizing the dispatch cost and pre-

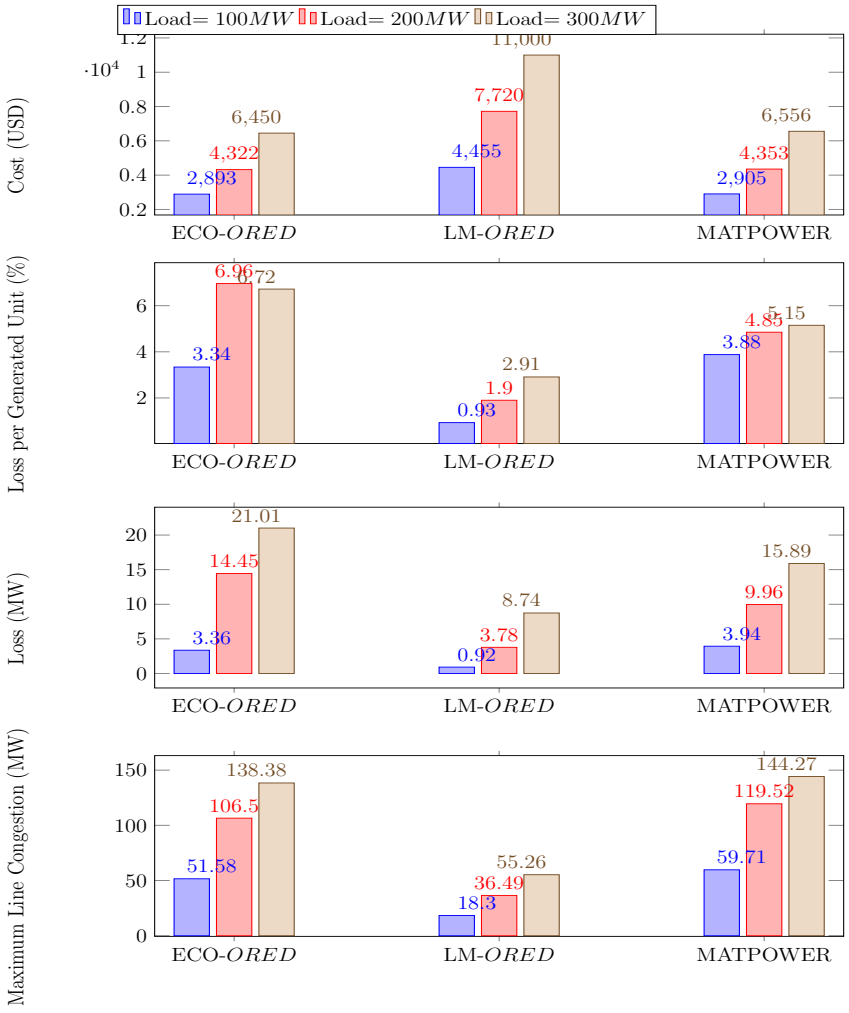


Figure 4.6: Comparison of the proposed algorithms (green and economic ones) with MATPOWER on the IEEE 57-bus test system.

venting the R-edge congestion in transmission lines. Our method has also the versatility of solving the ED problem considering two different approaches, i.e., loss minimizing approach (denoted by LM-$\mathcal{ORED}$) and economic approach (denoted by ECO-$\mathcal{ORED}$)

The plots shown in Figure 4.6 experimentally points out that by increasing the load, the difference between the generation capacity and total demand is reduced which makes the algorithms not to have various options to choose in order to dispatch the electricity and all of the three studied algorithms (MATPOWER, LM-$\mathcal{ORED}$

and ECO-$\mathcal{ORED}$) come up with more similar results (total cost, loss, and edge congestion) and eventually, they converge to the same value, i.e., we have a fully-congested network.

Furthermore, the plots show that there exists a trade off between the generation cost and total loss which is proportional to gas emission and air pollution. The ECO-$\mathcal{ORED}$ has a very competent total cost (better than what MATPOWER proposes) while the energy loss is not as low as what MATPOWER offers. This means that ECO-$\mathcal{ORED}$ compromises the gas-emission minimization in order to reach a more economic solution. This trade-off happens in reverse direction for the LM-$\mathcal{ORED}$ where the total energy loss is substantially lower than MATPOWER; but the generation cost is not as low as what MATPOWER suggests. The very important point here is that the maximum edge-congestion in both $\mathcal{ORED}$ algorithms are less than what we see in MATPOWER. As we see in Figure 5, the recent claim is true for any power load rate.

By comparing the plots of Figure 5 and 4.6, it is clear that the $\mathcal{ORED}$ algorithms generally work better when the graph is more dense. The reason is that for each two arbitrary nodes in the graph there are more available paths to dispatch the power through when the network distribution is more dense, i.e., the proposed $\mathcal{ORED}$ algorithms have more alternatives to decrease the edge-congestion.

4.6 Summary and Conclusion

In this chapter, we introduced a novel economic dispatch algorithm for distribution of generated power and congestion management in smart distribution networks. The proposed $\mathcal{ORED}$ method is inspired by the oblivious network design which can be utilized for large-scale networks with time-varying topology and dynamic source-

to-sink flows. In our model, there are several feeders which have been designed to provide electricity for communities of customers and are connected via a smart power distribution network spread over a city. Two different optimizers have been explained in this chapter which both are based on oblivious network routing schemes: (1) ECO-$\mathcal{ORED}$ which proves to dispatch the electric power with competent generation cost; it also assures that the total power loss doesn't asymptotically exceed a higher-threshold comparing to the best possible solution; (2) ENV-$\mathcal{ORED}$ which dispatches the electric power with competent power loss while guaranteeing that the total generation cost doesn't asymptotically exceed a high-threshold times the cost incurred by the best solution. Both algorithms were experimentally tested on IEEE standard 57-bus system and shown to have superior performance over the common tool for solving ED (MATPOWER). The proposed algorithms work for every network topology either radial or non-radial. Moreover, it is theoretically guaranteed that the algorithms dispatch power in a way that the maximum congestion of power transmission lines doesn't exceed an asymptotic poly-logarithmic high-threshold.

Bibliography

[1] S. Kar, G. Hug, J. Mohammadi, and J.M.F. Moura, "Distributed State Estimation and Energy Management in Smart Grids: A Consensus+ Innovations Approach," *IEEE Transactions on Selected Topics in Signal Processing* , vol. 8, no. 6, 1022 - 1038, Dec. 2014.

[2] U.S. Department of Energy, *The Smart Grid: An Introduction*, 2008.

[3] A. Zidan and E. F. El-Saadany, "A cooperative multi-agent framework for self-healing mechanisms in distribution systems" , *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1525–1539, 2012.

[4] R. E. Brown, "Impact of Smart Grid on distribution system design" , in *Proc. IEEE Power and Energy Society General Meeting*, pp. 1–4, Pittsburgh, PA, July 2008.

[5] B. H. Chowdhury and S. Rahman, "A review of recent advances in economic dispatch," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1248-1259, Nov. 1990.

[6] Vijay Pappu, Marco Carvalho, and Panos Pardalos. *Optimization and Security Challenges in Smart Power Grids*. Springer, 2013.

[7] J. Zhu. *Optimization of power system operation*. John Wiley & Sons, 2014.

[8] N. Padhy, "Unit commitment-a bibliographical survey" , *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 1196–1205, 2004.

[9] A. I. Selvakumar and K. Thanushkodi, "A New Particle Swarm Optimization Solution to Nonconvex Economic Dispatch Problems," *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 42-51, Feb. 2007.

[10] M.H. Amini, B. Nabi, and M. -R. Haghifam, "Load management using multi-agent systems in smart distribution network," in *Proc. IEEE Power and Energy Society General Meeting*, pp. 1–5, Vancouver, BC, Canada, July 2013.

[11] J. Lavaei, D. Tse, and B. Zhang, "Geometry of power flows and optimization in distribution networks," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 572-583, Mar. 2014.

[12] M.J. Sanjari, H. Karami, and H. B. Gooi."Micro-generation dispatch in a smart residential multi-carrier energy system considering demand forecast error." *Energy Conversion and Management*, vol. 120, pp. 90–99, 2016.

[13] E. Kellerer and F. Steinke, "Scalable Economic Dispatch for Smart Distribution Networks," *IEEE Transactions on Power Systems*, no. 99, pp. 1-8, Sep. 2014.

[14] M. Carrion and J. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem", *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1371–1378, Aug. 2006.

[15] A. Botterud, Z. Zhi, J. Wang *et al*, "Demand Dispatch and Probabilistic Wind Power Forecasting in Unit Commitment and Economic Dispatch: A Case Study of Illinois", *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 250–261, Oct. 2012.

[16] Z. Li, Q. Guo, H. Sun, and J. Wang, 'Sufficient Conditions for Exact Relaxation of Complementarity Constraints for Storage-Concerned Economic Dispatch", *IEEE Transactions on Power Systems*, no. 99, pp. 1–2, Apr. 2015.

[17] S. K. Khator and L.C. Leung, "Power distribution planning: a review of models and issues," *IEEE Transactions on Power Systems*, vol.12, no.3 , pp. 1151–1159, Aug. 1997.

[18] Yi Liu, Naveed Ul Hassan, Shisheng Huang, and Chau Yuen, "Electricity Cost Minimization for a Residential Smart Grid with Distributed Generation and Bidirectional Power Transactions," *IEEE Innovative Smart Grid Technologies (ISGT)*, pp. 1–6, Washington, DC, USA, Feb. 2013.

[19] A. Papavasiliou and S. S. Oren, "Supplying Renewable Energy to Deferrable Loads: Algorithms and Economic Analysis, " in *Proc. IEEE Power and Energy Society General Meeting*, pp. 1–8, Minneapolis, MN, USA, July 2010.

[20] K. G. Boroojeni *et al*, "Optimal Two-Tier Forecasting Power Generation Model in Smart Grids," *International Journal of Information Processing* , vol. 8, no. 4, pp. 79-88, 2014.

[21] S. S. Iyengar and K. G. Boroojeni. *Oblivious Network Routing: Algorithms and Applications*. MIT Press, 2015.

[22] A. Gupta, M. T. Hajiaghayi, and H. Racke, "Oblivious network design," in SODA 06: Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm. New York, NY, USA: ACM, 2006, pp. 970979.

[23] A. Maknouninejad, W. Lin, H. G. Harno, Z. Qu, and M. A. Simaan, "Cooperative control for self-organizing microgrids and game strategies for optimal dispatch of distributed renewable generations," Energy Systems, March 2012, Vol. 3, No. 1, pp 23-60.

[24] M. Tuffaha and J. T. Gravdahl, "Discrete state-space model to solve the unit commitment and economic dispatch problems," Energy Systems, May 2016, pp. 1-23, in Press.

[25] C. Nolden, M. Schönfelder, A. Eßer-Frey, V. Bertsch, W. Fichtner, "Network constraints in techno-economic energy system models: towards more accurate modeling of power flows in long-term energy system models," Energy Systems, September 2013, Vol. 4, No. 3, pp 267-287.

[26] S. Rebennack, Bruno Flach, Mario V. F. Pereira, and Panos M. Pardalos, "Stochastic hydro-thermal scheduling under CO2 emissions constraints," IEEE Transactions on Power Systems, Jan 2012 Vol. 27, No. 1, pp 58-68.

[27] J. Shortle, et al. "Transmission-capacity expansion for minimizing blackout probabilities." IEEE Transactions on Power Systems, January 2014, Vol. 29, No. 1, pp 43-52.

[28] A. Tucker, "A note on convergence of the Ford-Fulkerson flow algorithm." Mathematics of Operations Research, Vol. 2, No. 2, pp.143-144, 1977.

[29] A. Kumar, A. Gupta, and T. Roughgarden, "Simpler and Better Approximation Algorithms for Network Design," in Proc. of the 35th STOC, 2003, pp. 365-372.

[30] Yair Bartal, "On Approximating Arbitrary Metrics by Tree Metrics," in Proc. of the 30th STOC, 1998, pp. 161-168.

[31] F. S. S Alman, J. C Heriyan, R. R Avi, and S. S Ubramanian, Approximating the single-sink link-installation problem in network design, SIAM J. Optimization, 11 (2000), pp. 595-610.

[32] D. R. K Arger and M. M Inkoff, Building Steiner trees with incomplete global knowledge, in Proc. of the 41st FOCS, 2000, pp. 613-623.

[33] S. G Uha, A. M Eyerson, and K. M Ungala, Hierarchical placement and network design problems, in Proc. of the 41st FOCS, 2000, pp. 603-612.

[34] S. G Uha, A. M Eyerson, and K. M Ungala, A constant factor approximation for the single sink edge installation problem, in Proc. of the 33rd STOC, 2001, pp. 383-388.

[35] A. M Eyerson, K. M Ungala, and S. A. P Lotkin, Cost-distance: Two metric network design, in Proc. of the 41st FOCS, 2000, pp. 624-630.

[36] S. Rebennack, Artyom Nahapetyan, and Panos M. Pardalos. "Bilinear modeling solution approach for fixed charge network flow problems," Optimization Letters, July 2009, Vol. 3, Issue 3, pp 347-355.

[37] Srivathsan Srinivasagopalan, Costas Busch, and S.S. Iyengar, "An Oblivious Spanning Tree for Single-Sink Buy-at-Bulk in Low Doubling-Dimension Graphs", IEEE Transactions On Computers, Vol. 61, No. 5, MAY 2012.

[38] A. Kumar, A. Gupta, and T. Roughgarden, "Approximations via Cost-Sharing: A Simple Approximation Algorithm for the Multicommodity Rent-or-Buy Problem," in Proc. of the 44th FOCS, 2003, pp. 606-615.

[39] N. Garg, R. Khandekar, G. Konjevod, R. Ravi, F. S. Salman, and A. Sinha, "On the Integrality Gap of a Natural Formulation of the Single-Sink Buy-at-Bulk Network Design Formulation," in Proc. of the 8th IPCO, 2001, pp. 170-184.

[40] M. Charikar and A. Karagiozova, "On Non-Uniform Multicommodity Buy-at-Bulk Network Design," in Proc. of the 37th STOC, 2005, pp. 176-182.

[41] K. T Alwar, "Single-Sink Buy-at-Bulk LP Has Constant Integrality Gap," in Proc. of the 9th IPCO, 2002, pp. 475-486.

[42] A. Kumar, A. Gupta, and T. Roughgarden, "A Constant-Factor Approximation Algorithm for the Multicommodity Rent-or-Buy Problem, in Proc. of the 43rd FOCS, 2002, pp. 333-342.

[43] B. A Werbuch and Y. A Zar, Buy-at-bulk network design, in Proc. of the 38th FOCS, 1997, pp. 542-547.

[44] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: a bibliographic survey I: Formulations and deterministic methods," Energy Systems, September 2012, Vol. 3, No. 3, pp 221-258.

[45] S. Frank, I. Steponavice, and S. Rebennack, "Optimal power flow: a bibliographic survey II: Non-deterministic and hybrid methods," Energy Systems, September 2012, Vol. 3, No. 3, pp 259-289.

[46] S. Frank and S. Rebennack, "An Introduction to Optimal Power Flow: Theory, Formulation, and Examples," IIE Transactions, to appear (2016).

[47] M. H. Amini, et al. "Distributed security constrained economic dispatch," IEEE Innovative Smart Grid Technologies-Asia (ISGT ASIA), 2015.

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

[49] J. Fakcharoenphol, S. B. Rao, and K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, in Proc. of the 35th STOC, 2003, pp. 448-455.

[50] MATLAB version 8.5. Miami, Florida: The MathWorks Inc., 2015.

[51] (2015) Power systems test case archive. [Online]. Available: http://www.ee.washington.edu/research/pstca/

[52] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education," *Power Systems, IEEE Transactions on*, vol. 26, no. 1, pp. 12-19, Feb. 2011.

CHAPTER 5

A NOVEL COULD-BASED PLATFORM FOR IMPLEMENTATION OF OBLIVIOUS POWER ROUTING FOR CLUSTERS OF MICROGRIDS

There has been an increasing demand for connectivity of the clusters of microgrids to increase their flexibility and security. This chapter presents a framework for implementation, simulation, and evaluation of a novel power routing algorithm for clusters of microgrids. The presumed cluster is composed of multiple direct current (DC) microgrids connected together through multi-terminal DC (MTDC) system in a meshed network. In this structure, the energy is redirected from the microgrid with excessive power generation capacity to the microgrid which has power shortage to supply its internal loads. The key contribution of this study is that each microgrid in the cluster is unaware of the current state and other flows of the cluster. In this approach, the optimal power flow problem is solved for the system, while managing congestion and mitigating power losses. The proposed methodology works for both radial and non-radial networks regardless of the network topology, scale and number of microgrids involved in the cluster. Therefore, it is also well suited for large-scale optimal power routing problems that will emerge in the future clusters of microgrids. The effectiveness of the proposed algorithm is verified by Matlab simulation. We also present a comprehensive cloud-based platform for further implementation of the proposed algorithm on the OPAL-RT real-time digital simulation (RTDS) system. The communication paths between the microgrids and the cloud environment can be emulated by OMNeT++.

[0]Part of this chapter has been reprinted with permission from Kianoosh G. Boroojeni, et al., "A Novel Cloud-based Platform for Implementation of Oblivious Power Routing for Clusters of Microgrids," IEEE Acess, vol. 5, doi: 10.1109/ACCESS.2016.2646418, 2016.

5.1 Introduction

5.1.1 Motivation

Smart microgrids can be defined as a new generation of smart power networks that incorporate actions from all connected end-users. Many researchers have investigated different aspects of microgrids, including their penetration into electric power system, integration issues of distributed energy resources (DERs), role of power electronics, stability and reliability of microgrids [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. Although all of these investigations and their associated models are very useful in understanding the performance and operation of the system, little or no efforts have been done in co-simulation of the communication network and power systems. The communication platform is an essential factor, which is unexploited in most of researches in the areas related to microgrids, such as [2]. The role of information and communication technology (ICT) and the required infrastructures in the future power systems simulation is investigated in [11].

In [3], a comprehensive energy management system for off-grid and on-grid microgrids is described. The proposed approach is very competent when the power system under test is operating in steady state mode. However, disregarding the transient response of power systems may lead to the instability and cascading failures of the network. In an efficient energy management system, there is a bidirectional flow of data between the DERs. Therefore, communication delays added to the transient conditions may deteriorate the stability margin of the network in the transient period. In [4, 5, 6], a state-of-the-art planning method is proposed for AC and DC microgrids. This method investigates the economic viability of microgrids deployment and optimal generation of DERs. Khodaei *et al.* in [6], addressed the microgrid planning under uncertainty. The cooperative control of power electronics

converters within a power network consisting of several microgrids and sub-grids is investigated in [7, 8, 9]. The proposed algorithms can provide a robust optimization approach for the planning and operation of the old fashioned AC and DC microgrids. However, the advent of cloud computing has brought the networking and optimization principals to the next era, which is a highly integrated centralized controllers within a microgrids or amongst multiple interconnected microgrids in a cluster.

There has been an increasing demand for connectivity of the microgrids to establish a secure cluster of power networks. The idea of self-sustaining energy islands that can stay on even during grid-wide blackouts is of obvious value to entities and consumers that cannot let power outages keep them from performing their missions. In a typical microgrid system, two main types of uncertainty affect the reliable and secure operation: outage of the generation units and deviation from the forecasts [13, 12]. The outage of generation units can lead to a supply shortage in system, which is usually met from both spinning and non-spinning operating reserves. Departures from the forecasts resulting from uncertain loads and the integration of intermittent sources of energy, i.e. DERs introduce additional operational uncertainty. Wind and solar generation depend on wind speed and solar irradiance, respectively, which are difficult to predict accurately. Indeed, the effects of the uncertainty associated with wind and solar generation are more significant in microgrids due to the high penetration levels of such resources and typically small intertia. There are numerous reports of a variety of methods that take into account the uncertainties arising from load, wind, and solar power forecasting errors [14, 13, 15]. These uncertainties may lead to partial or overal blackout in the migrogrid system. In order to provide more resilience to such uncertainties, the promotion of the clusters of several interconnected microgrids are becoming more popular [16]. The new concept of cluster of microgrids introduced based on the idea of accommodating

the excesses and shortfalls of power between *prosumers* (producers and consumers), which enable the system to be able to accept more than 50% of required power from renewable energy without system stabilization and defines a highly secure system that is independent from blackouts.

Since the microgrids can act as both demand side and generator side, power flows among them can be significant. Therefore, the cluster of microgrids may face power congestion issue in some transmission lines connecting microgrids. As a result, it is critical to design power routing algorithms which prevent or mitigate power congestion in clusters of microgrids. To this end, there are some studies such as [17, 18, 19] which have proposed different congestion management techniques.

5.1.2 Related Work

A smart power system requires modern monitoring, analysis and control to deliver the electricity in a more reliable, economical, and sustainable way. Advent of the phasor measurement units (PMUs) and modern supervisory control and data acquisition (SCADA) systems resolved some of the conventional issues related to the monitoring and control of the smart grids. However, without a seamless communication technology, operators (computerized or hand-operated) cannot get an insight in the current grid states and use the monitored data to operate the system effectively [26]. Therefore, there is a need to analyze the performance of the proposed communication algorithm. However, due to multiple types of latency in the communication links, simulation should be considered as an exigent tool to give a realistic insight to the performance of the system. OMNeT++ is a discrete event simulator for modeling the communication network and distributed systems that can be used for modeling and testing the wide-area communication protocols, where the

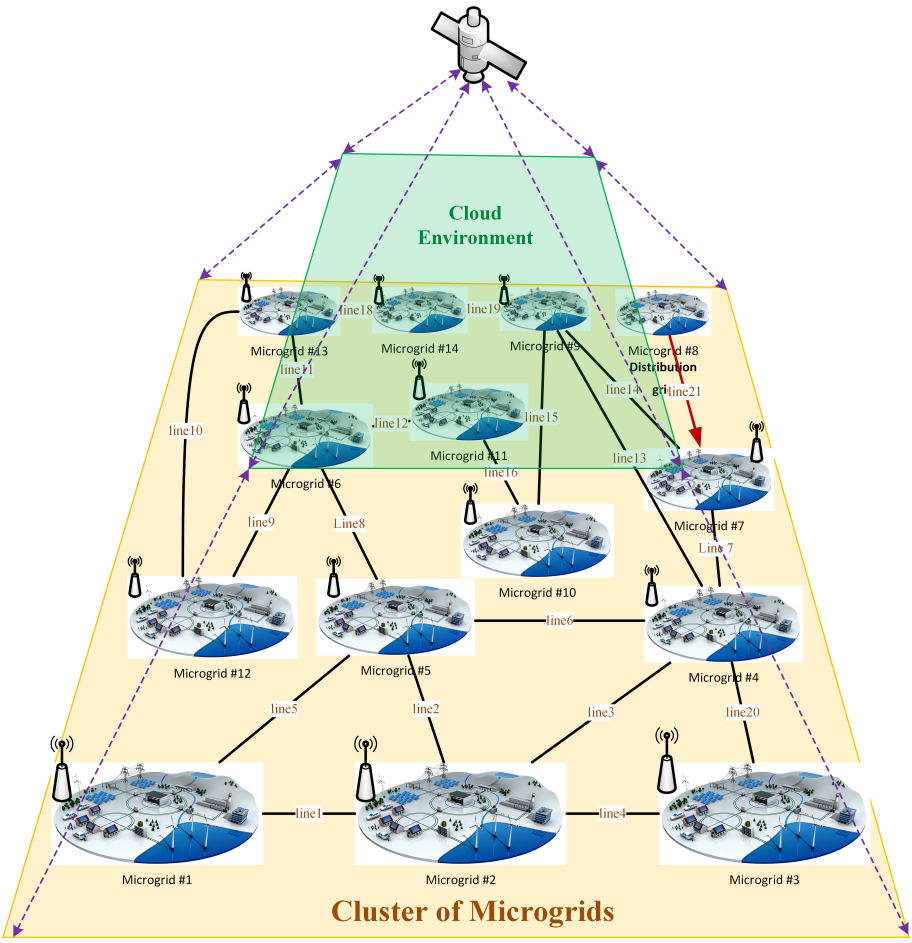


Figure 5.1: Schematic View of the Interconnected Clusters of Microgrids which Communicate Along with a Cloud Environment.

data propagation delays become significant. OMNeT++ was initially introduced in [27] and [28] as a design tool for wired and wireless networks of computer clusters, telecommunications and distributed/parallel systems. OMNeT++ have been used by many researchers to simulate different events on the smart grid [29]. In [30], OMNeT++ has been used to simulate the cyber layer of the smart grid. In [31, 32], a methodology for co-simulation of OMNeT++ and OpenDSS (electric power Distribution System Simulator) has been proposed, in which OMNeT++ and OpenDSS are running in parallel and the events are synchronized at certain time slots. The authors in [33] have utilized the OMNeT++ to analyze the measurements from a

real testbed. NREL (National Renewable Energy Laboratory) have utilized OMNeT++ as a network simulator-in-the-loop, which simulates the network and links with real computers and virtual hosts [34]. OMNeT++ platform was later used in [35, 36, 37, 38, 39, 40, 41] to model and simulate their proposed communication protocols. This helps to ensure the proposed methodology can be utilized in the real world applications.

The key components of microgrid include loads, distributed energy resources, master controller, smart switches, and protective devices. Furthermore, communication networks, control and automation systems play a pivotal role in microgrid operation [1]. In our application, due to the fact that the microgrids within a same cluster may be located hundreds of meters away from each other, the instant communication between the nodes (microgrids) is required to ensure seamless power flow between the nodes. Therefore, OMNeT++ is used to check on the practicality of our proposed method. The schematic view of the clusters of microgrids along with the cloud environment is depicted in Fig. 5.1.

Economic Dispatch (ED) is one of the most studied optimization problems in power systems and microgrids operation [20, 21, 22, 23, 24, 25]. The goal of ED is to obtain the efficient scheduling of the clusters of microgrids. Furthermore, in large-scale power systems, we need to deal with the unit commitment (UC) problem which is solved on a day-ahead basis. In this context, ED is normally solved at each hour using the results of UC as the main input[42, 43]. The nonconvex ED problem is solved utilizing a novel particle swarm optimization proposed by [44]. Amini *et al.* in [45] proposed a load manamgement scheme for the smart distribution networks. A graph-based modeling of the power flow problem is introduced in [46]. Several studies explored ED problems [47, 48, 49, 50, 51]. Boroojeni et al. [47] present a novel oblivious routing economic dispatch ($\mathcal{ORED}$) algorithm for power systems.

According to [47], $\mathcal{ORED}$ is the first ED algorithm built based on oblivious network design which is the most fit for networks with oblivious sources and destinations while they are unaware of the current network state and other flows. In our study, every microgrid can either send (source) or receive (destination) power through the available DC lines. In [48] and [49], a Mixed Integer Linear Programming (MILP) is employed in order to solve the ED problem. A demand dispatch in a large penetration of wind power is proposed by Botterud *et al.* in [50]. The effect of high penetration of energy storage units under ED problem was investigated in [51]. Yi Liu *et al.* in [53] proposed a two-tier pricing scheme for electricity trading between the agents which are responsible for renewable power resources. They also considered both defferable and non-defferable load demands in their optimization problems.

Oblivious Network Routing Design: Despite the traditional Minimum Cost Flow Problems (MCFPs) that are defined with a specific set of commodities (with given source, sink, and size) and pre-defined flow cost in each edge as a function of the flow size in that edge, oblivious network routing design solves a set of MCFP problems in which at-least one of the following conditions hold:

- Source, sink, or size of the commodities are not specified in advance, i.e. there is neither deterministic nor stochastic information regarding the commodities of the problem which are going to be routed through the network. In such circumstances, the MCFP is referred to as *commodity oblivious* [19].
- The cost of flowing commodities in an edge cannot be defined as a deterministic or stochastic function of the flow size in the same edge. In such circumstances, the MCFP is referred to as *flow cost oblivious* [20].

Oblivious routing design provides routing solution to the oblivious MCFPs so that the resulting routing scheme is flexible to the obliviousness of the commodities and flow cost functions. Also, oblivious routing schemes mitigate traffic congestion in small subgraphs of the network by distributing the flow throughout the network. These types of routing schemes provide a low-cost flow routing in long term despite the fact that some elements of the MCFP are not defined either deterministically or stochastically. In other words, oblivious routing design is most suited for the networks in which we have little/no knowledge regarding their current and future states [19]. Therefore, it goes well in line with the microgrid concept, as they are largely independent entities by definition.

Oblivious network routing is considered as a heuristic method as it is utilized to expedite the process of finding a sufficiently low-cost solution for the MCFPs with oblivious elements. In fact, since computing the global optimal solution is too complex to be done practically, oblivious network routing design finds a satisfactory solution for the oblivious MCFPs by deploying a practical means not guaranteed to be globally-optimum, but satisfactory enough for achieving immediate goal of finding a near-optimum approximation of the best solution.

Oblivious network routing is different from Adaptive Routing (AR) as AR responds to any change in the unknown elements of the MCFP by recalculating the solution and possibly providing different solution than the one calculated initially. However, oblivious routing employs a single routing scheme for a wide range of obliviousness in the MCFP in order to sufficiently approximate the best solution of the oblivious MCFP [19]. As a result, in highly oblivious circumstances where AR is not computationally-feasible, oblivious routing can solve the MCFP with much less time complexity.

Since prior art research concerning multiple microgrid clusters has by now been carried out only on conceptual level, interaction of communication interference in microgrids is an important direction for future research. To that end, only the most basic low level control functionalities such as DC link voltage regulation and power flow exchange have been investigated[57]. While this approach provides an invaluable base for further research, it also neglects many practical limitations that may occur in the real world. For instance, it considers only a limited number of DC buses and the power exchange among them is designed to equalized states of charge of local energy storage systems (ESSes). Moreover, the connection of the microgrid cluster to the overhead power system is not considered [58]. In the actual physical system, this might not hold as the future power systems are predicted to be comprised of high number of microgrids which are required to exchange energy among themselves. This exchange of energy makes the overhead system subject to many other metrics. For instance, realistic microgrids cannot be considered to comprise a single ESS but a number of prosumers which can actively complement with ESS [59, 60]. Therefore, state of available energy within the particular microgrids is deemed as more realistic metric than a simple State of Energy (SOE) [61].

5.1.3 Our Contribution

This chapter presents a novel cloud-based approach for solving optimal power routing problem in clusters of DC microgrids, utilizing oblivious network routing design. The presumed cluster is composed of multiple DC-microgrids connected together through multi-terminal DC (MTDC) system in a meshed network. In this methodology, the energy is redirected from the microgrid with excessive power generation (high state of energy) capacity to the microgrid which has power shortage to supply

the internal loads. The key contribution of this work is that all of the microgrids in the cluster are unaware of the current cluster state and other flows, i.e. each microgrid optimized its operation. This approach solves the optimal power flow problem, while managing congestion and mitigating power losses. The proposed methodology works for both radial and non-radial networks regardless of the network topology, scale and number of microgrids involved in the cluster. Therefore, it is well-suited for the large-scale economic dispatch problems that will emerge in the future smart distribution grids. The effectiveness of the proposed algorithm has been verified in MATLAB simulation as well as OPAL-RT real-time digital simulation (RTDS) system. The communication path between the microgrids are implemented on a cloud-based environment emulated by OMNeT++. The results verify the superior performance of the proposed method over the current methods in the literature in terms of congestion management and power loss minimization.

5.1.4 Organization of the Chapter

The rest of the chapter is organized as follows. Section II specifies the general overview of the proposed framework. Section III explains how oblivious network design can be utilized in order to solve the optimal power routing problem for clusters of microgrids. Section IV introduces RTDS as a powerful real-time digital power system simulator and addresses how it helps in the process of developing the proposed approach. Section V states the model of the cloud environment in which the proposed method works properly. In Section VI, OMNeT++ is introduced as an effective means for simulating modern communication enabled by several networking protocols. Finally, we present the summary and conclusion of our chapter in Section VII.

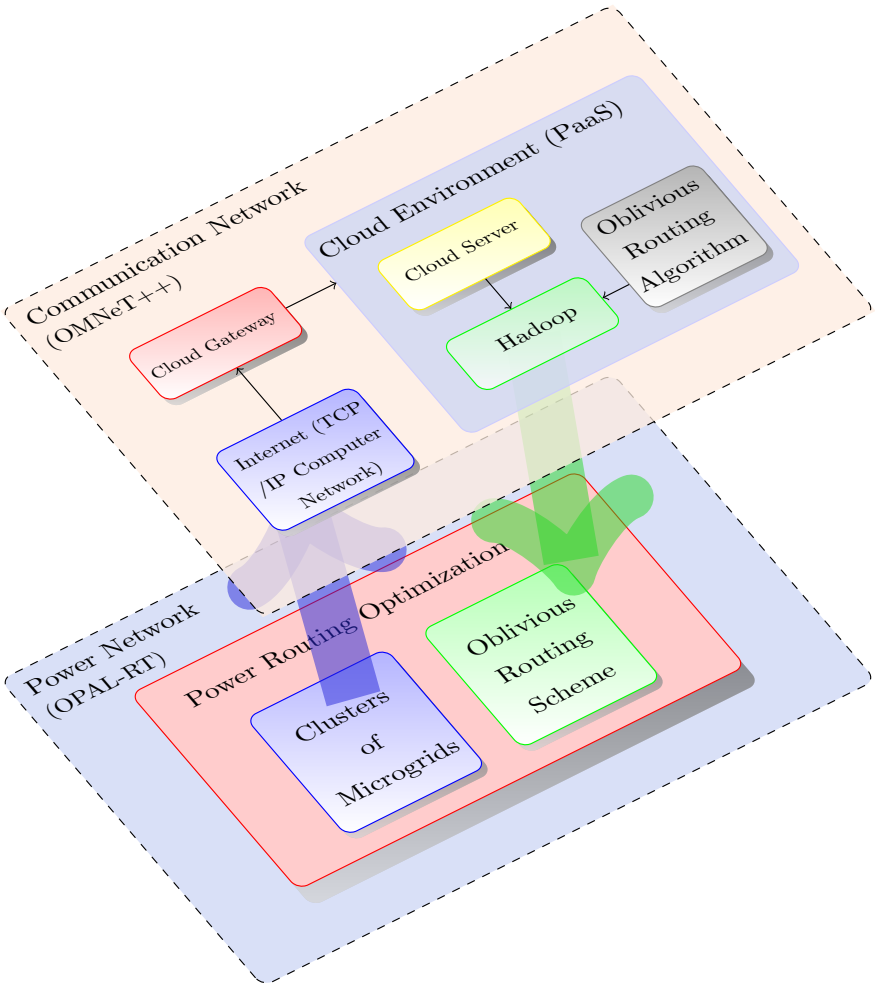


Figure 5.2: The schematic view of microgrids and their corresponding cloud system for communication.

5.2 General Overview of the Proposed Framework

Here, we provide the schematic view of microgrid and it corresponding cloud system for the communication. As Fig. 5.2 illustrates, we classify the elements of our proposed system into two layers: power network layer and communication network layer. In the first layer, we aim to solve the optimal power routing problem for the clusters of microgrids utilizing an oblivious routing scheme provided by the communication layer. In the communication layer, every microgrid is able to communicate

with the cloud environment and send a snapshot of its energy level to the cloud. This communication is performed through the conventional TCP/IP computer network as a subnet of the Internet. After receiving the energy level information of every microgrid, the cloud server utilizes Hadoop and an appropriate oblivious routing algorithm (to be discussed later) in order to obtain an efficient routing scheme which solves the problem of optimal power routing for clusters of microgrids.

5.3 The Proposed Power Routing Method on Clusters of Microgrids

5.3.1 Preliminaries to Oblivious Network Design

First, we explain some basic definitions, assumptions and preliminary data structures needed to construct the oblivious routing scheme, which is responsible for power routing in clusters of microgrids. Assuming that M denotes the set of microgrids and L represents the set of DC power lines connecting neighboring microgrids, we model the topology of the clusters of microgrids utilizing the graph (M, L) of vertex set M and edge set L. Additionally, since every line may have different physical characteristics (in general case), we consider a weighted graph (M, L, w) to formulate the clusters of microgrids, where for every line $l \in L$ connecting the couple of neighboring microgrids m_1 and m_2, $w_l \in \mathbb{R}^+$ is linearly proportional to the amount of power loss flowing from m_1 to m_2 (we address the detailed loss model later in this section). In the rest of the chapter, we consider (M, L, w) as a weighted graph of diameter[1] 2^n (for some integer n) where $w_l > 1$ for every $l \in L$. The

[1]Diameter of a graph is defined as the maximum distance between any pair of nodes in the graph.

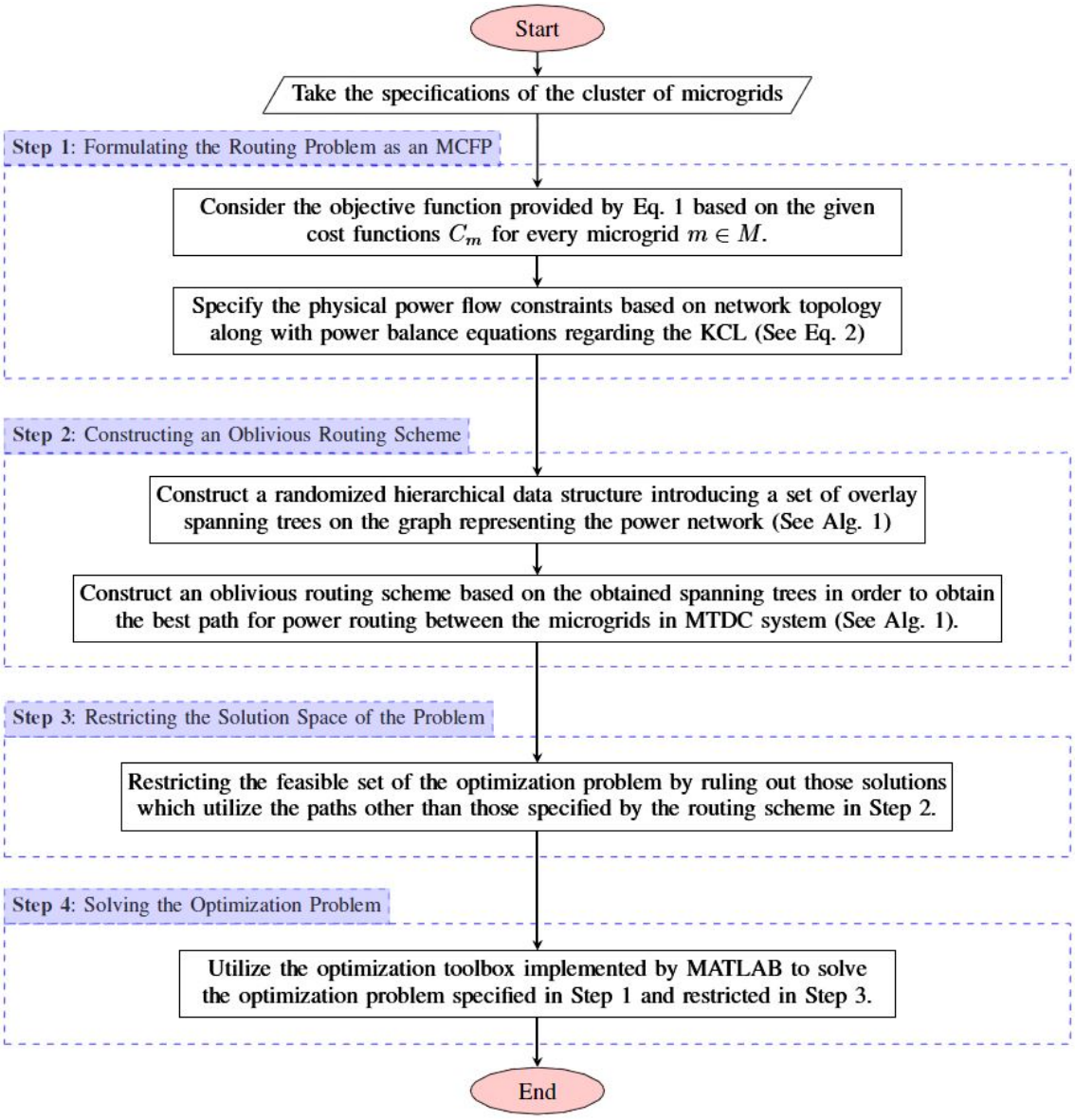


Figure 5.3: Flowchart of the Proposed Oblivious Routing Algorithm for Optimal Power Routing Problem for Clusters of Microgrids.

mentioned conditions on the graph diameter and its weight function don't reduce the generality of our model because any weighted connected graph can be converted to (M, L, w) by linearly scaling its weight function.

Let $\bar{\mathcal{S}}$ denote the microgrid sequence $\bar{\mathcal{S}} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_n)$, where $\mathcal{S}_n = \{M\}$, and for every i less than n, $\mathcal{S}_i$ is a 2^i-partition[2], and for every $i = 0, 1, \ldots, n-1$, if microgrid set $\mathcal{M}$ belongs to $\mathcal{S}_i$, there exists some set $\mathcal{M}' \in \mathcal{S}_{i+1}$ such that $\mathcal{M} \subseteq \mathcal{M}'$. Microgrid set $\mathcal{S}_i$ is called the i^{th} member of sequence $\bar{\mathcal{S}}$. Microgrid v is called α-padded in $\bar{\mathcal{S}}$ $(0 < \alpha < 1)$ if for every i, the i^{th} microgrid partition $\mathcal{S}_i$ contains a subset of microgrids $\mathcal{S}$ such that $\mathcal{S}$ completely covers ball[3] $B_G(v, \alpha \cdot 2^i)$; in other words, for every level i, there exists some $\mathcal{S} \in \mathcal{S}_i$ such that $B_G(v, \alpha \cdot 2^i) \subseteq \mathcal{S}$ [19].

Finally, we define an Overlay Routing Tree (ORT) T based on the microgrid sequence $\bar{\mathcal{S}}$ in the following way: T is an n-level tree where its i^{th} level nodes are the members of $\mathcal{S}_i$ (for every $i = 0, 1, \ldots, n$) and any i^{th}-level tree node $\mathcal{S} \in \mathcal{S}_i$ is connected to its parent $\mathcal{S}' \in \mathcal{S}_{i+1}$ in upper-level $(i+1)$ if and only if $\mathcal{S} \subseteq \mathcal{S}'$. Later in this section, we describe the role of ORT on how each microgrid participates in the process of power routing in the proposed system.

5.3.2 The Proposed Oblivious Routing Algorithm

Fig. 5.3 illustrates the details of our proposed oblivious routing method. In the first step, the power routing problem is formulated as an MCFP with some oblivious elements including the objective function and source/sink nodes. Then, an oblivious routing scheme is constructed in the form of a set of overlay spanning trees on the graph representing the network of microgrids. In this step, we show how Algorithm 4 constructs the oblivious routing scheme (mentioned in [19]). In the third step, we restrict the solution space of the problem by ruling out the routing solutions which

[2] 2^i-partition of a weighted graph is defined as a partition of its vertex set into a number of subsets like $\mathcal{S}$ such that there exist no pair of nodes in $\mathcal{S}$ with distance of more than 2^i from each other.

[3] Set $B_G(v, r) \in V$ is called a ball of center $v \in V$ of radius r if $u \in B_G(v, r)$ iff $d_G(u, v) \leq r$.

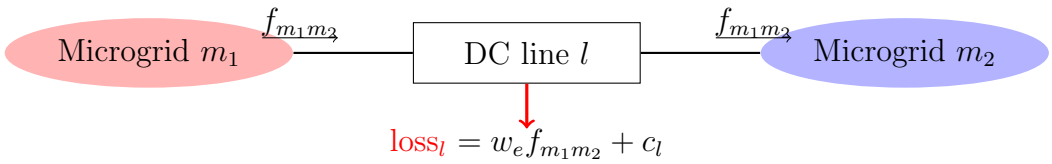


Figure 5.4: Power loss model for DC line connecting two neighboring micorgrids [47].

utilize the paths other than those specified by the previously constructed routing scheme. In the last step, we utilize the optimization toolbox of MATLAB in order to solve the MCFP specified in the first step and restricted in the third one.

Formal Specification of Power Routing Problem

Optimal power routing for cluster of microgrids $M = \{m_1, m_2, \ldots, m_n\}$ is the problem of specifying the generation level of microgrids to satisfy their load demands while minimizing the generation cost. Note that we don't consider the apparent power $S_{m_1m_2}$ of a line between two neighboring micorgrids m_1 and m_2 as the optimization parameters; instead, we only consider the power flow magnitude $f_{m_1m_2} = \mathcal{R}\{S_{m_1m_2}\}$ in the connecting lines between microgrids. In addition, we consider the following model for the power loss in each line connecting microgrids m_1 and m_2 [70]:

$$\text{loss}_{m_1m_2} = w_{m_1m_2} f_{m_1m_2} + c_{m_1m_2};$$

where $w_{m_1m_2}(c_{m_1m_2})$ denote the resistance (constant loss) of the line connecting m_1 and m_2 (see Fig. 5.4 for illustration).

The problem of optimizing power routing for clusters of microgrids can be formally modeled in the following way:

$$min_{f_{.,.}} \Big\{ \sum_{m \in M} C_m(-\sum_{r \neq m} f_{r,m}) \Big\} \tag{5.1}$$

subject to

$$\begin{cases} (i) & -D_m \leq \sum_{r \neq m} f_{r,m} \leq S_m \\ (ii) & f_{m_1 m_2} + f_{m_1 m_2} + w_{m_1 m_2} f_{m_1 m_2} + c_{m_1 m_2} = 0 \\ & \forall \text{ neighboring microgrids } m_1 \text{ and } m_2 \\ (iii) & f_{m_1 m_2} = 0, \text{ if } \{m_1 m_2\} \text{ are not neighbors} \\ (iv) & |f_{m_1 m_2}| \leq \mathcal{L}_{m_1 m_2} \end{cases} \tag{5.2}$$

where C_m denotes the generation cost function for microgrid m, D_m and S_m denote the high-threshold demand and supply of microgrid m (respectively), and $\mathcal{L}_{m_1 m_2}$ represents the capacity of line connecting m_1 and m_2.

Constructing Oblivious Routing Scheme

Algorithm 4 gets the weighted graph $\mathcal{G} = (M, L, w)$ as its input and returns function $\mathbb{S} : M^2 \mapsto \mathcal{P}(L)$ which represents the oblivious routing scheme and specifies a path[4] in graph $\mathcal{G}$ for every pair of source-sink microgrids. The algorithm starts with generating $27 \log |M|$ random microgrid sequences $\{\mathcal{S}^{(k)}_{0 \ldots n-1}\}_{k=1}^{27 \log |M|}$ utilizing

[3]Consider γ_l as the shortest path between the incident nodes of arbitrary edge $l \in p$ in graph G. The projection of tree path p on the graph is obtained by concatenating all of the shortest paths γ_ls back to back. In the case that the concatenation result is not a simple path and has crossed some nodes more than once, the projected path will be the shortest simple path corresponding to the concatenation result.

[4]In this chapter, path of a graph is considered as a simple path and represented by a subset of edge set L such that there exist a permutation of edges in a path where the first edge is incident to the start node of the path, each two consecutive edges are incident to a common node, and the last edge is incident to the end node of the path.

the lines 3-16 of Algorithm 4. The reason for generating multiple random microgrid sequences is to assure the existence of at-least one sequence $\bar{S}$ for every pair of mircrogrids in M. In fact, Iyengar *et al.* in [19] proved that the probability of existing at-least one α-padding sequence among the $c \log |V|$ sequences generated by Algorithm 4 is more than $1 - e^{-\frac{(c-2)^2}{2c}}$ (for every $\alpha \leq 1/8$). By considering $c \geq 27$, the mentioned probability would be greater than $1 - 10^{-5}$ which provides a reasonable theoretical guarantee that Algorithm 4 will find at-least one α-padding sequence for every source-sink pair of microgrids.

After creating $27 \log |V|$ random sequences and their corresponding ORTs (in line 17), Algorithm 4 runs its main loop in lines 19 to 24 for every pair of source-sink nodes. Assume T as the ORT corresponding to an α-padding sequence of microgrids. Tree T would have a pair of leaves (level-zero nodes) corresponding to the source and sink (as $\mathcal{G}$ is supposed to be a weighted graph of the weight function greater than one for every edge). Also, let p denote the only path in ORT connecting the mentioned leaves together. The routing scheme is computed in line 25 where the path between source and sink nodes $\mathbb{S}(source, sink)$ is obtained by projecting the overlay path p on graph $\mathcal{G}$.

Restricting the Solution Space of the Problem

In the third step, we restrict the feasible set of the optimization problem (defined by Eq. 1 and 2) by ruling out those solutions which utilize the paths (power lines) other than those specified by the routing scheme $\mathbb{S}$ (obtained by Alg. 1).

Table 5.1: Comparison of the Proposed Algorithms with MATPOWER on the Designed 14-Bus Microgrid Test System.

Cost (USD)			Loss per Generated Unit (%)			Loss (MW)		
Load	Proposed work	MATPOWER	Load	MATPOWER	Proposed work	Load	MATPOWER	Proposed work
$100MW$	3,073	2,936	$100MW$	3.92	3.76	$100MW$	3.92	3.76
$200MW$	4,210	4,278	$200MW$	4.77	4.72	$200MW$	9.54	9.44
$500MW$	10,332	10,823	$500MW$	5.10	5.16	$500MW$	25.5	25.80

Solving the Problem on a Multi-Terminal DC Test System including the Cluster of 14 Microgrids

Fig. 5.1 depicts the topology of a designed 14-MTDC test network which is a sample of non-radial electric grids. Our simulation procedure on this test network is implementing the oblivious power routing algorithm presented in Alg. 1 for some given demand/supply values and compare the results with the output of MATPOWER[5] software for the same given input. This comparison has illustrated by three plots shown in Fig. 5.5. Also, the detailed values of the obtained resluts for the total cost (in USD) and loss (in MW) are specified in Table 9.1. Also, we applied our proposed routing scheme on an MTDC system with a large-scale non-radial topology inspired by the IEEE-118 bus standard test system [71]. To design the simulation platform, we modified the IEEE-118 bus standard test system by replacing both the demand and generation buses with microgrids of random-valued (possitive and negative) SOEs. The obtained results is compared with the output of MATPOWER software for the same given input. This comparison is illustrated by the two plots shown in Fig. 5.6.

[5]MATPOWER is a package of MATLAB®M-files for solving power flow and optimal power flow problems [70].

Source: http://www.pserc.cornell.edu/matpower/

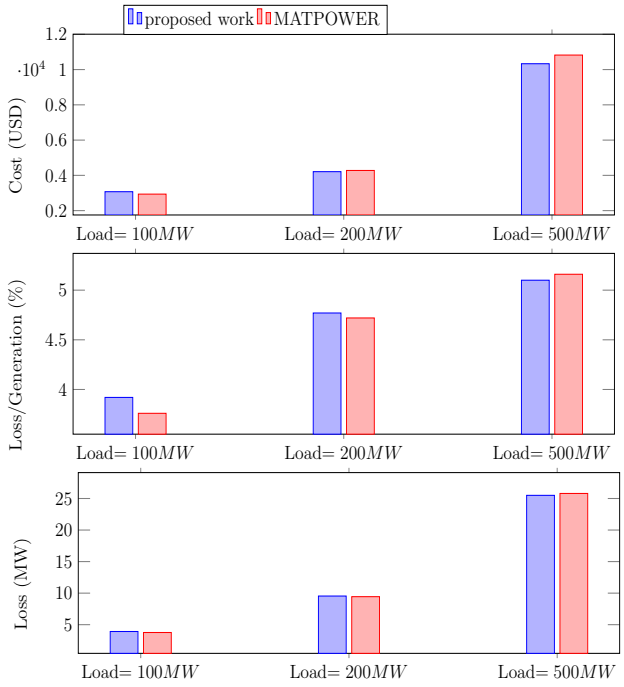


Figure 5.5: Comparison of the proposed algorithm with MATPOWER on the designed 14-bus microgrid test system.

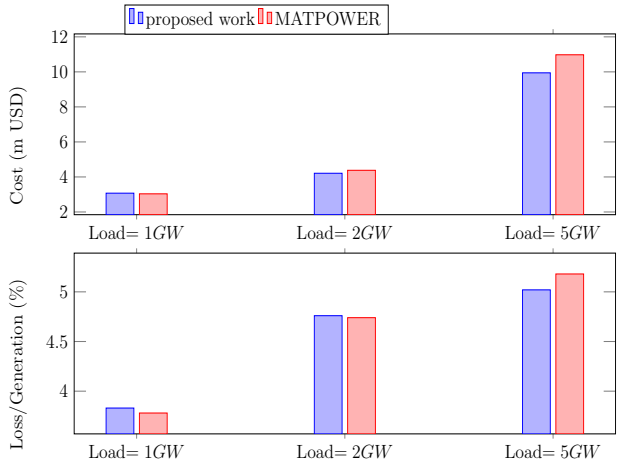


Figure 5.6: Comparison of the proposed algorithm with MATPOWER on the designed microgrid system based on IEEE-118 bus standard test system.

5.4 Real-Time Digital Power System Simulator

Communication interface is of instrumental importance in clusters of microgrids [62].

However, practical communication interfaces are characterized by inherent latencies

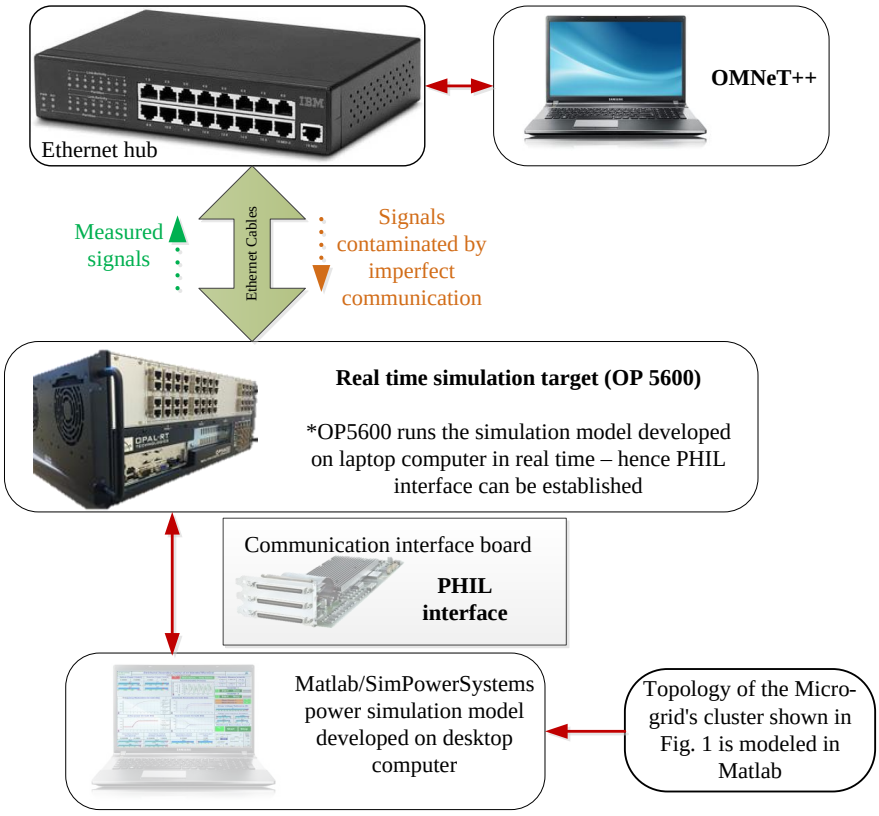


Figure 5.7: General structure of OPAL-RT implementation

and limited bandwidths [63]. This means that in realistic situations, we cannot expect that the control algorithms, which are normally designed in continuous time domain (or in the some cases, designed based on synchronized discrete time domain), behave as expected in the reality. While there has been tremendous amount of attention devoted to analytical investigation of the impacts of communication delays on modern power systems in the automatic control research community, derived methods are overly complicated and unapproachable in practical scenarios [64]. Therefore, real time simulation plays an indispensable role in analysis and design of future cyber-physical systems. OPAL-RT[6] eMEGASim is the cutting-edge simulator which integrates distributed processing software and hardware platforms for high speed and real-time simulation of electromagnetic transients [65]. Furthermore, it comprises fully customized I/O channels which allow seamless integration of physical hardware within the simulation loop or a third party software to emulate certain parts of the system. This latter functionality will be utilized in this chapter to integrate OMNeT++, which emulates communication protocols.

Fig. 5.7 represents the general schematic overview of the proposed real time simulation (RTS) platform. According to this platform, the topology of each cluster can be modeled in OPAL-5600 system. Also, the communication network is simulated in OMNeT++ on a separate computing system. Each microgrid within the cluster is assigned to one of the Ethernet ports of the OPAL interface to emulate the input and output gates of each microgrid. All the Ethernet wires are connected to the OM-

[6]OPAL-RT is the world leader in the development of PC/FPGA Based Real-Time Digital Simulators, Hardware-In-the-Loop (HIL) testing equipment and Rapid Control Prototyping (RCP) systems. Their systems are utilized to design, test and optimize control and protection systems for power systems, power electronics, motor drives, automotive, railway, aircraft and industries, as well as R&D centers and universities. Source: http://www.opal-rt.com/

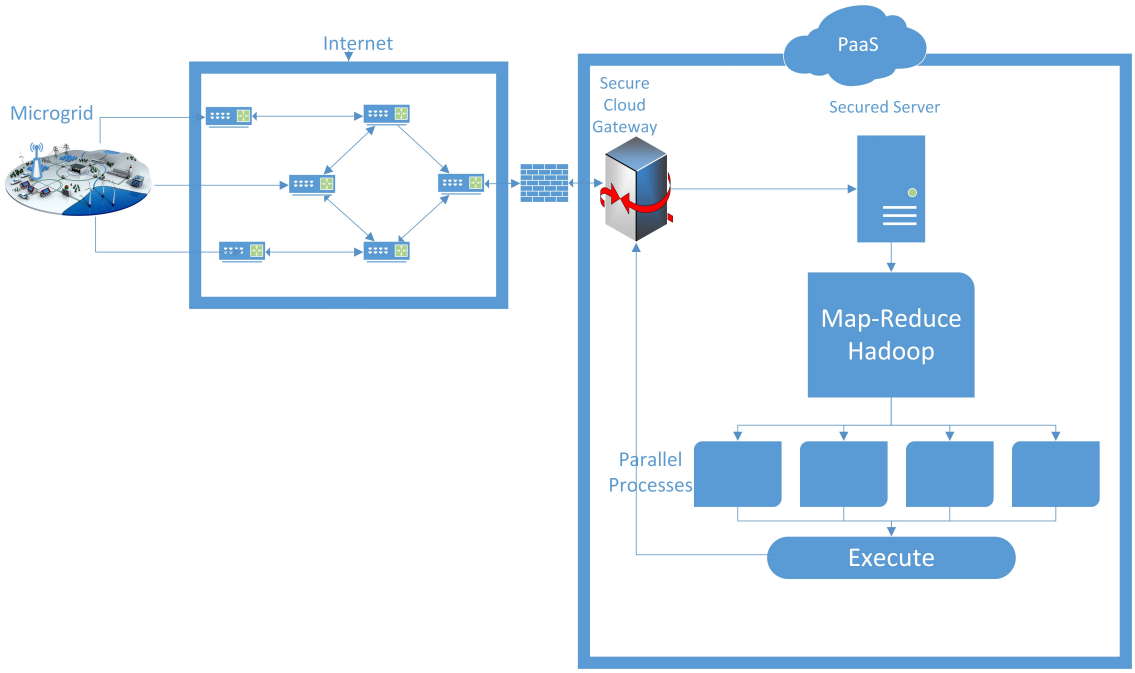


Figure 5.8: Schematic View of the Cloud-based Server for Secure Communication in Clusters of Microgrids.

NeT++ platform through the Ethernet hub[7]. OPAL-5600 communicates with the power simulation model utilizing an interface based on power hardware-in-the-loop (PHIL)[8].

5.5 Cloud-based Information Network for Microgrids Communication

Consider that in a cloud system, serving as Platform as a Service (PaaS), the cloud server is located in the private cloud which has a connecting point (cloud gateway)

[7]The IEEE 802.3 Ethernet communication protocol is followed in this schema.

[8]Power hardware-in-the-loop (PHIL) simulation is an extended version of hardware-in-the-loop (HIL) simulation where the simulation environment exchanges power with real hardware in a virtual fashion. However, the usual case in HIL simulation only involves the signal exchange rather than considering the power exchange [65].

to the Internet. The cloud server is equipped with a Hadoop framework in order to have high computational power needed for running the proposed routing algorithm. Every microgrid is considered to be a cloud customer which is located in the public cloud and communicates with the cloud servers via the cloud gateway (See Fig. 5.8 for schematic view of the system). A two-way communication between microgrids and the gateway is done through a network of routers connected with pre-established TCP connections[72].

PaaS is a category of cloud computing services that provide a platform allowing customers to develop, run, and manage Web applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. PaaS can be delivered in two ways: as a public cloud service from a provider, where the consumer controls software deployment and configuration settings, and the provider provides the networks, servers, storage and other services to host the consumer's application; or as software installed in private data centers or public infrastructurex as a service and managed by internal IT departments.

In PaaS, the provider might give some control to the people to build applications on top of the platform. But any security below the application level such as host and network intrusion prevention will still be in the scope of the provider and the provider has to offer strong assurances that the data remains inaccessible between applications [73]. PaaS is intended to enable developers to build their own applications on top of the platform. As a result, it tends to be more extensible than Software as a Service (SaaS), at the expense of customer-ready features. This trade-off extends to security features and capabilities. In fact, the built-in capabilities are not complete, but they are flexible enough for more security extension.

5.6 OMNet++: An Effective Means For Enabling Modern Communication

Combination of several simulator to realize a particular modeling objective is called co-simulation. Based on whether the simulations are real time or off-line, there may be a need to synchronize simulation time of the simulators involved in the simulation. In smart grid studies, this coordination of simulators represents a promising scientific contribution as it enables researchers to study a variety aspects of smart grid operation [74]. So many efforts have been done in co-simulation of the power grid and communication links. Most of the work highlight the integration of the different simulator types as the main issue. OMNeT++ itself is not a simulator of any communication network, but rather provides infrastructure and tools for writing simulations. One of the fundamental ingredients of this infrastructure is a modular architecture for simulation models. These modules enables OMNeT++ to emulate several communication and networking protocols, such as IPv4, IPv6, TCP, UDP, and Ethernet.

One major component of the smart power grid is the communication protocol. Without a proper communication network, the element of intelligence loses its sensibleness in the power system. So far, many simulations have been done by researchers to investigate certain conceptual designs and intellectual ideas, however, they are mostly impractical due to negligence of the communication limitations in their system. In order to fulfill this limitation, OMNeT++ simulation tool is used in conjunction with the real-time simulator of smart grid, to model: 1) the wireless communication networks, 2) oblivious network protocol, 3) distributed hardware system, and 4) validating the hardware architecture.

In our proposed work, the clusters of microgrids are emulated in OPAL-RT and RTDS real-time simulation platforms. This is an essential task since the queuing networks and propagation delay impose a significant constraint in realization of the actual system. In fact, we need to deploy a network simulation platform in order to evaluate the performance of the two-way communication network connecting the clusters of microgrids to the cloud server. To this end, we propose OMNeT++ which is a discrete event simulator for modeling and performance evaluation of the communication network. The co-simulation of RTDS and OMNeT++ enables us to run the power system models concurrently with real-time network simulator. Therefore, the simulations can be done simultaneously and the results can be fed into the calling application automatically. The structure of the system simulation is shown in Fig. 1.

5.7 Summary and Conclusion

This chapter presents a framework for implementation, simulation, and evaluation of a novel power routing algorithm for clusters of microgrids. We utilized an oblivious routing algorithm which is an efficient tool for network optimization in large-scale real world systems. Oblivious routing design is most suited for the networks in which we have little/no knowledge regarding their current and future states. Therefore, it goes well in line with the microgrid concept, as they are largely independent entities by definition. In order to validate the effectiveness of the oblivious routing algorithm, the proposed algorithm was implemented on a cluster composed of 14 microgrids and compared the performance results with the output of MATPOWER for the same input specifications. In order to implement the oblivious network routing algorithm, we presented a cloud environment in the form of PaaS which is an

economic and secure tool for computing the power routing scheme for clusters of microgrids. In order to simulate our novel routing algorithm in a large-scale and realistic system, we proposed to integrate RTDS in our framework for further implementation. Our comprehensive framework deployed a network simulator in order to evaluate the performance of the two-way communication network connecting the clusters of microgrids to the cloud server. To this end, we introduced OMNeT++ which is a discrete event simulator for modeling and performance evaluation of the communication network.

We plan to extend this work in the following three major directions:

- implementing an oblivious power routing algorithm on OPAL-RT and evaluating its performance in a real-time simulation environment;
- extending the cloud environment as an effective means for communication in the network of microgrids;
- deploying OMNeT++ which is a discrete event simulator for modeling the communication network and distributed systems that can be used for modeling and testing a wide-area communication protocols.

Bibliography

[1] S. Parhizi, H. Lotfi, A. Khodaei, and S. Bahramirad "State of the art in research on microgrids: A review," *IEEE Access*, vol. 3, pp. 890-925, 2015.

[2] M. N. Faqiry and S. Das, "Double-sided energy auction in microgrid: Equilibrium under price anticipation," *IEEE Access*, vol. 4, pp. 3794-3805, 2016.

[3] M. N. Faqiry and S. Das, "Generation applications package for combined heat power in on-grid and off-grid microgrid energy management system," *IEEE Access*, vol. 4, pp. 3444-3453, 2016.

[4] A. Khodaei, "Provisional microgrid planning," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, 2015.

[5] H. Lotfi and A. Khodaei, "AC versus DC microgrid planning," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, 2015.

[6] A. Khodaei, S. Bahramirad, and M. Shahidehpour, "Microgrid planning under uncertainty," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2417-2425, 2015.

[7] P. C. Loh, D. Li, Y. K. Chai, and F. Blaabjerg, "Autonomous operation of hybrid microgrid with AC and DC subgrids," *IEEE Transactions on Power Electronics*, vol. 28, no. 5, pp. 2214-2223, 2012.

[8] J. Rocabert, A. Luna, F. Blaabjerg, and P. Rodriguez, "Control of power converters in AC microgrids," *IEEE Transactions on Power Electronics*, vol. 27, no. 11, pp. 4734-4749, 2012.

[9] X. Wang, J. M. Guerrero, F. Blaabjerg, Z. Chen, "A review of power electronics based microgrids ," *Journal of Power Electronics*, vol. 12, no. 1, pp. 181-192, 2012.

[10] Y. Li, et al., "Design, analysis, and real-time testing of a controller for multibus microgrid system," *IEEE Transactions on Power Electronics*, vol. 19, no. 5, pp. 1195-1204, 2004.

[11] S.C. Mueller, H. Georg, J. J. Nutaro, E. Widl, Y. Deng, P. Palensky, M. U. Awais, M. Chenine, M. Kuch, M. Stifter, H. Lin, S. K. Shukla, C. Wietfeld, C. Rehtanz, C. Dufour, X. Wang, V. Dinavahi, MD. O. Faruque, W. Meng, S. Liu, A. Monti, M. Ni, A. Davoudi, and A. Mehrizi-Sani, "Interfacing Power System and ICT Simulators: Challenges, State-of-the-Art, and Case Studies," *IEEE Transactions on Smart Grid*, 2016, to appear.

[12] N. Ahmed Khan, G. A. S. Sidhu, and F. Gao, " Optimizing combined emission economic dispatch for solar integrated power systems," *IEEE Access*, vol. 4, pp. 3340-3348, 2016.

[13] K. G. Boroojeni *et al.*, "Optimal two-tier forecasting power generation model in smart grids," *International Journal of Information Processing* , vol. 8, no. 4, pp. 79-88, 2014.

[14] K. G. Boroojeni, M. H. Amini, S. Bahrami, S.S. Iyengar, A. I. Sarwat, and O. Karabasoglu, "A novel multi-time-scale modeling for electric power demand forecasting: From short-term to medium-term horizon," *Electric Power Systems Research*, vol. 142, pp. 58-73, 2017.

[15] K. G. Boroojeni, S. Mokhtari, S.S. Iyengar, "A hybrid model for forecasting power and demand in smart grids," in Proc. *Eighth Conference on Communication Networks*, July 2014.

[16] H. Zhou, et al. "An information management platform for smart microgrid cluster based on ASOA," *Automation of Electric Power Systems*, vol. 34, no. 13, pp. 66-71, 2010.

[17] J. Hazra, and K. S. Avinash, "Congestion management using multiobjective particle swarm optimization," *IEEE Transactions on Power Systems*, vol. 22, no. 4, pp. 1726-1734, 2007.

[18] S. Huang *et al.*, "Uncertainty management of dynamic tariff method for congestion management in distribution networks," *IEEE Transactions on Power Systems*, vol. 31, no. 6, pp. 4340-4347, 2016.

[19] R.S. Fang, and A. K. David, "Transmission congestion management in an electricity market," *IEEE Transactions on Power Systems*, vol. 14, no. 3, pp. 877-883, 1999.

[20] S. Kar, G. Hug, J. Mohammadi, and J.M.F. Moura, "Distributed state estimation and energy management in smart grids: A Consensus+ Innovations approach," *IEEE Transactions on Selected Topics in Signal Processing* , vol. 8, no. 6, 1022 - 1038, Dec. 2014.

[21] U.S. Department of Energy, *The Smart Grid: An Introduction*, 2008.

[22] A. Zidan and E. F. El-Saadany, "A cooperative multi-agent framework for self-healing mechanisms in distribution systems," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1525-1539, 2012.

[23] R. E. Brown, "Impact of smart grid on distribution system design" , in *Proc. IEEE Power and Energy Society General Meeting*, July 2008, pp. 1–4.

[24] B. H. Chowdhury, and S. Rahman, "A review of recent advances in economic dispatch," *IEEE Transactions on Power Systems*, vol. 5, no. 4, pp. 1248-1259, Nov. 1990.

[25] V. Pappu, M. Carvalho, and P. Pardalos, *Optimization and security challenges in smart power grids*. Springer, 2013.

[26] K. Mets, J. A. Ojea, and C. Develder, "Combining power and communication network simulation for cost-effective smart grid analysis," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1771-1796, 2014.

[27] A. Varga, "The OMNeT++ discrete event simulation system," in *Proc. the European Simulation Multi-conference (ESM2001)*, vol. 9, no. S 185. sn, 2001.

[28] A. Varga, "Using the OMNeT++ discrete event simulation system in education," *IEEE Transactions on Education*, vol. 42, no. 4, pp. 1-11, Nov 1999.

[29] J. Dede *et al.*, "OMNeT++ and Mosaik: Enabling simulation of smart grid communications", arXiv:1509.03067, 2015 Sep 10.

[30] M. A. H. Sadi, M. H. Ali, D. Dasgupta, and R. K. Abercrombie, "Opnet/simulink based testbed for disturbance detection in the smart grid," in *Proc. the 10th Annual Cyber and Information Security Research Conference* , ser. CISR-15. New York, NY, USA: ACM, 2015, pp. 17:1-17:4.

[31] M. Levesque, D. Q. Xu, G. Joos, and M. Maier, "Communications and power distribution network co-simulation for multidisciplinary smart grid experimentations," in *Proc. the 45th Annual Simulation Symposium*, 2012, pp. 2-7.

[32] D. Bhor, K. Angappan, and K. Sivalingam, "A co-simulation framework for smart grid wide-area monitoring networks," in *Proc. Sixth International Conference on Communication Systems and Networks (COMSNETS)*, Jan 2014, pp. 1-8.

[33] S. Bocker, C. Lewandowski, C. Wietfeld, T. Schluter, and C. Rehtanz, "ICT-based performance evaluation of control reserve provision using electric vehicles," in *Proc. Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Oct 2014, pp. 1-6.

[34] "Energy System Integration", Microgrid Testing, National Renewable Energy Laboratory (NREL), [Online]. Available: http://www.nrel.gov/docs/fy16osti/65839.pdf

[35] D. Chen, J. Brown and J. Y. Khan, "Performance analysis of a distributed 6LoWPAN network for the Smart Grid applications," in *Proc. IEEE Ninth*

International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2014, pp. 1-6.

[36] Y. Qu, K. Xu, J. Liu and W. Chen, "Toward a practical energy conservation mechanism with assistance of resourceful mules," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 145-158, April 2015.

[37] A. Y. Privalov and A. Tsarev, "Analysis and simulation of WAN traffic by self-similar traffic model with OMNeT," in *Proc. International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2014, pp. 629-634.

[38] J. L. Kuo, C. H. Shih and Y. C. Chen, "Performance analysis of real-time streaming under TCP and UDP in VANET via OMNET," in *Proc. 13th International Conference on ITS Telecommunications (ITST)*, 2013, pp. 116-121.

[39] S. N. Khan, M. A. Kalil and A. Mitschele-Thiel, "crSimulator: A discrete simulation model for cognitive radio ad hoc networks in OMNeT ++," in *Proc. Wireless and Mobile Networking Conference (WMNC)*, 2013, pp. 1-7.

[40] S. Wang, K. Z. Liu and F. P. Hu, "Simulation of wireless sensor networks localization with OMNeT," in *Proc. 2nd Asia Pacific Conference on Mobile Technology, Applications and Systems*, 2005, pp. 1-6.

[41] L. I. Niar and H. Haffaf, "Graphical analysis for monitoring in a sensor network (WSN). Simulator: OMNeT++," in *Proc. International Conference on Education and e-Learning Innovations (ICEELI)*, 2012, pp. 1-6.

[42] J. Zhu. *Optimization of power system operation*. John Wiley & Sons, 2014.

[43] N. Padhy, "Unit commitment-a bibliographical survey," *IEEE Transactions on Power Systems*, vol. 19, no. 2, pp. 1196-1205, 2004.

[44] A. I. Selvakumar and K. Thanushkodi, "A new particle swarm optimization solution to nonconvex economic dispatch problems," *IEEE Transactions on Power Systems*, vol. 22, no. 1, pp. 42-51, Feb. 2007.

[45] M.H. Amini, B. Nabi, and M. -R. Haghifam, "Load management using multi-agent systems in smart distribution network," in *Proc. IEEE Power and Energy Society General Meeting*, July 2013, pp. 1–5.

[46] J. Lavaei, D. Tse, and B. Zhang, "Geometry of power flows and optimization in distribution networks," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 572-583, Mar. 2014.

[47] K. G. Boroojeni, M. H. Amini, S. S. Iyengar, M. Rahmani, P. M. Pardalos, "An economic dispatch algorithm for congestion management of smart power networks," *Energy Systems*, doi: 10.1007/s12667-016-0224-6.

[48] E. Kellerer and F. Steinke, "Scalable economic dispatch for smart distribution networks," *IEEE Transactions on Power Systems*, vol. 30, no. 4, pp. 1739-1746, 2015.

[49] M. Carrion and J. Arroyo, "A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem" , *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1371-1378, Aug. 2006.

[50] A. Botterud, Z. Zhi, J. Wang *et al*, "Demand dispatch and probabilistic wind power forecasting in unit commitment and economic dispatch: A case study of Illinois" , *IEEE Transactions on Sustainable Energy*, vol. 4, no. 1, pp. 250-261, Oct. 2012.

[51] Z. Li, Q. Guo, H. Sun, and J. Wang, "Sufficient conditions for exact relaxation of complementarity constraints for storage-concerned economic dispatch" , *IEEE Transactions on Power Systems*, vol. 31, no. 2, pp. 1653-1654, 2016.

[52] S. K. Khator and L.C. Leung, "Power distribution planning: a review of models and issues," *IEEE Transactions on Power Systems*, vol.12, no.3 , pp. 1151-1159, Aug. 1997.

[53] Y. Liu, N. Ul Hassan, S. Huang, and C. Yuen, "Electricity cost minimization for a residential smart grid with distributed generation and bidirectional power transactions," in *Proc. IEEE Innovative Smart Grid Technologies (ISGT)*, 2013, pp. 1–6.

[54] A. Papavasiliou and S. S. Oren, "Supplying renewable energy to deferrable loads: Algorithms and economic Analysis, " in *Proc. IEEE Power and Energy Society General Meeting*, July 2010, pp. 1–8.

[55] S. S. Iyengar and K. G. Boroojeni, *Oblivious network routing: Algorithms and applications*. MIT Press, 2015.

[56] A. Gupta, M. T. Hajiaghayi, and H. Racke, "Oblivious network design," in *Proc. of the 17th annual ACM-SIAM symposium on Discrete algorithm*, 2006, pp. 970-979.

[57] Q. Shafiee, T. Dragicevic, J. C. Vasquez, and J. M. Guerrero, "Hierarchical Control for Multiple DC-Microgrids Clusters," *IEEE Transactions on Energy Conversion*, vol. 29, no. 4, pp. 922-933, 2014.

[58] R. Zamora and A. K. Srivastava, "Multi-layer architecture for voltage and frequency control in networked microgrids," *IEEE Transactions on Smart Grid,* 2016, to appear.

[59] A.-H. Mohsenian-Rad, V. W. S. Wong, J. Jatskevich, R. Schober, and A. Leon-Garcia, "Autonomous demand-side management based on game-theoretic energy consumption scheduling for the future smart grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 3, pp. 320-331, 2010.

[60] D. S. Callaway and I. A. Hiskens, "Achieving controllability of electric loads," *Proceedings of IEEE*, vol. 99, no. 1, pp. 184-199, 2011.

[61] S. Bhattacharya and P. Bauer, "Requirements for charging of an electric vehicle system based on state of power (SoP) and state of energy (SoE)," in *Proc. 7th International Power Electronics and Motion Control Conference (IPEMC)*, 2012, pp. 434-438.

[62] S. Sucic, T. Dragicevic, T. Capuder, and M. Delimar, "Economic dispatch of virtual power plants in an event-driven service-oriented framework using standards-based communications," *Electric power Systems Research*, vol. 81, no. 12, pp. 2108-2119, 2011.

[63] Q. Shafiee, C. Stefanovic, T. Dragicevic, P. Popovski, J. C. Vasquez, and J. M. Guerrero, "Robust networked control scheme for distributed secondary control of islanded microgrids," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 10, pp. 5363-5374, Oct. 2014.

[64] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," Proceedings of IEEE, vol. 95, no. 1, pp. 138162, 2007.

[65] OPAL-RT, "eMEGAsim PowerGrid Real-Time Digital Hardware in the Loop Simulator." [Online]. Available: http://www.opal-rt.com/product/emegasim-powergrid-real-time-digital-hardware-in-the-loop-simulator.

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

[67] J. Fakcharoenphol, S. B. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," in *Proc. the 35th STOC*, 2003, pp. 448-455.

[68] MATLAB version 8.5. Miami, Florida: The MathWorks Inc., 2015.

[69] (2015) Power systems test case archive. [Online]. Available: http://www.ee.washington.edu/research/pstca/

[70] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12-19, Feb. 2011.

[71] IEEE 118-bus, 54-unit, 24-hour system, available online: motor.ece.iit.edu/data/JEAS_IEEE118.doc

[72] K. G. Boroojeni, M. H. Amini, and S.S Iyengar, "Cloud Network Data Security," Smart Grids: Security and Privacy Issues. Springer International Publishing, 2017, pp. 71-82.

[73] K. G. Boroojeni, M. H. Amini, and S.S. Iyengar, "Overview of the Security and Privacy Issues in Smart Grids," Smart Grids: Security and Privacy Issues. Springer International Publishing, 2017, pp. 1-16.

[74] M. O. Ezeme, "A multi-domain co-simulator for smart grid: Modeling interactions in power, control and communications." PhD dissertation, 2015.

CHAPTER 6

AN OBLIVIOUS ROUTING SCHEME FOR CONTENT-CENTRIC NETWORKS

Contemporary Internet architecture is based on the host-based model whereby the data flow is driven by explicit addresses assigned to the communicating hosts. However, future Internet structure is expected to be focused on content and not the physical location of machines. This leads to a very efficient use of the network when many people are interested in the same content. On the other words, the content-centric networking seeks to adapt the Internet architecture to the current usage pattern.

One of the most critical and basic issues in designing the content-centric networks is the security of the system. Considering the data network as a service, the users has to be assured regarding their security and privacy when using the service. In today Internet, the security preserving is already implemented in different layers including IP, TCP, and the application layer. However, there are many unsolved challenges regarding the security issues in a host-oblivious network as the identity of each host may be unknown. For the content-centric networks as a network of oblivious hosts, the network designers need to come up with an appropriate security scheme which is not based on the IPs/locations of the hosts.

The other important challenges in the data distribution systems is how to distribute the data flow throughout the network. More specifically, in a content-centric network, the data traffic pattern is usually in a way that some of the hosts get congested because of a lot of data that is flowing through them. As illustration, consider a situation that some special content gets extraordinarily hot and interesting in some

[0]Part of this chapter has been reprinted with permission from S. S. Iyengar and Kianoosh G. Boroojeni, "Oblivious Network Routing: Algorithms and Applications," MIT Press, 2015 [1].

period of time. In this case, the hosts that are sort of related to that content and their neighbors may get congested with a huge data flow. As the result, the network delay and energy consumption may substantially increase.

Regarding the two aforementioned issues in the content-centric networks, we are proposing a model of a data distribution system which is a peer-to peer network and appropriately deals with the both issues. In this model, the data requests and the level of security are managed in a content-centric based way; however, the data flows are routed in a node-congestion preventing routing scheme which is location/host based. Subsequently, we call the model as the *hybrid* one.

In the first section, we present an introduction of the security issues in content-centric networks. Then, we will address the details of our hybrid model in the context of its security and routing schemes in Section 6.2 and 6.3. In Sections 6.4 and 6.5, we will explain the details of how our routing scheme prevents the node congestion and in the meanwhile, it doesn't increase the routing cost. In order to prove our claim, we provide some mathematical and geometrical analysis in Sections 6.6 and 6.7.

6.1 Security Preliminaries

The first step of understanding the information security is to understand the basic principals concerning it. Confidentiality, integrity and availability (CIA) comprises all the principles on which every security program is based. Also, some other key concepts like accounting, auditing and non-reputation have been proposed but they are not as essential as CIA.

Confidentiality

Confidentiality determines the secrecy of the information asset. It refers to preventing the disclosure of some information to unauthorized individuals or systems. For instance, a credit card transaction on the Internet requires the card number to be transmitted from the purchaser to the merchant and from the merchant to a transaction-processing network. The system attempts to enforce the confidentiality by encrypting the card number during the transmission. This is done by limiting the places where it might appear (in databases, log files, backups, printed receipts, and so on), and also by restricting the access to the places where it is stored. If an unauthorized party obtains the card number in some way, a breach of confidentiality has occurred.

Confidentiality is necessary for maintaining the privacy of the people whose personal information is held by a system. Authentication methods like the user-IDs and passwords (which uniquely identify the users and control the data access to the system resources) underpin the goal of confidentiality.

Integrity

Integrity refers to the trustworthiness of information resources. In information security, data integrity means maintaining and assuring the accuracy and consistency of data over its entire life-cycle. In fact, the data cannot be modified in an unauthorized or undetected manner. This is not the same as what is called the referential integrity in the databases; however, it can be viewed as a special case of the consistency as understood in the classic model of transaction processing. Integrity is violated when a message is actively modified in transit. Information security systems typically provide the message integrity in addition to the data confidentiality.

Availability

Availability refers to the condition in which the information resources are available. In any information system, the information must be available when it is needed; otherwise, the system may fail to reach its goal. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly. High-availability systems aim to remain available at all times and prevent the service disruptions due to the power outages, hardware failures, and system upgrades. Ensuring availability also involves the prevention of the denial-of-service (DoS) attacks.

Privacy

Privacy relates to all the elements of the CIA triad. It considers which information can be shared with others (confidentiality), how that information can be accessed safely (integrity), and how it can be accessed (availability).

Host-Oblivious Security Schemes

Ensuring security and privacy in a host oblivious network is a challenging problem, especially in the content-based network. We distinguish a host-oblivious network security paradigm from a conventional host-dependent network security from two perspectives, a host-oblivious security association and multiple security levels. Host-dependent network security is based on the conventional networks themselves, which are running using a host-based architecture. A source initiates a communication on the assumption that the source is able to identify a destination. Due to the destination awareness, for cryptographic functions, the source simply uses a shared

symmetric secret key or a public key of the destination. Because the destination also can identify the source, a security association for both nodes is easily established. That is, regardless of the diversity of the application data, only a single security level is provided during the entire communication (which is based on the hosts identities).

There are two basic principles of the host-oblivious network security. The first is that a security association is independent from the host identification, which is caused by blindness of hosts in the content-based network architecture. The second is the provision of multiple security associations for diverse security-sensitivity of the various contents.

An important challenge in oblivious Content Centric Network Security is the protocol design to bootstrap the establishment of secure communication infrastructure from a collection of nodes which may have been pre-initialized with some secret information without any prior direct contact among them. This problem called the *bootstrapping problem*. The key point in the host oblivious security is the keys selection which must be independent from each other and the host. As mentioned before, there are two methods for generating session keys including asymmetric and symmetric crypto-algorithms. The prerequisite of applying public-key cryptography is the awareness of the destination address which is in contradiction with the content centric networks.

There are three types of general key agreement schemes: trusted-server scheme, self-enforcing scheme and key pre-distribution scheme. In the latest one, the key information is distributed among all nodes prior to deployment. Random pool-based (RPB) scheme is a proposed random key pre-distribution scheme to address the bootstrapping problem. Its method is briefly discussed as follow: A random pool of keys is selected from the key space; then, each node receives a random subset of keys from the key pool prior to any connection which it wants to be involved in.

Any node which has a common key within its subset can apply it as a shared secret key to initiate communication.

There is a key distribution center (KDC) in Random pool-based scheme which manages a key pool of keys. Before deploying the nodes, an initialization phase is performed. In the initialization phase, the basic scheme picks a random pool (set) of keys S out of the total possible key space. For each node, m keys are randomly chosen from the key pool S; then, KDC distribute the selected keys to the nodes. When the nodes are deployed, a key-setup phase is performed. When two nodes want to generate a link key, they exchange the keys indexes. In the case that two nodes share at least one key, they use one of the commonly shared keys as a *link key*. The mentioned method has been extended to the *q-composite random pool based scheme*. In this scheme, both nodes are required to share at least q numbers of keys and the link key is generated by combining q common keys.

6.2 The Hybrid Model Description

In this section, we propose a hybrid model of the content-centric networks that will be analyzed in the context of security and cost efficiency through the chapter.

We assume that there are a number of *cyber devices* scattered over a plane. These devices which may have time-varying locations (like mobile devices) have various interests to special data contents. Additionally, each device has some stored data which may satisfy others interests. The data transmission between the cyber devices is performed through a network of *routing nodes* which are deployed on a convex subset of the Euclidean plane and are wired together. Let $r_1, r_2, \ldots, r_n$ denotes the

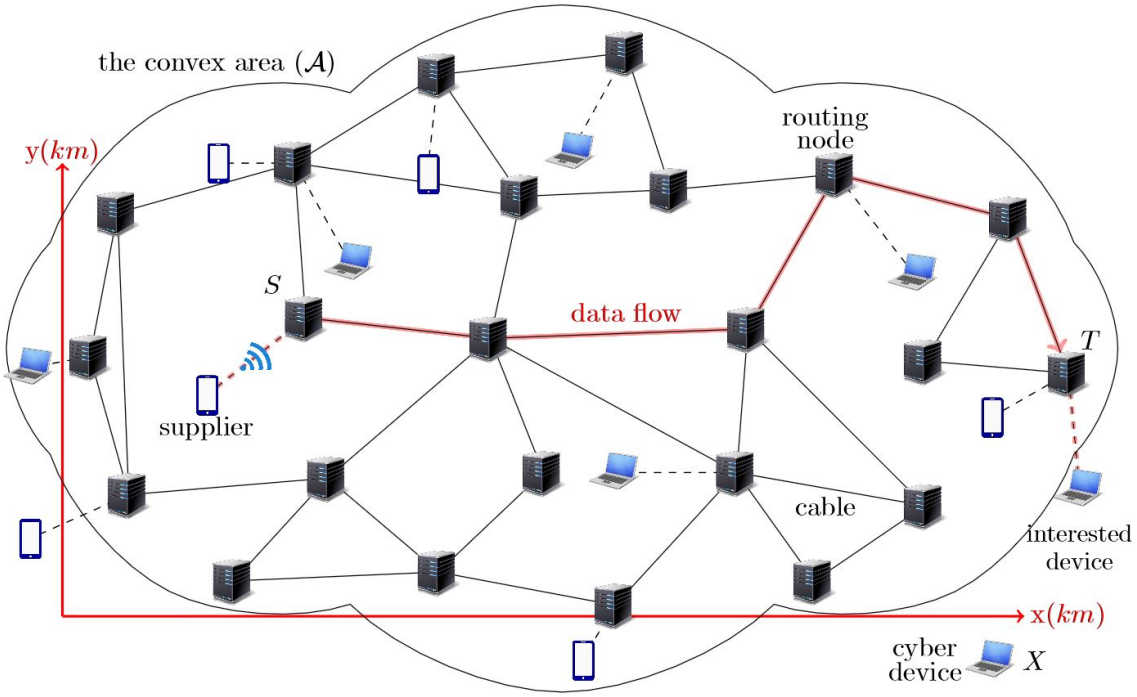


Figure 6.1: Schematic representation of the network topology and the nodes deployment in the Euclidean plane. As you see, device X is not connected to any routing node.

n routing nodes in the network. Despite the cyber devices, every routing node has a fixed location on the conveying convex subset.

Moreover, we assume that if a cyber device gets close enough to some routing node, a wired or wireless connection will be established between them; however, every device can only be connected to one routing node at a given time. Note that, the cyber devices may get connected and disconnected to the network of the routing nodes as they are moving through the plane over time. More specifically, every cyber device may get connected to different routing nodes as it moves in the plane and also, it may move far from the network such that it gets connected to no routing node. See Figure 6.1 for illustration.

Now consider the following scenario: some cyber device is interested in receiving some data, say a movie. We call this device as the "interested device". Assuming

that this device is close enough to some routing node (T), a *request* including the movie name is sent from the interested device to T via a wired or wireless connection between them. Then, node T will broadcast the request through the network of routing nodes in the following way: "Each routing node that receives the request, will forward it to its connected cyber devices and neighboring routing nodes". In the case that there exists a connected cyber device which has the requested movie, it notifies the interested device with a message through the reverse path. When the notification messages are delivered to the interested device, one of the cyber devices that has replied back the request will be chosen as the "supplier" (for simplicity, we assume that the sender of the first received notification will be chosen as the supplier). Then, the interested device asks the supplier to send the movie. Finally, the supplier transmits the movie through a path between S and T which is obtained by a versatile routing scheme (note that the data has to be transmitted from the supplier to S in the first step and from T to the interested device in the last step).

In this model, we assume that every cyber device wants to satisfy its demand only by receiving the interesting content from an authenticated device. In addition, the supplier will share its data only with the authenticated devices. To deal with these security issues, we use a security paradigm which uses a trusted third party to provide such authentication between devices. Furthermore, we will thoroughly address the cost efficiency and the node congestion issues regarding the data flows through the network of our model.

Now we will look at the detailed four phases of the protocol which is used in this model and is mainly based on the security paradigm proposed by Chan et al. in 2003 and Jeong et al. in 2010. In this protocol, we assume that every cyber device connected to the network has a secure connection with a trusted third party in the network called *the trusted node*. This node has a *key pool* which contains a

number of keys which are used to establish authenticated connections in the network. Additionally, we assume that the routing nodes are already able to route data through the network using a location-based routing scheme; i.e. in the case that the final destination of an arrived packet is specified, the routing node will forward the packet toward a suitable interface. We specify the destination of a packet using the ID of the routing node in which the packet is supposed to be in the last step.

Phase 0: Key Distribution

When a cyber device gets connected to the network of the routing nodes, it asks the trusted node for m keys. Then, the trusted node assigns m keys of the key pool to the cyber device as its key ring. This is done in such a way that for every given pair of connected devices x and y, device x authenticates y if and only if a specific q-size subset of x's key ring completely belongs to the key ring of y ($q \leq m$).

Here is the exact format of the "registration message" which is sent by the trusted node using the location-based routing scheme and includes the key ring of the new connected device:

$$\text{REG} = \left(\text{RNI}, \{k_1, k_2, \ldots, k_m\}_{\text{pvt-key}}\right) \tag{6.1}$$

where RNI is the index of the routing node directly connected to the new device (i.e. if the new device is connected to r_3, RNI= 3). Also, keys $k_1, k_2, \ldots, k_m$ belong to the key pool and are assigned to x as its key ring. These keys are encrypted in the trusted node before sending the REG message. The encryption of the key ring is done by the private key of the trusted node (pvt-key) using a symmetric key-based authentication algorithm.

Phase 1: Requesting a Content

The interested device makes a "request message" containing a unique ID of the request, the identifying information of the exact content in which it is interested, a TTL[1] integer value, and the indexes of the q randomly chosen keys belonging to its key ring ($q \leq m$); for example, the device wants to receive content c, $TTL = 15$, and it chooses the keys of indexes $i_1, i_2, \ldots, i_q$ among the m keys belonging to its key ring ($\forall j \in [1, q] : 1 \leq i_j \leq m$). Then, the message will be sent to the routing node directly connected to the device. This node will broadcast the request message through the network by forwarding it to its connected devices and neighboring routing nodes. Before forwarding the message, the routing node decrease the TTL by one. Every other routing node which receives the request message, decrements its TTL and broadcasts it through the network in the same way until the TLL value becomes zero.

Here is the detailed content of the request message:

$$\text{REQ} = \big(\text{RID}, \{\text{CID}\}_k, \text{TTL}, (i_1, i_2, \ldots, i_q)\big) \tag{6.2}$$

where RID and CID respectively denote the request ID and the content ID (and uniquely specify the request and the content). Additionally, in order to preserve the privacy of the interested device, value CID is included to the request message after getting encrypted by key k. This key is obtained by applying a globally known hash function (h) to the q chosen keys of the key ring; i.e. considering the key ring as the set of keys in the form $\{k_i | i = 1 \ldots m\}$, value key will be in the following form:

$$k = h(k_{i_1}, k_{i_2}, \ldots, k_{i_q}) \tag{6.3}$$

[1]time to live

Phase 2: Responding to the Request

Every cyber device which receives the request message checks whether its key ring includes the q keys specified in the message. If yes, it then checks its own data storage to see if it has the requested content. In the case that it doesn't have all the q keys or the exact requested content, the request message will be simply ignored; otherwise, the device makes a "notification message" containing the index of the routing node to which it is already connected, the TTL value of its associated request message (represented by RES[2]), and the ID of the request. Then, this message will be sent to the interested device using the reverse path of the one through which the corresponding request message was previously sent (later, we will address the details of how the reverse path is computed).

Here is the detailed content of the notification message:

$$\text{NOTIF} = \big(\text{RID}, \text{RNI}, \text{RES}\big) \tag{6.4}$$

where RNI is the index of the routing node directly connected to the device which is sending the notification message. Note that the value of RID is the same as what was included in the corresponding REQ message.

Phase 3: Choosing the Supplying Device

When all of the notification messages are delivered to the interested device, it will choose the one with the largest RES value; i.e. the message with the closest sender. We call the sender of the chosen message as the *supplier*. Then, the interested device sends an "acknowledgment message" to the supplier to ask for content c. The acknowledgment message includes the ID of the routing node connected to the interested device and is sent through a path obtained by the location-based

[2]residue

routing scheme. Here is the details of the acknowledgment message where RNI_I and RNI_S respectively denote the indexes of the routing nodes directly connected to the interested device and the supplier:

$$\mathrm{ACK} = \big(\mathrm{RID}, \mathrm{RNI}_I, \mathrm{RNI}_S\big). \tag{6.5}$$

Again, note that the value of RID is the request ID which was put in the corresponding REQ and NOTIF messages.

Phase 4: Data Transmission

When the supplier receives the acknowledgment message, it starts sending the requested content to the interested device through a congestion prevention routing scheme which will be discussed in detail later. The data transmission occurred in this phase is usually much larger than the messages transmitted previously. Henceforth, a number of "data messages" will be sent in this phase in a way that each message contains only a portion of the whole data that has to be transmitted toward the interested device. Here is the parameters of a data message:

$$\mathrm{DATA} = \big(\mathrm{RID}, \mathrm{RNI}_I, \mathrm{RNI}_M, \{\mathrm{FIRST}, \mathrm{LAST}, \mathrm{DP}\}_{k'}\big) \tag{6.6}$$

where FIRST and LAST respectively denote the indexes of the first and the last bytes of the data portion which is included in the data message. Additionally, DP represents the data portion of size (LAST − FIRST) and value RNI_M specifies the index of the "intermediate routing node" through which the data message will be routed toward the interested device (in Section 4, we will describe the way of choosing the intermediate routing node in more details). As you see in Equation 6.6, the data included in the message is encrypted using key k' which is obtained by the following equation:

$$k' = h_{\text{content-type}}(k_{i_1}, k_{i_2}, \ldots, k_{i_q}) \tag{6.7}$$

such that the hash function $h_{\text{content-type}}$ depends on the type of the requested data content. For example, in the case that the interested device requests a multimedia file, we use a hash function that generates a short key; however, when the interested device requests a file that urges a higher level of security, we have to use another hash function which generates longer keys.

When the data message is delivered to the interested device, it is decrypted using key k' computed by Equation 6.7 in the interested device.

6.3 Message Forwarding in the Routing Nodes

In the previous section, we described the hybrid protocol for distribution of interesting data through the network model proposed before. Additionally, different types of messages used in this protocol were precisely determined. In the current section, we will focus on the implementation of the message forwarding in the routing nodes. We also illustrate the way that the cyber devices communicate with each other using an state machine diagram.

In this section, we assume that for a pair of connected routing nodes X and Y, node X needs to have a dedicated interface to handle its connection with Y and vice versa. Additionally, every cyber device connects to a routing node through a "port" of the node.

Forwarding REQ and NOTIF

As mentioned before, the message REQ is broadcast through the network in Phase 1. At the first hop, the interested device sends the message to the p^{th} port of its directly connected routing node T. Node T stores the message RID and port index p in a table before forwarding the message toward other devices and the neighboring

v	RID	Interface #
1	111	11
0	001	10
1	011	10
1	101	00

Broadcast Table of node r

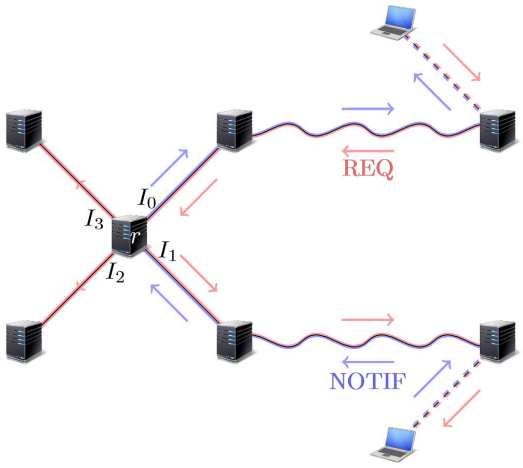


Figure 6.2: How routing node r forwards the REQ and NOTIF messages using its broadcast table. Note that node r has four interfaces.

routing nodes. During the process of broadcasting the message, every routing node that receives the message via its i^{th} interface will forward it to the routing nodes connected to other interfaces (than the i^{th} one). The routing node stores the RID (request ID) of the message REQ and the index (i) of the interface through which the message was received in the "broadcast table" which is depicted in Figure 6.2.

The forwarding process of a NOTIF message is the same as what is done for its associated REQ message, but in the reverse order. This is because the NOTIF message has to be routed in the reverse path of the one through which the request message was sent. In order to route through the reverse path, every routing node that receives the NOTIF message, searches through the broadcast table to find the RID of the message. Then, it will forward the message through the corresponding interface of the RID in the table (see Figure 6.2 for illustration). This process continues until when the NOTIF message reaches to the routing node T which is directly connected to the interested device. In this step, there is no valid entry associated with the message RID in the broadcast table of node T; however, node T finds the message RID in another table where the corresponding port index of the

interested device has been already stored. Finally, the message gets delivered to the interested device via the appropriate port.

Note that the first routing node which receives the NOTIF via its j^{th} port stores the message RID and the value j in a table so that it can remember the port index if a response message (ACK) will come into it in the future.

Forwarding REG and ACK

As mentioned before, we assume that REG and ACK are routed by a location-based routing scheme. This scheme is implemented using a number of tables known as "forwarding tables". Every routing node has a forwarding table which specifies the outgoing interface of any incoming message based on its destination.

Messages REG and ACK both have the indexes of the routing nodes directly connected to their target devices (RNI in REG and RNI_I in ACK). Henceforth, Every routing node which receives a registration message searches through the forwarding table to specify the outgoing interface corresponding to the RNI value of the message and then forwards it through the specified interface. Finally, when the routing node of index RNI receives the REG message, it will deliver it to the recently connected device. An ACK message is forwarded in the similar way. However, when the message reaches its final routing node, it is forwarded based on the table which has already stored the port index associated with the message RID.

Forwarding DATA

In a data message, there are two destinations: the intermediate routing node and the ultimate one. The index of the earlier one is specified by RNI_M; while the later node is determined by index RNI_I.

When a routing node gets a DATA message, it follows Algorithm 5 to forward the message. In this algorithm, forwarding the DATA message a routing node is assume to be done based on the aforementioned location-based routing scheme; however, the message will be forwarded to the target device based on the table which has already stored the port index associated with the RID value of DATA.

Finally, in this section, we provide the detailed interaction of every cyber device in the proposed protocol. Figure 6.3 specifies a state machine which illustrates how a connected device communicates with other elements of the network during different phases of the protocol.

Algorithm 5 DATAFORWARDING

input: DATA message
$k \leftarrow$ the routing node index
if DATA.RNI$_I = k$ **then**
  Forward DATA to the target device
  return
else if DATA.RNI$_M = k$ **then**
  DATA.RNI$_M \leftarrow null$
  Forward DATA to the routing node of index RNI$_I$
else if DATA.RNI$_M = null$ **then**
  Forward DATA to the routing node of index RNI$_I$
else
  Forward DATA to the routing node of index RNI$_M$
end if

As you see in Figure 6.3, when a cyber device gets connected to the network, it asks the trusted node for m keys and goes to the "start" state. Then, it transits to the "ready" state if it receives an appropriate REG message. If the user asks for content c, the device broadcasts a request message, sets the timer to constant $\mathcal{T}$, and goes to the "demand" state. Additionally, if the device is in the ready state and a REQ message arrives at the device, it goes to the "supply" state.

Now, consider the case that the user is in the demand state. In this case, every arrived NOTIF message will be added to a message buffer called "buff". As the timer was set to $\mathcal{T}$ at the moment of state transition from ready to demand, the device stays in the demand state for $\mathcal{T}$ units of time and then, it will go to the ready state again. During this state transition, if the buffer is empty, the user gets informed that the requested data is unavailable. However, in the case that the buffer is not empty, the device will send an ACK message and initiates a parallel thread called "data receiver" to receive the requested data.

Finally, if the device goes to the supply state, it checks its data storage to see whether it has the content of ID $REQ.RID$. If it doesn't have the requested content, it simply ignores the REQ message and goes back to the ready state. However, if it has the content specified in the REQ message, it sends a NOTIF message to the interested device, sets a timer to $\mathcal{T}$ and goes to a new state called "wait". If an ACK message arrives at the device within $\mathcal{T}$ units of time after the sate transition, the device initiates another parallel thread called "data sender" which sends the requested data toward the interested device (in the form of some DATA messages); otherwise, it simply goes back to the ready state without doing anything.

6.4 Oblivious Routing Problem Specification

In this section, we precisely define the oblivious routing problem we encounter in sending data through the hybrid network. To do this, we need to specify the network topology and characteristics using the concept of *geometric* graphs. At first, some preliminary definitions are presented.

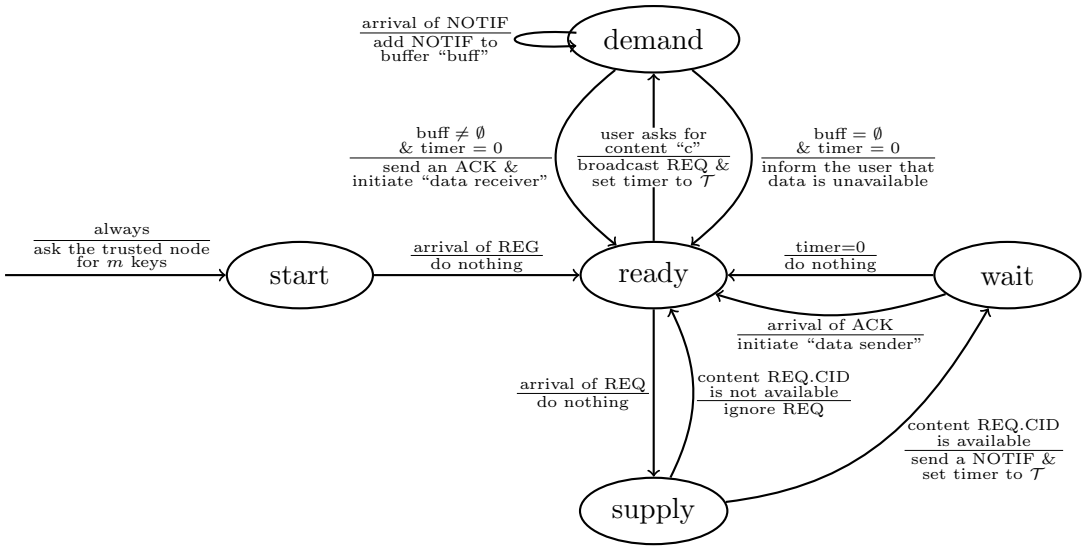


Figure 6.3: How routing node r forwards the REQ and NOTIF messages using its broadcast table. Note that node r has four interfaces.

6.4.1 Definitions

The triple (V, E, loc) is an unweighted geometric graph in the n-dimensional Euclidean space $\mathcal{S}$ if (V, E) specifies an unweighted graph; and loc denotes a function in the form $\text{loc} : V \mapsto \mathcal{S}$. For every (unweighted) geomtric graph, we define three types of distances for each pair of vertices. Assuming that u and v are two vertices of graph $G = (V, E, \text{loc})$, this is the case that:

- The *graph distance* between u and v is denoted by $d_G(u, v)$ and defined as $d_{G'}(u, v)$ where $G' = (V, E)$ represents the unweighted graph corresponding to G. Moreover, if p denotes a path in G, its length $\text{len}_G(p)$ is defined as $\text{len}_{G'}(p)$.
- The (Euclidean) distance between u and v is denoted by $d_{\text{loc}}(u, v)$ and defined as $||\text{loc}(v) - \text{loc}(u)||$ ($||X - Y||$ represents the Euclidean distance[3] between points X and Y in space $\mathcal{S}$).

[3]Euclidean distance between points $X \in \mathcal{S}$ and $Y \in \mathcal{S}$ is denoted by $||X - Y||$ and defined as the length of line segment $\overline{XY}$: $||X - Y|| = |\overline{XY}|$.

- The *hop-by-hop distance* between u and v is represented by $\delta_G(u, v)$ and defined inductively in the following equation:

$$\delta_G(u,v) = \begin{cases} 0 & u = v \\ \min\limits_{\{w,v\}\in E}\{\delta_G(u,w) + ||\mathrm{loc}(w) - \mathrm{loc}(v)||\} & \text{otherwise} \end{cases}$$

Regarding the above defined distances in a geometric graph, it is inferred by the triangular inequality that for every pair of vertices u and v, the Euclidean distance $d_{\mathrm{loc}}(u, v)$ is not greater than $\delta_G(u, v)$. Also, the *pseudo-diameter* Ψ of geometric graph $G = (V, E, \mathrm{loc})$ is defined in the following way:

$$\Psi(G) = \max_{u\in V}\max_{v\in V}\frac{d_G(u,v)}{d_{\mathrm{loc}}(u,v)}$$

For every point $X \in \mathbb{R}^n$ and value $r \in \mathbb{R}_{\geq 0}$, the n-dimensional ball $B(X, r) \subseteq \mathbb{R}^n$ is defined by the following equation:

$$B(X,r) = \left\{Y \in \mathbb{R}^n \middle| ||X - Y|| \leq r\right\}$$

The n-dimensional space $\mathcal{S}$ is called to be *thoroughly $\mathcal{R}$-covered* by geometric graph $G = (V, E, loc)$ if this is the case that:

$$\mathcal{S} \subseteq \bigcup_{v\in V} B(\mathrm{loc}(v), \mathcal{R})$$

6.4.2 Graph Representation of the Hybrid Model

As mentioned before, the network of routing nodes consists of n vertices $r_1, r_2, \ldots, r_n$ which are wired together in the form of a connected graph (r_i represents the i^{th} routing node in our model). Additionally, we assume that the upper-bound of the distance between cyber device and its directly connected routing node is positive constant value $\mathcal{R}$; in fact, those devices can get connected to node r_i that are located in $B(\mathrm{loc}(r_i), \mathcal{R})$.

Let $G = (V, E, \text{loc})$ denote the graph representation of our system where the vertex set V is equal to $\{v_i | i \in [1, n]\}$ and function $\text{loc} : V \mapsto \mathbb{R}^2$ specifies the deployment of r_is in the Euclidean plane. We consider the following constraint for the node locations on the plane:

$$\forall u, v \in V : d_{\text{loc}}(u, v) > \mathcal{R}$$

Any geometric graph that holds such constraint is called to have an $\mathcal{R}$-*distant* deployment. Also, for some $c \geq 1$, this is the case that:

$$\forall \{u, v\} \in E : d_{\text{loc}}(u, v) \leq c\mathcal{R}$$

In addition, we assume that there exists some convex subset[4] $\mathcal{A} \subseteq \mathbb{R}^2$ such that:

$$\begin{cases} \text{loc}(v) \in \mathcal{A} & \forall v \in V \\ \mathcal{A} \subseteq \bigcup\limits_{v \in V} B(\text{loc}(v), \mathcal{R}) \end{cases}$$

which means that set $\mathcal{A}$ completely contains the network and is *thoroughly* $\mathcal{R}$-covered by the graph representation (G) of the network. This means that if a cyber device is located in area $\mathcal{A}$, there exists a routing node which is not farther than $\mathcal{R}$ than the device and consequently, they can get connected together (see Figure 6.4 for illustration).

In order to prove the feasibility of the aforementioned constraints regarding the deployment of vertices in the Euclidean plane, we will show that for any subset $\mathcal{S}$ of the Euclidean plane, there is a deployment of nodes in $\mathcal{S}$ that is $\mathcal{R}$-distant and thoroughly R-covers $\mathcal{S}$. The following algorithm specifies a way of constructing such deployment:

[4]Set $\mathcal{S} \subseteq \mathbb{R}^n$ is a convex subset of the n-dimensional Euclidean space if for every $X, Y \in \mathcal{S}$, line segment $\overline{XY}$ completely lies inside $\mathcal{S}$.

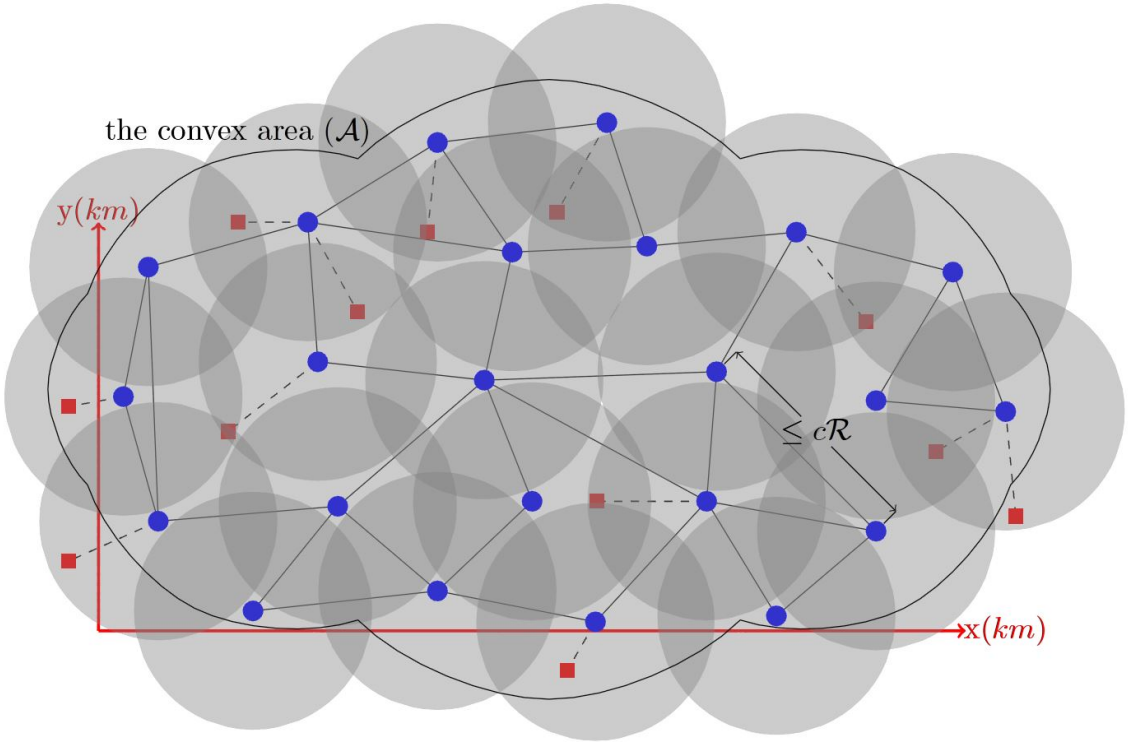


Figure 6.4: Deployment of the routing nodes (blue circles) and cyber devices (red squares) on the Euclidean plane. Note that the radius of any gray circle is $\mathcal{R}$. As you see, the routing nodes are $\mathcal{R}$-distant and at the same time, they thoroughly $\mathcal{R}$-cover the convex area $\mathcal{A}$. Additionally, area $\mathcal{A}$ completely includes the network of routing nodes; however, there may exist some cyber devices located outside of $\mathcal{A}$.

In a greedy manner, arbitrarily choose the uncovered point X in set $\mathcal{S}$ and fix the node there. Then cover the points belonging to ball $B(X, \mathcal{R})$ (At first, every point in $\mathcal{S}$ is uncovered). Do this repetitively until there remains no uncovered point in $\mathcal{S}$.

It is already clear that this greedy algorithm makes a thoroughly R-covered node deployment in $\mathcal{S}$. So, we only need to show that the generated deployment by the above algorithm is $\mathcal{R}$-distant. By contradiction, assume that it is not; i.e. there are two nodes located at points $X \in \mathcal{S}$ and $Y \in \mathcal{S}$ such that the value $|\overline{XY}|$ is not greater than $\mathcal{R}$; or equivalently, Y is in the ball $B(X, \mathcal{R})$. Without loss of generality, assume that X is chosen earlier than Y in the greedy algorithm. So, when we chose

Y as a node location, all the points of $\mathcal{S}$ inside the ball $B(X, \mathcal{R})$ (including Y) had been earlier covered. This means that when we chose Y in the algorithm, it had been already covered; however, according to the algorithm, each node location should be uncovered at the time of being chosen.

6.4.3 Oblivious Routing Cost Environment

Note that in the proposed protocol of our model, we assume that any cyber device can ask for data in some specific moment, and its request may be satisfied by other devices in some time interval. Additionally, we have assumed that our cyber devices may move through the network and get connected to different routing nodes. These assumptions makes the distribution of the data traffic pattern in our model dynamic and time-sensitive.

Here, we consider the data flow of our model in ϵ-length time intervals (for some small $\epsilon > 0$). Assume that in interval $I_x = [x, x + \epsilon]$, there are $k(x)$ data flows through the network in a way that routing node $s_i(x)$ sends $b_i(x)$ bits of data to node $t_i(x)$ for every $i \in [1, k(x)]$ (note that $s_i(x)$ and $t_i(x)$ can be any two vertices of graph G and $b_i(x)$ is an arbitrary positive number). Henceforth, in order to find the data paths in time interval I_x, we need to solve a general routing problem of commodities $\bar{K} = (K_1(x), K_2(x), \ldots, K_{k(x)}(x))$ in graph G such that:

$$K_i(x) = \big(s_i(x), t_i(x), b_i(x)\big) \qquad \forall i \in k(x)$$

As you see, the sequence of commodities is a function of time x and may change during the time. To illustrate, assume that one device always asks for a huge bunch of data which leads to the traffic congestion in the routing node connected to the device. As the device moves through the network over time, it gets connected to different routing nodes which leads to changes in the target of some commodities.

At the rest of this chapter, we will focus on data routing through the network while satisfying the following couple of concerns:

i. Preventing data congestion in some specific routing node (*vertex-level* cost optimization): Data congestion increases the delay and energy consumption of our network. Moreover, as the cameras are power consumers, distributing the electrical load through the network will improve the energy efficiency.

ii. Keeping the total number of data sending/forwarding small (*network-level* cost optimization) as the total energy consumption of our system is substantially dependent to the amount of data transmitted between the routing nodes.

If our goal is to lessen the node traffic congestion, as mentioned in Chapter 1, we have to consider the edge routing cost function as the summation ($cost_e = \sum_f$); and also the nrc_π will be in the following form:

$$\mathrm{nrc}_\pi\big(c(e_1), c(e_2), \ldots, c(e_{|E|})\big) = \frac{1}{2} \max_{v \in V} \sum_{\substack{e \in E \\ v \in e}} c(e)$$

Consequently, we have to deal with a general routing problem in a time-varying routing cost environment in the form $\mathcal{E}(x) = (\sum, \frac{1}{2}\max_v \sum_e, \bar{K}(x))$. This implies that the general routing problem turns into an oblivious one when we consider a long time interval containing multiple ϵ-length intervals (note that for every time interval I_x, we may have a different unpredictable sequence of commodities $\bar{K}(x)$). Henceforth, the set of all the possible routing cost environments will be in the following form:

$$\mathbb{E} = \Big\{ \big(\sum, \frac{1}{2} \max_v \sum_e, \mathfrak{K} \big) \Big| \mathfrak{K} \in \mathcal{K} \Big\} \tag{6.8}$$

where $\mathcal{K}$ is the set of the commodity sequences which may contain any number of commodities and each of the commodities may have any source, target, and value.

Additionally, in order to address the second mentioned concern, we need to route data in a way that the following value doesn't exceed some high-threshold:

$$\mathcal{N}_{\mathbb{S}} = \max_{\substack{s,t \in V \\ s \neq t}} \frac{len_G(\mathbb{S}(s,t))}{d_G(s,t)} \tag{6.9}$$

where $\mathbb{S}$ is the oblivious routing scheme used for the data routing through the network and will be described in the next section. Value $\mathcal{N}_{\mathbb{S}}$ which is called as the "network cost factor" presents an upper-bound on the number of hops of a data flow routed by scheme $\mathbb{S}$ (which is equal to $\text{len}_G(\mathbb{S}(s,t))$) in comparison with the minimum possible number of hops ($d_G(s,t)$).

In the rest of this chapter, the oblivious routing scheme of our model will be addressed in more details. Moreover, we show that how the aforementioned concerns are considered in our proposed routing scheme.

6.5 An Oblivious Routing Scheme

As mentioned in Section 6.2, the cyber devices in our hybrid model communicate with each other using four types of messages during different phases of the proposed protocol: REQ, NOTIF, ACK, and DATA. The first three types are *control* messages while the last one contains the requested data. There are few number of control messages in a connection session; however, we may have multiple of DATA messages in a single connection. Additionally, the DATA messages are usually much larger in size than the control ones. Henceforth, we will restrict our routing cost discussion to DATA messages only.

As mentioned before, in order to route a DATA message in the "data transmission" phase, we need to first choose an intermediate routing node. Then, we use the aforementioned location-based routing scheme to send the DATA message from the supplier to the intermediate node and subsequently, forward it toward the

interested node which is the message ultimate target. In this section, we describe the routing process in more detail. More specifically, we explain the way of choosing the intermediate node.

Busch's Randomized Algorithm

Here, we present an algorithm to compute an oblivious routing scheme for our model. This algorithm which was originally proposed by Busch et al. in 2005 [2] uses a randomized solution to make the traffic pattern of an existing routing scheme distributed (node congestion free) and at the meanwhile, avoid a considerable increase in the total cost of data routing in the network level.

Remember that in our model, the network of routing nodes is represented by geometric graph $G = (V, E, loc)$ which is deployed in convex area $\mathcal{A}$ belonging to the Euclidean plane. In addition, let symbol $\mathbb{Q}$ denote the existing location-based routing scheme which is assumed to be implemented in the form of some non-centralized forwarding tables in the routing nodes. We call this scheme as the "default routing scheme". Algorithm 6 describes more details of the data routing process. Note that for every pair of vertices $s, t \in V$, we call path $\mathbb{Q}(s, t)$ as the *default path* between s and t; in fact, we call the paths proposed by the existing routing scheme as the default ones.

Algorithm 6 RandomizedRoutingAlgorithm

input: Vertices s and t (that are source and target vertices respectively)
output: Randomized path $\mathbb{S}(s, t)$ between s and t in graph G
$\overline{ST} \leftarrow$ the line segment connecting points $S = loc(s)$ and $T = loc(t)$
$\overline{UV} \leftarrow$ the perpendicular bisector line segment of $\overline{ST}$
/*Line segments a and b are perpendicular bisectors of each other if $a \perp b$ and each one bisects the other.*/
$\overline{XY} \leftarrow \overline{UV} \cap \mathcal{A}$
$x \leftarrow$ IntermediateVertexSelection($\overline{XY}$)
$\mathbb{S}(s, t) \leftarrow \mathbb{Q}(s, x) \oplus \mathbb{Q}(x, t)$

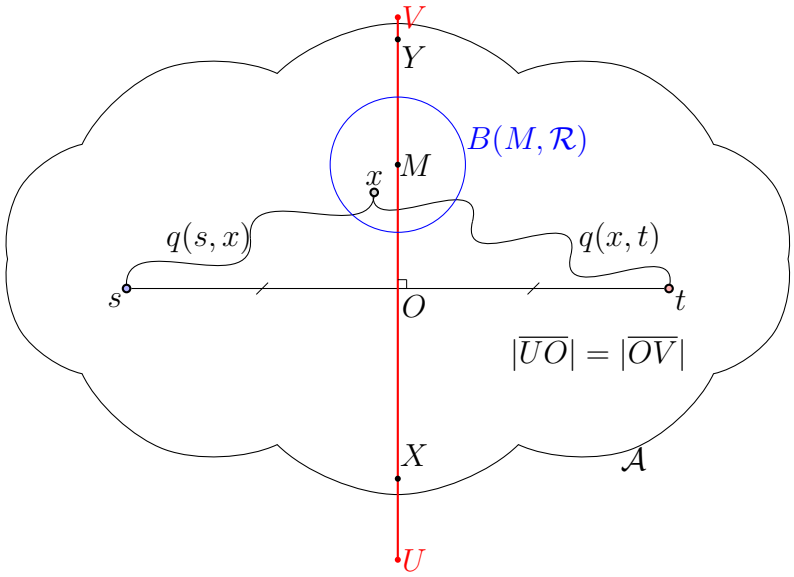


Figure 6.5: How the randomized algorithm find a path between source s and target t.

As you see in this algorithm, $\overline{ST}$ denotes the line segment connecting the source and target vertices. Moreover, $\overline{UV}$ represents the perpendicular bisecting line segment of $\overline{ST}$; i.e. $\overline{UV}$ is perpendicular to $\overline{ST}$ and each one bisects the other (See Figure 6.5). Since vertices s and t are located in a convex area ($\mathcal{A}$), the result of expression $\overline{UV} \cap \mathcal{A}$ is a line segment[5] ($\overline{XY}$). Moreover, function INTERMEDIATE VERTEX SELECTION determines the intermediate vertex x. Note that, the output path is obtained by merging paths $\mathbb{Q}(s, x)$ and $\mathbb{Q}(x, t)$ and is denoted by $\mathbb{S}(s, t)$. In other words, we use Algorithm 6 to create a randomized routing scheme $\mathbb{S}$ for the given default routing scheme $\mathbb{Q}$. Now, we address the details of function INTERMEDIATE VERTEX SELECTION in Algorithm 7.

[5]In fact, the value of $\overline{UV} \cap \mathcal{A}$ can also be a point; but we consider a point as a line segment of length zero.

Algorithm 7 IntermediateVertexSelection

input: Line segment l
output: The intermediate vertex x inside the area $\mathcal{A}$
$M \leftarrow$ A uniformly distributed random point on l
for every vertex x of graph G **do**
  if $loc(x) \in B(M, \mathcal{R})$ **then**
    return x
  end if
end for
/*x is one of the graph vertices that is in ball $B(M, \mathcal{R})$*/

As you see, in this algorithm, point M is chosen randomly on the input line segment. Since the input line segment completely belongs to area $\mathcal{A}$, point M is also inside this area. Consequently, as area $\mathcal{A}$ is thoroughly $\mathcal{R}$-covered by the network of routing nodes, there exists a routing node at-most $\mathcal{R}$ units far from point M. Lines 2 to 6 of the algorithm searches for the representing vertex of such routing node.

In the next two sections, we analyze the described routing scheme in the context of the vertex-level cost (node congestion) and the network-level routing cost which is evaluated using factor $\mathcal{N}_{\mathbb{S}}$ defined in Equation 6.9.

6.6 Node-Congestion Prevention

Here, we analyze the vertex-level cost of the randomized scheme presented in the previous section. To do this, we obtain an upper-bound for the expected competitiveness ratio of the versatile routing scheme in the oblivious routing problem defined in Section 6.4:

$$\mathbf{E}\big[CR(\mathbb{E}, \mathbb{S})\big] = \frac{\mathbf{E}\big[C_{\mathbb{S}}\big]}{C^*}$$

In the above equation, $\mathbb{E}$ is the set of possible routing cost environments defined in Equation 6.8, C^* is the optimal cost of the oblivious routing problem, and $C_{\mathbb{S}}$ is the cost of solution proposed by scheme $\mathbb{S}$.

6.6.1 Preliminary Definitions

For every convex subset in the Euclidean plane, a number is defined as its *pseudo-convexity factor* which is formally defined in Definition 6.6.1.

Definition 6.6.1. *If $\mathcal{S}$ denotes a convex set of points on the Euclidean plane, the pseudo-convexity factor $\gamma^{\mathcal{S}}$ is defined as:*

$$\gamma^{\mathcal{S}} = \inf_{A,B \in \mathcal{S}} \frac{|\overline{AB}^{\perp} \cap \mathcal{S}|}{|\overline{AB}|}$$

such that line segment $\overline{AB}^{\perp}$ denotes the perpendicular bisecting line segment[6] of $\overline{AB}$; and $|\overline{AB}^{\perp} \cap \mathcal{S}|$ is the length of the part of line segment $\overline{AB}^{\perp}$ that is inside set $\mathcal{S}$.

In the above definition, note that as A and B belong to the convex set $\mathcal{S}$, the result of expression $\overline{AB}^{\perp} \cap \mathcal{S}$ is always a line segment or a point (line segment of length zero). In Figure 6.6, you can see the pseudo-convexity factor of some familiar convex sets in the Euclidean plane. In this section, assume that γ represents the pseudo-convexity factor of convex area $\mathcal{A}$ over which the routing nodes are distributed .

Definition 6.6.2. *Let $G = (V, E, loc)$ denote a geometric graph in the Euclidean plane and p represent a path in G. Deviation of path p from line l in the plane is denoted by $dev(p, l)$ and defined in the following form:*

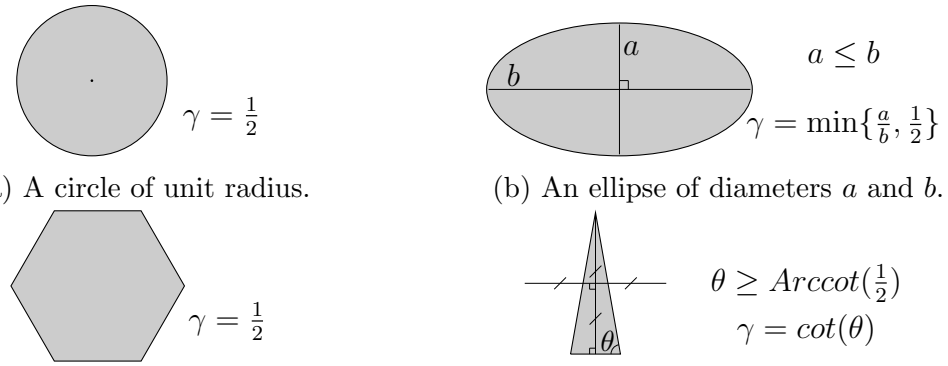
$$dev(p, l) = \max_{v \in N_p} distance(loc(v), l) \tag{6.10}$$

such that set $N_p \subseteq V$ denotes the set of vertices participating in path p and $distance(O, l)$ denotes the Euclidean distance from point O to line l[7].

---

[6] Line segment $\overline{AB}^{\perp}$ is perpendicular bisecting line segment of $\overline{AB}$, if and only if $\overline{AB}^{\perp} \perp \overline{AB}$, $\overline{AB}$ bisects $\overline{AB}^{\perp}$, and $\overline{AB}^{\perp}$ bisects $\overline{AB}$.

[7] Euclidean distance from point O to line l is defined as the length of the line segment perpendicular to line l from point O



(a) A circle of unit radius.

(b) An ellipse of diameters a and b.

(c) A regular hexagon.

(d) A bilateral triangle.

Figure 6.6: Pseudo-convexity of some familiar convex sets.

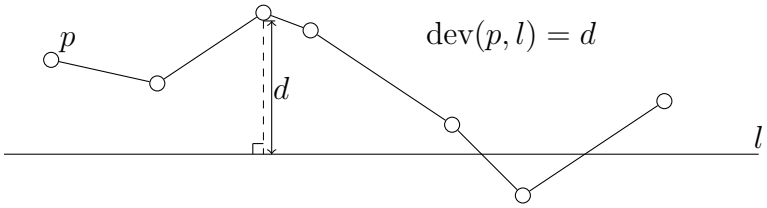


Figure 6.7: Deviation of path p from line l.

At the rest of this section, we consider the following value (δ) as the *deviation factor* of the default scheme $\mathbb{Q}$:

$$\delta = \max_{u,v \in V} \mathrm{dev}(\mathbb{Q}(u, v), \overleftrightarrow{UV})$$

such that $U = \mathrm{loc}(u)$, $V = \mathrm{loc}(v)$, and $\overleftrightarrow{UV}$ denotes the line passing through U and V. In fact, deviation factor δ specifies the maximum deviation of every default path from the straight line connecting its end vertices.

6.6.2 The Expected Competitiveness Ratio

At first, we compute a high-threshold for the probability of using some vertex $v \in V$ in path $\mathbb{S}(s, t)$ between vertices s and t where $\mathbb{S}$ is the oblivious scheme described in Section 6.4. As mentioned before, path $\mathbb{S}(s, t)$ is the union of two default paths $\mathbb{Q}(s, x)$ and $\mathbb{Q}(x, t)$ such that x is the intermediate vertex selected by Algorithm 7.

So, if path $\mathbb{S}(s,t)$ is passing through v, vertex v is either on the first part ($\mathbb{Q}(s,x)$) or on the second one ($\mathbb{Q}(x,t)$). We will focus on finding a high-threshold for the probability that v participates on $\mathbb{Q}(s,x)$; then, we extend our result to path $\mathbb{Q}(x,t)$.

Theorem 6.6.3. *Let x denote the intermediate vertex of path $\mathbb{S}(s,t)$ in Algorithm 6. The probability that vertex v participates in default path $\mathbb{Q}(s,x)$ has the following upper-bound:*

$$\boldsymbol{Pr}\Big[v \text{ is on } \mathbb{Q}(s,x)\Big] \leq \frac{5}{\gamma}\big(\frac{\mathcal{R}}{d_{loc}(s,t)} + \frac{\delta}{d_{loc}(s,v)}\big) \tag{6.11}$$

Proof. Let $X, S \in \mathcal{A}$ respectively denote the points that vertices x and s are located at. Regarding the Equation 6.10, this is the case that:

$$\text{dev}(\mathbb{Q}(s,x), \overleftrightarrow{SX}) \leq \delta$$

or equivalently,

$$\textit{distance}(\text{loc}(z), \overleftrightarrow{SX}) \leq \delta$$

for every vertex z on path $\mathbb{Q}(s,x)$. The last inequality implies that if path $\mathbb{Q}(s,x)$ passes through vertex v, the Euclidean distance of point $V = \text{loc}(v)$ from line $\overleftrightarrow{SX}$ will be less than or equal δ. As the result, point X must belong to the area $\mathcal{M}$ highlighted in Figure 6.8 (Note that in this figure, lines l and l' are passing through S and are tangent to the circle of radius δ and center V). Moreover, concerning Algorithm 7, vertex x must also be in ball $B(M,\mathcal{R})$ where M is a randomly distributed point on $\overline{S'T'}$ which is the perpendicular bisecting line segment of ST. As the result, point X should also be located in area $\mathcal{M}'$ which implies that:

$$\left.\begin{array}{l} X \in \mathcal{M} \\ X \in \mathcal{M}' \end{array}\right\} \rightarrow X \in \mathcal{M} \cap \mathcal{M}'$$

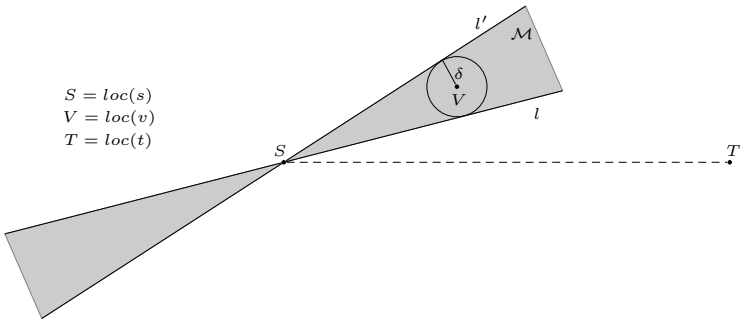


Figure 6.8: The highlighted area shows the set of points that vertex x can be possibly located in.

Area $\mathcal{M} \cap \mathcal{M}'$ has been highlighted by red color in Figure 6.9. Additionally, as the distance $||X - M||$ is not larger than $\mathcal{R}$, the point M can only be located on line segment $\overline{AB}$ specified in Figure 6.9. Consequently, we obtain the following proposition:

$$\big(v \text{ is on } \mathbb{Q}(s,x)\big) \rightarrow \big(M \in \overline{AB}\big)$$

which implies that:

$$\mathbf{Pr}\Big[v \text{ is on } \mathbb{Q}(s,x)\Big] \leq \mathbf{Pr}\Big[M \in \overline{AB}\Big] \tag{6.12}$$

As point M is uniformly distributed over line segment $\overline{S'T'} \cap \mathcal{A}$, this is the case that:

$$\begin{aligned}\mathbf{Pr}\Big[M \in \overline{AB}\Big] &= \frac{|\overline{AB}|}{|\overline{S'T'} \cap \mathcal{A}|} \\ &= \frac{|\overline{AB}|}{|\overline{S'T'}| \times \frac{|\overline{S'T'} \cap \mathcal{A}|}{|\overline{S'T'}|}}\end{aligned}$$

Moreover, regarding the definition of pseudo-convexity factor, we obtain the following inequality:

$$\begin{aligned}\gamma &\leq \frac{|\overline{S'T'} \cap \mathcal{A}|}{|\overline{ST}|} \\ &\leq \frac{|\overline{S'T'} \cap \mathcal{A}|}{|\overline{S'T'}|}\end{aligned}$$

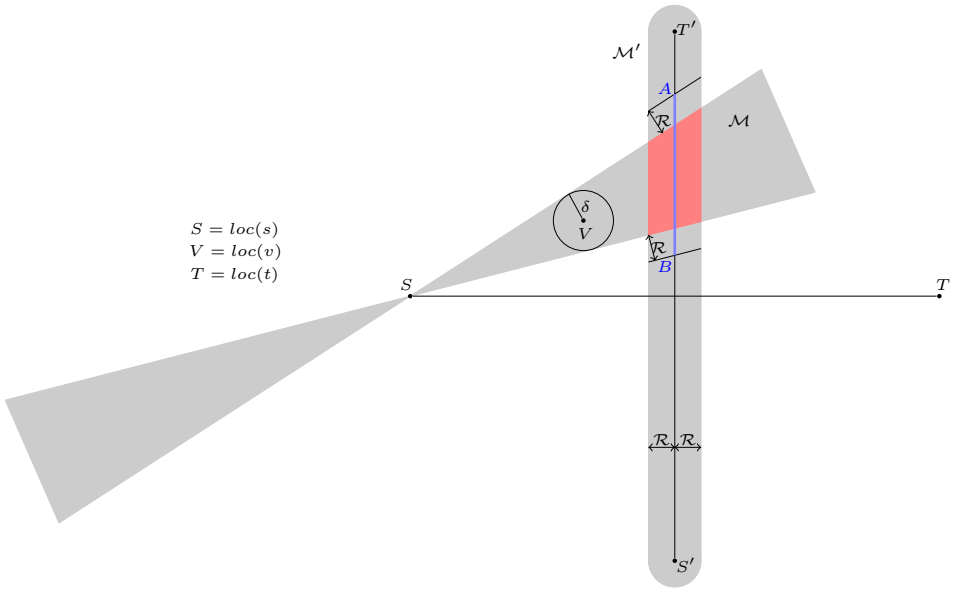


Figure 6.9: The blue line segment specifies the set of points to which point M belongs.

Henceforth, we obtain the following relation:

$$\mathbf{Pr}\Big[M \in \overline{AB}\Big] \leq \frac{|\overline{AB}|}{\gamma \times |\overline{S'T'}|}$$
$$\leq \frac{|\overline{AB}|}{\gamma \times d_{\text{loc}}(s,t)}$$

Regarding Inequality 6.12, we conclude that:

$$\mathbf{Pr}\big[v \text{ is on } \mathbb{Q}(s,x)\big] \leq \frac{|\overline{AB}|}{\gamma \times d_{\text{loc}}(s,t)}$$

In exercises, you are asked to geometrically prove the following upper-bound for the value of $|\overline{AB}|$:

$$|\overline{AB}| < 5(\mathcal{R} + \delta\frac{|\overline{ST}|}{|\overline{SV}|}) \tag{6.13}$$

As the result, we obtain Inequality 6.11.

□

Concerning Theorem 6.6.3, the expected value of $CR(\mathbb{E},\mathbb{S})$ is bounded to the following value:

$$\mathbf{E}\big[CR(\mathbb{E},\mathbb{S})\big] = \Theta\big(c\mathcal{R} \cdot \Psi(G) \cdot \max\{\mathcal{R},\delta\}^2\big) \tag{6.14}$$

6.7 Routing Cost Analysis

In this section, we address the network-level cost issues of routing scheme $\mathbb{S}$ presented in Section 6.5. In fact, we obtain an upper-bound for the value of the network cost factor $\mathcal{N}_{\mathbb{S}}$ defined in Equation 6.9. This upper-bound is a scaled value of the network cost factor of the default routing scheme ($\mathcal{N}_{\mathbb{Q}}$).

Let x denote the intermediate vertex chosen by Algorithm 6 when the input vertices are s and t. Since the randomized output path $\mathbb{S}(s,t)$ is equal to $\mathbb{Q}(s,x) \cup \mathbb{Q}(x,t)$, this is the case that:

$$\text{len}_G(\mathbb{S}(s,t)) = \text{len}_G(\mathbb{Q}(s,x)) + \text{len}_G(\mathbb{Q}(x,t))$$

Moreover, assuming $\mathcal{N}_{\mathbb{Q}}$ as the network-level cost factor of default scheme $\mathbb{Q}$, we obtain the following inequalities

$$\frac{\text{len}_G(\mathbb{Q}(s,x))}{d_G(s,x)} \leq \mathcal{N}_{\mathbb{Q}}$$
$$\frac{\text{len}_G(\mathbb{Q}(x,t))}{d_G(x,t)} \leq \mathcal{N}_{\mathbb{Q}}$$

The above relations lead us to the following inequality for $\text{len}_G(\mathbb{S}(s,t))$.

$$\text{len}_G(\mathbb{S}(s,t)) \leq \mathcal{N}_{\mathbb{Q}}(d_G(s,x) + d_G(x,t)) \tag{6.15}$$

In the next step, we will find an upper-bound for the RHS[8] expression of Inequality 6.15. Assuming $\Psi(G)$ as the pseudo-diameter of G, this is the case that:

$$\frac{d_G(s,x)}{d_{\text{loc}}(s,x)} \leq \Psi(G)$$
$$\rightarrow d_G(s,x) \leq \Psi(G) d_{\text{loc}}(s,x)$$

On the other hand, concerning Figure 6.10, if M denotes the point chosen in line 1 of Algorithm 7, and $S = \text{loc}(s)$, this is the case that:

$$d_{\text{loc}}(s,x) = ||S - \text{loc}(x)||$$

[8]Right Hand Side

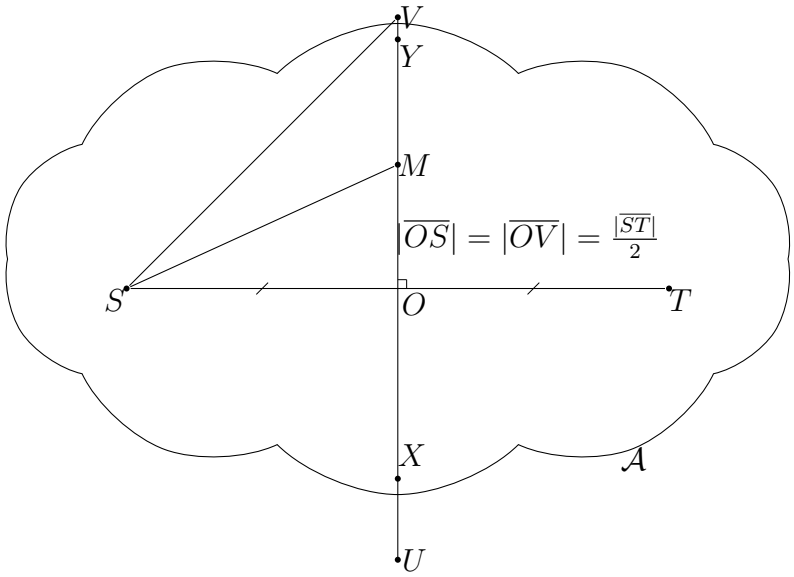


Figure 6.10: The geometric approach for computing an upper-bound for the network-level cost factor $\mathcal{N}_{\mathbb{S}}$.

and

$$||\text{loc}(x) - M|| \leq \mathcal{R}$$

As the result, regarding the triangular inequality, we obtain the following relation:

$$\begin{aligned} d_{\text{loc}}(s, x) &\leq |\overline{SM}| + ||\text{loc}(x) - M|| \\ &\leq |\overline{SM}| + \mathcal{R} \end{aligned}$$

In addition, regarding Figure 6.10 and Pythagorean Theorem, it is inferred that:

$$|\overline{SM}|^2 = |\overline{OM}|^2 + |\overline{SO}|^2$$

Without loss of generality, we assume that M is a point on line segment $\overline{YO}$ (in the similar case, it may be on $\overline{XO}$). Since $\overline{XY} \subseteq \overline{UV}$, this is the case that $|\overline{OM}| \leq |\overline{OY}| \leq |\overline{OV}|$; consequently, we obtain the following inequality:

$$\begin{aligned} |\overline{SM}|^2 &\leq |\overline{OV}|^2 + |\overline{SO}|^2 \\ &= 2(\frac{|\overline{ST}|}{2})^2 \end{aligned}$$

In the above relations, note that $|\overline{OV}| = |\overline{SO}| = \frac{|\overline{ST}|}{2}$.

So, we obtain the following inequality regarding $d_{\text{loc}}(s, x)$:

$$d_{\text{loc}}(s, x) \leq \frac{1}{\sqrt{2}}|\overline{ST}| + \mathcal{R}$$

which leads to the following upper-bound on the value of $d_G(s, x)$:

$$d_G(s, x) \leq \Psi(G)\frac{\sqrt{2}|\overline{ST}| + 2\mathcal{R}}{2}$$

In the similar way, we conclude the following inequality for $d_G(x, t)$:

$$d_G(x, t) \leq \Psi(G)\frac{\sqrt{2}|\overline{ST}| + 2\mathcal{R}}{2}$$

As the result, according to Inequality 6.15, this is the case that:

$$\begin{aligned}\text{len}_G(\mathbb{S}(s, t)) &\leq \mathcal{N}_{\mathbb{Q}}\Psi(G)(\sqrt{2}|\overline{ST}| + 2\mathcal{R}) \\ &\leq \mathcal{N}_{\mathbb{Q}}\Psi(G)(\sqrt{2}d_{\text{loc}}(s, t) + 2\mathcal{R})\end{aligned}$$

At the next step, we will find a relation between $d_{\text{loc}}(s, t)$ and $d_G(s, t)$. Considering p as the shortest path from s to t in G, the value $d_G(s, t)$ is by definition equal to $|p|$. Moreover, using the triangular inequality, we obtain the following relation:

$$\begin{aligned}d_{\text{loc}}(s, t) &\leq \delta_G(s, t) \\ &\leq \sum_{\{u,v\}\in p} d_{\text{loc}}(u, v) \\ &\leq \sum_{\{u,v\}\in p} c\mathcal{R} \\ &= c\mathcal{R} \sum_{\{u,v\}\in p} 1 = c\mathcal{R}d_G(s, t)\end{aligned}$$

Consequently, we find an upper-bound for $\text{len}_G(\mathbb{S}(s, t))$ in the following form:

$$\begin{aligned}\text{len}_G(\mathbb{S}(s, t)) &\leq \mathcal{N}_{\mathbb{Q}}\Psi(G)(\sqrt{2}c\mathcal{R}d_G(s, t) + 2\mathcal{R}) \\ &= \mathcal{N}_{\mathbb{Q}}\Psi(G)\mathcal{R}(\sqrt{2}c + \frac{2}{d_G(s, t)})d_G(s, t)\end{aligned}$$

Since $d_G(s,t)$ is a positive integer, $d_G(s,t) \geq 1$; henceforth:

$$\mathrm{len}_G(\mathbb{S}(s,t)) \leq \mathcal{N}_{\mathbb{Q}} \Psi(G) \mathcal{R}(\sqrt{2}c + 2) d_G(s,t)$$

Finally, we obtain the following upper-bound for $\mathcal{N}_{\mathbb{S}}$:

$$\begin{aligned} \mathcal{N}_{\mathbb{S}} &= \mathrm{len}_G(\mathbb{S}(s,t))/d_G(s,t) \\ &\leq (2 + \sqrt{2}c)\mathcal{R}\Psi(G)\mathcal{N}_{\mathbb{Q}} \\ &= \Theta(R\Psi(G))\mathcal{N}_{\mathbb{Q}} \end{aligned}$$

or equivalently,

$$\frac{\mathcal{N}_{\mathbb{S}}}{\mathcal{N}_{\mathbb{Q}}} \leq \Theta(R\Psi(G)) \tag{6.16}$$

As you see in Inequality 6.16, the network cost factor of the oblivious routing scheme obtained by Algorithm 6 will not be $\Theta(R\Psi(G))$ times more than the cost factor of the default one.

6.8 Summary and Outlook

In this chapter, we developed a hybrid model of a peer-to-peer network which provides a secure congestion-free way of distributing data over a network of routing nodes deployed in a convex subset of the Euclidean plane. This model uses a content-centric protocol for sending the requests; while a host-based scheme is used for routing the data flow through the network.

We used a security scheme for the model to manage the security issues based on the content which is being transformed. Additionally, the routing scheme applied in the model makes the data traffic pattern of our network distributed and node-congestion free.

Bibliography

[1] S. S. Iyengar and Kianoosh G. Boroojeni, "Oblivious Network Routing: Algorithms and Applications," MIT Press, 2015.

[2] Busch, Costas, Malik Magdon-Ismail, and Jing Xi. "Oblivious routing on geometric networks." Proceedings of the seventeenth annual ACM symposium on Parallelism in algorithms and architectures. ACM, 2005.

PART III

SECURITY AND PRIVACY ISSUES IN SMART GRIDS

CHAPTER 7

LOCATION PRIVACY ISSUES IN SMART GRIDS: A CASE STUDY ON ELECTRIC VEHICLES

Smart grid (SG) concept is introduced to achieve a sustainable, secure and environmentally friendly power system by using new elements such as distributed renewable resources, advanced metering infrastructure, and modern transportation in terms of electric vehicle (EV) utilization [1][2]. In recent years, he U.S. government targets to increase the penetration of modern EVs[3]. From a critical point of view, utilizing large number of EVs connected to the future power grid may threaten the reliability and stability of power grid [4] [5]. The society of automotive engineers (SAE) established some standards about the utilization of EVs including SAE J2847 which institutes requirements and specifications for communication between EVs and power system. This standard specifies interactions between EVs and power system operators [6]. According to [1], from the utilities perspective, it is not elaborately specified whether EV utilization in terms of vehicle to grid (V2G) is cost-effective [9]. In [10], authors introduced a comparison between direct and deterministic communication structure and proposed an aggregative command transmit architecture considering three influential factors, reliability, availability, and participating EVs in ancillary services.

Chapter 7 addresses the location privacy concerns that mobile components of the modern smart grid (e.g. electric vehicles) would have when they use a variety of location-based services on which the smart grid controllers rely for their main functionalities. Section 7.1 introduces the mobile components in smart grids like electric vehicles and their importance in a given smart grid. Section 7.2 introduces the preliminary concepts of location privacy and the trajectory revealing attacks.

Section 7.3 introduces a location privacy preservation mechanism. At the end, a summary and outlook of Chapter 7 will be presented.

7.1 Introduction

Based on the literature, in the modern transportation context, EVs require communicating information about the state of charge, desired charging rate, and their location data so that charging stations and EV aggregators estimate expected power consumption for next hours and provide an acceptable level of reliable service for EVs. There has been several studies about communication protocols, including, but not limited to: ZigBee which is implementable for small mesh networks, has low price, and high redundancy which requires less maintenance [7]; and Cellular Network which is applicable in long-range wireless systems and regarding to [8] this type of communication is feasible for EVs' data transfer. Additionally, collaborations between EVs and power system infrastructures, called V2X, are visualized to ameliorate traffic efficiency and driver welfare [11]. In order to implement V2X in a more secure way, there is an exigent need to propose a structure for concealing location and exact driving path of EVs.

Consequently, in the PHEV charging context, the vehicles usually transmit some data, such as identity, state of charge, usage pattern and location and these data used to be accessed by charging stations [12]. Therefore, one of the main privacy concerns is to protect customers' data, especially the vehicles' location [13]. The location privacy of EVs includes, but not limited to, drivers home address, working location, and favorite places to travel [14][15]. Some efforts focused on evaluating the effectiveness of location privacy methods in V2X [16][17]. This work introduces a novel algorithmic structure not only to conceal the location of mobile devices

Item	Description
knowledge	Important data about EV drivers including their location, user ID for each car,current SOC, desired SOC, and car model.
Interest of adversary	If a third party have access to this data, it can be used to make unpredictable peak load in a specific region/feeder in power network
Possible adversaries	1) High penetration of EVs means more sensitivity especially in the power system demand estimation considering EV's charging demand . Therefore, accessing to this kind of information can help attacers to increase the demand in a specific region by manipulating the transferred data. 2) The exact driving patterns can be extracted based on location information. Therefore it can be used to classify car drivers' behavior by advertising companies without drivers' permission. 3) Location of cars is critical for traffic management and it should not be accessible conveniently by third parties.

Table 7.1: Preliminary Definitions regarding Location Privacy Issues Facing Electric Vehicles

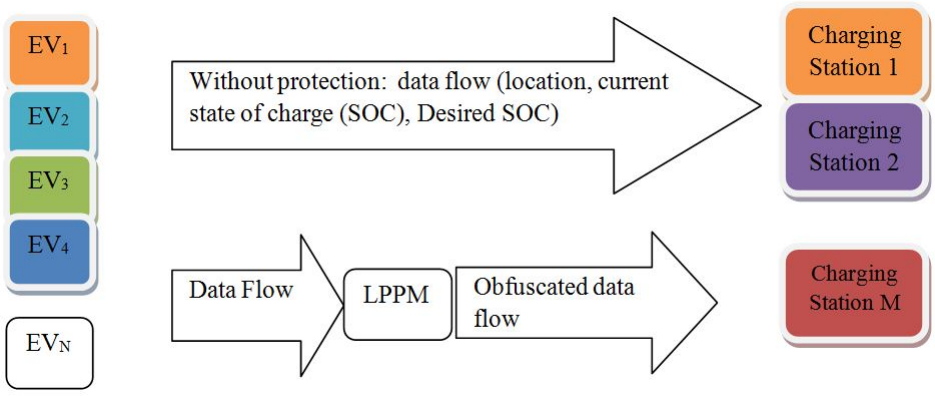


Figure 7.1: Schematic View of Location Obfuscation Methods in Electric Vehicle Networks.

but also to obfuscate their path movement information. Table 7.1 summarizes the preliminary definitions regarding location privacy issues facing EVs. Also, Figure 7.1 depicts a schematic view of location obfuscation methods in electric vehicle networks.

7.2 Preliminaries on Mobile Nodes Trajectory Privacy

In 2011, Shokri et al. [20,21] formalized localization attacks using Bayesian inference for Hidden Markov Processes. They mathematically modeled a location privacy-preserving mechanism, users mobility pattern and adversary knowledge-base. They also divide the LBSs into two classes: those who *sporadically* ask their users to expose their location, and those who *continuously* do. Additionally, they quantified the user location privacy as the expected distortion of adversary's guess from the reality of user's location. They used this quantification method to evaluate the effectiveness of location obfuscation and fake location injection mechanisms in the improvement of location privacy level.

Let c denote a mobile node in an Euclidean plane ($\mathbb{R}^2$) that wants to use a location-based service (say A). Assume that c is walking on the plane to do some job; for example, let c be a *sensor* which wants to control some criteria on the 2-D plane. Additionally, assume that there is no obstacle on the plane and node c can go anywhere on the plane (we leave the case with obstacles for future work). We discretize the plane by partitioning it into an infinite countable number of congruent distinct regions. As the result, node c is located in one of the regions at any given moment. Let r_t denote a random process which specifies the region in which c is located at moment $t \geq 0$.

Location-Based Service

Assume that node c sends a query message containing its location to location-based service A every one time unit (which can be any value) and gets benefit of its service. Here is the format of the query message that node c sends to A at time $t = n$:

$$\mathcal{Q}_c(n) = \langle \mathrm{ID}_c, r_n, \mathrm{DATA} \rangle \tag{7.1}$$

where ID_c specifies the ID of node c, r_n denotes the region where c is located in at time n, and DATA represents the other information that c may need to send to the LBS.

This kind of communication makes the LBS able to keep track of node c during the communication time. This may compromise its privacy (as the LBS or some third party (like data sniffer) may abuse this information). Consequently, we need to define a Location Privacy-Preserving Mechanism (LPPM) to filter the query message before sending it to the LBS. In this paper, we use a location-obfuscation method to filter this information and increase the privacy level of node c.

Location Privacy-Preserving Mechanism

In order to preserve the location privacy, node c obfuscates its current location using an LPPM before sending it to the LBS. In fact, instead of sending query $\mathcal{Q}_c(n)$ at time $t = n$, it sends $\mathcal{Q}'_c(n)$ such that:

$$\mathcal{Q}'_c(n) = \langle \mathrm{ID}_c, \mathrm{obf}(r_n), \mathrm{DATA} \rangle \tag{7.2}$$

where obf: $\mathcal{R} \mapsto 2^{\mathcal{R}}$ and $\mathcal{R} = \{r_0, r_1, r_2, \ldots\}$ (note that function obf maps every region to a set of regions). In this way, node c sends an obfuscated location each time instead of revealing its exact location to the LBS. This obfuscated location is obtained by the following equation:

$$\mathrm{obfuscated}(r_t) = \bigcup_{\rho \in \mathrm{obf}(r_t)} \rho \tag{7.3}$$

We call the sequence of regions $\bar{r} = r_0, r_1, r_2, \ldots$ as the *(actual) track* of node c; while the sequence: $\mathrm{obfuscated}(r_0), \mathrm{obfuscated}(r_1), \mathrm{obfuscated}(r_2), \ldots$ is the corresponding *obfuscated* track of c.

Adversary

As mentioned before, the LBS itself is considered to be a potential adversary. Additionally, some third party may eavesdrop the query messages originated by node c to use them for some malicious purposes. In this paper, we assume that the adversary knows the LPPM (which is function obf) used by node c to filter the query message. Additionally, at time $t = n$, she is aware of the obfuscated track sent by c in time interval $[0, n]$. Using this information, the adversary wants to reasonably guess the actual track of node c. Let $\bar{\hat{r}} = \widehat{r}_0, \widehat{r}_1, \ldots$ denote the adversary guess of the actual track such that for every i, $\widehat{r}_i \in \mathcal{R}$ specifies the adversary guess of region r_i in which node c is located at time $t = i$.

7.3 Privacy Preservation Mechanisms and Quantification: A Probabilistic Approach

This section describes a novel location privacy-preserving mechanism which obfuscates the location information of mobile node c to increase its privacy. In order to specify such a mechanism, we need to define the output value of the aforementioned function obf for every given input region $r \in \mathcal{R}$.

In this section, we assume that set $\mathcal{R}$ partitions the Euclidean plane into an infinite number of unit squares such that:

$$\forall \rho \in \mathcal{R}, \exists x, y \in \mathbb{R} : \rho = [x, x+1) \times [y, y+1) \tag{7.4}$$

Additionally, we discretize the time space into the set of non-negative integer (as we have already assumed that the query messages are sent every single time unit).

Algorithm 8 specifies how the mechanism works.

Algorithm 8 RegionObfuscator

input: region ρ, time n, & scale factor α
output: obfuscated region o & boundary factor $\mathcal{B}$
if $n = 0$ **then**
  $x \leftarrow \mathcal{U}\text{nif}(-\frac{1}{6}, \frac{1}{6})$
  $y \leftarrow \mathcal{U}\text{nif}(-\frac{1}{6}, \frac{1}{6})$
  $o \leftarrow$ A square of edge length α and
  centroid
  ρ.centroid
  /* edges are parallel to x-y axes*/
  $o \leftarrow \textsc{Translation}\big(o, (x\alpha, y\alpha)\big)$
  $\mathcal{B} \leftarrow \mathcal{U}\text{nif}(\frac{1}{3}, 1)$
else
  $o' \leftarrow$ the obfuscated region of time $n-1$
  $\mathcal{B}' \leftarrow$ the boundary factor of time $n-1$
  $\mathcal{S} \leftarrow$ the square of edge length $\mathcal{B}'\alpha$
  and centroid $o'.centroid$
  if $o \subset \mathcal{S}$ **then**
    $o \leftarrow o'$
    $\mathcal{B} \leftarrow \mathcal{B}'$
  else
    $o \leftarrow \textsc{Update}(\rho, o')$
    $\mathcal{B} \leftarrow \mathcal{U}\text{nif}(\frac{1}{3}, 1)$
  end if
end if
return $(o, \mathcal{B})$

As you see, function RegionObfuscator gets region $\rho \in \mathcal{R}$, time $t = n$, and scale factor α as its input and calculates the corresponding obfuscated region and *boundary factor* (which will be addressed later) as the output.

In the case that $n = 0$, the obfuscated region is a square of edge length α with parallel edges to x-y axes. The square is centered at a point obtained by randomly translating the ρ's centroid:

$$o.\text{centroid} = \rho.\text{centroid} + (x\alpha, y\alpha) \tag{7.5}$$

where x and y are uniformly distributed over interval $(-1/6, 1/6)$. Additionally, the boundary factor is obtained by generating a sample of random variable $\mathcal{U}\text{nif}(1/3, 1)$.

As you see in Algorithm 8, in the case that $n > 0$, function RegionObfuscator first needs to check whether the current obfuscated region and boundary factor (which were generated at time $t = n - 1$) are *valid* for the current region ρ. If yes, it simply returns the previous values; otherwise, it generates a new boundary factor ($\mathcal{B}$) and calls function Update to generate a new region (o).

Here is the validation rule: if region ρ lies inside the square $\mathcal{S}$ (which is centered at o'.centroid and has the edge length of $\mathcal{B}'\alpha$), the current obfuscated region (o') and boundary factor ($\mathcal{B}'$) are still valid at time $t = n$.

Now, we consider the case that the validation rule doesn't hold. In other words, node c is no longer inside square $\mathcal{S}$. Let l denote the line segment connecting points r_{n-1}.centroid and r_n.centroid (note that $\rho = r_n$). Assuming that $\mathcal{B}' > 2/3$, there are two possible cases:

Case 1: If l intersects b_i, region o will be the area inside a square of edge length α and centroid M_i' (for every $i = 1, 2, 3, 4$).

Case 2: If l intersects b_i', region o will be the area inside a square of edge length α and centroid N_i (for every $i = 1, 2, 3, 4$). Assuming that random variable Z is uniformly distributed over interval $(-1/6, 1/6)$, point $N_i = (X_i, Y_i)$ is obtained by the following equations:

$$X_i = \begin{cases} \rho.\text{centroid}.x + \alpha Z & i = 1 \\ o'.\text{centroid}.x + \frac{\alpha}{3} & i = 2 \\ \rho.\text{centroid}.x + \alpha Z & i = 3 \\ o'.\text{centroid}.x - \frac{\alpha}{3} & i = 4 \end{cases} \tag{7.6}$$

$$Y_i = \begin{cases} o'.\text{centroid}.y + \frac{\alpha}{3} & i = 1 \\ \rho.\text{centroid}.y + \alpha Z & i = 2 \\ o'.\text{centroid}.y - \frac{\alpha}{3} & i = 3 \\ \rho.\text{centroid}.y + \alpha Z & i = 4 \end{cases} \tag{7.7}$$

In addition, if $\mathcal{B}' \leq 2/3$, there exists only one possible case (Case 2), as $b_i = \emptyset$ for every $i = 1, 2, 3, 4$.

This section presents an extension of the Vornoi-based scheme, described in Chapter 6, for obfuscating the location and trajectory of a mobile node. We restrict our consideration to the case that the node has subsequent random close by destinations. More precisely, the node has a sequence of independently chosen random destinations in the form $\mathcal{D}_0 = \text{loc}_c(0), \mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \ldots$ such that for every $i \geq 0$, the node moves from point $\mathcal{D}_i \in \mathbb{R}^2$ to $\mathcal{D}_{i+1} \in \mathbb{R}^2$ in time interval $[t_i, t_{i+1})$ such that $t_0 = 0$,

$$0 < t_{i+1} - t_i \leq \epsilon \qquad \forall i = 0, 1, \ldots$$

for some small real number $\epsilon > 0$, and assuming that the maximum speed of the node is represented by M_s, this is the case that:

$$\epsilon M_s \ll \mu$$

or equivalently, the distance between two consequent destinations $||\mathcal{D}_{i+1} - \mathcal{D}_i||$ is negligible (compared to the scale factor).

7.3.1 A Stochastic Model of the Node Movement

We define random processes x_t and y_t in the following form:

$$\begin{cases} x_t = x_c(t) - x_c(0) \\ y_t = y_c(t) - y_c(0) \end{cases} \qquad \forall t \geq 0 \tag{7.8}$$

where $\big(x_c(t), y_c(t)\big)$ denotes the Cartesian coordinates of the node's location in time t ($\mathrm{loc}_c(t)$).

Considering the aforementioned assumptions regarding the node movement on the plane, we estimate x_t and y_t using random processes $\hat{x}_t$ and $\hat{y}_t$ which have the following properties:

1. Maps $g : t \mapsto \hat{x}_t(\omega)$ and $g' : t \mapsto \hat{y}_t(\omega)$ are continuous for every ω and $t > 0$.
2. For every $k \in \mathbb{N}$, assuming that $0 \leq t_1 \leq t_2 \leq \ldots \leq t_k$, the random variables belonging to the following set are mutually independent:

   $$\{(\hat{x}_{t_{i+1}} - \hat{x}_{t_i}) | i = 1 \ldots k-1\}$$

   The same proposition is true for the following set:

   $$\{(\hat{y}_{t_{i+1}} - \hat{y}_{t_i}) | i = 1 \ldots k-1\}$$

3. Every increment of processes $\hat{x}_t$ and $\hat{y}_t$ is stationary; i.e. the probability distributions of $\hat{x}_t - \hat{x}_s$ and $\hat{y}_t - \hat{y}_s$ only depend on $t - s$ for every $t, s > 0$.

Note that since $\mathrm{loc}_c(t) = \big(x_t + x_c(0), y_t + y_c(0)\big)$, the first mentioned property is also true for x_t and y_t:

$$\begin{cases} x_t = \lim\limits_{w \to t^+} x_w = \lim\limits_{w \to t^-} x_w \\ y_t = \lim\limits_{w \to t^+} y_w = \lim\limits_{w \to t^-} y_w \end{cases}$$

However, the other two properties are reasonably estimated regarding processes x_t and y_t.

Random processes $\hat{x}$ and $\hat{y}$ are known as Brownian Motion process where $\hat{x}_t \sim \mathcal{N}(0, \sigma^2 t)$ and $\hat{y}_t \sim \mathcal{N}(0, \sigma^2 t)$. Also, regarding Equation 7.8, we find an estimation stochastic process for $x_c(t)$ and $y_c(t)$:

$$\begin{cases} x_c(t) \sim \mathcal{N}(x_c(0), \sigma^2 t) \\ y_c(t) \sim \mathcal{N}(y_c(0), \sigma^2 t) \end{cases} \tag{7.9}$$

7.3.2 Proposed Scheme for A Mobile Node

Now, we extend our scheme to the case that node c is assumed to be moving in the way mentioned previously.

Algorithm 9 proposes an appropriate procedure which generates an anonymity zone for mobile node c at time $t = 0$ and keeps it updated as the node is moving for every $t \geq 0$. In this procedure, we assume that function AZGenerator(O, n, μ, γ_n) works in the following way: consider Algorithm 9 which generates the initial anonymity zone for a static node. If we change the second line of this algorithm to the form of Expression 7.10, we obtain another function called MobileZoneGenerator(O, n, μ, γ_n) where γ_n is a positive real number less than $2\sin^2(\frac{\pi}{n})$. Function AZGenerator returns the zone generated by applying the greedy algorithm mentioned in Section three on the output zone of function MobileZoneGenerator.

$$d \leftarrow \text{Unif}\Big(\mu\big(\cos(\frac{2\pi}{n}) + \gamma_n\big), \mu\Big) \tag{7.10}$$

Algorithm 9 MobileZoneUpdater

```
privacy level λ & odd integer n ≥ 3 & scale factor μ > 0 & γ_n < 2 sin^2(π/n)
while true do
  t ← Now()
  /*Function Now() returns the current time t ≥ 0.*/
  if loc_c(t) ∈ S_c then
    continue
  end if
  S_c ← AZGenerator(loc_c(t), n, μ, γ_n)
end while
```

7.3.3 Computing the Instantaneous Privacy Level

Now, we find a lower-bound for the node's privacy level at any given time $t > 0$. Without loss of generality, we assume that the anonymity zone S_c has been generated

at time $t = 0$ and kept unchanged in time interval $[0, T]$.

$$\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\mathrm{loc}_c(t) \in S_c\Big] = \iint\limits_{\substack{(x_0,y_0)\\ \in S_c}} \mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\mathrm{loc}_c(0) = (x_0, y_0) \wedge \mathrm{loc}_c(t) \in S_c\Big] \times \mathbf{Pr}\big[\mathrm{loc}_c(0) = (x_0, y_0)\big]\mathrm{d}x_0\mathrm{d}y_0 \tag{7.11}$$

Additionally, considering the estimated stochastic model of $\mathrm{loc}_c(t) = \big(x_c(t), y_c(t)\big)$ in Relation 7.9, we obtain the following relations:

$$\begin{aligned}
&\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\begin{matrix}\mathrm{loc}_c(0) = (x_0, y_0)\\ \wedge \mathrm{loc}_c(t) \in S_c\end{matrix}\Big]\\
&\quad = \mathbf{Pr}\Big[\mathrm{loc}_c(t) \in B\big(G, r\big)\Big|\begin{matrix}\mathrm{loc}_c(0) = (x_0, y_0)\\ \wedge \mathrm{loc}_c(t) \in S_c\end{matrix}\Big]\\
&\quad = \iint\limits_{\substack{(x,y)\\ \in B(G,r)}} \mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) = (x, y)\Big|\big(x_c(t), y_c(t)\big) \in S_c\Big]\\
&\quad \leq \frac{\iint_{\substack{(x,y)\\ \in B(G,r)}} \mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) = (x, y)\Big]}{\mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) \in S_c\Big]}
\end{aligned}$$

Since for every $t \leq T$, $x_c(t)$ and $y_c(t)$ are normal distributed random variables of mean x_0 and y_0 respectively, this is the case that $(O = loc_c(0) = (x_0, y_0))$:

$$\iint\limits_{\substack{(x,y)\\ \in B(G,r)}} \mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) = (x, y)\Big] \leq \iint\limits_{\substack{(x,y)\\ \in B(O,r)}} \mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) = (x, y)\Big]$$

which implies that:

$$\begin{aligned}
\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\begin{matrix}\mathrm{loc}_c(0) = (x_0, y_0)\\ \wedge \mathrm{loc}_c(t) \in S_c\end{matrix}\Big] &\leq \frac{\iint_{\substack{(x,y)\\ \in B(O,r)}} \mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) = (x, y)\Big]}{\mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) \in S_c\Big]}\\
&\leq \frac{\int_{r'=0}^{r}\int_{\theta=0}^{2\pi} \frac{r'}{2\pi\sigma^2} \cdot \mathrm{e}^{-\frac{r'^2}{2\sigma^2 t}}\mathrm{d}r'\mathrm{d}\theta}{\mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) \in S_c\Big]}
\end{aligned} \tag{7.12}$$

Here, we make a claim which will be proved later:

$$B\big(O, r_{\min}\big) \subseteq S_c \tag{7.13}$$

where:

$$r_{\min} = \frac{\mu\gamma_n}{2\sin(\frac{2\pi}{n})} \tag{7.14}$$

Regarding Relation 7.13, we obtain the following inequality:

$$\begin{aligned}
\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big| \begin{matrix}\mathrm{loc}_c(0) = (x_0, y_0)\\ \wedge \mathrm{loc}_c(t) \in S_c\end{matrix}\Big] \leq & \frac{\int_{r'=0}^{r}\int_{\theta=0}^{2\pi} \frac{r'}{\sqrt{2\pi t\sigma^2}} \cdot \mathrm{e}^{-\frac{r'^2}{2t\sigma^2}}\mathrm{d}r'\mathrm{d}\theta}{\mathbf{Pr}\Big[\big(x_c(t), y_c(t)\big) \in B\big(O, r_{\min}\big)\Big]} \\
\leq & \frac{\int_{r'=0}^{r}\int_{\theta=0}^{2\pi} \frac{r'}{\sqrt{2\pi t\sigma^2}} \cdot \mathrm{e}^{-\frac{r'^2}{2t\sigma^2}}\mathrm{d}r'\mathrm{d}\theta}{\int_{r'=0}^{r_{\min}}\int_{\theta=0}^{2\pi} \frac{r'}{\sqrt{2\pi t\sigma^2}} \cdot \mathrm{e}^{-\frac{r'^2}{2t\sigma^2}}\mathrm{d}r'\mathrm{d}\theta} \\
\leq & \frac{1 - e^{-\frac{r^2}{2t\sigma^2}}}{1 - e^{-\frac{r_{\min}^2}{2t\sigma^2}}}
\end{aligned} \tag{7.15}$$

Using Equation 7.11 and Inequality 7.15, we conclude that:

$$\begin{aligned}
&\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\mathrm{loc}_c(t) \in S_c\Big] \\
&\leq \frac{1 - e^{-\frac{r^2}{2t\sigma^2}}}{1 - e^{-\frac{r_{\min}^2}{2t\sigma^2}}} \times \iint\limits_{\substack{(x_0,y_0)\\ \in S_c}} \mathbf{Pr}\big[\mathrm{loc}_c(0) = (x_0, y_0)\big]\mathrm{d}x_0\mathrm{d}y_0
\end{aligned} \tag{7.16}$$

We replace $\mathbf{Pr}\big[\mathrm{loc}_c(0) = (x_0, y_0)\big]$ by its upper-bound in Inequality 7.16:

$$\begin{aligned}
&\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\mathrm{loc}_c(t) \in S_c\Big] \\
&\leq \frac{1 - e^{-\frac{r^2}{2t\sigma^2}}}{1 - e^{-\frac{r_{\min}^2}{2t\sigma^2}}} \times \iint\limits_{\substack{(x_0,y_0)\\ \in S_c}} \mathbf{Pr}\big[\bigwedge_{i=0}^{m-1} X'_i = \mathrm{dist}((x_0, y_0), l'_i)\big] \\
&\qquad\qquad\qquad\qquad\qquad\qquad \mathrm{d}x_0\mathrm{d}y_0
\end{aligned} \tag{7.17}$$

where line l'_i and function dist are defined as the same as what mentioned previously. In addition, similar to the proof of recent claim, we can get the following inequality:

$$\mathbf{Pr}\big[\bigwedge_{i=0}^{m-1} X'_i = \mathrm{dist}((x_0, y_0), l'_i)\big] \leq \zeta_n\big(\frac{r}{\mu}\big)^3 \tag{7.18}$$

for some real positive sequence ζ_n. Inequalities 7.17 and 7.18 imply that:

$$\mathbf{Pr}\Big[G \in B\big(\mathrm{loc}_c(t), r\big)\Big|\mathrm{loc}_c(t) \in S_c\Big] \leq \frac{1 - e^{-\frac{r^2}{2t\sigma^2}}}{1 - e^{-\frac{r_{\min}^2}{2t\sigma^2}}} \times \zeta_n\big(\frac{r}{\mu}\big)^3|S_c|$$

It is easy to see that the recent claim is also true for the mobile case; i.e area S_c belongs to ball $B\big(O, \mu\sin(\frac{\pi}{n})\big)$. As the result, this is the case that:

$$\begin{aligned} |S_c| &\leq |B\big(O, \mu\sin(\frac{\pi}{n})\big)| \\ &\leq \pi\mu^2\sin^2(\frac{\pi}{n}) \end{aligned}$$

Henceforth, we obtain a lower-bound for the instantaneous privacy level of mobile node c:

$$\lambda(t) \leq \pi\mu^2\sin^2(\frac{\pi}{n})\zeta_n\frac{1-e^{-\frac{r^2}{2t\sigma^2}}}{1-e^{-\frac{r_{\min}^2}{2t\sigma^2}}} \times \big(\frac{r}{\mu}\big)^3$$

or,

$$\lambda(t) \leq \zeta'_n\frac{1-e^{-\frac{r^2}{2t\sigma^2}}}{1-e^{-\frac{r_{\min}^2}{2t\sigma^2}}} \times \big(\frac{r^3}{\mu}\big) \tag{7.19}$$

for some positive real sequence ζ'_n.

To complete our analysis, we need to show Relation 7.13. Remember the notation X'_i which specifies the Euclidean distance between the static node's location O and the i^{th} edge of the polygon S_c for every $i = 0 \ldots m-1$. The change we made in this algorithm will increase the minimum possible value of random variable X'_i from zero to $r_{\min}$:

$$r_{\min} = \frac{\mu\gamma_n}{2\sin(\frac{2\pi}{n})}$$

Henceforth, we conclude Relation 7.13.

7.3.4 Concealing the Movement Path

In order to preserve the location privacy of a mobile node, not only we need to hide its instantaneous location, but we have to conceal its movement path in some extent. In our stochastic scheme, we quantifies the privacy level of the node's path by calculating a probabilistic low-threshold for random variable T that is the length

of the time interval in which function MobileZoneUpdater (Algorithm 9) keeps the anonymity zone unchanged.

As mentioned before, $B(O, r_{\min}) \subseteq S_c$. This implies that:

$$\sup_{t \leq t'} \big\{ ||\mathrm{loc}_c(t) - \mathrm{loc}_c(0)|| \big\} \leq r_{\min} \rightarrow T \geq t'$$

Subsequently, we obtain the following proposition:

$$\begin{aligned} \sup_{t \leq t'} \big\{ x_c(t) - x_c(0) \big\} \leq \frac{r_{\min}}{\sqrt{2}} \wedge \sup_{t \leq t'} \big\{ y_c(t) - y_c(0) \big\} \leq \frac{r_{\min}}{\sqrt{2}} \\ \rightarrow T \geq t' \end{aligned} \tag{7.20}$$

Now, we defined processes M_t and M'_t in the following form:

$$\begin{cases} M_{t'} = \sup_{t \leq t'} \big\{ x_c(t) - x_c(0) \big\} \\ M'_{t'} = \sup_{t \leq t'} \big\{ y_c(t) - y_c(0) \big\} \end{cases} \tag{7.21}$$

Concerning Proposition 7.20, we obtain the following inequality:

$$\mathbf{Pr}\big[T \geq t'\big] \geq \mathbf{Pr}\big[M_{t'} \leq \frac{r_{\min}}{\sqrt{2}}\big] \times \mathbf{Pr}\big[M'_{t'} \leq \frac{r_{\min}}{\sqrt{2}}\big] \tag{7.22}$$

Processes M_t and M'_t respectively represent the running maximum[1] of processes $\big(x_c(t) - x_c(0)\big)$ and $\big(y_c(t) - y_c(0)\big)$ which has been previously estimated by two Brownian motion processes of variance σ^2. As the result, this is the case that:

$$\begin{cases} \mathbf{Pr}\big[M_t \leq m\big] = \mathrm{erf}\big(\frac{m}{\sqrt{2t\sigma^2}}\big) \\ \mathbf{Pr}\big[M'_t \leq m\big] = \mathrm{erf}\big(\frac{m}{\sqrt{2t\sigma^2}}\big) \end{cases} \tag{7.23}$$

Consequently, we obtain a probabilistic low-threshold for random variable T:

$$\mathbf{Pr}\big[T \geq t'\big] \geq \mathrm{erf}^2\big(\frac{r_{\min}}{2\sqrt{t'\sigma^2}}\big) \tag{7.24}$$

[1]Running maximum M_t of the Brownian Motion process B_t is a random process which has the following cumulative density function at the arbitrary time $t > 0$ (σ^2 represents the variance of process B_t): $F_{M_t}(m) = \mathrm{erf}(\frac{m}{\sqrt{2t\sigma^2}})$ for every $m \geq 0$.

7.4 Summary and Conclusion

In this chapter, we addressed the location privacy of mobile devices connected to smart grids, especially a randomly walking node on the Euclidean plane which continuously exposes its location to an LBS (or possibly an adversary). Then, we quantified the privacy-level of the (sensor) node over time by computing the expected distortion of adversary's guess from the reality of node's trajectory. Additionally, the trade-off between the privacy level and maximum error tolerance of the mobile node was examined closely. As a result, the minimum privacy level occurs at the moments that the obfuscated location is being updated and we obtained that $1 - \lambda_{\min} = \Theta(\varepsilon_{\max}^{-2})$. Finally, we used some simulations to support our theoretical results and prove the efficacy of our method in practice.

Bibliography

[1] Rinku Dewri, "Location Privacy and Attacker Knowl- edge: Who Are We Fighting Against?," Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 96, pp 96-115, 2012.

[2] Ming Li, Sergio Salinas, Arun Thapa, Pan Li, "n-CD: A Geometric Approach to Preserving Location Privacy in Location-Based Services," Proc. IEEE INFOCOM, 2013.

[3] S. S. Iyengar, Kianoosh G. Boroojeni, and N. Balakrishnan, "Mathematical Theories of Distributed Sensor Networks," Springer, pp 111-145, 2014.

[4] Rui Shi, Mayank Goswami, Jie Gao, and Xian- feng Gu, "Is Random Walk Truly Memory-less - Traffic Analysis and Source Location Privacy under Random Walks," INFOCOM, 2013 Proceedings IEEE, Pages 3021-3029, Turin, Italy, 2013.

[5] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in ACM Mobisys'03, May 2003.

[6] B. Gedik and L. Liu, "Protecting location privacy with personalized kanonymity: Architecture and algorithms," IEEE Transactions on Mobile Computing, vol. 7, no. 1, pp. 118, January 2008.

[7] J. Meyerowitz and R. R. Choudhury, "Hiding stars with fireworks: Location privacy through camouflage," in Proceedings of ACM MobiCom, Beijing, China, September 2009.

[8] M. F. Mokbel, C. Y. Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in Proceedings of VLDB, 2006.

[9] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias, "Preventing location-based identity inference in anonymous spatial queries," IEEE Transactions on Knowledge and Data Engineering, vol. 19, no. 12, pp. 1719 1733, December 2007.

[10] B. Gedik and L. Liu, "Location privacy in mobile systems: A personalized anonymization model," in Proceedings of IEEE ICDCS, Columbus, Ohio, June 2005.

[11] Chi-Yin Chow, Mohamed F. Mokbel, Xuan Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in Proceedings of ACM GIS, Arlington, Virginia, November 2006.

[12] A. Beresford and F. Stajano, "Location privacy in pervasive computing," IEEE Pervasive Computing, vol. 2, no. 1, pp. 4655, 2003.

[13] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady, "Preserving privacy in gps traces via uncertainty-aware path cloaking," in Proceedings of ACM CCS 2007, Alexandria, VA, US, January 2007.

[14] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in Proceedings of IEEE ICPS, Santorini, Greece, July 2006.

[15] H. Lu, C. S. Jensen, and M. L. Yiu, "Pad: privacy-area aware, dummybased location privacy in mobile services," in Proceedings of ACM MobiDE, Vancouver, Canada, June 2008.

[16] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in Proceedings of International Conference on Pervasive Computing, Munich, Germany, May 2005.

[17] C. A. Ardagna, M. Cremonini, S. D. C. di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 1, pp. 1327, January 2011.

[18] A. Pingley, W. Yu, N. Zhang, X. Fu, and W. Zhao, "Cap: A contextaware privacy protection system for location-based services," in Proceedings of IEEE ICDCS, Montreal, Canada, June 2009.

[19] M. Damiani, E. Bertino, and C. Silvestri, "Probe: An obfuscation system for the protection of sensitive location information in lbs," Technical Report 2001-145, CERIAS, 2008.

[20] Zhenqiang Gong, Guang-Zhong Sun, and Xing Xie, "Protecting Privacy in Location-based Services Using K-anonymity without Cloaked Region," Eleventh International Conference on Mobile Data Management, Kansas City, MO, USA, 2010.

[21] Reza Shokri, George Theodorakopoulos, George Danezis, Jean-Pierre Hubaux, and Jean-Yves Le Boudec, "Quantifying Location Privacy: The Case of Sporadic Location Exposure," S. Fischer-Hubner and N. Hopper (Eds.): PETS 2011, LNCS 6794, pp. 5776, 2011.

[22] Reza Shokri, George Theodorakopoulos , Carmela Troncoso, Jean-Pierre Hubaux, and Jean-Yves Le Boudec, "Protecting Location Privacy: Optimal Strategy against Localization Attacks," CCS '12 Proceedings of the 2012 ACM conference on Computer and communications security, pp. 617-627, New York, NY, USA, 2012.

[23] http://www.interdependentnetworks.com/contributorsresources/

CHAPTER 8

A NOVEL MULTI-TIME-SCALE LOAD FORECASTING FOR STATE ESTIMATION: A DETECTION METHOD AGAINST DATA FALSIFICATION ATTACKS

One of the major roles that Smart Grid has promised to play is to provide a power to satisfy power demand with environmentally-friendly source of energy while maintaining an acceptable level of adequacy and security that traditional systems promise. As a result, there have been many efforts to develop estimation algorithms of the power system states which are the core of the time-sensitive grid management. In this chapter, we address auto-regressive load forecasting methods which play pivotal role in creating an accurate state estimator for the power grid management.

Short-term load forecasting is essential for reliable and economic operation of power systems. Short-term forecasting covers a range of predictions from a fraction of an hour-ahead to a day-ahead forecasting. An accurate load forecast results in establishing appropriate operational practices and bidding strategies, as well as scheduling adequate energy transactions. This chapter presents a generalized technique for modeling historical load data in the form of time-series with different cycles of seasonality (e.g., daily, weekly, quarterly, annually) in a given power network. The proposed method separately models both non-seasonal and seasonal cycles of the load data using auto-regressive (AR) and moving-average (MA) components, which only rely on historical load data without requiring any additional inputs such as historical weather data (which might not be available in most cases). The accu-

[0]Part of this chapter has been reprinted with permission from Kianoosh G. Boroojeni et al., "A Novel Multi-Time-Scale Modeling for Electric Power Demand Forecasting: From Short-Term to Medium-Term Horizon, Electric Power System Research, vol. 142, no. 1, pp 58-73, Jan. 2017.

racy of data modeling is examined using the Akaike/Bayesian information criteria (AIC/BIC) which are two effective quantification methods for evaluation of data forecasting. In order to validate the effectiveness and accuracy of the proposed forecaster, we use the hourly-metered load data of PJM network as a real-world input dataset.

8.1 Introduction

8.1.1 Motivation

Electricity demand forecasting plays a pivotal role in power systems management, especially for operation and maintenance purposes [1]. It is particularly more important for deregulated power systems, where the forecast inaccuracies have significant implications for market operators, transmission owners, and market participants. Load forecasting is categorized based on the time scale into short-term, medium-term, and long-term forecasting. These three types are utilized for power systems scheduling and control, operation and planning, and generation/transmission expansion planning respectively [1, 2]. Short-term load forecasting (STLF) is required for generation scheduling, security assessment of system operation, and hourly economic dispatch information [3]. From the demand side's point of view, there is an exigent requirement to accurately estimate demand to achieve a more reliable operation of the power systems [4, 5]. Future power systems, namely smart grids, are emerging with the concept of advanced metering infrastructure to ameliorate the reliability of the conventional power systems in demand side [6, 7]. Furthermore, demand side management is widely used for residential [8, 9] and industrial load control [10] and smart energy hub applications [11]. Although the utilization of demand response and other demand side resources improves the reliable operation of

power systems, accurate demand forecasting is an inevitable obligation to maintain the load-generation balance. The accurate demand forecast will improve the real-time and long-term performance of power systems based on the available historical data.

8.1.2 Literature Review

A comprehensive survey on demand forecasting approaches has been provided in [12]. This paper classified the methods into four groups: very short term load forecast (VSTLF) [13, 14], STLF [15, 16, 17], medium term load forecast (MTLF) [18], and long term load forecast (LTLF) [19, 20] that a literature survey represented 6%, 58%, 20% and 16% of the past research efforts focused on these load forecast horizons, respectively. The authors in [13] applied artificial neural networks to model load dynamics for VSTLF application. The proposed VSTLF approach was tested for online load forecasting in a power utility in the United States. Bagged neural network is deployed for STLF in [21]. In [14], 5-min moving time windows are used to determine the hourly ahead load. This paper adopted wavelet neural networks with data pre-filtering to forecast the load in very short time slots by minimizing the effect of noisy data. ISO New England data set has been employed for validation of the proposed approach in [14]. Piras *et al.* in [15] proposed heterogeneous neural network architecture composed of an unsupervised part to detect some features of the data and suggest regression variables. To obtain a smooth transition between submodels, a weighted fuzzy average was deployed to integrate the outputs of each submodel.

Amjady in [16] composed a forecast-aided state estimator (FASE) and the multi-layer perceptron (MLP) neural network to build a short term load forecaster. The

method trains the MLP to determine the mapping function which is required for FASE (input features) and the output (real load). Dove *et al.* implemented different regularization procedures for training purposes in neural networks for medium term load forecasting. This approach referred to as feed forward neural network (FNN) model [18]. In [19], a knowledge-based expert system was developed for choosing the most appropriate annual prediction model. The selected model was then utilized medium/long term power system planning. In [22], kernel-based multi-task learning methods are employed to forecast the electric power demand at the distribution network.Authors in [20], proposed a forecasting method to predict the long-term peak demand. Different uncertainties that can affect the peak demand were considered during the LTLF, including population growth, economic conditions, and weather conditions. Furthermore, large-scale utilization of electric vehicles can increase the uncertainty due to different driving behaviors and charging patterns [23, 24, 25, 26].

In addition to the time slot based classification of forecasting methods, we can classify the existing approaches according to the applied techniques. One method is to use historical load pattern as a time-series to forecast the demand using time-series analysis methods. The second method is based on the correlation of load pattern and weather variables. This approach constructs the relation between historical load and weather conditions to forecast demand.

Traditional load forecasting approaches, such as regression and interpolation, may not lead to accurate results. On the other hand, complex forecasting methods, which are computationally-burdened converge slower. Several number of studies have been focused on prediction techniques, including fuzzy logic approach and artificial neural network [16, 27], linear regression [28], and data mining [29], transfer functions [30], Bayesian statistics [31], judgmental forecasting [32], and grey dynamic models [33]. In [20], a methodology to forecast electricity demand up to ten

years ahead was proposed. Artificial neural networks (ANN) has been widely used for electricity demand forecasting [34, 35, 36]. According to [34], the generalized Delta rule (GDR) was utilized for training neurons in ANN and the output vector is used as an input pattern to the network. In [35], ANN is used for short term demand forecasting. Lee *et al* used ANN for weekdays and weekends separately to achieve more accurate results. In [37], artificial neural network is utilized to predict the interruptions in smart grids based on weather condition and historical interruptions in the analyzed network. A comprehensive literature survey of the neural network application in short term demand forecasting is provided in [36]. Amjady in [38] proposed a short term forecaster that differentiates between weekdays, weekends, and public holidays to improve the accuracy. There are four major concerns related to the previous studies on the demand forecasting: 1) Long process of load forecasting (flow of load forecast process is too long) such as [39]; 2) Small data set is used for model validation [40]; 3) Large amount of data is required in the training phase [41]; and 4) Big error in forecasting [42].

Electricity load demand over a period of time is a seasonal non-stationary time-series. Many attempts have been made by statisticians to create forecasting models for this kind of time-series. Dudek in [43] proposed linear regression models for pattern-based short-term load forecasting in which multiple seasonal cycles of the forecasting time-series is filtered out and non-stationary in mean and variance is eliminated. A step-by-step procedure has been developed [44] for applied seasonal non-stationary time-series modeling following the Box-Jenkins methodology which is a known modeling methodology first created by Box and Jenkins in 1976 [45]. Additionally, Pappas et al. in [46] presented a new method for electricity demand load forecasting using the multi-model partitioning theory, first filters out the seasonality and non-stationary of the actual data using the modeling and forecasting electricity

loads and prices (MFE) toolbox for Matlab and then applies a multi-variate auto-regressive moving average (ARMA) model to forecast the electricity demand of the Hellenic power system.Moreover, Desouky and Elkateb in [47] utilized a combination of ARMA and ANN methods in order to obtain a more promising forecaster compared with previous works using time-series method. In addition, Huang et al. proposed a short-term load forecaster based on ARMA model including non-Gaussian process considerations [48]. Furthermore, Contreras et al. in [49] analyzed different ARIMA models for predicting next-day electricity load.

8.1.3 Our Contribution

This chapter presents a generalized technique for modeling historical load data in the form of time-series with different cycles of seasonality (e.g., daily, weekly, quarterly, annually) in a given power network. The proposed method separately models both non-seasonal and seasonal cycles of the load data using auto-regressive (AR) and moving-average (MA) components, which only rely on historical load data without requiring any additional inputs such as historical weather data (which might not be available in most cases). The accuracy of data modeling is examined not only by calculating the conventional forecasting errors but utilizing the Akaike/Bayesian information criteria (AIC/BIC) which are two effective quantification methods that penalize the complexity of a model and reward its fitness and accuracy. If the forecaster's flexibility is our most important concern, i.e. our main objective is to develop a forecaster which can be broadly used for different data training sets without deteriorating the forecaster's performance, the BIC is a better alternative (compared with AIC) for quantifying the forecaster's utility since BIC penalizes the complexity of forecasting models more than AIC. However, AIC is a better criterion

if we choose to compromise model flexibility by some degree to gain more fitness and improve the accuracy of forecaster by reducing the forecasting error. The main contributions of this chapter are as follows:

- Enhancing the Box-Jenkins methodology for modeling of the historical electricity load data over a period of time in order to create an accurate electric power demand forecaster.
- Adding multiple seasonality cycles to create a multi-time-scale modeling for electricity power demand forecasting which is more flexible to the daily, weekly, and annual seasonal nature of the electricity data. Consequently, our model can be considered as an extended seasonal auto-regressive integrated moving average model (SARIMA) [50].
- Using Bartlett's periodogram-based test to prove that the residual time-series (forecasting error) is a white noise and is meaningless. That is, the proposed model has extracted any meaningful information from the input training set in order to obtain the most accurate forecast.
- Using Akaike/Bayesian information criteria (AIC/BIC) instead of the conventional error measurements for model evaluation and fine-tuning: AIC/BIC deals with the trade-off between appropriate fitness of the model and the model complexity. It considers some penalty value for complex models, which may only work precisely for the under-studied training set; however, the conventional error measurements are considerably dependent on the given training set and measure the fitness of the model merely with respect to the training set.
- Finally, we show that by changing the size or time of training set, the model remains robust and the forecasting error almost remains constant.

8.1.4 Organization of the Chapter

The rest of this chapter is organized as follows. Section 2 introduces preliminaries of the time-series modeling. Section 3 presents a detailed explanation of the proposed multi-time-scale model. In Section 4, a practical case study is presented and the superior performance of our novel methodology is illustrated. Section 5 concludes the chapter and outlooks the forecasting results. The detailed representation of used data set is provided in the appendix.

8.2 Preliminaries

A time-series is a sequence of data points, typically consisting of successive measurements made over a time interval. Time-series are very frequently plotted via line charts and are used in any field which involves temporal measurements. Time-series based analysis comprises methods for analyzing data in order to extract meaningful statistics and other characteristics of the data. It also used for forecasting which is based on the utilization of a model to predict future values based on previously observed values. Time-series x_t is wide-sense stationary (WSS) if its mean doesn't vary over time ($\mathbb{E}[x_t] = \mu_x$) and its autocorrelation function, defined as:

$$R_x(t_1, t_2) = \mathbb{E}[(x_{t_1} - \mu_x)(x_{t_2} - \mu_x)]$$

depends only on the value of $(t_2 - t_1)$ for every $t_1, t_2 \in \mathbb{R}_{\geq 0}$. A seasonal time-series of seasonal cycle T is called stationary over its seasonal cycle if

$$\mathbb{E}[x_t] = \mathbb{E}[x_{t+T}],$$

and $R_x(t, t + nT)$ is independent of t for every $t > 0$; e.g. if a time-series with daily value and weekly seasonality is *weekly-stationary* if it has constant autocorrelations on multiples of 7.

Autoregressive Models

In statistics, an auto-regressive (AR) model is a representation of a type of random process showing (wide-sense) stationary behavior. In contrast with the regression analysis which models the output variable as a function of multiple independent variables, the autoregressive model considers the output variable as a linear function of its own previous values and on a stochastic term (a non-deterministic, imperfectly predictable term). Thus, the model is in the form of a stochastic difference equation:

$$x_t = (\sum_{i=1}^{p} \phi_i L^i) x_t + \varepsilon_t,$$

where x_t specifies the representing random process of a given time-series, ϕ_i is the coefficient of the i^{th} AR term, L is the lag operator acting on x_t as $L : x_t \mapsto x_{t-1}$, and ε_t denotes the stochastic term called the *error value* of the model.

AR model is a special case of the more general ARMA model of time-series, which has a more complicated stochastic structure. Given a time-series of data x_t where t is an integer index and the x_t are real numbers, an ARMA(p, q) model is given by:

$$x_t = \underbrace{\left(\sum_{i=1}^{p} \phi_i L^i\right) x_t}_{\text{AR-Part}} + \underbrace{\left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t}_{\text{MA-Part}}, \tag{8.1}$$

where θ_i's are the parameters of the moving average part.

These models are fitted to time-series data to predict future points in the series (forecasting). An ARIMA model is an extension of an ARMA. They are applied in some cases where data are non-stationarity and an initial differentiating step (corresponding to the "integrated" part of the model) can be applied to reduce the non-stationarity. In fact, the ARIMA(p, d, q) model can be viewed as a "cascade" of two models. The first applies on a non-stationary time-series x_t:

$$x'_t = (1 - L)^d x_t$$

while the second part models the WSS time-series x'_t using ARMA(p, q) described in Equation 8.1.

When two out of the three terms are zeros, the model may be referred to based on the non-zero parameter, dropping "AR", "I" or "MA" from the corresponding acronym. For example, ARIMA (1,0,0) is AR(1), ARIMA(0,1,0) is I(1), and ARIMA(0,0,1) is MA(1). Before explaining the proposed methodology, we introduce two functions that are assigned to a WSS time-series x_t: auto-correlation function[1] (ACF) and partial auto-correlation function[2] (PACF). The behavior of these two functions reveals that how a WSS time-series can be formulated as an ARMA model. More details regarding how the behavior of ACF and PACF is interpreted is presented in the next section.

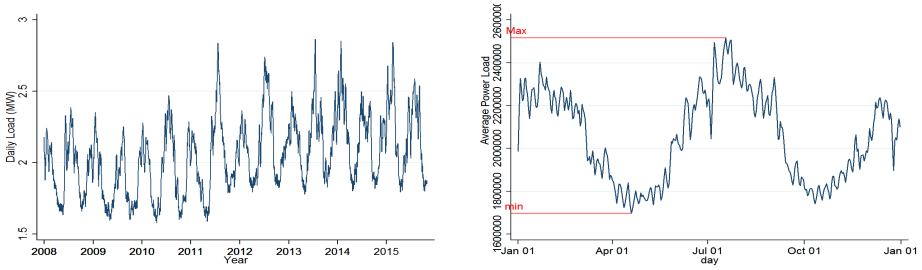
8.3 The Proposed Methodology

The chronologically-ordered power load values of a specific part of the power systems is a time-series that occurs over a period of time in a seasonal manner. To model such time-series as a function of its past values, we analyze the pattern with the assumption that the general pattern will persist in the future. This sections is devoted to identify the best model that matches the statistical behavior that the observed electricity demand data shows over a few years. The quality of modeling the power load time-series determines how accurate the power load data is estimated

[1]The ACF of a WSS time-series x_t is defined as function $\text{AC}_x(l) = \text{corr}(x_{t+l}, x_t)$, where $\text{corr}(\cdot, \cdot)$ is the correlation function defined as $\text{corr}(x_{t+l}, x_t) = \mathbb{E}\left[\frac{(x_t - \mu_x)(x_{t+l} - \mu_x)}{\sigma_x^2}\right]$, where μ_x and σ_x^2 are the mean and variance of stationary process x_t respectively. Variance of the wide-sense stationary process x_t is defined as $\sigma_x^2 = \mathbb{E}[(x_t - \mu_x)^2]$, where μ_x is the mean of x_t.

[2]The PACF of a wide-sense stationary time-series x_t is represented by $\text{PAC}_x(l)$ and is defined as $\text{AC}_x(l)$ if $l = 1$, and $\text{corr}\left(x_{t+l} - P_{t,l}(x_{t+l}), x_t - P_{t,l}(x_t)\right)$ otherwise; where $P_{t,l}(x)$ denotes the projection of x onto the space spanned by $x_{t+1}, x_{t+2} \ldots, x_{t+l-1}$.

and forecast in the future. In order to evaluate the model identification, the section utilizes the Akaike/Bayesian information criteria which help us to figure out how close the estimated time-series fits the observed power load data. Analyzing the residual time-series (the actual difference between the observed and estimated load power) can also determine the quality of a model where if the residual time-series is non-deterministic and has no meaningful part, the model accurately represents the observed data. During this section, we utilize the actual daily load data of PJM network over the time period of 2008-2014 as our training-set in order to illustrate the steps of creating the model. Then, we use the load data of the same network in 2015 to evaluate the model and compare the observed and forecast values. Figure 8.1 shows a statistical insight into the load data of PJM network.



(a) Seasonality of daily load (big picture).

(b) Average annual load data of PJM

Figure 8.1: Statistical analysis of the load data of PJM network over a period of 8 years.

8.3.1 Outline of the Proposed Methodology

First, we check whether the time-series representing the load data has homogeneous variance. If not, we apply an appropriate logarithmic transformation to the time-series to make it variance homogeneous. To obtain the effective transformation, we apply the Box-Cox method [51], which is explained later.

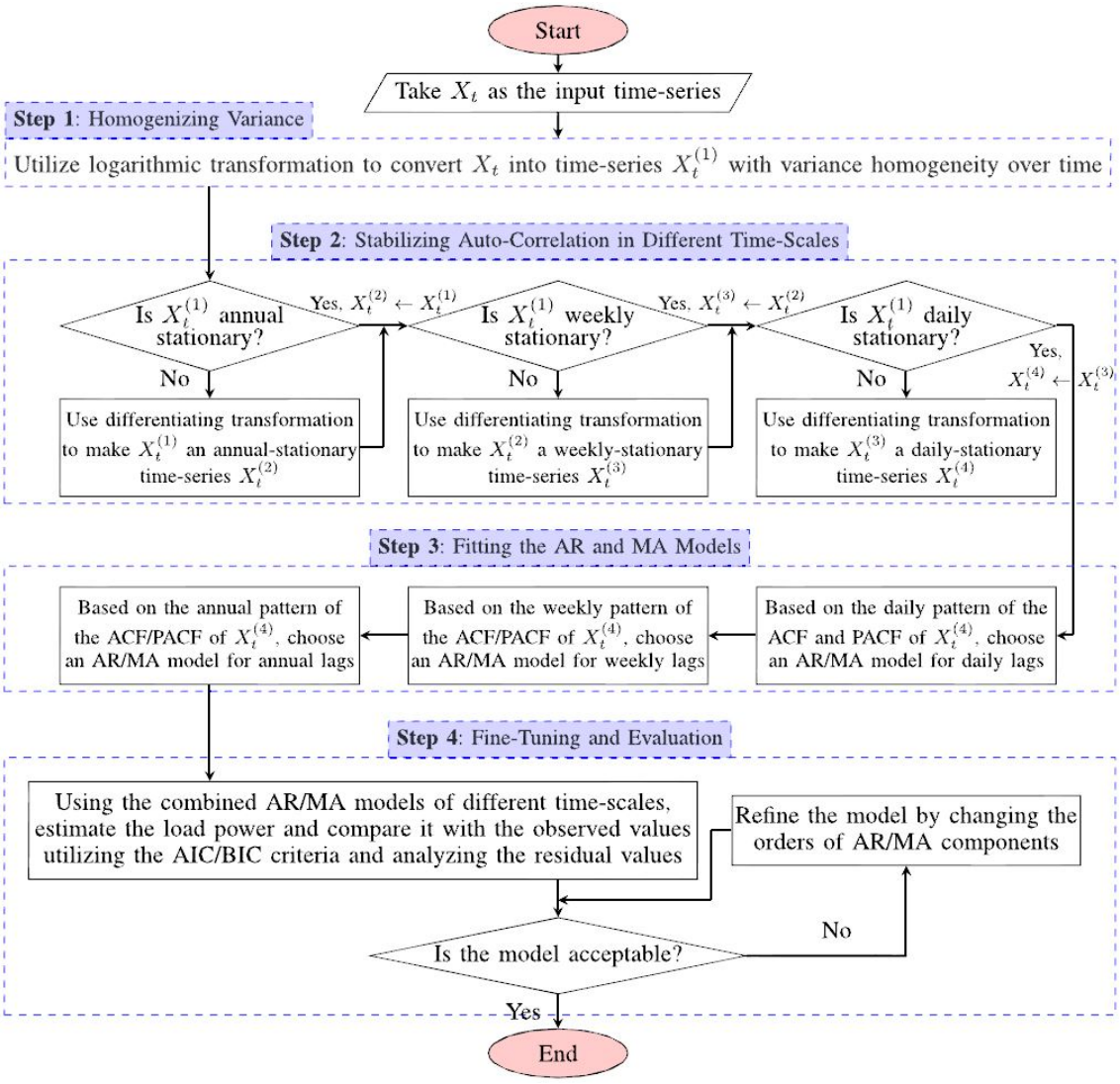


Figure 8.2: The flowchart representation of the proposed methodology for creating a forecasting model for daily load values.

Second, we examine the ACF plot of the transformed time-series at the non-seasonal level and seasonal cycles (weekly, annually, etc) to find any indication of being non-stationary. If the ACF of the time-series either falls off or declines in the first few lags, it is considered as stationary. If the ACF values either falls off after a considerable number of lags or declines quite slowly, it should be marked as non-stationary. Notice that the ACF of a time-series may show different behaviors in

different seasonal cycles; e.g. the ACF of a time-series with weekly lags may show stationary sign; while, considering daily lags, the same ACF may die down very slowly (non-stationary behavior). In order to filter the non-stationary indications, we repeatedly apply differencing transformations on the time-series.

Finally, the transformed stationary time-series is modeled to a moving average and/or autoregressive model based on how ACF and PACF of the time-series behave. Finally, the detailed parameters of the model is estimated by trying multiple values and validating the forecasts using the AIC. Figure 8.2 shows the flowchart representation of the proposed methodology.

8.3.2 Homogenizing Variance

When a time-series does not have variance homogeneity[3], some transformations are applied on the time-series in order to homogenize (stabilize) its variance over time. Here, we use a special case of Box-Cox transformation [51] to obtain a homogeneous time-series. It is common that the standard deviation of a time-series (especially electric load data) proportionally changes with its mean over time, i.e. $Var[x_t] = (a\mu_t + b)^2$ for some a and b. Let τ_d denote the function that transforms x_t to a time-series with homogeneous variance: $Var[\tau_d(x_t)]$ =constant. Assuming that τ'_d represents the derivative function of τ_d, we approximate $\tau_d(x_t)$ by its first-order Taylor series around its mean μ_t:

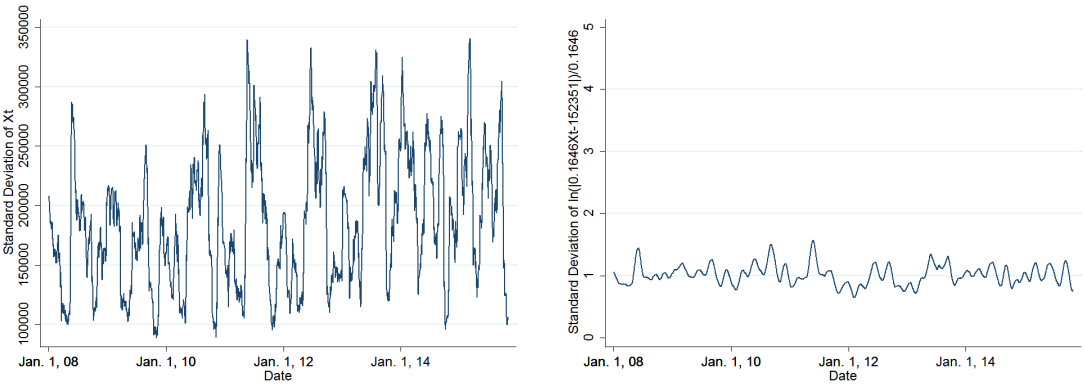
$$\tau_d(x_t) \approx \tau_d(\mu_t) + \tau'_d(\mu_t)(x_t - \mu_t)$$

[3]A time-series has variance homogeneity if its variance doesn't change over time.

. Consequently, we obtain the following approximation for the variance of transformed time-series $\tau_d(x_t)$:

$$Var[\tau_d(x_t)] \approx Var[\tau_d(\mu_t) + \tau'_d(\mu_t)(x_t - \mu_t)] = (\tau'_d(\mu_t))^2 Var[x_t] = \big(\tau'_d(\mu_t)(a\mu_t + b)\big)^2. \tag{8.2}$$

To obtain time-series $\tau_d(x_t)$ with variance-homogeneity, the following condition



(a) The instantaneous standard deviation before applying logarithmic Box-Cox transformation.

(b) The stabilized standard deviation after applying logarithmic Box-Cox transformation.

Figure 8.3: Homogenizing the variance of daily values of electric load in PJM power network over the period of 2008-2014

should hold: $\tau'_d(\mu_t)(a\mu_t + b) = \text{constant}$. That is, τ_d is a logarithmic transformation: $\tau_d(x_t) = \ln(|ax_t + b|)/a$. Values of a and b are obtained by linear regression analysis of σ_t (standard deviation of x_t); where μ_t is the independent variable.

In order to illustrate the aforementioned variance-homogenizer method, consider the daily values of electric load in PJM power network over the period of 2008-2014. Let X_t denote the time-series representation of the daily load values. Figure 8.3 shows the plot of standard-deviation σ_{X_t} of X_t ($\sqrt{\text{Var}(X_t)}$) over time. It can be seen that the time-series is not stationary in variance. Using a linear regression, we obtain the relation of $\sigma_{X_t} = 0.1646\mu_{X_t} - 152351$ between instantaneous mean and standard-deviation of the time-series where the coefficient of determination

for linear regression is $R^2 = 0.86$. As a result, the Box-Cox transformation for making the time-series stationary is in the following form: $\tau_d(X_t) = \ln(0.1646X_t - 152351)/0.1646$.

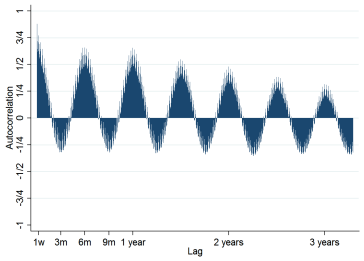
8.3.3 Stabilizing Auto-Correlation in Different Time Scales

Any time-series that shows periodic similarities over time is referred to as seasonal. The electric power demand values follow multiple periodic patterns. For example, the load demand values on Sundays follow the same pattern; while they are different from Mondays'. Also, special holidays have specific annually-repetitive load patterns. The seasonal behaviors appear at the exact seasonal lags L, 2L, 3L, 4L, and 5L. For daily load data, the time-series shows repetitive pattern in the lags 7, 14, 21, etc (weekly seasonality). Also, for hourly load time-series, data shows daily seasonality with L=24, i.e. the seasonal lags are 24, 48, 72, and so on. In general, the transformed time-series values are considered stationary if the ACF satisfies the following conditions:

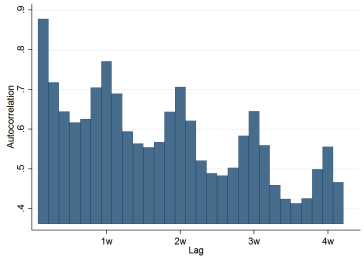
1. Suddenly falling off or declining quickly at the non-seasonal level;
2. Suddenly falling off or declining quickly at the seasonal cycles (exact seasonal lags or near seasonal lags).

Otherwise, these values are not considered as (wide-sense) stationary. For example, consider the daily load data of PJM network again. As mentioned before, $X_t^{(1)}$ is the logarithmic transformed load time-series with stabilized variance. Figure 8.4(a) shows the ACF plot of $X_t^{(1)}$ for the first 1200 lags. It can be seen that the auto-correlation of data does not fall off or decline after the first few annual lags; instead, it decreases with a slow pace. Consequently, $X_t^{(1)}$ shows non-stationary behavior in annual seasonal cycle. Additionally, considering Figure 8.4(b) which depicts the

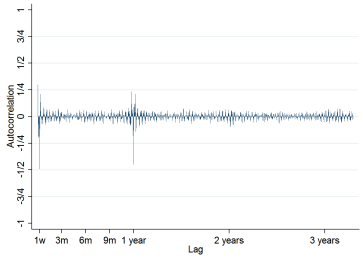
ACF plot of $X_t^{(1)}$ in the first few weekly lags, the auto-correlation $X_t^{(1)}$ in weekly and daily lags declines very slowly. As a result, $X_t^{(1)}$ is not a stationary time-series in both weekly lags (weekly seasonal cycle) and daily lags (non-seasonal level).



(a) ACF of $X_t^{(1)}$. The quarterly and annul seasonal patterns are illustrated.



(b) ACF of $X_t^{(1)}$ showing its non-stationary behavior in the first few weekly and daily lags.

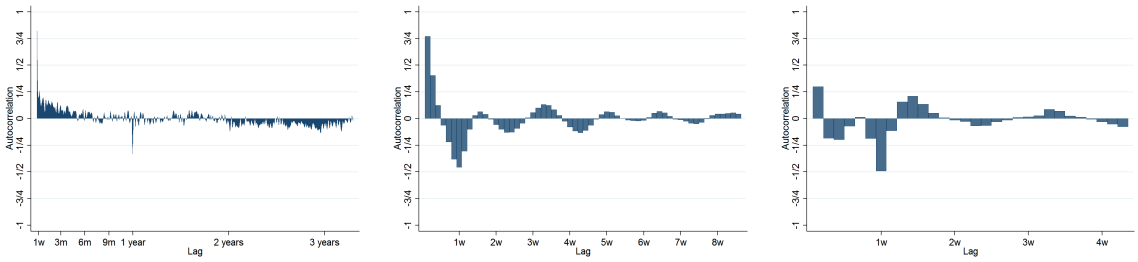


(c) ACF of $X_t^{(4)}$ which is obtained by equation $X_t^{(4)} = D^{(d)}_{\text{annual}}(D^{(d)}_{\text{weekly}}(D^{(d)}_{\text{daily}}(X_t^{(1)})))$.

Figure 8.4: ACF plots of daily load data in PJM network. Sub-figure (a) show the quarterly, and annual seasonality of variance stabilized load data. Sub-figure (b) shows the non-stationary behavior of data in both daily level and weekly cycle. Sub-figure (c) illustrates how multiple differentiating transformations have led us to obtain a stationary time-series in all daily, weekly, and annual levels.

Henceforth, in order to make $X_t^{(1)}$ stationary, we need to transform it using multiple differentiating transformations in annual, weekly and daily levels. By transforming the time-series $X_t^{(1)}$ using the annual differentiation $D^{(d)}_{\text{annual}} = 1 - L^{364}$, the annual auto-correlation values fall off after the first annual lag (See the ACF plot of $X_t^{(2)} = D^{(d)}_{\text{annual}}(X_t^{(1)})$ depicted in Figure 8.5(a)). Notice that we consider 364 as the number of days in a year as it is the closest multiple of 7 to the size of a year and it will make the later computations simpler.

Additionally, by transforming the time-series $X_t^{(2)}$ using the differentiation $D^{(d)}_{\text{weekly}} = 1 - L^7$, the weekly auto-correlation values fall off substantially after the first-week lag (See the ACF of $X_t^{(3)} = D^{(d)}_{\text{weekly}}(X_t^{(2)})$ in Figure 8.5(b)). The daily differentiation $D = 1 - L$ transforms $X_t^{(3)}$ into a time-series with stationary behavior in all annual, weekly, and daily lags. As Figure 8.5(c) depicts, the auto-correlation of



(a) ACF of $X_t^{(2)}$ (b) ACF of $X_t^{(3)}$ (c) ACF of $X_t^{(4)}$

Figure 8.5: Stabilizing the Auto-Correlation of daily load data of PJM network in daily, weekly, and annual levels utilizing differentiating transformations. Sub-figures (a), (b), and (c) show the ACF of $X_t^{(1)}$ transformed by annual, weekly and daily differentiation respectively.

$X_t^{(4)}$ exponentially decreases three or four days of every weekly lag. It shows that $X_t^{(4)}$ is a stationary time-series in non-seasonal level. Also, the plot in Figure 8.5(c) verifies that the autocorrelation values of $X_t^{(4)}$ in weekly lags is negligible except in the seventh lag (proof of weekly stationary). Finally, the annual auto-correlation values of $X_t^{(4)}$ shown in Figure 8.4(c) prove that $X_t^{(4)}$ shows stationary behavior in annual lags too (the auto-correlation in annual lags except the first one has very low values).

In summary, the series of multi-time-scale differentiating transformations $D^{(d)}_{\text{annual}}$ o $D^{(d)}_{\text{weekly}}$ o D has converted the variance-stabilized time-series $X_t^{(1)}$ into a (wide-sense) stationary time-series $X_t^{(4)}$:

$$X_t^{(4)} = D^{(d)}_{\text{annual}} \text{o} D^{(d)}_{\text{weekly}} \text{o} D(X_t^{(1)}) = (1 - L^1 - L^7 + L^8 - L^{364} + L^{365} + L^{371} - L^{372}) X_t^{(1)}. \tag{8.3}$$

8.3.4 Fitting the AR and MA Models

This step creates a model of the transformed stationary time-series δ_t (obtained in Equation 8.3) with a moving average and/or autoregressive components. The model

is made based on the behavior of the ACF and PACF of δ_t. Table 8.1 shows the values of auto-correlation and partial auto-correlation of δ_t in multiple non-seasonal and seasonal levels.

Table 8.1: Behavior of autocorrelation and partial-autocorrelation functions of the daily load data in non-seasonal and multiple seasonal levels.

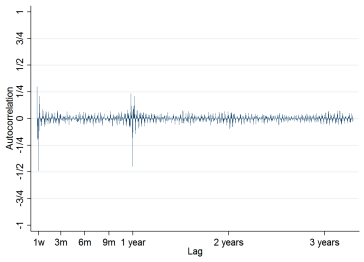
Non-Seasonal (Daily)			Weekly Seasonality			Annual Seasonality		
Lag	ACF	PACF	Lag	ACF	PACF	Lag	ACF	PACF
	Sine-wave declining	Falling off at $Lag = 2$		Falling off at $Lag = 7$	Exponentially declining		Falling off at $Lag = 364$	Exponentially declining
1	+0.300	+0.300	7	−0.494	−0.457	364	−0.450	+0.249
2	−0.187	−0.304	14	−0.017	−0.276	728	+0.030	+0.106
3	−0.199	−0.042	21	+0.015	−0.252	1092	+0.033	+0.031
4	−0.074	−0.048	28	−0.034	−0.190	–	–	–
5	+0.014	−0.010	35	+0.068	−0.118	–	–	–
6	−0.189	−0.275	42	−0.053	−0.151	–	–	–

The autocorrelation of the subsequent days (the non-seasonal part of the time-series) drops down suddenly after the second lag (see the values of the second column of Table 8.1). Additionally, the partial autocorrelations of δ_t in subsequent days are reduced exponentially (values of the third column of the same table supports the claim). The combinational behavior of the ACF and PACF functions at the non-seasonal level, which is depicted in Figures 8.6c and 8.6f, implies that the transformed daily load values (δ_t) follows a moving-average of order two; i.e. the time-series consists of two moving-average terms:

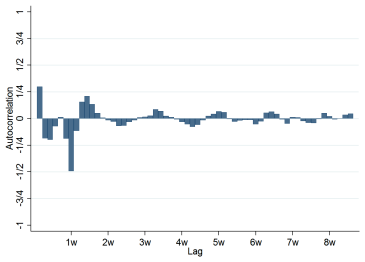
$$\gamma_1^{(d)}\varepsilon_{t-1}^{(d)} + \gamma_2^{(d)}\varepsilon_{t-2}^{(d)},$$

where $\gamma_1^{(d)}$ and $\gamma_2^{(d)}$ denote the two coefficients of the auto-regressive terms that will be computed in the model fine-tuning step (next step). Moreover, $\epsilon_t^{(d)}$ shows the non-deterministic part of the whole signal (δ_t) and is traditionally called the "error value" of the ARMA model that will be built subsequently.

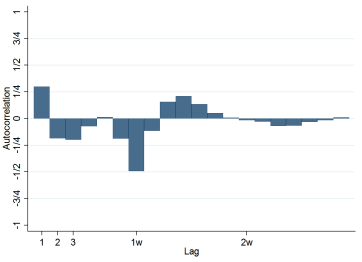
The values of the fifth and sixth columns of Table 8.1 illustrate that the autocorrelation values of the transformed daily load time-series in weekly lags ($L =$



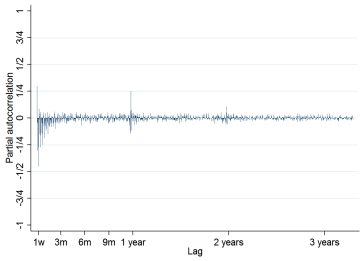
(a) Annual view to the ACF of δ_t



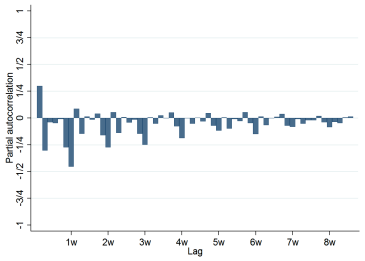
(b) Weekly view to the ACF of δ_t



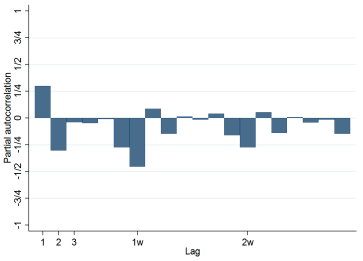
(c) Daily view to the ACF of δ_t



(d) Annual view to the PACF of δ_t



(e) Weekly view to the PACF of δ_t



(f) Daily view to the PACF of δ_t

Figure 8.6: ACF and PACF plots of stationary time-series δ_t obtained by transforming daily load values of PJM network.

$7, 14, 21, \ldots$) will substantially decrease after the first week; while its partial autocorrelation dies down exponentially with respect to the weekly lags. This behavior, which is depicted in Figures 8.6b and 8.6e, implies that the time-series includes the moving-average of order one in the weekly seasonal cycle, i.e., the transformed daily load data contains the following moving-average signal which represents the non-deterministic identity of the time-series: $\gamma_1^{(w)}\varepsilon_{t-7}^{(d)}$; where $\gamma_1^{(w)}$ represents the coefficient of the moving-average term of the model that will be specified in the next step of the model creation process.

The results of Table 8.1 and Figure 8.6 validate that the best real-time forecaster should have a moving-average element of lag two in the non-seasonal level and one

moving-average term at the weekly cycle:

$$\delta_t = \underbrace{\psi_1 \delta_{t-1}^{(d)} + \psi_2^{(d)} \delta_{t-2}}_{\text{non-seasonal}} + \underbrace{\gamma_1^{(w)} \varepsilon_{t-7}^{(d)}}_{\text{weekly-seasonal}} + \underbrace{\gamma_1^{(y)} \varepsilon_{t-364}^{(d)}}_{\text{annual-seasonal}} + \varepsilon_t^{(d)}. \tag{8.4}$$

8.3.5 Fine-Tuning and Evaluation

In the final step of constructing the model, the detailed parameters of the AR/MA components are determined by trying multiple values and finding the forecasting model with minimum AIC/BIC values.

AIC/BIC

The AIC is a measure of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates the quality of each model, relative to each of the other models. Hence, AIC provides a means for model selection. AIC is founded on information theory: it offers a relative estimate of the information lost when a given model is used to represent the process that generates the data. To this end, it deals with the trade-off between the appropriate fitness of the model and the model complexity. AIC does not provide a test of a model in the sense of testing a null hypothesis, i.e. AIC can tell nothing about the quality of the model in an absolute sense. If all the candidate models fit poorly, AIC will not give any warning of that. Suppose that we have a statistical model of some data. Let $\mathcal{L}$ be the maximum value of the likelihood function for the model; let $\mathcal{K}$ be the number of estimated parameters in the model. Then, the AIC value of the model is the following:

$$\text{AIC} = 2\mathcal{K} - 2\ln(\mathcal{L}).$$

Given a set of candidate models for the data, the preferred model is the one with the minimum AIC value. Hence, AIC rewards goodness of fit (as assessed by the

likelihood function), but it also includes a penalty that is an increasing function of the number of estimated parameters. The penalty discourages overfitting (increasing the number of parameters in the model almost always improves the goodness of the fit).

BIC is a criterion for model selection among a finite set of models; the model with the lowest BIC is preferred. It is based, in part, on the likelihood function and it is closely related to AIC. When fitting models, it is possible to increase the likelihood by adding parameters, but doing so may result in over-fitting. Both BIC and AIC resolve this problem by introducing a penalty term for the number of parameters in the model; the penalty term is larger in BIC than in AIC. The BIC is formally defined as

$$\mathrm{BIC} = -2 \cdot \ln \mathcal{L} + \mathcal{K} \cdot \ln(n),$$

where n is the size of historical data which the model is based on.

Fine-Tuning the Model

As mentioned before in Equation 8.4, the behavior of ACF and PACF of time-series δ_t suggests that there should be two daily-AR terms, one weekly-MA term, and one annual-MA term in the model. In this step, we construct different forecasting models with multiple number of AR and MA terms and compare their corresponding AIC/BIC values to find out which model has the least AIC/BIC and therefore forecasts in the most accurate way. In other words, a sensitivity analysis is necessary in order to find the best combination of M (number of daily-AR terms), N (number of weekly-MA terms) and Q (number of annual-MA terms) where the corresponding

model has the following form:

$$\delta_t = \underbrace{\sum_{i=0}^{M} \psi_i^{(d)} \delta_{t-i}}_{\text{daily-AR}} + \underbrace{\sum_{j=0}^{N} \gamma_j^{(w)} \varepsilon_{t-7j}^{(d)}}_{\text{weekly-MA}} + \underbrace{\sum_{k=0}^{Q} \gamma_Q^{(y)} \varepsilon_{t-364k}^{(d)}}_{\text{annual-MA}} + \varepsilon_t^{(d)}, \tag{8.5}$$

and $\psi_0^{(d)} = \gamma_0^{(w)} = \gamma_0^{(y)} = 0$. By changing any of these three parameters, we obtain a different forecasting model with different evaluation criteria (AIC, BIC, RMSE, MAE, and MAPE). Figures 8.7, 8.8, 8.9 and 8.10 visualize the results of sensitivity analysis. In Figures 8.7 and 8.8, the AIC and BIC values of $10 \times 6 \times 3 = 180$ different forecasters are depicted in terms of three thermal graphs since the triple of parameters (M, N, Q) belongs to the set $\{0, 1, \ldots, 9\} \times \{0, 1, \ldots, 5\} \times \{0, 1, 2\}$. In these two figures, the changing pattern of AIC and BIC is represented over multiple settings of the forecaster parameters. The darkest blue regions in these figures, marked by red spots, specify the parameter settings of the forecaster which lead to the lowest AIC/BIC values. Figure 8.9 shows the changes of AIC over three different values of Q. Each of sub-figures also represents a sensitivity analysis over M and N for a specific Q. Similarly, in Figure 8.10, the changing pattern of BIC values is shown over three different values of Q. Each of sub-figures also represents a sensitivity analysis over M and N for a specific Q.

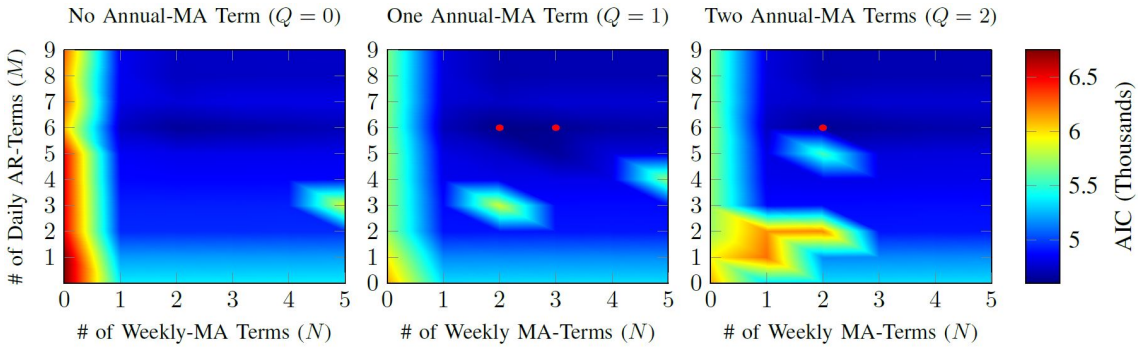


Figure 8.7: AIC values for different settings of the proposed AR/MA model. Red markers show three smallest AIC values.

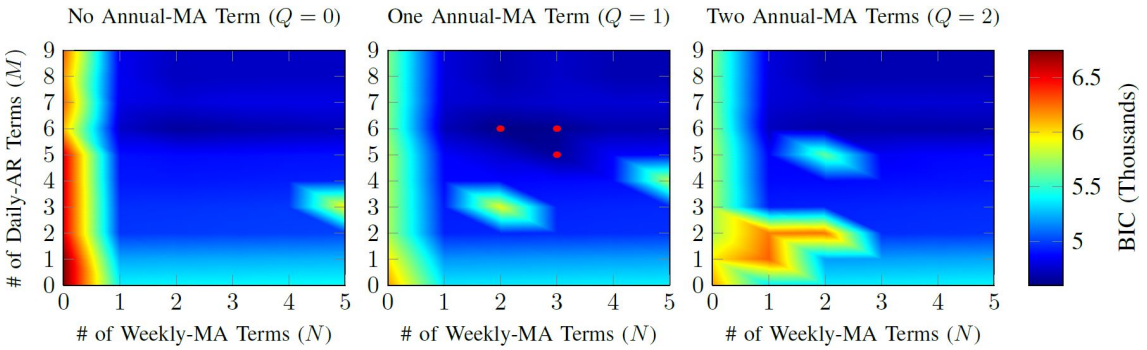


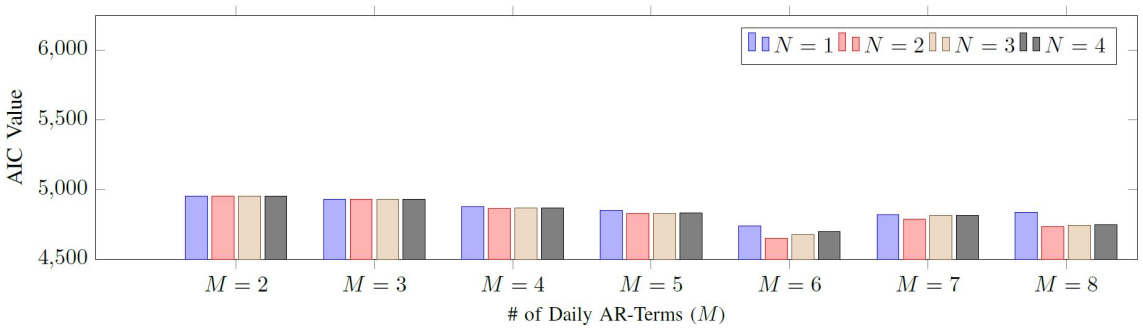
Figure 8.8: BIC values for different settings of the proposed AR/MA model. Red markers show three smallest BIC values.

Figure 8.11 compares the values of three different error types for multiple settings of the proposed model. Sub-figures 8.11a, 8.11b, and 8.11c specify the values of Root-Mean-Square Error (RMSE), Mean-Absolute-Percent-Error (MAPE), and Mean-Absolute-Error (MAE) [4] respectively for different values of M, N and Q. The minimum error values of our proposed model happens in the minimum points of the plots depicted in Figures 8.7, 8.8 and 8.11. Table 8.2 summarizes the best choices for triple (M, N, Q) which minimize the values of AIC/BIC and three different error types and AIC/BIC.

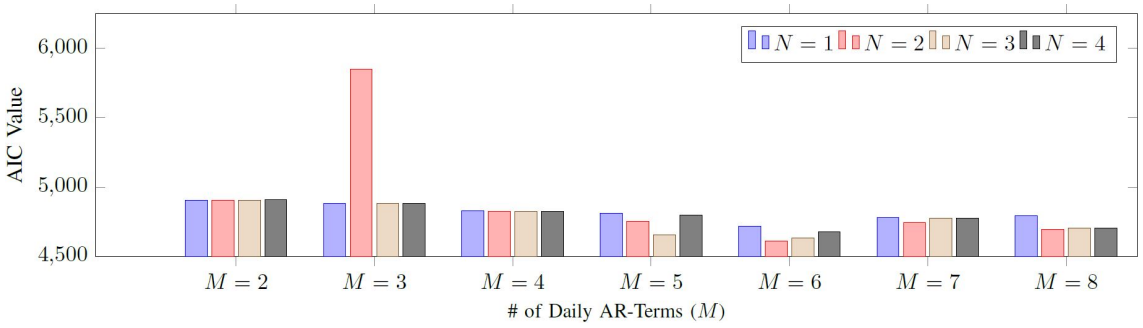
Table 8.2: Choosing the best setting of the proposed model for forecasting the daily load values of PJM network in 2015.

Selection Criteria	# of Daily-AR Terms (M), # of Weekly-MA Terms(N) and # of Annual-MA Terms (Q)	AIC (Thousands)	BIC (Thousands)	RMS Error (%)	MAP Error (%)	MA Error (%)
Minimizing AIC	$M = 2$, $N = 6$, $Q = 1$	4.61	4.76	6.60	4.49	4.68
	$M = 3$, $N = 6$, $Q = 1$	4.63	4.79	6.80	4.59	4.73
	$M = 2$, $N = 6$, $Q = 2$	4.64	4.80	7.02	4.75	5.42
Minimizing BIC	$M = 2$, $N = 6$, $Q = 1$	4.61	4.76	6.60	4.49	4.68
	$M = 3$, $N = 6$, $Q = 1$	4.63	4.79	6.80	4.59	4.73
	$M = 3$, $N = 5$, $Q = 1$	4.65	4.81	7.09	4.82	4.98
Minimizing RMS/MAP /MA Errors	$M = 5$, $N = 9$, and $Q = 1$	4.71	4.85	6.60	4.46	4.67
	$M = 2$, $N = 6$, and $Q = 1$	4.61	4.76	6.60	4.49	4.68
	$M = 4$, $N = 8$, and $Q = 2$	4.71	4.87	6.62	4.49	4.70

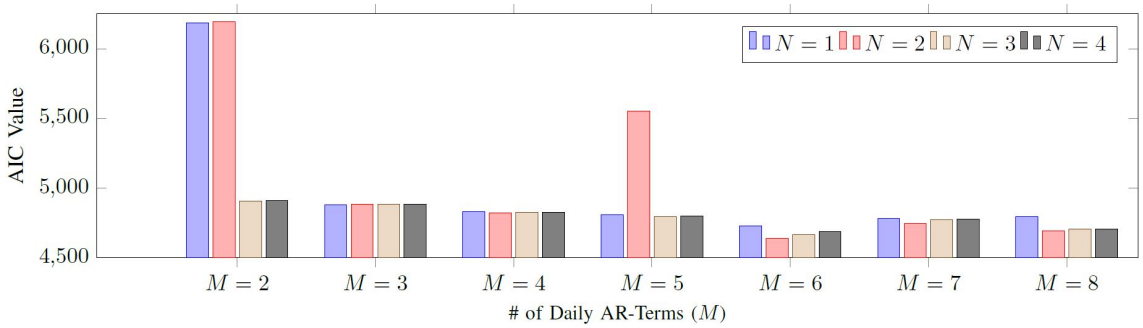
[4] Assuming o_i and f_i as the observed and forecast values ($i = 1, 2, \ldots, n$ and $o_i, f_i > 0$), the RMSE, MAPE, and MAE are respectively defined as $\frac{100}{n\bar{o}} \sum_{i=1}^{n} (f_i - o_i)^2$, $\frac{100}{n} \sum_{i=1}^{n} \frac{|f_i - o_i|}{o_i}$, and $\frac{100}{n\bar{o}} \sum_{i=1}^{n} |f_i - o_i|$.



(a) No Annual-MA Term ($Q = 0$)



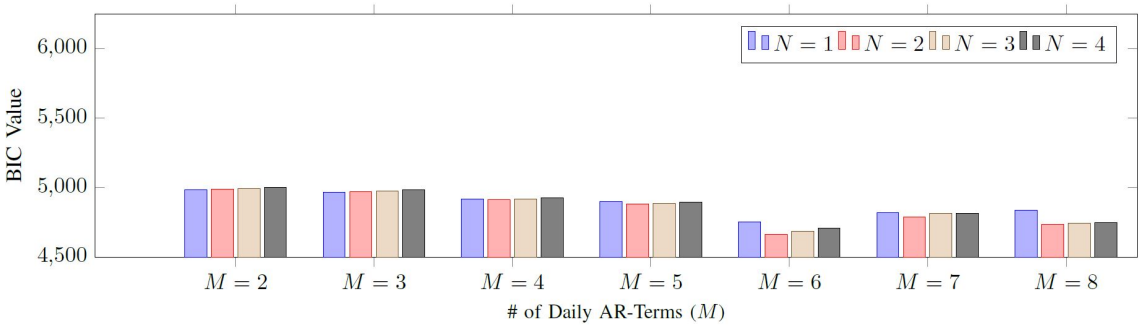
(b) One Annual-MA Term ($Q = 1$)



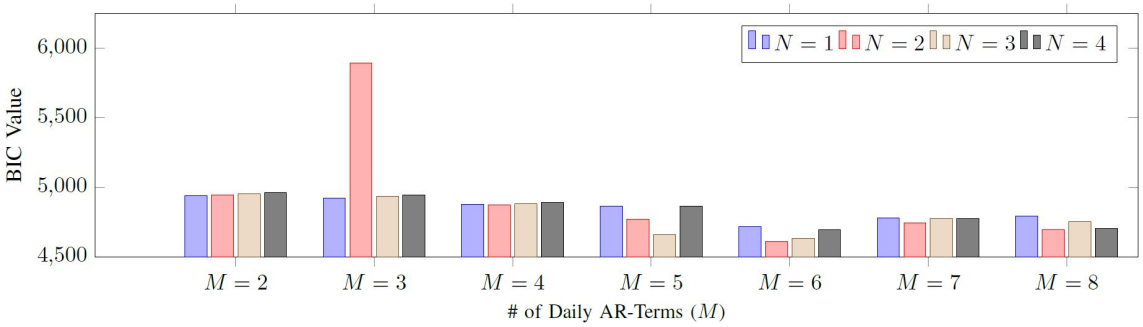
(c) Two Annual-MA Terms ($Q = 2$)

Figure 8.9: Bar chart of the AIC values corresponding to different settings of the proposed AR/MA model.

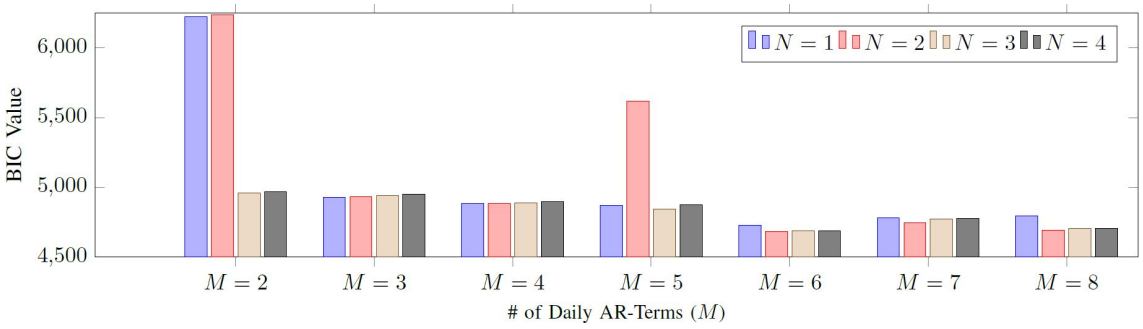
Table 8.2 suggests that the model with setting $(M, N, Q) = (2, 6, 1)$ has the least value of AIC and BIC and second-least error value among all of the considered settings. Additionally, as mentioned before, AIC and BIC not only reward the model goodness of fit, but they also include penalty function which are increasing functions of the number of estimated parameters and discourage overfitting. Consequently, they are better criteria for evaluating the forecasting models than error values which



(a) No Annual-MA Term ($Q = 0$)



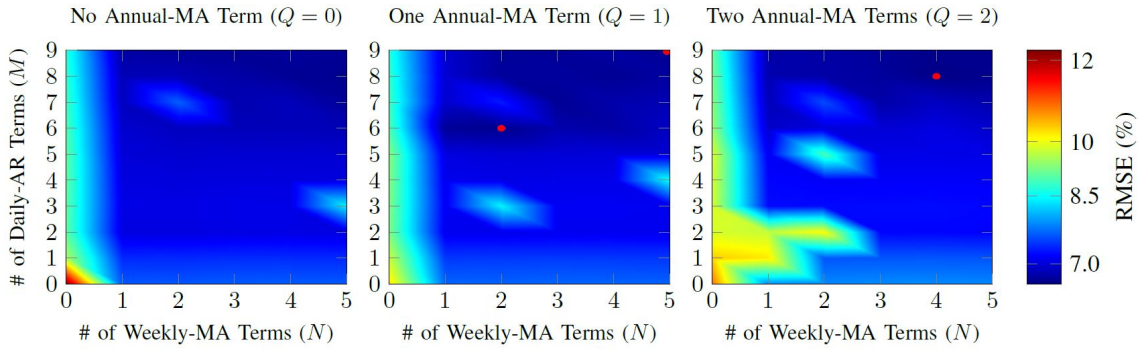
(b) One Annual-MA Term ($Q = 1$)



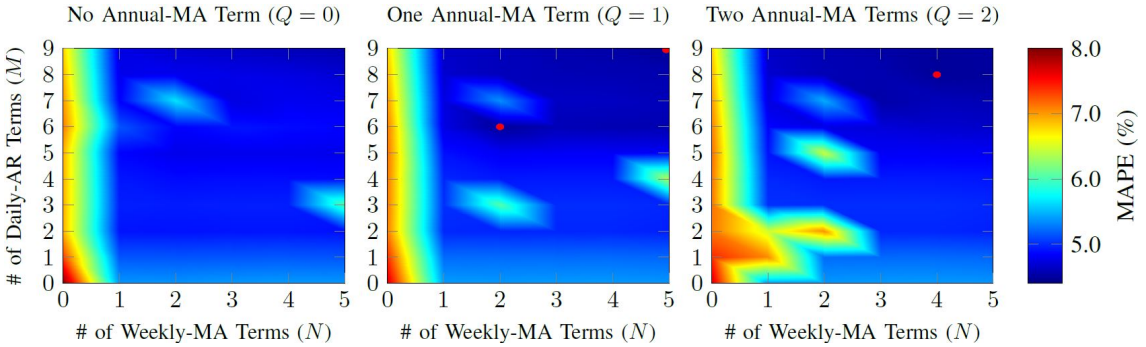
(c) Two Annual-MA Terms ($Q = 2$)

Figure 8.10: Bar chart of the BIC values corresponding to different settings of the proposed AR/MA model.

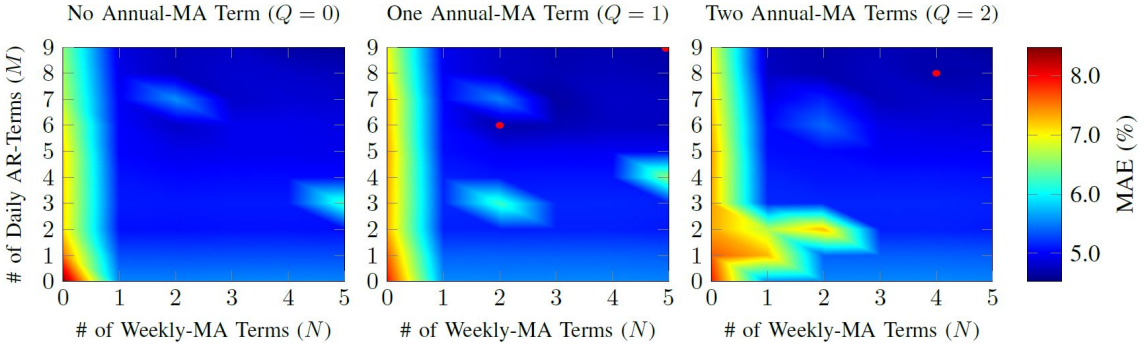
merely quantify how fit the model is for the training set. As a result, we consider $(M, N, Q) = (6, 2, 1)$ as the best setting of our proposed model for forecasting the daily electrical load. After creating the AR/MA-based model for δ_t and computing the forecast values of $\widehat{\delta_t}$, the forecast load values $\widehat{X_t}$ are obtained by applying the reverse transformations of steps one and two on $\widehat{\delta_t}$, i.e. regarding Equation 8.3,



(a) RMS Errors in forecasting the daily load of year 2015 when the proposed AR/MA model is applied with different settings.



(b) MAP Errors in forecasting the daily load of year 2015 when the proposed AR/MA model is applied with different settings.



(c) MAE values in forecasting the daily load of year 2015 when the proposed AR/MA model is applied with different settings.

Figure 8.11: Comparing the values of three different error types for different settings of the proposed AR/MA model. The points with three smallest error values have been specified with red markers.

time-series $\widehat{X}_t$ is computed by the following equation:

$$\widehat{X}_t = \tau_d^{-1}\Big(\widehat{\delta}_t + \Big[\big(L^1 + L^7 - L^8 + L^{364} - L^{365} - L^{371} + L^{372}\big) \circ \tau_d\Big] X_t\Big), \qquad (8.6)$$

where τ_d is the logarithmic Box-Cox transformation introduced earlier and

$$\widehat{\delta}_t = \sum_{i=1}^{6} \psi_i^{(d)} \delta_{t-i} + \sum_{j=1}^{2} \gamma_j^{(w)} \varepsilon_{t-7j}^{(d)} + \gamma_1^{(y)} \varepsilon_{t-364}^{(d)} + \varepsilon_t^{(d)}. \tag{8.7}$$

Comparing the Residual Time-series with White-Noise

As mentioned before, one way of evaluating the goodness of a forecaster is to compare its residual time-series $\rho_t^{(d)} = \widehat{X}_t - X_t$ (error values over time) with white noise. The ideal forecasting model extracts any meaningful information from the input training set to obtain the most accurate forecast. Henceforth, no meaningful signal will remain in the residual time-series, i.e. the residual time-series will be a *white noise*. Here, we use the cumulative periodogram of $\rho_t^{(d)}$ to show how close the residual time-series is to a *white noise*.

Let $\widehat{f}(\omega)$ denote the sample spectral density of the residual time-series $\rho_t^{(d)}$:

$$\widehat{f}(\omega) = \frac{1}{n} |\sum_{t=1}^{n} \rho_t^{(d)} e^{2\pi i (t-1)\omega}|^2,$$

where $\omega \in [0, 0.5]$. The cumulative periodogram of $\rho_t^{(d)}$ is defined as the plot of $\sum_{j=1}^{k} \widehat{f}(\omega_j) / \sum_{j=1}^{q} \widehat{f}(\omega_j)$ versus $\omega_k = (k-1)/n$ for $k = 1, 2, \ldots, q = \lfloor n/2 \rfloor + 1$. Figure 8.12 depicts the cumulative periodogram of $\rho_t^{(d)}$ with red color. This plot is used in Bartlett's test which is a test of the null hypothesis that the residual time-series comes from a white-noise process of uncorrelated random variables having a constant mean and variance. In this figure, two dashed lines specify the %99-confidence band for Bartlett's test; i.e. since the periodogram resides in the band, the residual time-series is a white-noise with confidence level of at-least %99.

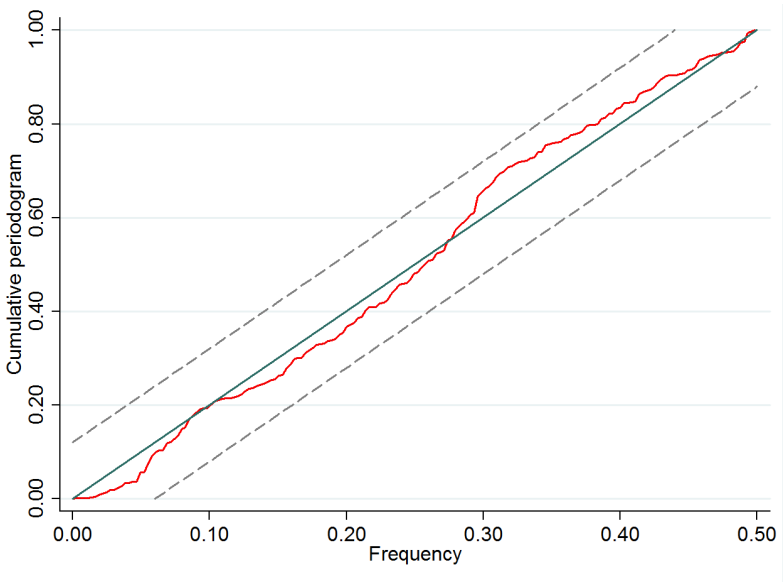


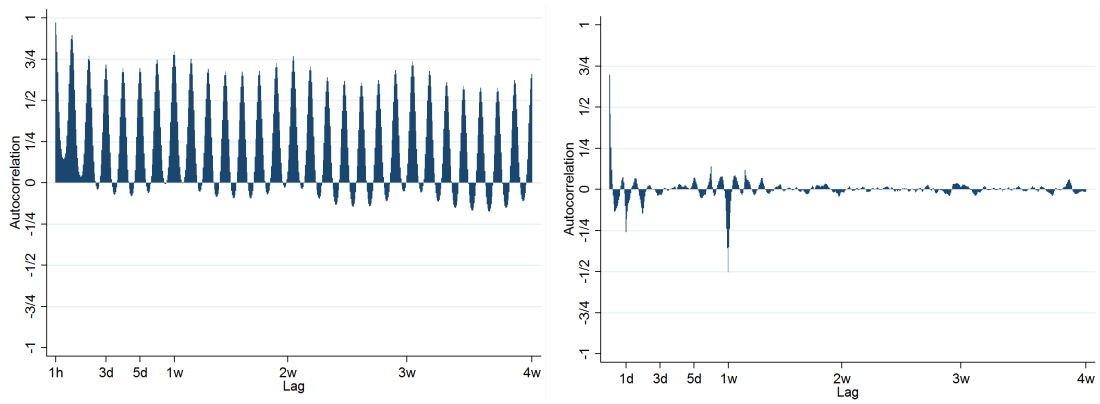
Figure 8.12: The cumulative periodogram of residual time-series $\rho_t^{(d)}$. The dashed lines specify the %99-confidence band for Bartlett's test to prove that the time-series is a white noise of constant mean and variance.

8.4 Case Study

In this section, our approach is implemented on the PJM hourly-metered load data available in [52] and the superior performance of our proposed methodology is illustrated compared with the currently existing models. Throughout this section, we build an hour-ahead forecaster for the PJM load data. Similar to the model built in previous section, the forecaster is trained using the historical hourly load data of PJM network in years 2013 to 2014; while it will be evaluated and fine-tuned using the same network load data in 2015.

Homogenizing the Variance

As mentioned in the previous section, the first step for modeling the load data is to transform the original load data Y_t to time-series $Y_t^{(1)} = \tau_h(Y_t)$ with variance homogeneity. To this end, we use the same technique utilized in previous section



(a) ACF plot of $Y_t^{(1)}$.

(b) ACF plot of η_t.

Figure 8.13: ACF plots of $Y_t^{(1)}$ and its transformed time-series η_t.

to create the following logarithmic Box-Cox transformation: $\tau_h(\cdot) = 0.1232\ln(\cdot) - 3880.71$.

Stabilizing Auto-Correlation in Different Time-Scales

In the next step, the time series $Y_t^{(1)}$ with variance-homogeneity is transformed to a WSS time-series η_t by three chained differentiating transformations $\eta_t = D^{(h)}_{\text{weekly}}$ o $D^{(h)}_{\text{daily}}$ o $D(Y_t^{(1)})$. Figure 8.13 respectively depicts the ACF plots of $Y_t^{(1)}$ and η_t for the first four weeks lags. As Figure 8.13a illustrates, $Y_t^{(1)}$ is a daily and weekly seasonal time-series and does not show WSS behavior as its ACF values decrease extremely slowly in daily and weekly cycles; however, the ACF plot of η_t in Figure 8.13b proves that η_t is a weekly-stationary, daily-stationary, and hourly-stationary time-series as its autocorrelation suddenly falls off after the first few hourly, daily, and weekly lags.

Fitting the AR and MA Models

In this step, the WSS time-series η_t obtained in previous step is modeled to an AR/MA model based on the behavior of its ACF and PACF. Table 8.3 shows the

values of ACF and PACF of η_t in the first few hourly, daily, and weekly lags. Ac-

Table 8.3: Behavior of autocorrelation and partial-autocorrelation functions of the hourly load data in non-seasonal and multiple seasonal levels.

Non-Seasonal (Hourly)			Daily Seasonality			Weekly Seasonality		
Lag	ACF	PACF	Lag	ACF	PACF	Lag	ACF	PACF
	Sine-wave declining	Falling off at $Lag = 1$		Falling off at $Lag = 48$	Exponentially declining		Falling off at $Lag = 168$	Exponentially declining
1	$+0.697$	$+0.697$	24	-0.281	-0.229	168	-0.490	-0.156
2	$+0.428$	-0.111	48	-0.135	-0.130	336	-0.009	-0.089
3	$+0.218$	-0.071	72	-0.033	-0.096	504	$+0.013$	-0.061
4	$+0.084$	-0.020	96	$+0.008$	-0.064	672	–	–
5	-0.001	-0.030	120	$+0.050$	-0.019	840	–	–
6	-0.060	-0.045	144	$+0.149$	$+0.090$	1008	–	–

cording to this table, in the first few hourly lags, the ACF substantially decreases in a Sine-wave manner with the respect to the lag parameter (See the values of the second column of Table 8.3). Additionally, the PACF falls off suddenly at lag $L = 1h$ (values of the third column of the same table supports the claim). The combinational behavior of the ACF and PACF functions at the non-seasonal level implies that the WSS time-series η_t follows an auto-regressive model of order one, i.e. the time-series consists of the auto-regressive term $\phi_1^{(h)}\eta_{t-1}$, where variable $\phi_1^{(h)}$ denotes the coefficient of the auto-regressive term.

The values of the fifth and sixth columns of Table 8.3 illustrate that the autocorrelation of η_t in daily lags ($L = 24, 48, 72, \ldots$) will substantially decrease after the first two days while its partial autocorrelation dies down exponentially with respect to the daily lags. This behavior induces that the time-series η_t should include the moving-average of the second order in the daily seasonal cycles; i.e. η_t includes the summation of two moving-average signals: $\theta_1^{(d)}\varepsilon_{t-24}^{(h)} + \theta_2^{(d)}\varepsilon_{t-48}^{(h)}$, where $\theta_1^{(d)}$ and $\theta_2^{(d)}$ represent the coefficients of the moving average terms of the model. Moreover, $\varepsilon_t^{(h)}$ shows the non-deterministic part of η_t.

The values of the last two columns of Table 8.3 determine that the ACF and PACF of η_t in weekly lags ($L = 168, 336, \ldots$) show similar behavior as in daily lags.

In fact, the values of ACF will suddenly falls off after the first weekly lag; while PACF declines exponentially with respect to the weekly lags. This combination of behaviors lead us to the conclusion that the time-series η_t should include the moving-average of the first order in the weekly seasonal cycle, i.e. η_t has a non-deterministic moving-average part in the form of $\theta_1^{(w)}\varepsilon_{t-168}^{(h)}$ where $\theta_1^{(w)}$ represents the coefficient of the first moving-average term in weekly lags.

Table 8.3 suggests the following AR/MA-based model for WSS time-series η_t

$$\eta_t = \underbrace{\phi_1^{(h)}\eta_{t-1}}_{\text{non-seasonal}} + \underbrace{\theta_1^{(d)}\varepsilon_{t-24}^{(h)} + \theta_2^{(d)}\varepsilon_{t-48}^{(h)}}_{\text{daily-seasonal}} + \underbrace{\theta_1^{(w)}\varepsilon_{t-168}^{(h)}}_{\text{weekly-seasonal}} + \varepsilon_t^{(h)}. \tag{8.8}$$

Fine-Tuning and Evaluation

As mentioned before, in the final step of constructing the model, the detailed parameters of the AR/MA components are determined by trying multiple values and finding the forecasting model with minimum AIC/BIC values. In the case of hourly forecaster, we figured out that the following model settings will give the minimum AIC/BIC when forecasting the hourly electrical load of PJM network in 2015:

$$\eta_t = \sum_{i=1}^{12} \phi_i^{(h)}\eta_{t-i} + \sum_{j=1}^{3} \theta_j^{(d)}\varepsilon_{t-24j}^{(h)} + \theta_1^{(w)}\varepsilon_{t-168}^{(h)} + \varepsilon_t^{(h)} \tag{8.9}$$

Table 8.4 summarizes the performance of our model comparing the other forecasters of current literature.

8.5 Summary and Outlook

In this chapter, we proposed a novel step-by-step technique for creating an AR/MA-based forecasting model. This model is used for electric power demand forecasting from short-term to medium-term horizon. In the first step, the time-series corresponding to the historical load data is evaluated to make sure that it has homo-

Table 8.4: Comparison the accuracy of forecasters presented in Pappas *et al.* [46], Shaker *et al* [53], and Dudek [43]

Work	Year	Location	Method	WN Test†††	MAPE (%)	RMSE (%)	MAE (%)	Type	AIC (1000's)	BIC (1000's)
Pappas *et al.*	2010	Greece	MMPF*	tested	1.87	not tested	not tested	short-term	not tested	not tested
Pappas *et al.*	2010	Greece	AICC**	not tested	1.98	not tested	not tested	short-term	not tested	not tested
Shaker *et al.*	2014	Alberta Canada	WNN***	not tested	not tested	1.328	0.983	short-term	not tested	not tested
Shaker *et al.*	2014	Alberta Canada	MLPNN+	not tested	not tested	3.724	2.556	short-term	not tested	not tested
Shaker *et al.*	2014	Alberta Canada	RBFNN++	not tested	not tested	2.287	1.712	short-term	not tested	not tested
Dudek	2016	Polish Network	PCR+++	not tested	1.15	not tested	not tested	short-term	not tested	not tested
Dudek	2016	Polish Network	PLSR†	not tested	1.09	not tested	not tested	short-term	not tested	not tested
This work	2016	PJM Network	MTS ARMA††	tested	0.86	1.24	0.92	short-term	tested	tested

*Multi-Model Partitioning Filter
**Corrected Aikaike Information Criterion
***Wavelet Neural Network
+Multi-Layer Perceptron Neural Network
++Radial Basis Function Neural Network
+++Principle Component Regression
†Partial Least-Square Regression
††Multi Time-Scale ARMA
†††WN Test: White Noise Test

geneous variance over time. In the case that the time-series is not variance homogeneous, we apply an appropriate logarithmic Box-Cox transformation in order to convert it into a time-series with variance homogeneity. In the next step, the ACF values of the transformed time-series are examined to see if there exists any indication of being non-stationary at the non-seasonal level and seasonal cycles (daily, weekly, and annually). If the ACF values either fall off or decline in the first few lags, it should be considered stationary. On the other hand, if the ACF values either fall off after a considerable number of lags or declines quite slowly, it should be considered as non-stationary. To eliminate the non-stationary indications, we iteratively apply differentiating transformations on the time-series. In the next step, the stationary time-series, resulted from the second step, is modeled to a moving average and/or autoregressive model based on the behavior of its corresponding ACF and

PACF in different time resolutions (hourly, daily, weekly, and annually). For any given time resolution, if the ACF values of the time-series cut off in the first few lags, while the PACF values are reduced exponentially or in a Sine-wave manner, the time-series in that specific time resolution is best fit to a moving-average model. On the other hand, if the PACF values of the time-series substantially fall off in the first few lags, while the ACF values are reduced exponentially or in a Sine-wave way, the best model for that specific time resolution is autoregressive.

In order to fine-tune the model, we utilize AIC/BIC which deals with the trade-off between appropriate fitness of the model and the model complexity. To this end, we consider a penalty value for complex models which may only work precisely for the analyzed training set; however, the conventional error measurements are considerably dependent on the given training set and measure the fitness of the model merely with respect to the training set. If the forecaster's flexibility is our most important concern, i.e. our main objective is to develop a forecaster which can be broadly used for different data training sets without deteriorating the forecaster's performance, the BIC is a better alternative (compared with AIC) for quantifying the forecaster's utility since BIC penalizes the complexity of forecasting models more than AIC. However, AIC is a better criterion if we choose to compromise model flexibility by some degree in order to gain more fitness and improve the accuracy of forecaster by reducing the forecasting error.

In order to validate the effectiveness of the proposed forecaster, we implement it for forecasting the electric power demand of a real-world example from PJM interconnection. The accuracy of the proposed forecaster is evaluated by Bartlett's periodogram-based test and quantified by multiple conventional error types. The ourperformance of the propsoed forecaster, compared with the previous studies, is validates using three error metrics (MAPE, RMSE, and MAE). For instance, the

MAPE value of our forecaster for PJM hourly-metered load data is 0.86%, which shows 21% reduction compared with Dudek's forecaster which is developed in 2016 [43].

Bibliography

[1] E. Gonzalez–Romera, M. A. Jaramillo-Moran, and D. Carmona–Fernandez, " Monthly electric energy demand forecasting based on trend extraction," *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1946-1953, Nov. 2006.

[2] J. W. Taylor and P. E. McSharry, " Short-term load forecasting methods: An evaluation based on European data," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 2213-2219, Nov. 2007.

[3] M. Shahidehpour, H. Yamin, and Z. Li, *Market operations in electric power systems: forecasting, scheduling, and risk management* , Wiley Online Library, 2002

[4] K. Moslehi and R. Kumar, " A reliability perspective of the smart grid," *IEEE Trans. on Smart Grid*, vol. 1, no. 1, pp. 57–64, Jun. 2010.

[5] M.H. Amini, O. Karabasoglu, Marija D. Ilić, Kianoosh G. Boroojeni, and S.S. Iyengar, "ARIMA-based Demand Forecasting Method Considering Probabilistic Model of Electric Vehicles' Parking Lots," *IEEE PES General Meeting 2015*, Denver, CO, USA, July 26-30, 2015.

[6] F. Rahimi and A. Ipakchi, " Demand response as a market resource under the smart grid paradigm," *IEEE Trans. on Smart Grid*, vol. 1, no. 1, pp. 82–88, Jun. 2010.

[7] S. Bahrami and V. W. S. Wong, "An autonomous demand response program in smart grid with foresighted users," *In Proc. of IEEE Smart Grid Communications (SmartGridComm)*, Miami, FL, 2015, pp. 205-210.

[8] F. Kamyab, M.H. Amini, S. Sheykhha, M. Hasanpour, and M.M. Jalali, "Demand Response Program in Smart Grid Using Supply Function Bidding Mechanism," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 1277-1284, May 2016.

[9] Nadali Mahmoudi, New Demand Response Framework and Its Applications for Electricity Markets. PhD Dissertation. *The University of Queensland*, 2015.

[10] Xiao Zhang, G. Hug, J. Zico Kolter, and I. Harjunkoski, "Model Predictive Control of Industrial Loads and Energy Storage for Demand Response," *IEEE PES General Meeting*, 2016.

[11] S. Bahrami and A. Sheikhi, "From Demand Response in Smart Grid Toward Integrated Demand Response in Smart Energy Hub," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 650-658, March 2016.

[12] L. Hernandez, et al, "A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart Buildings," *IEEE Communications Surveys & Tutorials*, pp. 1-36, Apr. 2014.

[13] W. Charytoniuk and M.-S. Chen, " Very short-term load forecasting using artificial neural networks," *IEEE Trans. Power Syst.*, vol. 15, no. 1, pp. 263-268, Feb. 2000.

[14] C. Guan, P. B. Luh, L. D. Michel, Y. Wang, and P. B. Friedland, " Very short-term load forecasting: wavelet neural networks with data pre-filtering," *IEEE Trans. Power Syst.*, vol. 28, no. 1, pp. 30-41, Jan. 2013.

[15] A. Piras, A. Germond, B. Buchenel, K. Imhof, and Y. Jaccard, " Heterogeneous artificial neural network for short term electrical load forecasting," *IEEE Trans. Power Syst.*, vol. 11, no. 1, pp. 397-402, 1996.

[16] N. Amjady, "Short-term bus load forecasting of power systems by a new hybrid method," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 333–341, Feb. 2007.

[17] L. Zjavka and V. Snášel, "Short-term power load forecasting with ordinary differential equation substitutions of polynomial networks." *Electric Power Systems Research*, vol. 137, pp. 113-123, 2016.

[18] E. Doveh, P. Feigin, D. Greig, and L. Hyams, " Experience with FNN models for medium term power demand predictions," *IEEE Trans. Power Syst.*, vol. 14, no. 2, pp. 538-546, May. 1999.

[19] M. S. Kandil, S. M. El-Debeiky, and N. E. Hasanien, " Long-term load forecasting for fast developing utility using a knowledge-based expert system," *IEEE Trans. Power Syst.*, vol. 17, no. 2, pp. 491-496, May. 2010.

[20] R. J. Hyndman and S. Fan, "Density forecasting for long-term peak electricity demand," *IEEE Trans. Power Syst.*, vol. 25, no. 2, pp. 1142-1153, May. 2010.

[21] A. S. Khwaja, M. Naeem, A. Anpalagan, A. Venetsanopoulos, and B. Venkatesh, "Improved short-term load forecasting using bagged neural networks." *Electric Power Systems Research*, vol. 125, pp. 109-115, 2015.

[22] J. Fiot and F. Dinuzzo, " Electricity Demand Forecasting by Multi-Task Learning," *IEEE Trans. Smart Grid*, vol. 99, 2016.

[23] M. H. Amini, and A. Islam, "Allocation of electric vehicles' parking lots in distribution network," *IEEE Innovative Smart Grid Technologies Conference (ISGT)*, 2014.

[24] I. Rahman, et al, "Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures." *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 1039-1047, 2016.

[25] S. Bahrami and M. Parniani, "Game Theoretic Based Charging Strategy for Plug-in Hybrid Electric Vehicles," *IEEE Transactions on Smart Grid*, vol. 5, no. 5, pp. 2368-2375, Sept. 2014.

[26] Shahab Bahrami and Vincent W. S. Wong, "A potential game framework for charging PHEVs in smart grid", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, Canada, pp. 28-33, 2015.

[27] K.B. Song, Y.S. Baek, D.H. Hong, and G. Jang, "Short-term load forecasting for the holidays using fuzzy linear regression method," *IEEE Transactions on Power Systems*, vol. 20, no. 1, pp. 96–101, Feb. 2005.

[28] P. E. McSharry, S. Bouwman, and G. Bloemhof,"Probabilistic forecast of the magnitude and timing of peak electricity demand," *IEEE Transactions on Power Systems*, vol. 20, no. 2, pp. 1166–1172, May 2005.

[29] B. J. Chen, M. W. Chang, and C. -J. Lin, "Load forecasting using support vector machines: A study on EUNITE competition 2001," *IEEE Trans. Power Syst.* , vol. 19, no. 4, pp. 1821–1830, Nov. 2004.

[30] A. Pardo, V. Meneu, and E. Valor, " Temperature and seasonality influences on Spanish electricity load," *IEEE Energy Econ.*, vol. 24, pp. 55–1830, 2002.

[31] A. P. Douglas, A. M. Breipohl, F. N. Lee, and R. Adapa, "Load forecasting using support vector machines: A study on EUNITE competition 2001," *IEEE Trans. Power Syst.* , vol. 13, no. 4, pp. 1507–1513, Nov. 1998.

[32] M. S. Kandil, S. M. El-Debeiky, and N. E. Hasanien, "Overview and comparison of long-term forecasting techniques for a fast developing utility: Part I ," *Elect. Power Syst. Res.* , vol. 58, no. 1, pp. 11–17, May 2001.

[33] H. Morita, T. Kase, Y. Tamura, and S. Iwamoto, "Interval prediction of annual maximum demand using grey dynamic model ," *Int. J. Elect. Power Energy Syst.* , vol. 18, no. 7, pp. 409–413, Oct. 1996.

[34] D. C. Park, *et al*, "Electric Load Forecasting Using An Artificial Neural Network," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 442449, May. 1991.

[35] K. Y. Lee, Y. T. Cha, and J. H. Park, "Short-term load forecasting using an artificial neural network," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 124132, Feb. 2001.

[36] H. S. Hippert, C. E. Pedreira, and R. C. Souza, "Neural networks for short-term load forecasting: a review and evaluation," *IEEE Trans. Power Syst.*, vol. 16, no. 1, pp. 4455, Feb. 2001.

[37] Arif. I. Sarwat, M. H. Amini, Alexander Domijan Jr., Aleksandar Damnjanovic, and F. Kaleem, "Weather-based Interruption Prediction in the Smart Grid Utilizing Chronological Data," *Journal of Modern Power Systems and Clean Energy*, vol. 4, no. 2, pp. 308-315, April 2016.

[38] N. Amjady, "Short-term hourly load forecasting using time-series modeling with peak load estimation capability," *IEEE Trans. Power Syst.*, vol. 16, no. 3, pp. 498–505, Aug. 2001.

[39] W. Charytoniuk and M. -S. Chen, " Very short-term load forecasting using artificial neural networks," *IEEE Trans. Power Syst.*, vol. 15, no. 1, pp. 263–268, Aug. 2002.

[40] Y. Wang, Q. Xia, and C. Kang, " Secondary Forecasting Based on Deviation Analysis for Short-Term Load Forecasting," *IEEE Trans. Power Syst.*, vol. 26, no. 2, pp. 500–507, Apr. 2011.

[41] C. Garca-Ascanio and C. Mate, " Electric power demand forecasting using interval time series: a comparison between VAR and iMLP," *Energy Policy* , vol. 38, no. 2, pp. 715–725, Feb. 2010.

[42] D. C. Park, M. A. El-Sharkawi, R. J. Marks II, L. E. Atlas, and M. J. Damborg, " Electric load forecasting using an artificial neural network," *IEEE Trans. Power Syst.*, vol. 6, no. 2, pp. 442–449, May. 1991.

[43] G. Dudek, "Pattern-based local linear regression models for short-term load forecasting," *Electric Power Systems Research*, vol. 130, January 2016, pp. 139-147.

[44] T. H. D. Ngo, "The Box-Jenkins methodology for time series models," SAS Global Forum, vol. 54, 2013.

[45] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, "Time-series analysis: forecasting and control," CA: Holden-Day, 1976.

[46] S. S. Pappas, L. Ekonomou, P. Karampelas, D. C. Karamousantas, S. K. Katsikas, G. E. Chatzarakis, and P. D. Skafidas, "Electricity demand load forecasting of the Hellenic power system using an ARMA model," *Electric Power Systems Research*, vol. 80, no. 3, , pp. 256-264, March 2010.

[47] A. A. El Desouky and M. M. Elkateb, "Hybrid adaptive techniques for electric-load forecast using ANN and ARIMA," *IEE Proceedings-Generation, Transmission and Distribution*, vol. 147, no. 4, pp. 213-217, 2000.

[48] S. J. Huang and K. R. Shih, "Short-term load forecasting via ARMA model identification including non-Gaussian process considerations," *IEEE Transactions on Power Systems*, vol. 18, no. 2, pp. 673-679, 2003.

[49] J. Contreras, R. Espinola, F. J. Nogales, and A. J. Conejo, "ARIMA models to predict next-day electricity prices," *IEEE Transactions on Power Systems*, vol. 18, no. 3, pp. 1014-20, 2003.

[50] J. W. Taylor, "Triple Seasonal Methods for Short-term Load Forecasting," *European Journal of Operational Research*, vol. 204, no. 1, pp. 139-152, 2010.

[51] G. E. P. Box and D. R. Cox, "An Analysis of Transformations," *Journal of the Royal Statistical Society*, Series B, vol. 26, pp. 211-252.

[52] http://www.pjm.com/markets-and-operations/ops-analysis/historical-load-data.aspx

[53] H. Shaker, H. Chitsaz, H. Zareipour, and D. Wood, "On Comparison of Two Strategies in Net Demand Forecasting Using Wavelet Neural Network," North American Power Symposium (NAPS), pp. 1-6, Sep. 2014.

CHAPTER 9

CLOUD NETWORK DATA SECURITY IN SMART GRIDS

The recent advances in cloud computing have substantially changed people's understanding of computing hardware/software infrastructure and development methods. This fast transition to the clouds has coincided with the transition of mainframe computers to client/server models which has facilitated the utilization of cloud computing in different enterprises. This world-wide transition has caused serious concerns regarding the confidentiality, integrity, and availability of communication and information systems participating in the cloud models as relocating data through the communication systems to the clouds causes various security and privacy issues that did not exist in traditional models before. This chapter proposes an Oblivious Routing-based Security Scheme (***ORSS***) which effectively addresses the security issues caused by DDoS zombie attacks related to the data communication in cloud systems. The proposed security scheme is mathematically proved to guarantee some security low-threshold in long-term run. The simulation results on a general case study support the theoretical bound while showing that the proposed solution enhances the cloud system response time.

9.1 Introduction

Cloud computing has become a growing architectural model for organizations seeking to decrease their computer systems maintenance costs by migrating their computational tasks to third party organizations who offer software-as-a-service, platform-as-a-service, etc. As the result of such transition, many critical security concerns

[0]Part of this chapter has been reprinted with permission from Kianoosh G. Boroojeni et al., "Smart Grids: Security and Privacy Issues," Springer Publication, 2017.

have emerged among the clients of clouds including data blocking and data leakage in the form of Distributed Denial of Service (DDoS) attacks [1]. As the cloud customers may have possibly thousands of shared resources in a cloud environment, the likelihood of DDoS attack is much more than a private architecture [2].

One of the main goals of DDoS attacks in cloud environments is to exhaust computer resources especially network bandwidth and CPU time so that the cloud service gets unavailable for the real legitimate clients [2]. In a general DDoS attack, the attacker usually mimics the legitimate web data traffic pattern to make it difficult for the victims to identify the attackers. This type of attack called "zombie attack" is a common type in cloud computing and is widely considered to be successful in circumventing the big portion of attack detection algorithms which work based on the abnormality of the traffic pattern generated by the DDoS attackers [2, 6, 7].

In recent years, many attempts have been made to mitigate the DDoS zombie attacks in general or specifically in the cloud environments. As we will see later in the literature review, these attempts have all relied on the conventional Internet routing algorithms and protocols (e.g. link-state, distance-vector) to route the data flow through the Internet. To the best of our knowledge, this is the first try ever made to mitigate the DDoS attacks by not utilizing the conventional Internet routing algorithms and creating a novel overlay network routing scheme based on the oblivious network design principles and algorithms. This project mainly focuses on the mitigation of the zombie attack and its destructive effect on the cloud services. The proposed mechanism is compatible with many of the DDoS detection and mitigation algorithms and can be integrated with them in order to enhance the security.

Related Work

Here, we review some of the recent studies on how to detect and mitigate the DDoS attacks in general and specifically in cloud environments.

Detection Mechanisms

The most common type of DDoS attacks in Internet is flooding. This popular attack has made many attempts in order to create effective countermeasures against it ([3, 4, 5]). The designed/implemented countermeasures have an attack detection algorithm which predicts the occurrence of attacks utilizing the fact that the pattern of data traffic made by the hackers is deviated from the normal (natural) traffic pattern. The greater this deviation is, the higher the precision of the attack detection algorithms are. In 2015, Wang et al. proposed a graphical model in order to detect the security attacks (distributed denial of service) and inferring the probability distribution of attack [16].

As a response to this types of counter attacks, the attackers may regulate their generated data traffic pattern in a way that it no longer distinguishable with the other network data traffic and confuses the attack detection algorithms. Zombie attack is a good example in which the attackers perform DDoS attacks through a Poisson process in order to confuse the detection algorithms [6, 7]. Joshi et al. designed a cloud trace back model in dealing with DDoS attacks and addressed its performance using back propagation neural network and experimentally showed that the model is useful in tackling DDoS attacks [2].

Mitigation Mechanisms

In 2013, Mishra et al. proposed Multi-tenancy and Virtualization as two major solutions to mitigate the DDoS attacks in cloud environments [8]. Zissis et al.

proposed a security solution to the security issues which helps the cloud clients make sure of their security by trusting a Third Party in a way that trust is created in a top-down manner where each layer of the cloud system trusts the layer lying immediately below it [9]. In 2011, Lua et al. reduced the possibility of DDoS attacks by utilizing a transparent and intelligent fast-flux swarm network which is a novel, efficient, and secure domain naming system.

This chapter proposes an Oblivious Routing-based Security Scheme ($\mathcal{ORSS}$) which mitigates the DDoS attack and related performance issues effectively while routing data through the Internet connecting the cloud servers and customers together. An oblivious routing algorithm usually proposes an overlay network in the form of a spanning tree or a set of trees on the graph representing the network [19]. These schemes are usually pretty versatile to the time-varying network topology and dynamic traffic pattern between different source-sink pairs by routing the traffic flows in a distributed way over the network and preventing the edge/node congestion. Moreover, the routing cost in such algorithms are usually proved to be asymptotically bounded to some coefficient of the optimal/minimum cost; this coefficient which is called the *competitiveness ratio* of the algorithm (or its corresponding routing scheme) has a value of greater than one [19, 20].

In this chapter, by defining the routing cost of data flow as a representation of security attack risk, we utilize an oblivious routing algorithm of competitiveness ratio $\boldsymbol{O(\log^2 |V|)}$ mentioned in [19] to design a secure overlay network routing scheme. The proposed security scheme is mathematically proved to guarantee some security low-threshold in long-term run. As the security attacks cause reduction in performance, we will show how $\mathcal{ORSS}$ enhance the system performance. The simulation results will support the theoretical bound.

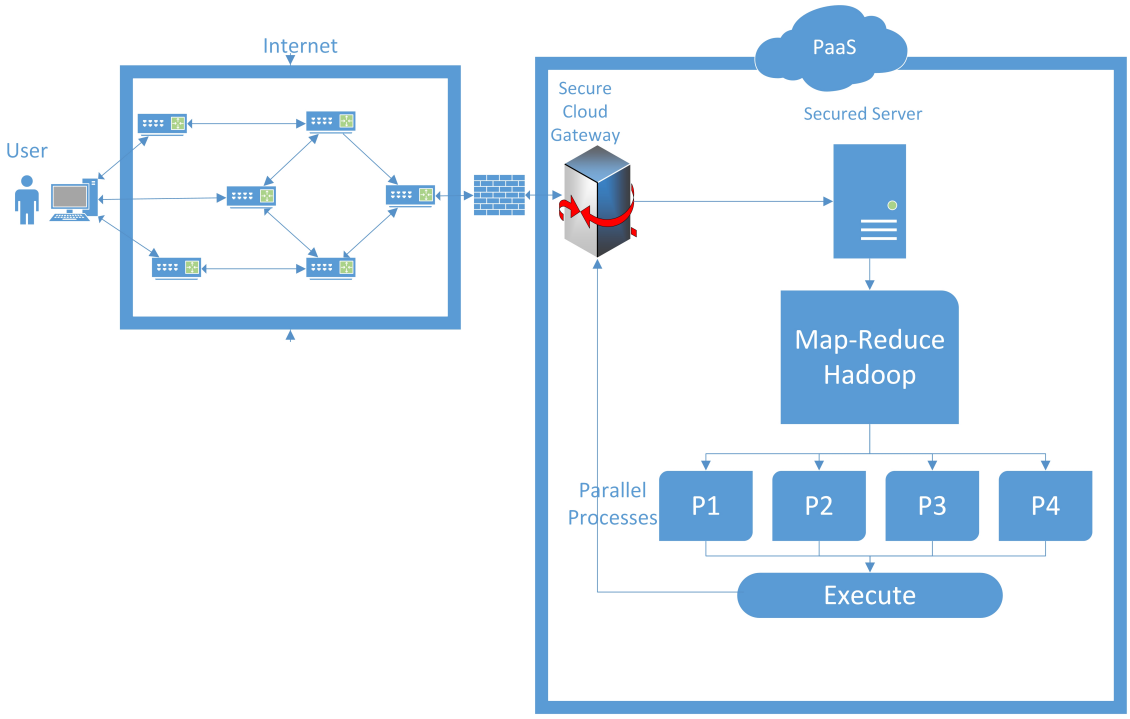


Figure 9.1: The schematic representation of the problem scope.

9.2 Data Security Protection in Cloud-connected Smart Grids

Consider that in a cloud system, the cloud servers are located in the private cloud which has a connecting point (cloud gateway) to the Internet. The cloud customers located in the public cloud communicate with the cloud servers via the cloud gateway. A two-way communication between customers and the gateway is done through a network of routers connected with pre-established TCP connections.

Let weighted graph $\boldsymbol{\mathcal{G} = (V, E, \lambda)}$ denote the network connecting cloud secured gateway $\boldsymbol{g \in V}$ and customers $\boldsymbol{c_1, c_2, \ldots, c_k \in V}$ together where $\boldsymbol{E}$ specifies the set of TCP connections connecting the customers, gateway and the internal network nodes together and $\boldsymbol{\lambda : E \mapsto \mathbb{R}_{\geq 0}}$ specifies the graph edge weights which is described later. Figure 9.1 depicts the cloud system, communication network, secured gateway, an example of cloud service (Platform as a Service) and the customer.

A security attack may be performed by a customer of a third party adversary by sniffing data or blocking data flow. Assuming that the data is encrypted by customer utilizing the elliptic curve cryptography and routed through the network of routers connected by TCP connections, it will make it very difficult for the hacker to identify the parameters used to draw the curve and then the point on the curve around which encryption is carried out. Furthermore, on receipt of the customer's task from the communicating network, the cloud gateway checks and verifies if any hacking took place. As a result, the security concern in such system is blocking data flow by the adversary in the communicating network (public cloud) before entering the cloud gateway through a distributed denial of service (DDoS) attack.

Attack Model

Distributed denial of service attacks are usually designed to be hard to detect. Attackers are mimicking network natural traffic statistical features to neutralize the detectors' effort which are based on these features. Zombie attack is a good example in which the attackers perform DDoS attacks through a Poisson process in order to confuse the detection algorithms [6, 7].

Consider random process $\boldsymbol{N_t^{(e)}}$ denotes the number of security attacks occurred in TCP connection $\boldsymbol{e \in E}$ during time interval $\boldsymbol{[0, t]}$ for every $\boldsymbol{t > 0}$. This is *homogeneous Poisson process* of intensity $\boldsymbol{\lambda_e}$ where $\boldsymbol{\lambda}$ is the weight function of the aforementioned weighted graph $\boldsymbol{\mathcal{G}}$. We call $\boldsymbol{\lambda_e}$ as the *attack rate* in connection $\boldsymbol{e}$. Note that for the sake of simplicity, we consider the attack rates as constant function of time; however, the proposed solution can be extended to the attacks with non-homogenous Poisson pattern.

Considering that there is a data flow of magnitude $\boldsymbol{f > 0}$ is passing through the

TCP connection e of transmission rate r_e, the data flow lasts for f/r_e time units. This implies that an average of $\lambda_e f/r_e$ takes place during the mentioned data flow session as the number of zombie attacks following a Poisson process. This value is formulated as the cost value incurred in edge e given a flow of f passing through it:

$$\text{cost}_e(f) = \frac{\lambda_e f}{r_e} \tag{9.1}$$

Assuming $\mathcal{N}_t$ as the random process of the total number of attack taking place in the TCP connections of the communicating network (i.e. $\mathcal{N}_t = \sum_e N_t^{(e)}$), then the expected value of the total number of attacks in the network, given that data flow of magnitude f_e is flowing through the connection e, in terms of cost value will then be equal to:

$$\overline{\mathcal{N}}(f.) = \sum_{e \in E} \text{cost}_e(f_e) = \sum_{e \in E} \frac{\lambda_e f_e}{r_e} \tag{9.2}$$

Performance Evaluation

When an attack takes place in a TCP connection which flows customer c's data, c needs to send the data again through the network which increases the system response time to his query. Assuming that the customer considers path $p \subseteq E$ for the data of size f to flow toward the cloud secured gateway, the increased response time because of the attack is given by $\sum_{e \in p} f/r_e$. Assuming that query q has been sent through the path p by customer c, and the cloud servers total response time to the query (excluding the communicating network delay) is ρ_q, the total response time is represented by a random variable R_{pq} defined as follows (for the sake of simplicity and without loss of generality, assume that the query itself and its response is sent through the network in a data packet of unit size and the network

transmission delay is the only considerable delay in the network):

$$\boldsymbol{R_{pq} = \rho_q + (2 + N_p) \sum_{e \in p} f_e / r_e,} \tag{9.3}$$

where $\boldsymbol{N_p}$ denotes the random variable counting the number of attacks taking place in path $\boldsymbol{p}$ given that flow $\boldsymbol{f_e}$ exists in edge $\boldsymbol{e}$ in the time that query $\boldsymbol{q}$ is submitted and responded by the cloud servers. Note that in Equation 9.3, $\boldsymbol{2 + N_p}$ specifies the total number of times that query $\boldsymbol{q}$ is sent to the cloud server or responded back to the customer (data flow would exist exactly twice through path $\boldsymbol{p}$ if no attack occurs).This variable is the summation of $\boldsymbol{|p|}$ Poisson distributed variables $\boldsymbol{N_p^{(e)}}$ corresponding to the edges (e) participating in the path $\boldsymbol{p}$ such that $\boldsymbol{N_p^{(e)}}$ has a mean of $\boldsymbol{\lambda_e \cdot f_e / r_e}$. Henceforth, the expected value of the total system response time for arbitrary query $\boldsymbol{q}$ is obtained in the following way:

$$\boldsymbol{E[R_{pq}] = \rho_q + (2 + \sum_{e \in p} \frac{\lambda_e f_e}{r_e}) \sum_{e \in p} f_e / r_e} \tag{9.4}$$

This section proposes the novel $\boldsymbol{\mathcal{ORSS}}$ method which utilizes a top-down oblivious routing scheme to solve security problem of the network connection between cloud server and its customers. Moreover, the method uses the optimization toolbox of MATLAB 2015a [18] made for minimizing the class of linearly-constrained optimization problems. Here, we show how Algorithm 4 illustrates the construction of the oblivious routing scheme mentioned in [19].

Algorithm 4 gets the weighted graph $\boldsymbol{\mathcal{G}}$ as its input and returns function $\boldsymbol{\mathbb{S} : \{G_i\}_1^n \times \{D_i\}_1^q \mapsto \mathcal{P}(E)}$ which represents the oblivious routing scheme and specify a path[1] in graph $\boldsymbol{\mathcal{G}}$ for every pair of source-sink nodes. The algorithm starts with generating $\boldsymbol{27 \log |V|}$ random HDS's utilizing randomized Algorithm 4. The

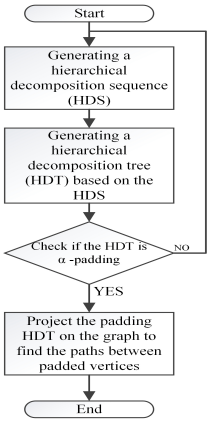
[1]In this chapter, path of a graph is considered as a simple path and represented by a subset of edge set $\boldsymbol{E}$ such that there exist a permutation of edges in a path where the first edge is incident to the start node of the path, each two consecutive edges are incident to a common node, and the last edge is incident to the end node of the path.

reason for generating multiple random HDS's is to assure the existence of at-least one HDS $\bar{\boldsymbol{H}}$ for every pair of nodes $\boldsymbol{source} \in \boldsymbol{F}$ and $\boldsymbol{sink} \in \boldsymbol{G}$. In fact, Iyengar *et al.* in [19] proved that the probability of existing at-least one $\boldsymbol{\alpha}$-padding HDS among the $\boldsymbol{c}\log|\boldsymbol{V}|$ HDS's constructed by Algorithm 4 is more than $\mathbf{1}-\boldsymbol{e}^{-\frac{(\boldsymbol{c}-\mathbf{2})^{\mathbf{2}}}{\mathbf{2c}}}$ (for every $\boldsymbol{\alpha} \leq \mathbf{1/8}$). By considering $\boldsymbol{c} \geq \mathbf{27}$, the mentioned probability would be greater than $\mathbf{1}-\mathbf{10}^{-\mathbf{5}}$ which provides a reasonable theoretical guarantee that Algorithm 4 will find at-least one HDS for every source-sink pair.

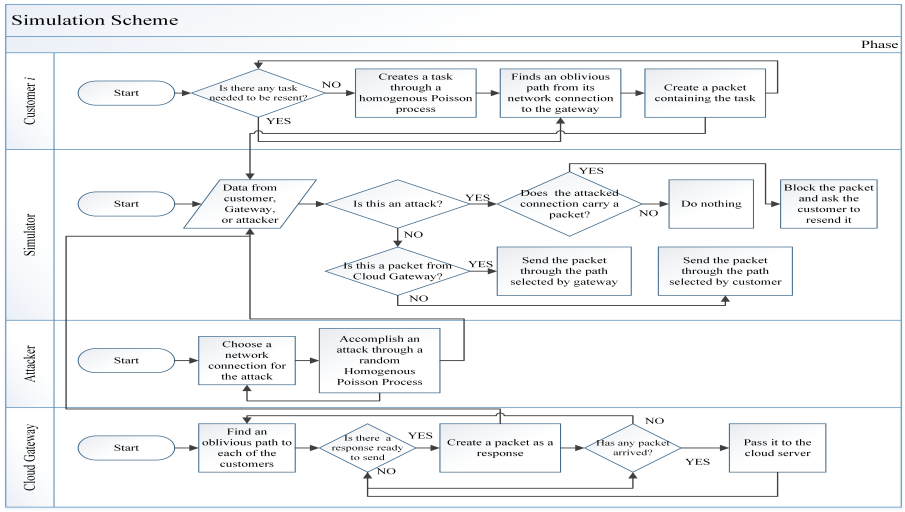
After creating $\mathbf{27}\log|\boldsymbol{V}|$ random HDS's and their corresponding HDTs, Algorithm 4 runs its main loop in lines 5 to 11 for every pair of source-sink nodes. Assume $\boldsymbol{T}$ as the HDT corresponding to the $\boldsymbol{\alpha}$-padding HDS of an arbitrary pair of source-sink nodes. Tree $\boldsymbol{T}$ would have a pair of leaves (level-zero nodes) corresponding to the source and sink (as $\boldsymbol{\mathcal{G}}$ is supposed to be a weighted graph of the weight function greater than one for every edge). Let $\boldsymbol{p}$ denote the only tree path connecting the mentioned leaves together. Finally, the resulted routing scheme is computed in line 11 where the path between source and sink nodes $\mathbb{S}(\boldsymbol{source}, \boldsymbol{sink})$ is obtained by projecting $\boldsymbol{p}$ on graph $\boldsymbol{\mathcal{G}}$. The projection of path $\boldsymbol{p}$ of HDT $\boldsymbol{T}$ on graph $\boldsymbol{\mathcal{G}}$ is defined in the following way:

Consider $\boldsymbol{\gamma_e}$ as the shortest path between the incident nodes of arbitrary edge $\boldsymbol{e} \in \boldsymbol{p}$ in graph $\boldsymbol{\mathcal{G}}$. The projection of tree path $\boldsymbol{p}$ on the graph is obtained by concatenating all of the shortest paths $\boldsymbol{\gamma_e}$s back to back. In the case that the concatenation result is not a simple path and has crossed some nodes more than once, the projected path will be the shortest simple path corresponding to the concatenation result.

As mentioned in Equation 1, by considering the average number of attacks as the traffic cost of a given edge, function $\text{cost}(\boldsymbol{e})$ is a sub-additive function of the data flow magnitude $\boldsymbol{f_e}$. Consequently, regarding [19, 20], if data flow is routed by the aforementioned oblivious routing scheme $\mathbb{S}$, the total cost in all of the graph



(a) Flowchart of the oblivious routing algorithm



(b) The Cross Flowchart of the simulation scheme.

Figure 9.2: The flowchart of oblivious routing algorithm

edges; i.e. the expected total number of attacks in the network connections doesn't exceed $\boldsymbol{\psi}$ times of the minimum possible expected attacks where $\boldsymbol{\psi = O(\log^2 |V|)}$.

In PaaS, the provider might give some control to the people to build applications on top of the platform. But any security below the application level such as host and network intrusion prevention will still be in the scope of the provider and the provider has to offer strong assurances that the data remains inaccessible between applications. PaaS is intended to enable developers to build their own applications on top of the platform. As a result it tends to be more extensible than SaaS, at the expense of customer-ready features. This tradeoff extends to security features and capabilities, where the built-in capabilities are less complete, but there is more flexibility to layer on additional security.

Platform as a service (PaaS) is a category of cloud computing services that provides a platform allowing customers to develop, run, and manage Web applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app. PaaS can be delivered in two ways: as

a public cloud service from a provider, where the consumer controls software deployment and configuration settings, and the provider provides the networks, servers, storage and other services to host the consumer's application; or as software installed in private data centers or public infrastructure as a service and managed by internal IT departments.

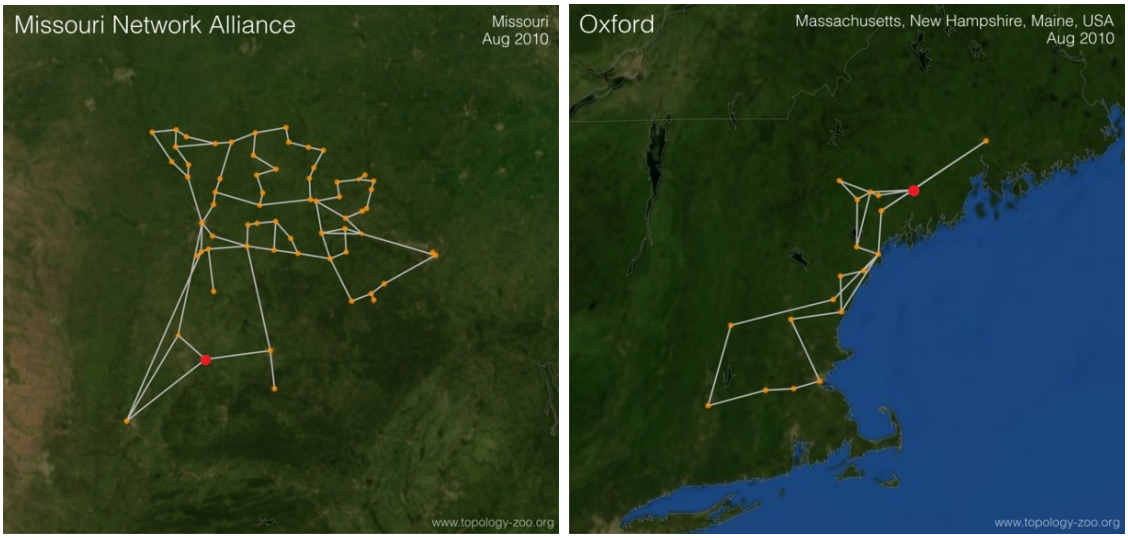
9.2.1 Simulation Scheme

Consider a cloud which provides its customers with a Platform as a Service (Paas). The customers have two-way communication with cloud's secured gateway through a network of routers connected by pre-established TCP connections (see Figure 9.1). The network topology is represented by graph $\boldsymbol{G = (V, E)}$ where $\boldsymbol{V}$ is the set of network routers and $\boldsymbol{E}$ is the TCP connections established between them. The secured gateway is connected to a server which is responsible to perform customers' tasks. The cloud secured server utilizes map-reduce model to perform customers tasks which are assumed to be oversized.

Assume that there are $\boldsymbol{k}$ customers spread over the network where the $\boldsymbol{i^{th}}$ one sends the average of $\boldsymbol{\mu_i}$ tasks to the cloud secured gateway within a homogeneous Poisson process. Each task is sent via a message of size $\boldsymbol{S_i \sim N(m_i, \sigma_i^2)}$.

For every TCP connection $\boldsymbol{e \in E}$, we consider $\boldsymbol{r_e}$ as its transmission rate and $\boldsymbol{\lambda_e}$ as the average number of security attacks which follows a homogeneous Poisson process.

Finally, we assume that every task takes $\boldsymbol{T \sim N(\tau, t^2)}$ time to be performed and the result will be transferred back through network $\boldsymbol{G}$ as a message of size $\boldsymbol{S' \sim N(m', \sigma'^2)}$.



(a) Missouri network alliance. Location: Missouri, 2010 [22].

(b) Oxford network. Location: MA, NH, MN, 2010 [22].

Figure 9.3: Real-world network topologies used in the case study. The big vertex in each graph representation of network topologies specifies the location of cloud server.

9.2.2 Simulation Results

We implemented the proposed oblivious routing algorithm on two real-world network topologies obtained by Knight et al.'s study of network topology [21] shown in Figure 9.3. The big red circular vertex represents the cloud gateway; while the other circles represent the cloud customers.

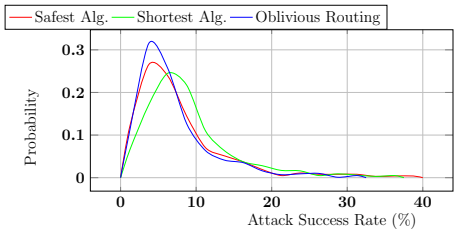
The following table (Table 9.1) specifies the performance of our novel routing scheme in contrast with the performance of the common traditional routing algorithms on the Oxford and Missouri network topologies. This table compares the percentage of successful attacks and the average increase in the system response time (obtained by Equation 4) because of DDoS zombie attacks. The comparison illustrates the superior performance of our novel scheme in both the average rate of successful attacks and response time increase because of zombie attacks. The algorithms that are compared to our novel oblivious routing algorithm are as follow:

- Safest path algorithm: the traditional single-source shortest path when the weight of each graph edge is considered as $\boldsymbol{\lambda_e}$.
- Shortest path algorithm: the traditional single-source shortest path when the graph is assumed to be unweighted.

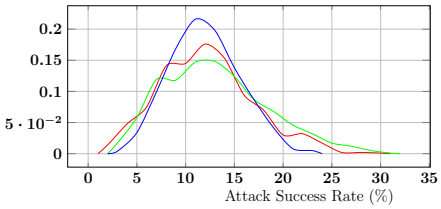
Table 9.1: Comparing the percentage of successful attacks and the average increase in the system response time (obtained by Equation 4) because of DDoS zombie attacks. The comparison illustrates the superior performance of our novel scheme.

		Oblivious Routing	Safest Path Routing	Shortest Path Routing
Oxford Topology$_{(n=20)}$	Average # of links in a path	6	5.75	4.8750
	Average rate of successful attack (%)	7.2257	7.9765	9.2090
	Response time increase because of attacks (%)	0.1036	0.1313	0.1611
Missouri Topology$_{(n=67)}$	Average # of links in a path	9.5417	6.9583	6.9167
	Average rate of successful attack (%)	11.9940	12.1463	13.0464
	Response time increase because of attacks (%)	0.3439	0.3525	0.3736

Finally, Figure 9.4 compares the two aforementioned algorithms based on the probability distribution of success rate of zombie attacks (6.a, 6-b, 6-c, 6-d) and the average expected network delay curve in the time horizon (6.e, 6-f). As you see in the first two plots, our routing algorithm has a lower mode/median attack success rate than its rivals. In the second two plots, our novel routing scheme outperforms the other two algorithms by creating a lower network delay through the whole simulation time slot.



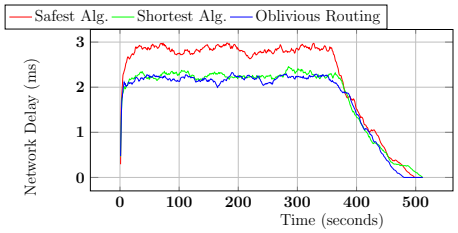
(a) Pdf of attack success rate (percentage) in Oxford network topology.



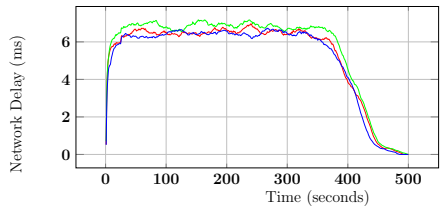
(b) Pdf of attack success rate (percentage) in Missouri network topology.

(c) Boxplot of attack success rate in Oxford network topology.

(d) Boxplot of attack success rate in Missouri network topology.



(e) Average expected total network delay over time in Oxford network topology.



(f) Average expected total network delay over time in Missouri network topology.

Figure 9.4: Comparison of different routing algorithms based on the average delay and probability distribution of successful attacks.

9.3 Summary and Outlook

The recent advances of cloud computing has substantially changed researcher's understanding of computing hardware/software infrastructure and development methods. This fast transition to cloud computing has enabled a plethora of enterprise services for client use, which has also opened new security challenges. More specifically, serious concerns regarding the confidentiality, integrity, and availability of communication and information systems have arisen as a result of rapid transition to the cloud. For example, relocating data through the communication systems to the clouds has caused various security and privacy issues that did not previously exist in traditional client/server models. This chapter proposed a novel Oblivious Routing-based Security Scheme ($\mathcal{ORSS}$) which effectively addressed the security issues caused by the Distributed Denial of Service (DDoS) security attacks related

to the data communication in cloud systems. A detailed theoretical model and the analysis including, an experimental work in the form of simulation showed the superior security and performance of the oblivious routing algorithm (utilized by the scheme) compared with conventional routing algorithms widely used today.

Bibliography

[1] C. Almond, "A Practical Guide to Cloud Computing Security," August 2009.

[2] Bansidhar Joshi, A. Santhana Vijayan, and Bineet Kumar Joshi, "Securing Cloud Computing Environment Against DDoS Attacks," 2012 International Conference on Computer Communication and Informatics (ICCCI-2012), Jan. 10-12, 2012, Coimbatore, India.

[3] J. J. B. Krishnamurthy and M. Rabinovich, "Flash crowds and denial of service attacks: characterization and implications for CDNs and Web sites," in Proc. International WWW conferences 2002.

[4] Y. Chen and K. Hwang, "Collaborative change detection of DDoS attacks on community and ISP networks," in Proc. IEEE CTS 2006.

[5] R. B. G. Carl, G. Kesidis, and S. Rai, "Denial of service attack detection techniques," IEEE Internet Computing, Jan. 2006.

[6] Shui Yu, Wanlei Zhou, and Robin Doss, "Information Theory Based Detection Against Network Behavior Mimicking DDoS Attacks," IEEE Communications Letters, Vol. 12, NO. 4, April 2008.

[7] Shui Yu and Wanlei Zhou, "Entropy-Based Collaborative Detection of DDoS Attacks on Community Networks," Sixth Annual IEEE International Conference on Pervasive Computing and Communications, Piscataway, N. J., pp. 566-571, 2008.

[8] Ankur Mishra, Ruchita Mathur, Shishir Jain, and Jitendra Singh Rathore, "Cloud Computing Security," International Journal on Recent and Innovation Trends in Computing and Communication, Volume 1, Issue 1, pp 36-39.

[9] Dimitrios Zissis and Dimitrios Lekkas, "Addressing Cloud Computing Security Issues," Future Generation Computer Systems, Volume 28, Issue 3, March 2012, pp 583-592.

[10] Farzad Sabahi, "Clou Computing Security Threats and Responses," 2011.

[11] http://cloudsecurity.trendmicro.com/

[12] "Security Management in the Cloud," http://mscerts.net/programming/Security %20Management%20in%20t he%20Cloud.aspx, 2010.

[13] N. Mead, et al., "Security Quality Requirements Engineering Methodology," Carnegie Mellon Software Engineering Institute.

[14] Dijiang Huang, Xinwen zhang, Myong Kang, and Jim Luo, "MobiCloud: Building Secure Cloud Framework for Mobile Computing and Communication," fifth IEEE International Symposium on Service Oriented System Engineering, 2010.

[15] Zhongjian Li, Naixue Xiong, Bo Yang, and Yuezhi Zhou, "SC-OA: A Secure and Efficient Scheme for Origin Authentication of Inter-domain Routing in Cloud Computing Networks," Parallel & Distributed Processing Symposium (IPDPS), pp 243-254, May, 2011.

[16] Bing Wang, Yao Zheng, Wenjing Lou, and Y. Thomas Hou, "DDoS Attack Protection in the Era of Cloud Computing and Software-Defined Networking," Computer Networks 81, pp 308-319, 2015.

[17] J. Fakcharoenphol, S. B. Rao, and K. Talwar, A tight bound on approximating arbitrary metrics by tree metrics, in Proc. of the 35th STOC, 2003, pp. 448-455.

[18] MATLAB version 8.5. Miami, Florida: The MathWorks Inc., 2015.

[19] S. S. Iyengar and K. G. Boroojeni. *Oblivious Network Routing: Algorithms and Applications*. MIT Press, 2015.

[20] A. Gupta, M. T. Hajiaghayi, and H. Racke, "Oblivious network design," in SODA 06: Proceedings of the 17th annual ACM-SIAM symposium on Discrete algorithm. New York, NY, USA: ACM, 2006, pp. 970979.

[21] Simon Knight, Hung X. Nguyen, Nickolas Falkner, Rhys Bowden, and Matthew Roughan, "The Internet Topology Zoo," IEEE Journal on Selected Areas in Communications, Vol. 29, No. 9, Oct. 2011.

[22] http://www.topology-zoo.org/dataset.html

CHAPTER 10

SUMMARY, OUTLOOK, AND FUTURE RESEARCH

Part I discussed the fundamental role of linked hierarchical data structures in providing the mathematical tools needed to construct rigorous versatile routing schemes and applied hierarchical routing tools to the process of constructing oblivious routing schemes.

Part II applied the routing tools generated in Part I to address real-world network optimization problems in the area of electrical power networks, clusters of microgrids, and content-centric networks. Chapter 4 described how an oblivious routing algorithm can be utilized in order to construct an overlay network routing scheme optimizing power generation cost in power networks. Also, Chapter 5 explained how an oblivious routing algorithm can be utilized in order to construct an overlay network routing scheme descreasing the cost of power routing in clusters of microgrids. Finally, Chapter 6 showed how an oblivious routing scheme can minimize the data congestion in data communication content-centric networks.

Part III of this dissertation utilized an advanced interdisciplinary approach to address existing security and privacy issues, proposing legitimate countermeasures for each of them from the standpoint of both computing and electrical engineering. The proposed methods were theoretically proven by mathematical tools and illustrated by real-world examples.

There are more open research questions regarding the application of oblivious network design in real-world network optimization problems. For example, can an oblivious routing algorithm be utilized in order to construct an overlay network routing scheme reducing the transportation cost in ground transportation networks of future smart cities in which driverless cars are predicted to become very popular?

Or, is there any oblivious solution for optimizing the flow costs in water, gas, and oil networks? We leave these problems for future research work.

PART IV

APPENDIX

APPENDIX A

PRELIMINARY DEFINITIONS REGARDING OBLIVIOUS NETWORK DESIGN

In contrast to the traditional MCFPs that are defined with a well-defined set of commodities (with specific size and source/destination nodes) and given flow cost function for every edge, oblivious network routing aims to solve an MCFP in which either the flow cost function is not specified (flow cost is oblivious), or the commodities size, source, and destination are not specified (commodities are oblivious).

Oblivious network routing approach solves oblivious MCFPs by employing a practical method not guaranteed to be perfect, but sufficient for the immediate goal which is obtaining an acceptable approximation of the optimal solution. In this regard, oblivious network routing is considered to be a heuristic method since it can be used to speed up the process of finding a satisfactory solution (while computing the optimal solution is impractical).

Oblivious network routing is different with Dynamic (adaptive) Routing (DR) since DR proposes a different solution in response to any change of the MCFP oblivious components; while, oblivious routing deploys a single routing scheme for an oblivious MCFP with the aim of approximating the optimal solution of the problem with oblivious parameters.

The common characteristics of the oblivious routing schemes includes being pretty flexible to the obliviousness of the environment in which the commodities are flowing and making the traffic flows distributed over the network and preventing the flow-congestion in some specific nodes or edges. Additionally, these types of routing schemes provide a low-cost flow routing in long term even if a wide range of unpredictable events occur in the network like bursty flow derived from a specific node or failure of some node in forwarding the flow through the network. In fact, the

versatile routing schemes best fit to those networks that we have little/no knowledge regarding their current and future states.

In Appendix A, we consider "network" as a *tool for flowing something from one place to another via the network connections in a step by step manner*. Each network flow will have the following properties: source point (initial place), target point (final place), flow path (the sequence of network connections that connect the end points), flow amount (the magnitude of whats flowing), and flow cost (the incurred cost of flowing). Additionally, we will focus on mathematically formulating the problem of routing flows through a network and the problems associated with cost minimization. To do this we will develop a general framework to precisely define flow routing cost in a network. Moreover, a formal definition of obliviousness[1] in the context of network flows will also be presented.

A.1 Single-Source Network Routing Problem

In this section, the single-source routing problem in an undirected graph, which is a common representation of some networks, will be defined and illustrated through a series of examples. In addition, the flow cost of its solution will be discussed in detail. We will also address an extended version of this problem with multi-commodity flow.

A.1.1 Preliminary Definitions

Before starting our discussion on the network routing problem, we need to examine a few preliminary definitions.

Undirected graph $\boldsymbol{G}$ is defined as the ordered pair $\boldsymbol{(V, E)}$ such that $\boldsymbol{V}$ and $\boldsymbol{E}$ are the set of vertices and edges in $\boldsymbol{G}$ respectively. Moreover, for every edge $\boldsymbol{e}$ in

[1]The state of not being certain about what is happening.

$\boldsymbol{E}$, $\boldsymbol{e} = \{\boldsymbol{u}, \boldsymbol{v}\}$ where $\boldsymbol{u}$ and $\boldsymbol{v}$ are two *distinct* vertices which are members of $\boldsymbol{V}$. Edge $\boldsymbol{e}$ is called the connecting edge of vertices $\boldsymbol{u}$ and $\boldsymbol{v}$ if $\boldsymbol{e} = \{\boldsymbol{u}, \boldsymbol{v}\}$.

For some undirected graph, a *simple path* is inductively defined in the following form:

Definition A.1.1. *In the undirected graph* $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$*, set* $\boldsymbol{p} \subseteq \boldsymbol{E}$ *is called a (simple) path from* $\boldsymbol{s} \in \boldsymbol{V}$ *to* $\boldsymbol{t} \in \boldsymbol{V}$ *if:*

$$\boldsymbol{p} = \begin{cases} \emptyset & \boldsymbol{s} = \boldsymbol{t} \\ \{\{\boldsymbol{s}, \boldsymbol{v}\}\} \cup \boldsymbol{p}' & \textit{otherwise} \end{cases}$$

where $\{\boldsymbol{s}, \boldsymbol{v}\} \in \boldsymbol{E}$ *and* $\boldsymbol{p}'$ *is a path from* $\boldsymbol{v}$ *to* $\boldsymbol{t}$ *such that:*

$$\forall \boldsymbol{e} \in \boldsymbol{p}' : \boldsymbol{s} \notin \boldsymbol{e} \tag{A.1}$$

Undirected graph $\boldsymbol{G}$ *is connected if for every pair of vertices* $\boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{V}$*, there is a path from* $\boldsymbol{u}$ *to* $\boldsymbol{v}$*.*

In the above definition, if we ignore the constraint mentioned in Equation A.1, the obtained set $\boldsymbol{p}$ is called a *walk* of *no repetitive edge*. Additionally, if $\boldsymbol{p}$ is a path from $\boldsymbol{s}$ to $\boldsymbol{t}$, it is also a path from $\boldsymbol{t}$ to $\boldsymbol{s}$; subsequently, path $\boldsymbol{p}$ is said to be *between* $\boldsymbol{s}$ *and* $\boldsymbol{t}$. Furthermore, the path between a pair of vertices in $\boldsymbol{G}$ is not necessarily unique.

Now, consider the problem of finding paths from a given *source* vertex to a number of given *target* vertices in a connected undirected graph. There are many real-world examples that can be represented by such a problem. Here is a simplified example in the context of ground transportation.

Ground Transportation Example

Assume that a company produces its products in a single factory and sells them in a number of retail outlets. The factory is located in the city $\boldsymbol{s}$, and there are a number

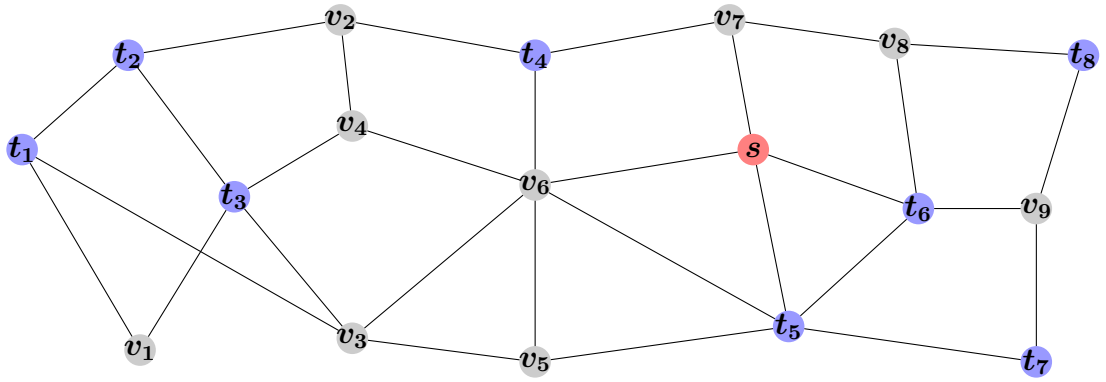


Figure A.1: The schematic view of the transportation network. The red circle s represents the city in which the factory located. The blue circles $t_1, t_2, \ldots, t_8$ are symbols of 8 cities associated with 8 retail outlets. The other circles represent the other cities which may participate in the paths from the factory to the outlets. The lines between circles show the available roads between cities.

of outlets scattered over different cities (there is at-most one outlet per city). After making the products in the factory, they have to be distributed among the outlets. To facilitate the distribution, the products are first packed into boxes of the *same size* in the factory. Then, the boxes are taken from the factory to the outlets using ground transportation. Figure A.1 schematically shows the ground transportation network through which the products are passed. This figure shows different cities including those in which the factory and outlets are located. It also specifies the available roads that can be used to take the products from one city to another.

Assume that each retail outlet needs a single box of a particular product to be taken from the factory. The goal is to satisfy the demand of each outlet by taking its needed box from the factory in city s to its designated target. This problem is a sample of the *single-source network routing problem* which is formally defined in the following form:

Definition A.1.2. *Consider connected undirected graph* $G = (V, E)$. *Let* $T = \{t_1, t_2, \ldots, t_k\}$ *denote the set of targets where* $t_1, t_2, \ldots, t_k$ *are* k *distinct target*

vertices which belong to $\boldsymbol{V}$. *Moreover, assume that* $\boldsymbol{s} \in \boldsymbol{V}$ *represents the source vertex. By definition, Single-Source Network Routing Problem is the problem of finding function* $\boldsymbol{p} : \boldsymbol{T} \mapsto \boldsymbol{2^E}$ *that maps target* $\boldsymbol{t_i}$ *to a simple path from* $\boldsymbol{s}$ *to* $\boldsymbol{t_i}$ *in graph* $\boldsymbol{G}$ *(*$\forall \boldsymbol{i} \in [\boldsymbol{1}, \boldsymbol{k}]$*). Function* $\boldsymbol{p}$ *and set* $\{\boldsymbol{p(t)} | \boldsymbol{t} \in \boldsymbol{T}\}$ *are respectively called the solution function and the (feasible) solution of the problem.*

Regarding Figure A.1, in the ground transportation example of the single-source network routing problem, $\boldsymbol{V}$ is the set of all cities, $\boldsymbol{E}$ is the set of roads between cities, and $\boldsymbol{T} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_8}\}$ is the set containing the cities in which the retail outlets are located; moreover, the factory is located in the source city $\boldsymbol{s}$. The solution is the set of paths $\{\boldsymbol{p(t_i)} | \boldsymbol{i} \in [\boldsymbol{1}, \boldsymbol{8}]\}$ such that $\boldsymbol{p}$ is the solution function. So, for every $\boldsymbol{i} \leq \boldsymbol{8}$, a box of product is routed through the path $\boldsymbol{p(t_i)}$ to reach the outlet $\boldsymbol{t_i}$. Figure A.2 represents one feasible solution by specifying all of the eight paths (note that the feasible solution of a single-source network routing problem is not unique). As you see in this figure, each road belongs to a number of solution paths. For example, the road $\{\boldsymbol{t_6}, \boldsymbol{v_9}\}$ belongs to the paths $\boldsymbol{p(t_7)}$ and $\boldsymbol{p(t_8)}$. On the other hand, no solution path uses the road $\{\boldsymbol{t_5}, \boldsymbol{t_7}\}$. The number of paths to which a road belongs is called its *traffic flow*.

Definition A.1.3. *Consider the single-source network routing problem of target set* $\boldsymbol{T}$ *in graph* $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$. *Moreover, assume* $\boldsymbol{p}$ *as a solution function of the problem. Function* $\boldsymbol{f_p} : \boldsymbol{E} \mapsto \mathbb{Z}_{\geq \boldsymbol{0}}$ *will be the flow function associated with* $\boldsymbol{p}$ *if this is the case that:*

$$\boldsymbol{f_p(e)} = |\{\boldsymbol{p(t)} | \boldsymbol{t} \in \boldsymbol{T} \wedge \boldsymbol{e} \in \boldsymbol{p(t)}\}| \qquad \forall \boldsymbol{e} \in \boldsymbol{E}$$

The value $\boldsymbol{f_p(e)}$ *is called the traffic flow of edge* $\boldsymbol{e}$ *in solution* $\boldsymbol{P} = \{\boldsymbol{p(t)} | \boldsymbol{t} \in \boldsymbol{T}\}$.

Regarding Definition A.1.3, the number of paths which use edge $\boldsymbol{e}$ in solution $\boldsymbol{P}$ is called the traffic flow of $\boldsymbol{e}$. For example, in the ground transportation problem,

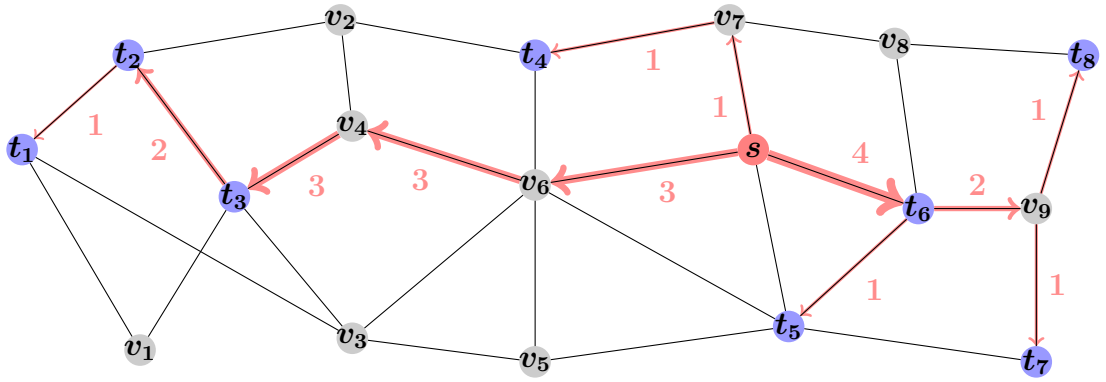


Figure A.2: A solution of the single-source network routing problem in the ground transportation example. The highlighted lines show the paths through which the boxes are routed from source to the designated targets. The number written on the each highlighted line shows its traffic flow.

since each path is used to take one box, the value $f_p(e)$ is the number of boxes which are routed through the edge e in solution P. In Figure A.2, each road has been labeled with its traffic flow computed by the flow function f_p where p is the solution function corresponding to the shown solution. Note that there is no flow in the edges with no label (their traffic flow value is zero).

A.1.2 Edge Routing Cost

Previously, the single-source network routing problem was defined, and the ground transportation problem was described to illustrate the problem and its solution. Later in this section, the routing cost in this type of problem will be introduced and addressed in detail.

Let's consider the ground transportation example again. As mentioned earlier, in this problem, there are a number of retail outlets, say k, such that each one needs a box of product supplied by the single factory. The goal is to take all of the boxes to their appropriate demanding outlets. To do this, the company uses a number of trucks. Each truck is used to carry at most t boxes from city x to its directly

connected city $\boldsymbol{y}$ using the road $\boldsymbol{\{x, y\}}$. Each box may be passed through different cities and carried by different trucks before reaching its demanding outlet. We will refer to the cost of using trucks to carry demands as the *routing cost*.

By modeling the problem as a single-source network routing problem, we have to specify $\boldsymbol{k}$ paths from the factory to each of the outlets. Assuming that $\boldsymbol{p}$ is a solution function of the problem, the total routing cost incurred by $\boldsymbol{p}$ can be obtained by first computing the cost incurred in each road, and then finding the total solution cost as an aggregation of all the computed costs.

First, our focus is on computing the cost incurred in each road. In this problem, moving the trucks containing some boxes of the factory products through the roads incurs some cost. Assuming the solution function $\boldsymbol{p}$, each road $\boldsymbol{e}$ belongs to $\boldsymbol{f_p(e)}$ paths in the solution (see Definition A.1.3). As a result, there are $\boldsymbol{f_p(e)}$ boxes that have to be passed through $\boldsymbol{e}$. Since each truck can carry at most $\boldsymbol{t}$ boxes, the number of trucks needed to take $\boldsymbol{f_p(e)}$ boxes through the road $\boldsymbol{e}$ is $\boldsymbol{\lceil f_p(e)/t \rceil}$.

Assume that passing a truck through each road incurs some cost directly proportional to the road length. Subsequently, passing trucks through different roads incurs different cost amounts. Let $\boldsymbol{c_e}$ denote the cost of passing one truck through road $\boldsymbol{e}$. Understanding that $\boldsymbol{\lceil f_p(e)/t \rceil}$ trucks are passing through $\boldsymbol{e}$, the total cost incurred in $\boldsymbol{e}$ will be $\boldsymbol{\lceil c_e \cdot f_p(e)/t \rceil}$. In addition, let $\boldsymbol{a}$ denote the cost of loading each box into a truck at the beginning of each road and unloading it from the truck at the end of it. As the result, the total cost of routing demand flows through road $\boldsymbol{e}$ can be written as a function of its traffic flow.

$$\boldsymbol{cost_e(x)} = \left\lceil \frac{\boldsymbol{x}}{\boldsymbol{t}} \right\rceil \cdot \boldsymbol{c_e} + \boldsymbol{a} \cdot \boldsymbol{x} \quad \text{where } \boldsymbol{x} = \boldsymbol{f_p(e)} \tag{A.2}$$

In this equation, function $\boldsymbol{cost_e}$ is called *cost function* of road $\boldsymbol{e}$ (different roads have different cost functions). Figure A.3 shows the scatter plot of the routing cost

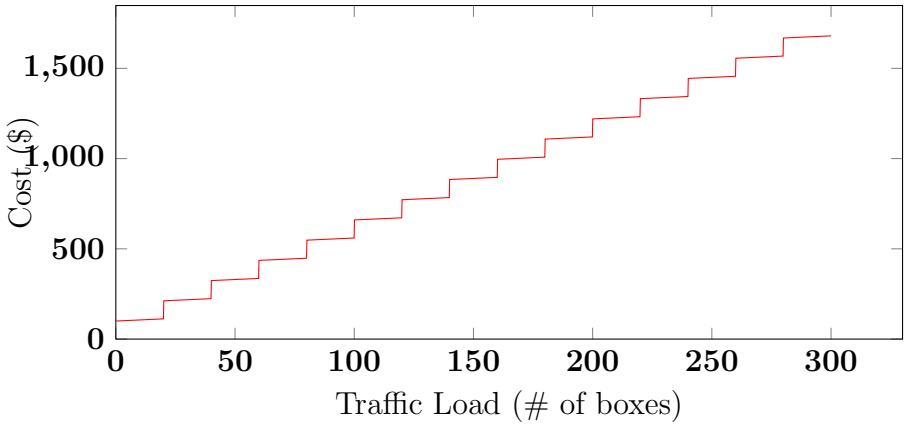


Figure A.3: The plot of routing cost function of road $\boldsymbol{e}$ in the ground transportation example for truck capacity of $\boldsymbol{t = 20}$, loading and unloading cost of $\boldsymbol{a = 0.6\$}$, and truck passing cost of $\boldsymbol{c_e = 100\$}$ through road $\boldsymbol{e}$.

incurred in road $\boldsymbol{e}$ versus its traffic flow for the specific configuration of our ground transportation example. In general, for single-source network routing problems, the routing cost incurred in each edge is defined as below:

Definition A.1.4. *Consider the single-source network routing problem of graph* $\boldsymbol{G = (V, E)}$. *Assuming that edge* $\boldsymbol{e}$ *has traffic flow of* $\boldsymbol{f}$ *in some feasible solution, the value* $\boldsymbol{cost_e(f)}$ *is defined as the routing cost of* $\boldsymbol{e}$ *incurred by the solution where function* $\boldsymbol{cost_e : \mathbb{Z}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}}$ *is called the edge routing cost function for every edge* $\boldsymbol{e \in E}$.

As you see in Definition A.1.4, the routing cost value of edge $\boldsymbol{e}$ is generally a function of both the edge and its traffic flow. In many real-world examples, the edge routing cost function is *multiplicatively separable*; i.e. it can be written as the product of two single input functions. For example, in the ground transportation problem, assuming that variable $\boldsymbol{a}$ in Equation A.2 is negligible (in comparison with $\frac{c_e}{t}$), $\boldsymbol{cost_e(x)}$ becomes the product of $\lceil \frac{x}{t} \rceil$ (which is a function of traffic flow) and $\boldsymbol{c_e}$ (that only depends on the road properties).

Definition A.1.5. *The edge routing cost function* $\boldsymbol{cost_e}$ *of the single-source network routing problem is said to be (multiplicatively) separable if:*

$$\boldsymbol{cost_e(x) = rrc(x) \cdot w_e} \tag{A.3}$$

In this equation, $\boldsymbol{w_e}$ *is called the edge weight of* $\boldsymbol{e}$ *for every* $\boldsymbol{e \in E}$*, and the function* $\boldsymbol{rrc : \mathbb{Z}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}}$ *is called the relative edge routing cost function or simply the relative routing cost function.*

In Equation A.3, function $\boldsymbol{rrc}$ determines how the routing cost of an edge is related to its traffic flow value. This relation is identical for every edges in the problems with separable edge routing cost function. The *absolute* routing cost of each edge in these problems is the weighted (scaled) value of its relative routing cost.

In the ground transportation problem with a separable edge routing cost function, the edge weight of each road is $\boldsymbol{c_e}$ which denotes the cost of passing a truck through that road. Moreover, the relative routing cost function in this problem is $\boldsymbol{\lceil x/t \rceil}$ which is a concave[2] function. Later in this chapter, we will see that $\boldsymbol{rrc}$ function is the critical parameter in routing cost optimization problems in such a way that its properties may significantly affect the complexity of problem solving algorithms. In spite of the ground transportation example, in the following example, we will see that the $\boldsymbol{rrc}$ function is convex[3].

Assume that there is a computer network that consists of a set of routers and links connecting each pair of routers together. Consider graph $\boldsymbol{G = (V, E)}$ as the model of the computer network where $\boldsymbol{V}$ represents the set of identical routers and

[2]The real-valued function $\boldsymbol{f}$ on the interval $\boldsymbol{[x, y] \in \mathbb{R}}$ is concave if and only if $\boldsymbol{\forall t \in [0, 1] : f(xt + y(1 - t)) \geq tf(x) + (1 - t)f(y)}$.

[3]The real-valued function $\boldsymbol{f}$ on the interval $\boldsymbol{[x, y] \in \mathbb{R}}$ is convex if and only if $\boldsymbol{\forall t \in [0, 1] : f(xt + y(1 - t)) \leq tf(x) + (1 - t)f(y)}$.

$\boldsymbol{E}$ models the set of identical links between them. Let $\boldsymbol{e}$ denote a link between the routers $\boldsymbol{a}$ and $\boldsymbol{b}$. Figure A.4 schematically shows the link and the routers and the data packet flow direction from $\boldsymbol{a}$ to $\boldsymbol{b}$. Although there is a two-way data packet flow in the link, our focus is on the shown direction. In this figure, the rate of exiting packets from the waiting queue of router $\boldsymbol{a}$ is assumed to be $\boldsymbol{\mu}$ packets per second which is referred as the link *transmission speed*. Moreover, the process of arriving packets at the waiting queue of router $\boldsymbol{a}$ is assumed to be a Poisson process[4] of intensity $\boldsymbol{\lambda}$ packets per second. Consequently, the average traffic flow of packets in the specific time interval $[\boldsymbol{t_0}, \boldsymbol{t_0} + \boldsymbol{n}]$ equals the product of intensity $\boldsymbol{\lambda}$ and the length of the interval ($\boldsymbol{n}$). As the result, considering $\bar{\boldsymbol{f}}_{[\boldsymbol{t},\boldsymbol{t}+\boldsymbol{n}]}$ as the average traffic flow in link $\boldsymbol{e}$, we obtain the following equation.

$$\boldsymbol{\lambda} = \frac{\bar{\boldsymbol{f}}_{[\boldsymbol{t},\boldsymbol{t}+\boldsymbol{n}]}}{\boldsymbol{n}} \tag{A.4}$$

In this example, the value of the edge routing cost function $\boldsymbol{cost_e}$ for traffic flow of $\boldsymbol{f}$ packets into edge $\boldsymbol{e}$ (which denotes the link shown in Figure A.4) equals $\boldsymbol{f}$ times the cost of routing one packet through $\boldsymbol{e}$. Let's assume that the cost of taking a packet from router $\boldsymbol{a}$ to router $\boldsymbol{b}$ is directly proportional to the length of time interval from the moment of entering the packet into the waiting queue of router $\boldsymbol{a}$ to the moment when it reaches the same point of router $\boldsymbol{b}$ (i.e. the entrance of waiting queue in router $\boldsymbol{b}$). This interval consists of three parts: waiting time in queue of router $\boldsymbol{a}$, transmission delay[5], and propagation delay[6]. Assuming that propagation delay

[4]Poisson process of intensity $\boldsymbol{\lambda}$ is a stochastic continuous-time process $\boldsymbol{N}$ that its value in the arbitrary time $\boldsymbol{t}$ is a stochastic, discrete, Poisson distributed variable such that: $\boldsymbol{\forall t > 0 : N_t \sim Pois(\lambda t)}$.

[5]The amount of time it takes for a router to transmit all the bits of a packet over a link connected to it.

[6]The amount of time it takes for all the bits of a packet to move from one end of a link to the other.

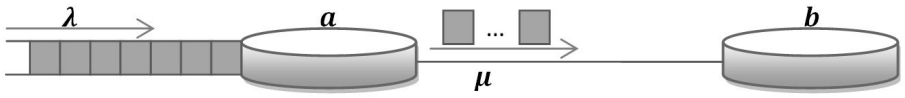


Figure A.4: The schematic view of a link and two routers. Data packets enter router a with the average rate λ and exits with the average rate μ.

is negligible compared with the other two delays, the cost of routing each packet through edge e and also the function $cost_e$ can be written in the following form:

$$\text{cost of routing each packet through } e = c \cdot (\text{transmission delay} + \text{queuing delay})$$

or

$$cost_e(f) = f \cdot c \cdot (\text{transmission delay} + \text{queuing delay}) \tag{A.5}$$

where c is the coefficient of the proportionality relation between the routing cost incurred and the routing time spent in the process of routing packets through a link.

To compute the queuing delay in router a, we use a preliminary theorem of queue theory. Since packets enter the queue of router a in a Poisson process of intensity λ packets per seconds, the waiting time of packets in such a queue follows the exponential distribution of rate $(\mu - \lambda)$ packets per second. Subsequently, the average queue waiting time for each packet is $1/(\mu - \lambda)$. Moreover, since the link transmission speed is μ, the average transmission delay for each packet is $1/\mu$. Using these computations, Equation A.5 can be rewritten in the following form if we use the average delays.

$$cost_e(f) = f \cdot c \cdot (\frac{1}{\mu} + \frac{1}{\mu - \lambda}) \tag{A.6}$$

Replacing λ in Equation A.6 with its equivalent expression (mentioned in Equation A.4) produces the following formula (again, we use the average value of traffic flow and f interchangeably).

$$cost_e(f) = f \cdot c \cdot (\frac{1}{\mu} + \frac{n}{n\mu - f}) \tag{A.7}$$

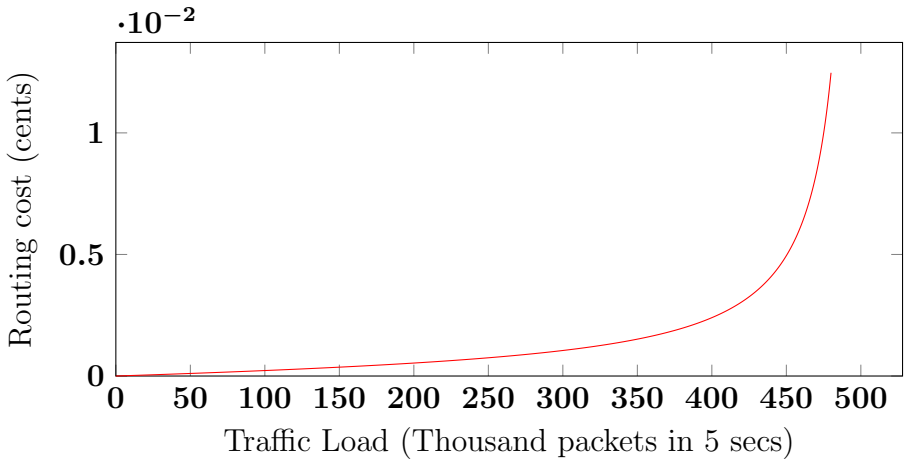


Figure A.5: The plot of the relative routing cost function of the computer network problem for $\boldsymbol{n = 5}$ sec, $\boldsymbol{c = 10^{-6}\$}$, and $\boldsymbol{\mu = 10^{5}}$packet/sec.

Equation A.7 represents the average routing cost of edge $\boldsymbol{e}$ as the function of its average traffic flow $\boldsymbol{f}$ in an interval of $\boldsymbol{n}$ seconds. As you see in this equation, because of the assumption that all the routers and the links are identical, the edge routing cost function is the same for every link $\boldsymbol{e}$ in the computer network. In fact, parameters $\boldsymbol{c}$ and $\boldsymbol{\mu}$ are independent of $\boldsymbol{e}$. Consequently, it is safe to assume that the edge weight $\boldsymbol{w_e = 1}$ for all the links. This implies that $\boldsymbol{rrc \equiv cost_e}$ in this problem. Moreover, concerning Equation A.7, the $\boldsymbol{rrc}$ function of the computer network problem is convex (despite the ground transportation problem). Figure A.5 plots the $\boldsymbol{rrc}$ function of the computer network problem.

A.1.3 Network Routing Cost

Previously, edge routing cost was defined by two functions $\boldsymbol{cost_e}$ and $\boldsymbol{rrc}$ in the single-source network routing problem. In this subsection, we will formally define the *network routing cost* which specifies how costly a solution of some network routing problem is. In spite of the edge routing cost which is defined for every edge of the graph, the network routing cost computes the aggregated cost incurred in all

of the edges. In fact, function $\boldsymbol{cost_e}$ determines the relation between the routing cost incurred in $\boldsymbol{e}$ and its traffic flow based on the characteristics of $\boldsymbol{e}$ rather than considering the whole graph's topology. On the other hand, the network routing cost of a solution depends on the routing cost of all of edges in the graph. Here is the formal definition of the network routing cost:

Definition A.1.6. *Consider the single-source network routing problem of graph* $\boldsymbol{G = (V, E)}$ *and solution function* $\boldsymbol{p}$*. By definition, function* $\boldsymbol{nrc_\pi : (\mathbb{R}_{\geq 0})^{|E|} \mapsto \mathbb{R}_{\geq 0}}$ *is called the network routing cost function such that* $\boldsymbol{\pi : \{1, 2, \ldots, |E|\} \mapsto E}$ *denotes an arbitrary permutation*[7] *of* $\boldsymbol{E}$ *members. Assuming that the routing cost of edge* $\boldsymbol{e}$ *corresponding to solution function* $\boldsymbol{p}$ *is represented as* $\boldsymbol{C_p(e)}$ *(for every* $\boldsymbol{e \in E}$*), the associated network routing cost* $\boldsymbol{C_p}$ *is then defined by the following equation:*

$$\boldsymbol{C_p = nrc_\pi(C_p(\pi_1), C_p(\pi_2), \ldots, C_p(\pi_{|E|}))}$$

Consider the ground transportation example again. In this example, since we are concerned with the total transportation cost, we define network routing cost of solution function $\boldsymbol{p}$ as the summation of the routing costs in all the roads. As the result, concerning Definition A.1.6, $\boldsymbol{C_p}$ in this example is in the following form:

$$\boldsymbol{C_p = \sum_{i=1}^{|E|} C_p(\pi_i) = \sum_{e \in E} C_p(e)} \tag{A.8}$$

where $\boldsymbol{C_p(e)}$ is the routing cost of edge (road) $\boldsymbol{e}$ incurred by $\boldsymbol{p}$. Regarding Definition A.1.4, the routing cost of edge $\boldsymbol{e}$ is the output of function $\boldsymbol{cost_e}$ which takes the traffic flow of edge $\boldsymbol{e}$ as the input. Since the traffic flow of $\boldsymbol{e}$ is shown by $\boldsymbol{f_p(e)}$,

$$\boldsymbol{C_p(e) = cost_e(f_p(e)).}$$

[7]Function $\boldsymbol{\pi : \{1, 2, \ldots, n\} \mapsto A}$ is a permutation of $\boldsymbol{A}$ members if $\boldsymbol{|A| = n}$, and $\boldsymbol{\pi}$ is a one-to-one, onto function. The notation $\boldsymbol{\pi_i}$ is commonly used to represent the output of function $\boldsymbol{\pi}$ corresponding to number $\boldsymbol{i}$.

By replacing the equivalent expression of $\boldsymbol{C_p(e)}$ in Equation A.8, we arrive at the following equation:

$$\boldsymbol{C_p = \sum_{e \in E} cost_e(f_p(e))}$$

Moreover, in the computer network example described previously, the network routing cost is assumed to be the total cost incurred in all of the routers and their connecting links. Since the incurred cost in every router and its connected link was previously defined as the edge routing cost, we define the network routing cost as the summation of all the edge routing costs. Subsequently, Equation A.8 also works for the computer network example. In fact, in both examples, function $\boldsymbol{nrc_\pi}$ outputs the simple addition of its input arguments:

$$\boldsymbol{nrc_\pi(C_p(\pi_1), C_p(\pi_2), \ldots, C_p(\pi_{|E|})) = \sum_{i=1}^{|E|} C_p(\pi_i)} \tag{A.9}$$

In general, the network routing cost function presented in Equation A.9 can be used in many examples of network routing problem. Another common $\boldsymbol{nrc_\pi}$ function is the **max** function that is used when the concern is about the *flow congestion* in network edges (connections).

$$\boldsymbol{C_p = nrc_\pi(C_p(\pi_1), C_p(\pi_2), \ldots, C_p(\pi_{|E|})) = \max_{i=1}^{|E|} C_p(\pi_i)}$$

A.2 General Network Routing Problem

Earlier in this chapter, we addressed the single-source network routing problem and the cost of its solutions. In addition, some examples were made to illustrate the discussion. This section is about a more general version of network routing problems. Despite the aforementioned problem, every graph vertex may send flow through the network in the general form. Moreover, each flow has some value that

affects the traffic flow amount of edges through which the flow is passing. Here, we present some basic definitions.

Definition A.2.1. *Let* $\boldsymbol{G = (V, E)}$ *denote an undirected graph.*

- *The triple* $\boldsymbol{(s, t, d) \in (V \times V \times \mathbb{R}^{+})}$ *is called a commodity in* $\boldsymbol{G}$*. The vertices* $\boldsymbol{s}$ *and* $\boldsymbol{t}$ *are the source and target of the commodity, respectively. Moreover,* $\boldsymbol{d}$ *is called the commodity value.*
- *Consider* $\boldsymbol{\bar{K} = (K_1, K_2, \ldots, K_k)}$ *as a sequence of commodities in graph* $\boldsymbol{G}$ *such that for every* $\boldsymbol{i, j \in [1, k]}$*, if* $\boldsymbol{K_i = (s, t, d)}$ *and* $\boldsymbol{K_j = (s', t', d')}$*, this is the case that:*

$$\boldsymbol{(s = s' \wedge t = t') \rightarrow i = j}$$

By definition, the general network routing problem or simply general routing problem is the problem of finding a sequence of paths like $\boldsymbol{\bar{p} = (p_1, p_2, \ldots, p_k)}$ *such that for every* $\boldsymbol{i \in [1, k]}$*,* $\boldsymbol{p_i}$ *represents a simple path from* $\boldsymbol{\Pi_1(K_i)}$[8] *to* $\boldsymbol{\Pi_2(K_i)}$ *in graph* $\boldsymbol{G}$*. Sequence* $\boldsymbol{\bar{p}}$ *is called an integral solution of the problem.*

Similar to the single-source network routing problem, here we define the traffic flow of edges in the general version.

Definition A.2.2. *Assuming that* $\boldsymbol{\bar{p}}$ *is an integral solution for the general routing problem of graph* $\boldsymbol{G = (V, E)}$*, the function* $\boldsymbol{f_{\bar{p}}^{(i)} : E \mapsto \mathbb{R}_{\geq 0}}$ *is the flow function of* $\boldsymbol{i^{th}}$ *commodity in solution* $\boldsymbol{\bar{p}}$ *if:*

$$\boldsymbol{f_{\bar{p}}^{(i)}(e) = \begin{cases} d_i & \text{if } e \in p_i \\ 0 & \text{otherwise} \end{cases} \qquad \forall e \in E, \forall i \in [1, k]} \tag{A.10}$$

The value of $\boldsymbol{f_{\bar{p}}^{(i)}(e)}$ *is called the traffic flow of commodity* $\boldsymbol{i}$ *through edge* $\boldsymbol{e}$*.*

[8] Assuming $\boldsymbol{x = (x_1, x_2, \ldots, x_n)}$ as an ordered $\boldsymbol{n}$-tuple, for every $\boldsymbol{k \in [1, n]}$, if projection operation $\boldsymbol{\Pi_k}$ gets $\boldsymbol{x}$ as the input, the output will be $\boldsymbol{\Pi_k(x) = x_k}$

As you see in Definition A.2.2, if there are $\boldsymbol{k}$ commodities, we define $\boldsymbol{k}$ flow functions for every solution $\bar{\boldsymbol{p}}$: $\boldsymbol{f}_{\bar{\boldsymbol{p}}}^{(1)}, \boldsymbol{f}_{\bar{\boldsymbol{p}}}^{(2)}, \ldots, \boldsymbol{f}_{\bar{\boldsymbol{p}}}^{(k)}$; however, in the single-source version, we only defined one flow function for every solution.

To define the solution cost, similar to the single-source version, we need to first define the routing cost incurred in every edge, and then compute the network routing cost based on the routing costs of all the edges.

Definition A.2.3. *Consider the general routing problem of graph* $\boldsymbol{G = (V, E)}$ *and solution* $\bar{\boldsymbol{p}}$*. Assuming that there are* $\boldsymbol{k}$ *commodities in the problem, function* $\boldsymbol{cost_e : (\mathbb{R}_{\geq 0})^k \mapsto \mathbb{R}_{\geq 0}}$ *is called the edge routing cost function for every edge* $\boldsymbol{e \in E}$*. Moreover, if* $\boldsymbol{f_i}$ *denotes the traffic flow of commodity* $\boldsymbol{i}$ *through edge* $\boldsymbol{e}$ *in* $\bar{\boldsymbol{p}}$ *(for every* $\boldsymbol{i \in [1, k]}$*), the routing cost of* $\boldsymbol{e}$ *incurred by solution* $\bar{\boldsymbol{p}}$ *is then defined as* $\boldsymbol{cost_e(f_1, f_2, \ldots, f_k)}$*.*

Note that in the above definition, the traffic flow $\boldsymbol{f_i}$ is obtained by the following equation:

$$\boldsymbol{f_i = f_{\bar{p}}^{(i)}(e)} \qquad \boldsymbol{\forall i \in [1, k]}$$

The network routing cost function $\boldsymbol{nrc_\pi}$ for the general routing problem is defined similar to what was mentioned in Definition A.1.6 for the single-source version of the problem.

As an example of the general routing problem, consider a computer network which connects some hosts together. To provide this connection, it uses some routers and links. Each link connects either a host to a router or a pair of routers to each other. Figure A.6 shows the graph representation of this network. As you see, the routers and hosts are represented as the graph vertices, and the links between them are modeled as the graph edges.

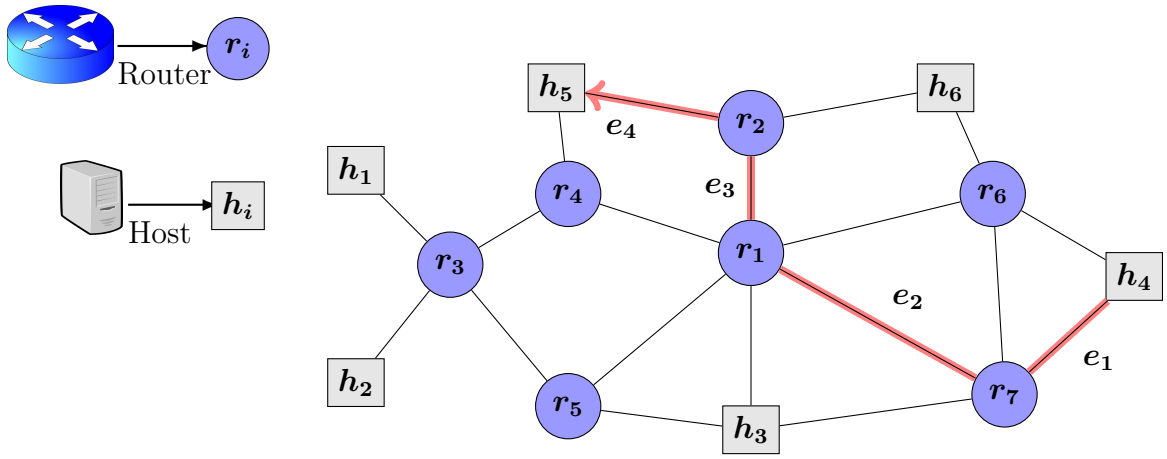


Figure A.6: Computer network example: In this figure, a path of data flow in a connection from host h_4 to host h_5 has been highlighted. The path is $\{e_1, e_2, e_3, e_4\}$.

Assume that hosts are communicating with each other using k different protocols $r_1, r_2, \ldots r_k$. Also, assume that k connections $c_1, c_2, \ldots, c_k$ have already been established between hosts such that connection c_i is between hosts s_i and t_i with protocol r_i for every $i \in [1, k]$. Hosts s_i and t_i are respectively called source host and target host of connection c_i. Note that each host can simultaneously be the source or target of more than one connection. Moreover, statistical study shows that protocol r_i sends d_i bits (on average) from the connection source to its target. For simplicity and without loss of generality, assume that hosts have one-way communications (our future formulating is extendable to the two-way communications). In addition, during a connection session, data is sent through a simple path specified at the connection start-up. The problem of routing the data flow of the connection sessions can be represented as a general routing problem.

To formulate this problem, consider graph $G = (V, E)$ as the problem graph such that V is the set of cyber nodes (i.e. routers and hosts) and E denotes the set of all the links connecting the nodes. Moreover, assume the sequence of commodities $\bar{K} = (K_1, K_2, \ldots K_k)$ such that K_i equals (s_i, t_i, d_i) for every $i \in [1, k]$ (s_i, t_i, and d_i are characteristics of the i^{th} connection c_i). Assume that

solution $\bar{p} = (p_1, p_2, \ldots, p_k)$ specifies k paths in the network such that p_i is the set of links connecting the pair of hosts s_i and t_i. Since each link of the network may participate in more than one connection, we present data flow through link e using k values $f_{\bar{p}}^{(1)}(e), f_{\bar{p}}^{(2)}(e), \ldots, f_{\bar{p}}^{(k)}(e)$ where $f_{\bar{p}}^{(i)}(e)$ is the average number of bits sent by connection c_i (which uses protocol r_i) through link e in one second.

Sending data through the network incurs some cost. Assume that there are two costly events during the data sending process:

(i) The incoming data to a router waits in its waiting queue.

(ii) The router transmits data through the appropriate link connected to it.

Note that for simplicity, we ignore the cost of propagating data through the links and processing data in routers. Assuming that link e has transmission speed of μ_e bits per second and the process of arriving bits at each router is a Poisson Process, the cost of data transmission and queue waiting delay of each router incurred in a one second length period is obtained by an equation similar to Equation A.7. We just need to replace μ by μ_e, and n by 1 in Equation A.7 to obtain the cost incurred in link e (note that in Equation A.7, the measurement unit of data amount was data packet; however, in this example, it is bit):

$$\text{The cost incurred in edge } e = f \cdot c \cdot \left(\frac{1}{\mu_e} + \frac{1}{\mu_e - f}\right) \tag{A.11}$$

In Equation A.11, c denotes the proportionality coefficient of the linear relation which is assumed to exist between the routing time spent and the routing cost incurred in link e. In addition, in this equation, f denotes the average of total number of bits sent by all the connections through link e in a one second length period of time; i.e.:

$$f = \sum_{i=1}^{k} f_{\bar{p}}^{(i)}(e) \tag{A.12}$$

Now, let's take the cost of processing data into consideration. Assume that it costs a_i units for any router to process one data bit of protocol r_i. Subsequently, processing $f_{\bar{p}}^{(i)}(e)$ bits of i^{th} connection data incurs the amount of $a_i f_{\bar{p}}^{(i)}(e)$ units for the router which is transmitting data into link e. As the result, regarding Equations A.11 and A.12, the edge routing cost function of the problem can be written in the following form:

$$cost_e(f_1, f_2, \ldots, f_k) = \sum_{i=1}^{k}(a_i f_i) + (c\sum_{i=1}^{k} f_i) \cdot (\frac{1}{\mu_e} + \frac{1}{\mu_e - \sum_{i=1}^{k} f_i}) \quad \text{(A.13)}$$

Note that Equation A.13 is only true for those links that are not connected to the source hosts of connections: $s_1, s_2, \ldots, s_k$. For the link connected to source host s_i ($i \in [1, k]$), since there is no router at the entrance end of the link, the queue waiting cost and the router processing cost is zero, and the only incurred cost is the transmission cost which equals (transmission speed in link e is μ_e):

$$cost_e(f_1, f_2, \ldots, f_k) = \frac{c\sum_{i=1}^{k} f_i}{\mu_e} \quad \text{(A.14)}$$

As an example, consider the network represented in Figure A.6. The function $cost_{e_1}$ is defined by Equation A.14; however, the routing costs of links e_2, e_3, and e_4 are obtained by Equation A.13.

The last step in modeling our example by the general routing problem is to define the network routing cost function nrc_π. If we are concerned with the data congestion in the connecting links, we have to define nrc_π as the **max** function. However, if we are concerned with the total routing cost, nrc_π must be the summation of all of the edge routing costs.

Routing Cost Optimization

In a general routing problem of graph $\boldsymbol{G = (V, E)}$, assume that:

- $\boldsymbol{g_e(f)}$ is the edge routing cost function for every edge $\boldsymbol{e \in E}$ and flow $\boldsymbol{f \geq 0}$;
- $\boldsymbol{h_\pi(x_1, x_2, \ldots, x_{|E|})}$ denotes the network routing cost function for $\boldsymbol{x_i \geq 0}$ and $\boldsymbol{i \in [1, |E|]}$;
- and $\boldsymbol{\bar{K} = (K_1, K_2, \cdots, K_k)}$ is the sequence of commodities.

By definition, the triple $\boldsymbol{\mathcal{E} = (g_e, h_\pi, \bar{K})}$ is called the *routing cost environment* of the problem. The routing cost environment specifies all the commodities in a general routing problem and also provides a tool for evaluating the routing costs of its solution. Here, we formulate the *routing cost optimization problem* using the mentioned routing cost environment.

Definition A.2.4. *Consider the general routing problem of graph* $\boldsymbol{G = (V, E)}$ *and routing cost environment* $\boldsymbol{\mathcal{E} = (g_e, h_\pi, \bar{K})}$. *The problem of finding an integral solution which minimizes the network routing cost in environment* $\boldsymbol{\mathcal{E}}$ *is called the routing cost optimization problem. The value* $\boldsymbol{C^*_{\mathcal{E}}}$ *which is called the minimum routing cost in environment* $\boldsymbol{\mathcal{E}}$ *is defined below.*

$$\boldsymbol{C^*_{\mathcal{E}} = \min_{\bar{p} \in \mathcal{I}_{\bar{K}}} h_\pi(C_{\bar{p}}(\pi_1), C_{\bar{p}}(\pi_2), \cdots, C_{\bar{p}}(\pi_{|E|}))}$$

In this equation, $\boldsymbol{\mathcal{I}_{\bar{K}}}$ *denotes the set of all the integral solutions of the problem with sequence* $\boldsymbol{\bar{K}}$ *of commodities; furthermore, function* $\boldsymbol{C_{\bar{p}}(e)}$ *outputs the routing cost of edge* $\boldsymbol{e}$ *in solution* $\boldsymbol{\bar{p}}$ *(*$\boldsymbol{f^{(i)}_{\bar{p}}}$ *denotes the flow function of* $\boldsymbol{i^{th}}$ *commodity in solution* $\boldsymbol{\bar{p}}$*):*

$$\boldsymbol{C_{\bar{p}}(e) = g_e(f^{(1)}_{\bar{p}}(e), f^{(2)}_{\bar{p}}(e), \cdots, f^{(k)}_{\bar{p}}(e)) \qquad \forall e \in E}$$

In Definition A.2.4, the integral solution(s) which has the minimum network routing cost in environment $\boldsymbol{\mathcal{E}}$ is called *the optimized integral solution* of the problem and denoted by $\bar{\boldsymbol{p}}^*_{\boldsymbol{\mathcal{E}}}$; hence:

$$\boldsymbol{C^*_{\mathcal{E}} = h_{\pi}(C_{\bar{p}^*_{\mathcal{E}}}(\pi_1), C_{\bar{p}^*_{\mathcal{E}}}(\pi_2), \cdots, C_{\bar{p}^*_{\mathcal{E}}}(\pi_{|E|}))}$$

A.3 Oblivious Routing Cost Environment

In the previous sections, we introduced two varieties of network routing problems. Here, we define another type of problem with an oblivious routing cost environment. To illustrate the obliviousness of the routing cost environment, we first make an example in the context of cyber networks, then the definition of *oblivious routing problem* will be presented formally.

Consider the Internet as a world-wide network. At any moment, large amount of data is being exchanged between computer devices scattered all around the world. This means that many data flows with different source and target hosts are passing through the Internet at any given moment. Moreover, the traffic flow amount in each point located in the Internet is permanently changing which eventuates in time-varying cost amount incurred by the data flows passing through it; for example, in a highly busy data connection, it takes relatively high amount of time for a data packet to be transferred through the connection. Subsequently, the Internet is a network which is being used in a wide range of routing cost environments; i.e. the network flows may have numerous sources, targets, and values, and also they incur varying amounts of cost while passing through the network over time. Now, consider the following constraint in the Internet:

It is a network through which the low-cost flow paths are too time-consuming to be computed online (upon request). This is basically because of the large size of the network and the necessity to route very quickly.

A practical way of routing flows in the Internet (and the networks with similar constraints) is to route the flows in an offline manner. In other words, as the flow routing process cannot be done in a *real-time* fashion, we have to *pre-compute* the flow paths. Additionally, as the routing cost environment in the Internet is varying from time to time, it is necessary to route the flows under *every possible* routing cost environment. This type of routing flows through a large-scale network is highly adaptable to the obliviousness of the routing cost environment and the huge problem size. Consequently, routing the flows in this way is said to be *versatile*.

Consider the computer network example again. Assume that each pair of hosts can unexpectedly initiate a connection with one of the protocols $\boldsymbol{r_1, r_2, \cdots, r_k}$ and start communicating with each other via the connection. In this network, the number of connections, and also the source, target, and protocol type of each connection is *non-deterministic*. In other words, concerning the aforementioned notations, the sequence of commodities $\boldsymbol{\bar{K}}$ in the associated general routing problem is oblivious. Moreover, the routing cost function of each connection ($\boldsymbol{cost_e}$) which depends on the number of connections of each protocol is also oblivious. For example, if we have $\boldsymbol{2k}$ connections $\boldsymbol{c_1, c_2, \cdots, c_{2k}}$ at any given moment, and we know that connections $\boldsymbol{c_{2i-1}}$ and $\boldsymbol{c_{2i}}$ use protocol $\boldsymbol{r_i}$ ($\forall \boldsymbol{i \leq k}$), this is the case that:

$$\boldsymbol{cost_e(f_1, f_2, \cdots, f_{2k}) = \sum_{i=1}^{2k} (a_{\lceil i/2 \rceil} f_i) + (c \sum_{i=1}^{2k} f_i)(\frac{1}{\mu_e} + \frac{1}{\mu_e - \sum_{i=1}^{2k} f_i})}$$

which is completely different from the cost function obtained in Equations A.13 and A.14. Consequently, we have to deal with a set of routing cost optimization problems with different edge routing cost functions for each.

Definition A.3.1. *Let $\boldsymbol{P_{G,\mathcal{E}}}$ denote a general routing problem of graph $\boldsymbol{G = (V, E)}$ in routing cost environment $\boldsymbol{\mathcal{E}}$. The following set of general routing problems is called an oblivious routing problem if $\boldsymbol{|\mathbb{E}| > 1}$.*

$$\boldsymbol{\{P_{G,\mathcal{E}} | \mathcal{E} \in \mathbb{E}\}}$$

Set $\mathbb{E}$ is called the set of possible routing cost environments of the oblivious routing problem.

As you see in Definition A.3.1, the total number of possible routing cost environments which equals $|\mathbb{E}|$ has to be more than one. In fact, if we only have one possible routing cost environment, the problem has no obliviousness and it turns into a general routing problem.

Definition A.3.2. *Consider an oblivious routing problem of graph $\boldsymbol{G = (V, E)}$ and the set of possible routing cost environments $\mathbb{E}$. An integral solution of the oblivious routing problem is defined as the function:*

$$\boldsymbol{S : \mathbb{E} \mapsto \bigcup_{\mathcal{E} \in \mathbb{E}} \mathcal{I}_{\bar{K}_{\mathcal{E}}}} \tag{A.15}$$

such that $\boldsymbol{S(\mathcal{E}) \in \mathcal{I}_{\bar{K}_{\mathcal{E}}}}$. In Equation A.15, for every $\boldsymbol{\mathcal{E}} \in \mathbb{E}$, $\boldsymbol{\bar{K}_{\mathcal{E}}}$ represents the third member of triple $\boldsymbol{\mathcal{E}}$ ($\boldsymbol{\Pi_3(\mathcal{E})}$), and also $\boldsymbol{\mathcal{I}_{\bar{K}_{\mathcal{E}}}}$ denotes the set containing all the integral solutions of the general routing problem with sequence $\boldsymbol{\bar{K}_{\mathcal{E}}}$ of commodities in graph $\boldsymbol{G}$.

In our recent definition, the integral solution $\boldsymbol{S(\mathcal{E})}$ is the solution suggested by function $\boldsymbol{S}$ for problem $\boldsymbol{P_{G,\mathcal{E}}}$. Additionally, note that instead of considering one path sequence as the solution of an oblivious routing problem, we have to specify a function that maps an integral solution to each of the possible routing cost environments. As there may be more than one solution for a general routing problem, the

solution function of the oblivious routing problem is not necessarily unique as well. To compute the integral solution of a given oblivious routing problem, we introduce two different approaches.

Dynamic Approach

In the dynamic approach, by considering one possible routing cost environment at a time, we compute the integral solution of the resulted general routing problem and do this process for all other possible routing cost environments in a repetitive manner. In other words, we take advantage of the fact that by restricting the routing cost environment to a special case, our oblivious routing problem turns into a general routing problem. Subsequently, the integral solution function obtained by the dynamic approach is in the following form:

$$\boldsymbol{S}_{dynamic}(\boldsymbol{\mathcal{E}}) = \bar{\boldsymbol{p}}_{\boldsymbol{\mathcal{E}}} \qquad \forall \boldsymbol{\mathcal{E}} \in \mathbb{E}$$

where $\bar{\boldsymbol{p}}_{\boldsymbol{\mathcal{E}}}$ is an integral solution of problem $\boldsymbol{P}_{\boldsymbol{G},\boldsymbol{\mathcal{E}}}$. Moreover, in the case that we are concerned with optimizing the routing cost of the oblivious routing problem, we use the following equation:

$$\boldsymbol{S}_{dynamic}(\boldsymbol{\mathcal{E}}) = \bar{\boldsymbol{p}}^{*}_{\boldsymbol{\mathcal{E}}} \qquad \forall \boldsymbol{\mathcal{E}} \in \mathbb{E}$$

where $\bar{\boldsymbol{p}}^{*}_{\boldsymbol{\mathcal{E}}}$ is the optimized integral solution of problem $\boldsymbol{P}_{\boldsymbol{G},\boldsymbol{\mathcal{E}}}$.

Although this approach results in the best answer and minimizes the network routing cost for any given set of possible routing cost environments, it is computationally impractical in most real-world cases.

Versatile Approach

On the other hand, the versatile approach is computationally much more efficient, but usually fails to minimize the network routing cost for every possible routing cost environment. This approach defines a *versatile routing scheme* to compute the integral solution function.

Definition A.3.3. *Consider an oblivious routing problem of graph* $\boldsymbol{G = (V, E)}$ *and set* $\mathbb{E}$ *of possible routing cost environments. The (integral) versatile routing scheme is defined as the following function:*

$$\mathbb{S} : \boldsymbol{V^2} \mapsto \text{ the set of all the paths in } \boldsymbol{G}$$

such that for every pair of vertices $\boldsymbol{u, v \in V}$, $\mathbb{S}(\boldsymbol{u, v})$ *specifies a simple path between* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *in* $\boldsymbol{G}$*. The solution function associated with the versatile routing scheme* $\mathbb{S}$ *is represented as* $\boldsymbol{S_{\mathbb{S}}}$*. Assuming* $\boldsymbol{(cost_e, nrc_\pi, \bar{K})}$ *as an arbitrary member of* $\mathbb{E}$*, this is the case that (*$\boldsymbol{\bar{K} = (K_1, K_2, \ldots K_k)}$*):*

$$\boldsymbol{S_{\mathbb{S}}(cost_e, nrc_\pi, \bar{K}) = (p_1, p_2, \ldots, p_k)}$$

such that $\boldsymbol{p_i = \mathbb{S}(\Pi_1(K_i), \Pi_2(K_i))}$ *for every* $\boldsymbol{i \in [1, k]}$.

Consequently, in the versatile approach, we use the paths specified by function $\mathbb{S}$ to route every commodity of the possible routing cost environments. In the third chapter, we will define and analyze different versatile routing schemes.

In order to evaluate the cost efficiency of solution $\boldsymbol{S_{\mathbb{S}}}$, we use the *competitive* approach which compares the incurred cost of solution $\boldsymbol{S_{\mathbb{S}}}$ with the cost incurred by the most efficient solution $\boldsymbol{S_{dynamic}}$ (which is obtained by the dynamic approach).

Consider Definition A.3.3 again. For every $\boldsymbol{\mathcal{E}} \in \mathbb{E}$, the competitive ratio of scheme $\mathbb{S}$ in the general routing problem $\boldsymbol{P_{G,\mathcal{E}}}$ is defined in the following form:

$$\boldsymbol{cr(\mathcal{E}, \mathbb{S}) = \frac{C_{\mathcal{E}}(S_{\mathbb{S}}(\mathcal{E}))}{C_{\mathcal{E}}(\bar{p}^*_{\mathcal{E}})}}$$

such that $\bar{p}^*_{\mathcal{E}}$ is the optimized integral solution of problem $P_{G,\mathcal{E}}$ and function $C_{\mathcal{E}}$ outputs the network routing cost of its input solution of problem $P_{G,\mathcal{E}}$. Moreover, the competitive ratio of scheme $\mathbb{S}$ in the oblivious routing problem of set $\mathbb{E}$ of the possible routing cost environments is represented by $CR(\mathbb{E}, \mathbb{S})$ and defined by the following equation:

$$CR(\mathbb{E}, \mathbb{S}) = \max_{\mathcal{E} \in \mathbb{E}} cr(\mathcal{E}, \mathbb{S})$$

As the result, the value of competitive ratio quantifies the cost efficiency of a versatile routing scheme in one or more routing cost environments.

A.4 Fractional vs. Integral Routing

In Section A.2, the general routing problem was formally defined. As you see in Definition A.2.1, in this problem, we have to specify a simple path for each commodity so that it flows through the path to reach its target. Since we had to route k commodities in this problem, sequence $\bar{p}$ of k simple paths was specified as the integral solution. Moreover, remember that for the integral solution $\bar{p}$, $f^{(i)}_{\bar{p}}(e)$ denotes the flow value of commodity $K_i = (s_i, t_i, d_i)$ through edge e and is defined in Equation A.10. As you see in this equation, $f^{(i)}_{\bar{p}}(e)$ can only get *two* values: zero and d_i. In the case that $f^{(i)}_{\bar{p}}(e) = 0$, as edge e doesn't belong to p_i, e has not been used to route the i^{th} commodity in solution $\bar{p}$. On the other hand, if $f^{(i)}_{\bar{p}}(e) = d_i$, since e is a member of p_i, the whole flow value is routed through e. Subsequently, our hidden assumption is that we don't break down a commodity into smaller parts before sending it through the network. Hence, in the *integral* solution of the general routing problem, the traffic flow of each commodity can only be routed through a *single* path.

On the other hand, there is another type of solution in which the flow of each commodity is not restricted to only one simple path. In other words, there may be multiple simple paths through which the commodity *fractions* flow from the source vertex to the target. As the result, the total value of each commodity ($\boldsymbol{d_i}$) will be carried from the source point to the target. This solution of the general routing problem is known as the *fractional solution*.

For instance, consider the computer network example in Section A.2 again. Assume that the network protocols allow the connection source to flow data toward the target through multiple paths rather than a predefined one. In this case, the fractional solution is also acceptable for the routing problem.

In this section, we will formally present the fractional flow, fractional solution of the general routing problem, and the fractional version of the routing cost optimization problem. The obliviousness in these problems will then be discussed.

Definition A.4.1. *Consider graph* $\boldsymbol{G = (V, E)}$ *such that* $\boldsymbol{V}$ *and* $\boldsymbol{E}$ *represent the vertex and edge set of* $\boldsymbol{G}$ *respectively. Fractional flow* $\boldsymbol{F}$ *of (total) value* $\boldsymbol{X}$*, source vertex* $\boldsymbol{u \in V}$*, and target vertex* $\boldsymbol{v \in V}$ *is defined in the following form:*

$$\boldsymbol{F = \{(p, x) | x \in (0, X] \wedge (p \text{ is a simple path in } G \text{ from } u \text{ to } v)\}}$$

such that:

i. $\boldsymbol{((p_1, x_1) \in F) \wedge ((p_2, x_2) \in F) \wedge (p_1 = p_2) \rightarrow (x_1 = x_2)}$

ii. $\boldsymbol{\sum_{(p,x) \in F} x = X}$

Henceforth, the fractional flow in graph $\boldsymbol{G}$ is a set of ordered pairs such that each one specifies a simple path ($\boldsymbol{p}$) in $\boldsymbol{G}$ and a fraction which specifies the weight of path $\boldsymbol{p}$. In Definition A.4.1, condition $\boldsymbol{i}$ forces each member of flow $\boldsymbol{F}$ to represent different paths. Additionally, the second one guarantees that the total flow value

($\boldsymbol{X}$) will be eventually carried by all the paths associated with $\boldsymbol{F}$ members. Now, we can extend the definition of the general routing problem to the problem with fractional solution.

Definition A.4.2. *Consider graph* $\boldsymbol{G = (V, E)}$ *and* $\boldsymbol{k}$ *distinct commodities* $\boldsymbol{\bar{K} = (K_1, K_2, \ldots, K_k)}$ *such that for every* $\boldsymbol{i \in [1, k] : K_i = (s_i, t_i, d_i)}$. *By definition, the fractional (general) routing problem is the problem of finding* $\boldsymbol{k}$ *fractional flows denoted by sequence* $\boldsymbol{\bar{F} = (F_1, F_2, \ldots, F_k)}$ *such that* $\boldsymbol{F_i}$ *is a fractional flow of value* $\boldsymbol{d_i}$ *from* $\boldsymbol{s_i}$ *to* $\boldsymbol{t_i}$ *in graph* $\boldsymbol{G}$. *Sequence* $\boldsymbol{\bar{F}}$ *is called a fractional solution of the problem.*

As recently defined, by applying the following constraint on the fractional solution, the problem will turn into the general routing problem with the integral solution.

$$\boldsymbol{|F_i| = 1} \qquad \boldsymbol{\forall i \in [1, k]}$$

Consequently, every fractional flow will only have one member in the form $\boldsymbol{(p_i, d_i)}$; i.e.

$$\boldsymbol{F_i = \{(p_i, d_i)\}} \qquad \boldsymbol{\forall i \in [1, k]} \tag{A.16}$$

Similar to the general routing problem, here we define the flow function of the fractional solution:

Definition A.4.3. *Let* $\boldsymbol{\bar{F}}$ *denote a solution of the fractional routing problem and* $\boldsymbol{G = (V, E)}$ *specifies the problem graph. Function* $\boldsymbol{f^{(i)}_{\bar{F}} : E \mapsto \mathbb{R}_{\geq 0}}$ *is the flow function of the* $\boldsymbol{i^{th}}$ *commodity in solution* $\boldsymbol{\bar{F}}$ *and is defined by the following equation:*

$$\boldsymbol{f^{(i)}_{\bar{F}}(e) = \sum_{(p,x) \in F_i} \sum_{e \in p} x} \qquad \boldsymbol{\forall e \in E} \ \textit{and} \ \boldsymbol{\forall i \in [1, k]} \tag{A.17}$$

such that $\boldsymbol{\bar{F} = (F_1, F_2, \ldots, F_k)}$. *The value of* $\boldsymbol{f^{(i)}_{\bar{F}}(e)}$ *is called the traffic flow of the* $\boldsymbol{i^{th}}$ *commodity through edge* $\boldsymbol{e}$ *in fractional solution* $\boldsymbol{\bar{F}}$.

Regarding Equation A.17, the flow function of the fractional solution can get the values between zero and $\boldsymbol{d_i}$. In other words, despite the integral solution, function $\boldsymbol{f^{(i)}_{\bar{F}}(e)}$ may output more than two values.

Now, consider the constrained version of the fractional routing problem which is equivalent to the one with the integral solution. In this special case, considering the form of each fractional flow shown in Equation A.16, we can simplify the flow function in the following form:

$$\begin{aligned} \boldsymbol{f^{(i)}_{\bar{F}}(e)} &= \sum_{(p,x)\in\{(p_i,d_i)\}} \sum_{e\in p} \boldsymbol{x} \\ &= \sum_{e\in p_i} \boldsymbol{d_i} = \begin{cases} \boldsymbol{d_i} & \boldsymbol{e \in p_i} \\ \boldsymbol{0} & \text{otherwise} \end{cases} \qquad \boldsymbol{\forall e \in E} \text{ and } \boldsymbol{\forall i \in [1,k]} \end{aligned}$$

where $\boldsymbol{d_i}$ is the total flow value of $\boldsymbol{F_i}$. As you see in this equation, $\boldsymbol{f^{(i)}_{\bar{F}}(e)}$ outputs the same value as flow function $\boldsymbol{f^{(i)}_{\bar{p}}(e)}$ does in the (integral) general routing problem.

Routing Cost of The Fractional Solution

For a fractional routing problem, function $\boldsymbol{cost_e}$ of the edge routing cost, function $\boldsymbol{nrc_\pi}$ of the network routing cost, and triple $\boldsymbol{\mathcal{E} = (cost_e, nrc_\pi, \bar{K})}$ of the routing cost environment are defined exactly the same as the general routing problem. Additionally, the aforementioned routing cost optimization problem can be extended to the problem of finding the optimized fractional solution which incurs the minimum network routing cost among all the possible fractional solutions of the problem; i.e.

$$\text{minimize}_{\bar{F}\in\mathcal{F}_{\bar{K}}}\{\boldsymbol{nrc_\pi}(\boldsymbol{C_{\bar{F}}(\pi_1), C_{\bar{F}}(\pi_2), \cdots, C_{\bar{F}}(\pi_{|E|}))}\}$$

such that $\boldsymbol{\mathcal{F}_{\bar{K}}}$ denotes the set of all the solutions of the fractional routing problem with commodities $\boldsymbol{\bar{K}}$ and also the routing cost incurred in every edge $\boldsymbol{e}$ of the network is obtained as $\boldsymbol{C_{\bar{F}}(e) = cost_e(f^{(1)}_{\bar{F}}(e), f^{(2)}_{\bar{F}}(e), \cdots, f^{(k)}_{\bar{F}}(e))}$. Assuming $\boldsymbol{\bar{F}^*_{\mathcal{E}}}$

as the optimized fractional solution, the minimum cost of routing the commodities is computed in the following form:

$$\boldsymbol{C^*_{\mathcal{E}} = nrc_{\pi}(C_{\bar{F}^*_{\mathcal{E}}}(\pi_1), C_{\bar{F}^*_{\mathcal{E}}}(\pi_2), \cdots, C_{\bar{F}^*_{\mathcal{E}}}(\pi_{|E|}))}$$

Obliviousness of The Routing Cost Environment

Now, we have prepared the tools needed to define the fractional routing problem in an oblivious environment. This problem is the set of fractional routing problems that each one has different routing cost environments (similar to what was mentioned in Definition A.3.1). The following definition presents the general form of the solution function of such routing problems that are defined in an oblivious routing cost environment.

Definition A.4.4. *Consider a fractional routing problem of graph* $\boldsymbol{G = (V, E)}$ *and set* $\mathbb{E}$ *of the possible routing cost environments. For this problem, a fractional solution function is defined in the following form:*

$$\boldsymbol{S : \mathbb{E} \mapsto \bigcup_{\mathcal{E} \in \mathbb{E}} \mathcal{F}_{\bar{K}_{\mathcal{E}}}}$$

such that:

- *for every environment* $\boldsymbol{\mathcal{E}}$ *belongs to* $\mathbb{E}$, $\boldsymbol{\bar{K}_{\mathcal{E}} = \Pi_3(\mathcal{E})}$;
- *set* $\boldsymbol{\mathcal{F}_{\bar{K}_{\mathcal{E}}}}$ *contains all the fractional solutions of the routing problem with commodities* $\boldsymbol{\bar{\mathcal{K}}}$ *in graph* $\boldsymbol{G}$;
- *and for every environment* $\boldsymbol{\mathcal{E}} \in \mathbb{E}$, *this is the case that:*

$$\boldsymbol{S(\mathcal{E}) \in \mathcal{F}_{\bar{\mathcal{K}}_{\mathcal{E}}}}$$

In the above definition, solution $\boldsymbol{S(\mathcal{E})}$ is called the solution suggested by function $\boldsymbol{S}$ for the fractional routing problem of graph $\boldsymbol{G}$ in the routing cost environment $\boldsymbol{\mathcal{E}} \in \mathbb{E}$.

Moreover, in order to find the optimized solution function using the dynamic approach, we have to compute the sequence $\boldsymbol{\bar{F}^*_{\mathcal{E}}}$ which minimizes the network routing cost in every possible routing cost environment $\boldsymbol{\mathcal{E}} \in \mathbb{E}$; i.e.

$$\boldsymbol{S_{dynamic}(\mathcal{E}) = \bar{F}^*_{\mathcal{E}}} \qquad \forall \boldsymbol{\mathcal{E}} \in \mathbb{E}$$

Subsequently, in the versatile approach of obtaining the solution function, we have to first define a *fractional versatile routing scheme* (like the integral version of the oblivious routing problem).

Definition A.4.5. *Consider a fractional routing problem of graph* $\boldsymbol{G = (V, E)}$ *and set* $\mathbb{E}$ *of the possible routing cost environments. The fractional versatile routing scheme is defined as function:*

$$\mathbb{S} : \boldsymbol{V^2} \mapsto \text{ the set of all the possible fractional flows in } \boldsymbol{G}$$

such that for every $\boldsymbol{u, v \in V}$, $\boldsymbol{\mathbb{S}(u, v)}$ *determines a fractional flow of unit value, source vertex* $\boldsymbol{u}$ *and target vertex* $\boldsymbol{v}$ *in graph* $\boldsymbol{G}$*. The solution function associated with* $\mathbb{S}$ *is represented by* $\boldsymbol{S_{\mathbb{S}}}$*. Assuming that* $\boldsymbol{(cost_e, nrc_\pi, \bar{K})}$ *is an arbitrary member of* $\mathbb{E}$ *and sequence* $\boldsymbol{\bar{K}}$ *equals* $\boldsymbol{(K_1, K_2, \ldots K_k)}$*, this is the case:*

$$\boldsymbol{S_{\mathbb{S}}(cost_e, nrc_\pi, \bar{K}) = (F_1, F_2, \ldots, F_k)}$$

such that:

$$\boldsymbol{F_i = \{(p, x \times \Pi_3(K_i))|(p, x) \in \mathbb{S}(\Pi_1(K_i), \Pi_2(K_i))\}} \qquad \boldsymbol{\forall i \in [1, k]}$$

Moreover, the competitive ratio of fractional scheme $\mathbb{S}$ is defined like the ratio of the integral one:

$$\boldsymbol{CR}(\mathbb{E}, \mathbb{S}) = \max_{\boldsymbol{\mathcal{E}} \in \mathbb{E}} \frac{\boldsymbol{C_{\mathcal{E}}(S_{\mathbb{S}}(\mathcal{E}))}}{\boldsymbol{C_{\mathcal{E}}(\bar{F}^*_{\mathcal{E}})}}$$

APPENDIX B

HIERARCHICAL DATA STRUCTURES PARTICIPATING IN OBLIVIOUS NETWORK DESIGN

Data structures are tools for storing and organizing data so that it can be used efficiently. There are a variety of data structures that are designed for different goals. The branch of data structures which use the "*Divide and Conquer*" technique to efficiently organize a large-scale bunch of data in a scalable way are known as the *hierarchical data structures*.

In the recent decades, during the world-wide process of globalization, the size of many real-world networks has been substantially increased. Subsequently, the graph representation of these networks has gradually become larger and more complicated. To deal with the routing problems defined in such large-size graphs, we need to have some well-designed data structures like hierarchical ones so that they prevent a huge growth in the cost of the routing algorithms.

Previously, different versions of the routing problem in a network were presented thoroughly. We used the graphs as a common type of linked data structures to represent an arbitrary network. Graph representation of a network made it easy for us to formulate the routing problems; however, we need more complex linked data structures to use in the solving algorithms of the formulated problems. These data structures are preferable to be highly flexible to the graph size as it may vary from tens to millions nodes. Moreover, in order to solve the problems in oblivious routing cost environments, it is important to design data structures that are versatile to the changing environment. Finally, we need efficient tools to reduce the cost and time of routing computations.

Our goal is to find the data structures which have the aforementioned criteria. More specifically, we will focus on the hierarchical data organizing to make our

design scalable. We define different levels of abstraction for the problem graph. The most abstract model of the graph represents the first (highest) level of abstraction; while the lower levels, model the graph in a more detailed way; and the lowest level is the graph itself.

The rest of Appendix A introduces two different data structures: "hierarchical decomposition tree" which organizes data in a *top-down* manner, and "hierarchical routing tree" which does it in a *bottom-up* fashion. Chapter two explains how these tools can be applied to make efficient routing algorithms. Let's define some preliminary terms: in definition A.1.1, the undirected graph was introduced. As an extension of that definition, here we define the *weighted* undirected graph.

Definition B.0.1. *Let* $\boldsymbol{G' = (V, E)}$ *denote an undirected graph. The triple* $\boldsymbol{G = (V, E, w)}$ *is defined as the weighted undirected graph corresponding to* $\boldsymbol{G'}$*, such that* $\boldsymbol{V}$*,* $\boldsymbol{E}$*, and* $\boldsymbol{w : E \mapsto \mathbb{R}^+}$ *are the vertex set, edge set, and the weight function of* $\boldsymbol{G}$*, respectively. Moreover, the (simple) path between two vertices in* $\boldsymbol{G}$ *is defined as the path between them in* $\boldsymbol{G'}$*. Also,* $\boldsymbol{G}$ *is connected if and only if* $\boldsymbol{G'}$ *is connected.*

Here we assume that any undirected graph $\boldsymbol{(V, E)}$ is equivalent to the weighted undirected graph $\boldsymbol{(V, E, unit)}$, where $\boldsymbol{\forall e \in E : unit(e) = 1}$. In other words, every definition or claim about the weighted undirected graph can be extended to the unweighted graph using this equivalence. Note, that in all of the following definitions, weighted undirected graphs and (unweighted) undirected graphs are simply referred to weighted graphs and (unweighted) graphs, respectively.

Considering $\boldsymbol{G = (V, E, w)}$ as a weighted graph, the length of simple path $\boldsymbol{p}$ in $\boldsymbol{G}$ is specified by function $\boldsymbol{len_G : 2^E \mapsto \mathbb{R}^+}$, and is defined in the following equation:

$$\boldsymbol{len_G(p) = \sum_{e \in p} w(e)}$$

Distance function $\boldsymbol{d_G : V^2 \mapsto \mathbb{R}^+}$ of weighted graph $\boldsymbol{G = (V, E, w)}$ is defined in the following form:

$$\boldsymbol{d_G(u, v) = \min_{p \in P(u,v)} len_G(p)} \tag{B.1}$$

where $\boldsymbol{P(u, v)}$ denotes the set of all the simple paths existing between vertices $\boldsymbol{u}$ and $\boldsymbol{v}$ in $\boldsymbol{G}$. Note that if weighted graph $\boldsymbol{G}$ is replaced with its unweighted equivalent, i.e. if we assume that the weight function of graph $\boldsymbol{G}$ is the $\boldsymbol{unit}$ function, the length of a path is obtained by the following equation:

$$\boldsymbol{len_G(p) = \sum_{e \in p} unit(e) = \sum_{e \in p} 1 = |p|}$$

Hence, the distance between any two vertices of a connected graph is the length of the shortest path existing between them. Additionally, if graph $\boldsymbol{G}$ is not connected, as set $\boldsymbol{P(u, v)}$ will be empty for some $\boldsymbol{u, v \in V}$, we need to redefine the distance function:

$$\boldsymbol{d_G(u, v)} = \begin{cases} \boldsymbol{\min_{p \in P(u,v)} len_G(p)} & \boldsymbol{P(u, v) \neq \emptyset} \\ \boldsymbol{+\infty} & \text{otherwise} \end{cases}$$

The maximum values of all the finite distances in a connected graph is called the *diameter of graph* $\boldsymbol{G}$ and is represented by $\boldsymbol{diam(G)}$:

$$\boldsymbol{diam(G)} = \begin{cases} \boldsymbol{\max_{u,v \in V} d_G(u, v)} & \boldsymbol{G} \text{ is connected} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Moreover, *an* $\boldsymbol{r}$*-neighborhood of vertex* $\boldsymbol{v}$ or *ball* $\boldsymbol{B_G(v, r)}$ in the weighted graph $\boldsymbol{G = (V, E, w)}$ is defined by the following set:

$$\boldsymbol{B_G(v, r) = \{u \in V | d_G(u, v) \leq r\}}$$

For any set $\boldsymbol{V' \subseteq V}$, the *subgraph* of graph $\boldsymbol{G = (V, E, w)}$ *induced by* $\boldsymbol{V'}$ is graph $\boldsymbol{G_{V'} = (V', E', w)}$ where:

$$\boldsymbol{E' = \{\{u, v\} | u, v \in V' \wedge \{u, v\} \in E\}}$$

Definition B.0.2. *For any* $\boldsymbol{\Delta \geq 0}$*, the* $\boldsymbol{\Delta}$*-partition of connected graph* $\boldsymbol{G = (V, E, w)}$ *is a partition*[1] *of vertex set* $\boldsymbol{V}$ *into a number of subsets (which are called as the clusters) such that:*

$$\forall \boldsymbol{C} \in \boldsymbol{\Delta}\text{-}\textit{partition}(\boldsymbol{G}) : \max_{\boldsymbol{u,v} \in \boldsymbol{C}} \boldsymbol{d_G(u, v)} \leq \boldsymbol{\Delta}$$

By setting the $\boldsymbol{\Delta}$ parameter of Definition B.0.2 to different values, we will get different partitions of the graph vertex set. Here are some properties of the $\boldsymbol{\Delta}$-partition of the weighted graph $\boldsymbol{G = (V, E, w)}$:

i. The $\boldsymbol{\Delta}$-*partition* of graph $\boldsymbol{G}$ is not unique.

ii. For every $\boldsymbol{\Delta \geq diam(G)}$, partition $\boldsymbol{\{V\}}$ specifies a $\boldsymbol{\Delta}$-partition of $\boldsymbol{G}$.

iii. Assuming that $\boldsymbol{\Delta_1 \leq \Delta_2}$, every $\boldsymbol{\Delta_1}$-partition of graph $\boldsymbol{G}$ is also a $\boldsymbol{\Delta_2}$-partition of $\boldsymbol{G}$.

iv. Assuming that $\boldsymbol{C}$ is a cluster of set $\boldsymbol{\Delta}$-partition$\boldsymbol{(G)}$, subgraph $\boldsymbol{G_C}$ is *not* necessarily connected.

Let's consider Definition B.0.2 again. Assuming the arbitrary $\boldsymbol{\Delta}$-partition $\boldsymbol{X_\Delta}$ of graph $\boldsymbol{G}$, by decreasing the value of $\boldsymbol{\Delta}$, the maximum possible distance between the vertices of a cluster in $\boldsymbol{X_\Delta}$ will also decrease. Subsequently, it seems that the cardinality of each cluster decreases as well, and subsequently, $\boldsymbol{X_\Delta}$ should have more members to cover all the graph vertices. However, this claim is not generally true. As a contradicting example, see Figure B.1 which represents a graph of diameter $\boldsymbol{2}$. In this graph, the $\boldsymbol{3}$-partition $\boldsymbol{X_3 = \{\{5\}, \{1, 2, 3, 4\}\}}$ divides the vertex set into more clusters than the $\boldsymbol{2}$-partition $\boldsymbol{X_2 = \{\{1, 2, 3, 4, 5\}\}}$.

[1]Partition of set $\boldsymbol{A}$ is a set of $\boldsymbol{A}$ subsets $\boldsymbol{\{A_i | i \in [1, k]\}}$ such that $\bigcup_{\boldsymbol{i=1}}^{\boldsymbol{k}} \boldsymbol{A_i} = \boldsymbol{A}$, $\boldsymbol{\forall i \in [1, k] : A_i \neq \emptyset}$, and $\boldsymbol{\forall i \neq j : A_i \cap A_j = \emptyset}$.

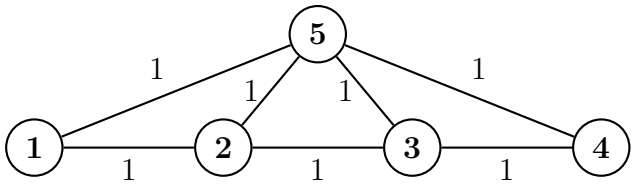


Figure B.1: A graph that has a **3**-partition of cardinality **2** and a **2**-partition of cardinality **1**.

As an example of the $\boldsymbol{\Delta}$-partitions, consider Figure B.2 which represents two different **4**-*partition*s of a weighted graph. In each partition, the way that clusters are scattered over the graph, and the subgraphs induced by them, has been specified.

B.1 Hierarchical Decomposition Tree

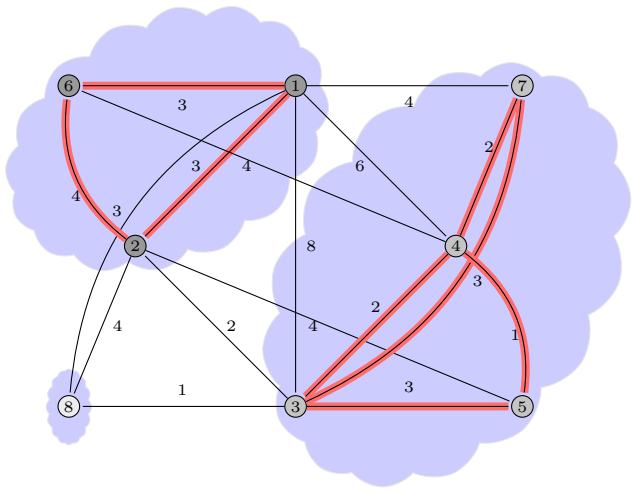
As mentioned before, we will present the *hierarchical decomposition tree* as an example of the hierarchical routing data structures which uses a top-down approach.

B.1.1 Hierarchical Decomposition Sequence

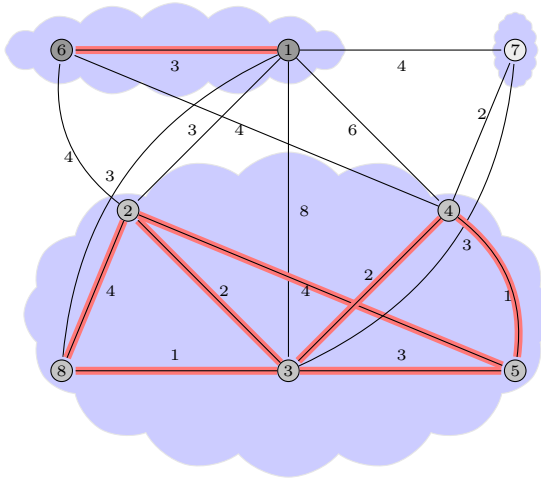
Let $\boldsymbol{G = (V, E, w)}$ denote a weighted connected graph. Hierarchical decomposition sequence (HDS) $\boldsymbol{\bar{H}}$ of graph $\boldsymbol{G}$ is defined as the sequence $\boldsymbol{\bar{H} = (H_0, H_1, \ldots, H_h)}$ where:

- $\boldsymbol{h = \lceil \log_2 diam(G) \rceil}$;
- $\boldsymbol{H_h = \{V\}}$;
- for every $\boldsymbol{i = 0, 1, \ldots, h-1}$, $\boldsymbol{H_i}$ is a $\boldsymbol{2^i}$-partition;
- and for every $\boldsymbol{i = 0, 1, \ldots, h-1}$, if $\boldsymbol{C}$ belongs to $\boldsymbol{H_i}$, then:

$$\boldsymbol{\exists C' \in H_{i+1} : C \subseteq C'}$$



(a) Vertex set partition: {{8},{1,2,6},{3,4,5,7}}



(b) Vertex set partition: {{7},{1,6},{2,3,4,5,8}}

Figure B.2: Two different **4**-partitions of a weighted graph

Partition $\boldsymbol{H_i}$ is called the $\boldsymbol{i^{th}}$ level partition of $\bar{\boldsymbol{H}}$.

Note that the HDS is only defined for connected graphs. In addition, there are $\lceil \mathbf{log_2}\, \boldsymbol{diam(G)} \rceil + \mathbf{1}$ partitions of set $\boldsymbol{V}$ in an HDS.

Each partition of a hierarchical decomposition sequence defines a level of abstraction in the graph. At the $\boldsymbol{h^{th}}$ level (highest level), the whole graph's vertex set is covered by only one cluster. At level $\boldsymbol{h-1}$, the distance high-threshold between the vertices of each cluster equals $\mathbf{2^{h-1}}$ (half of the threshold in the upper level).

Subsequently, in the lower-level partitions, the aforementioned high-threshold will exponentially decrease. This implies that there may exist more clusters in the lower levels of an HDS. In addition, as each cluster in the $(i+1)^{th}$ level partition is the union of a number of clusters in the i^{th} level partition (for every $i < h$), the number of clusters doesn't decrease in the lower levels. As the result, the partitioning of the vertex set gradually becomes *more refined* as we make out way to the lower levels.

In Figure B.3, you see one of the possible hierarchical decomposition sequences of the given graph. In general, the union of all the partitions of an HDS makes a *laminar family*[2] of the vertex set. An obvious HDS of any connected graph is found in the following form:

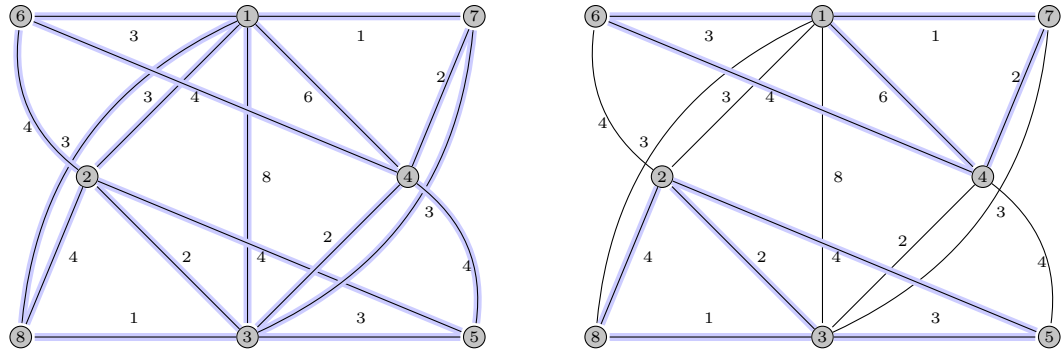
- $H_h = V$;
- and $\forall i = 0, 1, \ldots h-1 : H_i = \{\{v\} | v \in V\}$.

In the rest of this section, by graph we mean weighted connected graph. Any statement concerning the weighted graphs in this section is also extendable to the unweighted graphs (as (V, E) is equivalent to $(V, E, unit)$).

B.1.2 Tree Definition

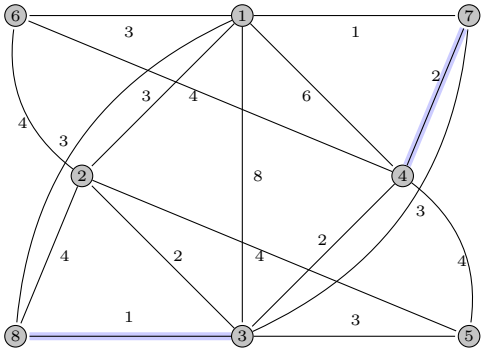
As mentioned before, the HDS of a graph is a sequence of vertex set partitions. The i^{th} partition (H_i) of the sequence contains a number of clusters that are more refined than those in H_{i-1} and coarser than the members of H_{i+1} (as you see in the definition of the HDS, clusters of the lower partitions are subsets of those in the higher ones). This relation between the clusters of subsequent partitions leads us

[2]Set $\mathcal{L} \subseteq 2^X$ is called a *laminar family* of set X if for every $A, B \in \mathcal{L}$, one of the following conditions holds: $A \subseteq B$, $B \subseteq A$, or $A \cap B = \emptyset$. Every partition of set X is a laminar family of X.

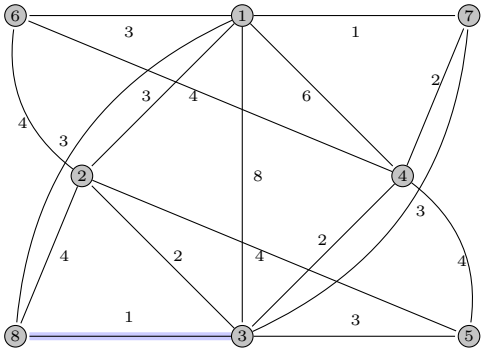


(a) level 3(Outermost level): $|\boldsymbol{H_3}| = \mathbf{1}$.

(b) level 2: $|\boldsymbol{H_2}| = \mathbf{2}$.



(c) level 1: $|\boldsymbol{H_1}| = \mathbf{6}$.



(d) level 0 (innermost): $|\boldsymbol{H_0}| = \mathbf{7}$.

Figure B.3: The hierarchical decomposition levels of a connected weighted graph of diameter eight. As you see, the number of clusters in the lower level partitions is more than or equal to the higher ones; i.e. $|\boldsymbol{H_3}| \leq |\boldsymbol{H_2}| \leq |\boldsymbol{H_1}| \leq |\boldsymbol{H_0}|$.

to define a laminar tree[3] for every HDS of a graph. Here is the formal definition of this tree:

Definition B.1.1. *Let* $\bar{\boldsymbol{H}} = (\boldsymbol{H_0}, \boldsymbol{H_1}, \ldots, \boldsymbol{H_h})$ *denote an HDS over the graph* $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{w})$. *Tree* $\boldsymbol{T_G}(\bar{\boldsymbol{H}}) = (\boldsymbol{V_T}, \boldsymbol{E_T})$ *of root* $\boldsymbol{H_h}$ *is called the Hierarchical Decomposition Tree (HDT) of HDS* $\bar{\boldsymbol{H}}$ *if this is the case that:*

$$\boldsymbol{V_T} = \{(\boldsymbol{C}, \boldsymbol{i}) | \boldsymbol{C} \in \boldsymbol{H_i} \wedge \boldsymbol{i} = \mathbf{0}, \ldots, \boldsymbol{h}\}$$

[3]Tree $\boldsymbol{T}$ of root $\boldsymbol{r}$ is a connected acyclic graph $(\boldsymbol{V}, \boldsymbol{E})$ with a distinguished vertex $\boldsymbol{r} \in \boldsymbol{V}$ called root. Considering $\boldsymbol{\mathcal{L}}$ as a laminar family of $\boldsymbol{X}$, *laminar tree* $\boldsymbol{T} = (\boldsymbol{V}, \boldsymbol{E})$ of root $\boldsymbol{r}$ and family $\boldsymbol{\mathcal{L}}$ is a tree such that every vertex $\boldsymbol{v} \in \boldsymbol{V}$ corresponds to set $\boldsymbol{S_v} \in \boldsymbol{\mathcal{L}}$, and for every edge $\{\boldsymbol{u}, \boldsymbol{v}\} \in \boldsymbol{E}$, if $\boldsymbol{d_T}(\boldsymbol{u}, \boldsymbol{r}) > \boldsymbol{d_T}(\boldsymbol{v}, \boldsymbol{r})$, $\boldsymbol{S_v} \subseteq \boldsymbol{S_u}$.

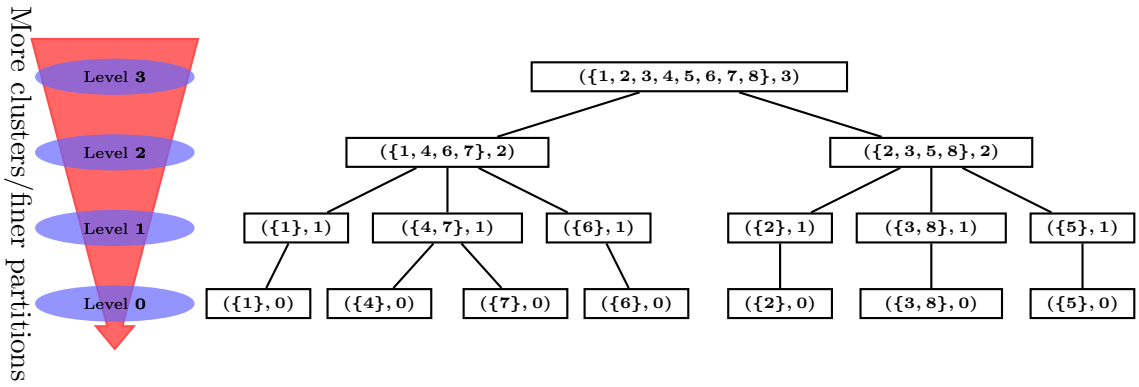


Figure B.4: Hierarchical decomposition tree of the HDS shown in Figure B.3

and

$$\boldsymbol{E_T = \{\{(C_1, i), (C_2, i+1)\} | C_1 \subseteq C_2 \wedge i = 0, 1, \ldots, h-1\}}$$

In the above relations, $\boldsymbol{h = \lceil \log_2 diam(G) \rceil}$. *For any tree vertex* $\boldsymbol{(C, 0) \in V_T}$, $\boldsymbol{C}$ *is called a basic cluster.*

Concerning the above definition, in the hierarchical decomposition tree of an HDS, every vertex is an ordered pair of a cluster and a number which specifies the level of the HDS partition containing the cluster. Additionally, each edge of the HDT connects a pair of vertices if their corresponding clusters belong to the partitions of the successive levels and also one cluster contains the other. Note that in the case that graph $\boldsymbol{G}$ is unweighted, the induced subgraph of every basic cluster in an HDT is a complete graph[4].

For an instance of the HDT, see Figure B.4 which shows the hierarchical decomposition tree of the HDS presented in Figure B.3. As you see, in the lower levels, the number of clusters increases and the partitions become finer.

[4]Unweighted graph $\boldsymbol{G = (V, E)}$ is complete if for every $\boldsymbol{u, v \in C}$, $\boldsymbol{d(u, v) \leq 1}$. Complete graphs are represented by symbol $\boldsymbol{K_i}$ where $\boldsymbol{i}$ is the cardinality of its vertex set.

B.2 Hierarchical Independence Tree Type-1

In this section, we introduce the "*hierarchical independence tree*" (HIT) as a hierarchical tool to organize the routing-related data in the bottom-up fashion. Before defining the HIT formally, the relation of independence between the vertices of a graph will be addressed thoroughly. Furthermore, the induced graph partitioning of the HIT will be discussed by comparing it with the HDT partitioning.

B.2.1 Independent Set of Vertices

Let $\boldsymbol{G = (V, E, w)}$ denote a connected graph. Set $\boldsymbol{I \subseteq V}$ is an $\boldsymbol{r}$*-independent set* of graph $\boldsymbol{G}$ if:

$$\forall \boldsymbol{u, v \in I : d_G(u, v) \geq r}$$

Now, consider set $\boldsymbol{J \subseteq V}$ as an $\boldsymbol{r}$-independent set of $\boldsymbol{G}$. Set $\boldsymbol{J}$ is the *maximum* $\boldsymbol{r}$*-independent set* of $\boldsymbol{G}$ if for every $\boldsymbol{r}$-independent set $\boldsymbol{J'}$ of graph $\boldsymbol{G}$, $\boldsymbol{|J| \geq |J'|}$. Similarly, the $\boldsymbol{r}$-independent set $\boldsymbol{K \subseteq V}$ is a *maximal* $\boldsymbol{r}$*-independent set* of $\boldsymbol{G}$ if:

$$\forall \boldsymbol{K' \subseteq V : (K'} \text{ is an } \boldsymbol{r}\text{-independet set} \wedge \boldsymbol{K \subseteq K') \rightarrow (K = K')}$$

In addition, assuming $\boldsymbol{I \subseteq V}$ as an $\boldsymbol{r}$-independent set of $\boldsymbol{G}$, set $\boldsymbol{J}$ is a *maximal* $\boldsymbol{r}$*-independent set of graph* $\boldsymbol{G}$ *including set* $\boldsymbol{I}$ if $\boldsymbol{J}$ specifies a maximal $\boldsymbol{r}$-independent set and $\boldsymbol{I \subseteq J}$.

Concerning the above definitions, the maximum $\boldsymbol{r}$-independent set of graph $\boldsymbol{G}$ has the maximum cardinality among all of the $\boldsymbol{r}$-independent sets of $\boldsymbol{G}$. For every $\boldsymbol{r_1, r_2 \in \mathbb{R}_{\geq 1}}$, assuming that $\boldsymbol{r_2 \geq r_1}$, the maximum $\boldsymbol{r_2}$-independent set is not larger than the maximum $\boldsymbol{r_1}$-independent set because every $\boldsymbol{r_2}$-independent set is also an $\boldsymbol{r_1}$-independent set (in the extreme case, the maximum $\boldsymbol{1}$-independent set of $\boldsymbol{G}$ is its vertex set, which has the maximum possible number of vertices). In

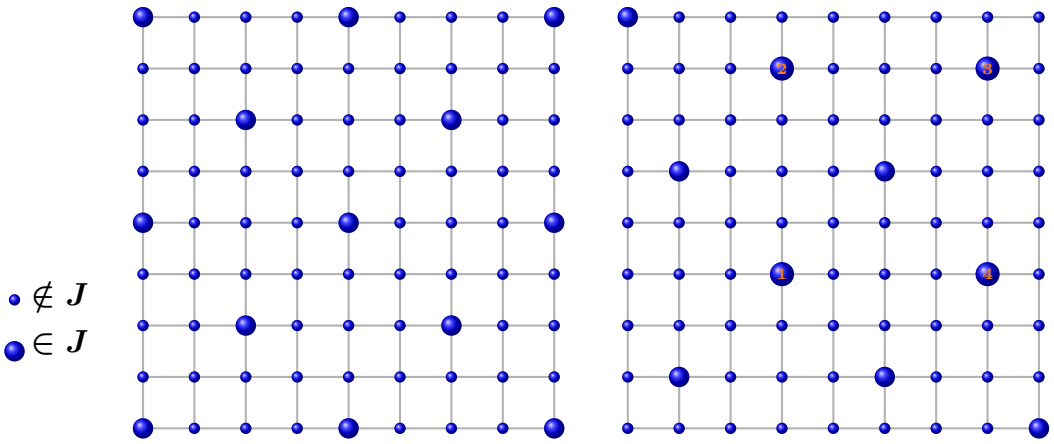


Figure B.5: Example of grid graph $G_{9\times 9}$. The left graph specifies set J as a maximum 4-independent set of the graph (in this case $|J| = 13$). The right one specifies J as a maximal 4-independent set of $G_{9\times 9}$ including set $I = \{1, 2, 3, 4\}$ (in this case, $|J| = 10$). Note that set I is a 4-independent set of the graph.

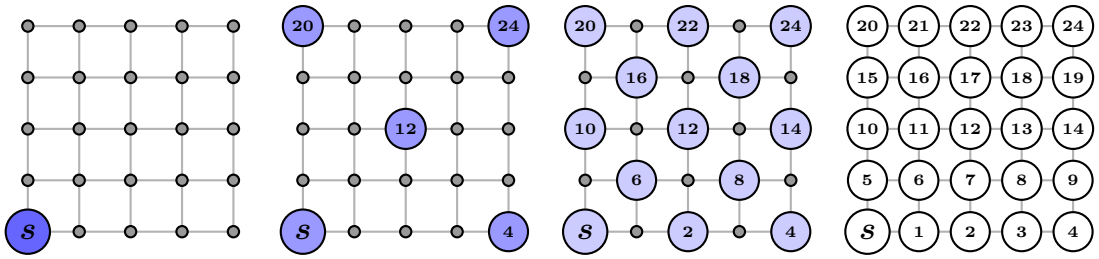
a similar manner, for every maximal r_1-independent set of G, there is no other r_2-independent set which contains it thoroughly.

To illustrate the independent sets, we use the family of two-dimensional grid graph[5]. As you see in Figure B.5, the maximum 4-independent set and the maximal 4-independent set, including set $\{1, 2, 3, 4\}$ of vertices, have been specified for graph $G_{9\times 9}$.

Assume $G = (V, E, w)$ as a connected graph and value $h = \lceil \log_2 diam(G) \rceil$. Considering vertex $s \in V$ as a distinguished vertex, sequence $\bar{I}_s = (I_0, I_1, I_2 \ldots, I_h)$ is called a *hierarchical independent sequence* (HIS) of source vertex s and graph G if:

- $I_h = \{s\}$; and
- $\forall i \in [0, h-1]$, I_i is a maximal 2^i-independent set of G including I_{i+1}.

[5]Two-dimensions grid graph $G_{n\times m}$ is a connected unweighted undirected graph (V, E) such that $V = \{v_{ij} | i \in [1, n] \wedge j \in [1, m]\}$ and $E = \{\{v_{ij}, v_{i(j+1)}\} | i \in [1, n] \wedge j \in [1, m-1]\} \cup \{\{v_{ij}, v_{(i+1)j}\} | i \in [1, n-1] \wedge j \in [1, m]\}$



(a) $\boldsymbol{I_3}$ (b) $\boldsymbol{I_2}$ (c) $\boldsymbol{I_1}$ (d) $\boldsymbol{I_0}$

Figure B.6: Considering $(\boldsymbol{I_0}, \boldsymbol{I_1}, \ldots, \boldsymbol{I_3})$ as a hierarchical independent sequence of source vertex $\boldsymbol{s}$ in grid graph $\boldsymbol{G_{5\times5}}$, sets $\boldsymbol{I_0}$, $\boldsymbol{I_1}$, $\boldsymbol{I_2}$, and $\boldsymbol{I_3}$ have been specified in the above figures ($\boldsymbol{I_0} = \boldsymbol{V}$). In each figure, the vertices included in the corresponding maximal $\boldsymbol{2^i}$-independent set are distinguished with the blue color. As you see, $\boldsymbol{I_3} \subseteq \boldsymbol{I_2} \subseteq \boldsymbol{I_1} \subseteq \boldsymbol{I_0}$.

Here is some basic properties of the hierarchical independence sequence:

i. Assuming that:

$$\boldsymbol{w(u,v) \geq 1} \qquad \boldsymbol{\forall \{u,v\} \in E},$$

$\boldsymbol{I_0}$ contains all the graph vertices; i.e. $\boldsymbol{I_0 = V}$.

ii. $\boldsymbol{I_h = \{s\}}$ is *not* necessarily a maximal $\boldsymbol{2^h}$-independent set.

iii. The HIS of a graph is not unique.

iv. For every $\boldsymbol{i, j \in [0, h]}$, this is the case that:

$$\boldsymbol{(i < j) \leftrightarrow (|I_i| > |I_j|)}$$

v. For every $\boldsymbol{i \in [0, h]}$, $\boldsymbol{s}$ belongs to set $\boldsymbol{I_i}$.

Figure B.6 schematically depicts an HIS of the shown source vertex in grid graph $\boldsymbol{G_{5\times5}}$.

Definition B.2.1. *Let* $\boldsymbol{G = (V, E, w)}$ *denote a connected graph and* $\boldsymbol{h = \lceil \log_2 diam(G) \rceil}$. *In addition, assume that* $\boldsymbol{\bar{I}_s = (I_0, I_1, I_2 \ldots, I_h)}$ *is a hierarchical independent sequence of source vertex* $\boldsymbol{s \in V}$ *in graph* $\boldsymbol{G}$. *Tree* $\boldsymbol{T^s = (V_{T^s}, E_{T^s})}$ *of root* $\boldsymbol{(s, h)}$ *is*

the hierarchical independence tree (HIT) type-1 associated with HIS $\bar{\boldsymbol{I}}_{\boldsymbol{s}}$ *if:*

$$\boldsymbol{V_{T^s}} = \{(\boldsymbol{v}, \boldsymbol{i}) | \boldsymbol{v} \in \boldsymbol{I_i} \wedge \boldsymbol{i} = \boldsymbol{0}, \boldsymbol{1}, \ldots \boldsymbol{h}\} \tag{B.2}$$

and $\boldsymbol{E_{T^s}} = \boldsymbol{E_1} \cup \boldsymbol{E_2}$ *such that:*

$$\boldsymbol{E_1} = \{\{(\boldsymbol{s}, \boldsymbol{h}), (\boldsymbol{v}, \boldsymbol{h}-\boldsymbol{1})\} | \boldsymbol{v} \in \boldsymbol{I_{h-1}}\} \tag{B.3}$$

and $\boldsymbol{E_2}$ *consists of the edges in the form* $\{(\boldsymbol{u}, \boldsymbol{i}+\boldsymbol{1}), (\boldsymbol{v}, \boldsymbol{i})\}$*, such that for every* $\boldsymbol{i} = \boldsymbol{0}, \boldsymbol{1}, \ldots, \boldsymbol{h}-\boldsymbol{2}$*, vertex* $\boldsymbol{v}$ *belongs to set* $\boldsymbol{I_i}$*, and* $(\boldsymbol{u}, \boldsymbol{i}+\boldsymbol{1})$ *is an arbitrarily chosen member of set* $\boldsymbol{Par_1}((\boldsymbol{v}, \boldsymbol{i}))$*. Function* $\boldsymbol{Par_1} : \boldsymbol{V_{T^s}} \mapsto \boldsymbol{2^{V_{T^s}}}$ *is called the parent function type-1 and is defined in the following form:*

$$\begin{aligned}\boldsymbol{Par_1}((\boldsymbol{v}, \boldsymbol{i})) = \{(\boldsymbol{u}, \boldsymbol{i}+\boldsymbol{1}) | \boldsymbol{u} \in \boldsymbol{I_{i+1}} \cap \boldsymbol{B_G}(\boldsymbol{v}, \boldsymbol{2^{i+1}}-\boldsymbol{1}) \\ \wedge \boldsymbol{d_G}(\boldsymbol{u}, \boldsymbol{s}) = \min_{\substack{\boldsymbol{x} \in \boldsymbol{B_G}(\boldsymbol{v}, \boldsymbol{2^{i+1}}-\boldsymbol{1}) \\ \boldsymbol{x} \in \boldsymbol{I_{i+1}}}} \boldsymbol{d_G}(\boldsymbol{x}, \boldsymbol{s})\} \qquad \forall \boldsymbol{i} \leq \boldsymbol{h}-\boldsymbol{2}\end{aligned} \tag{B.4}$$

As you see in Definition B.2.1, each tree vertex is an ordered pair of some vertex in $\boldsymbol{I_i}$ and index $\boldsymbol{i} = \boldsymbol{0}, \boldsymbol{1}, \ldots, \boldsymbol{h}$. Vertex set $\boldsymbol{L_i} = \{(\boldsymbol{v}, \boldsymbol{i}) | \boldsymbol{v} \in \boldsymbol{I_i}\}$ is called the set of the $\boldsymbol{i^{th}}$ level vertices[6] of the HIT, for every $\boldsymbol{i} \leq \boldsymbol{h}$ ($\boldsymbol{h}$ equals the tree height). Moreover, the edge set of the HIT is divided into two partitions $\boldsymbol{E_1}$ and $\boldsymbol{E_2}$. $\boldsymbol{E_1}$ consists of edges that connect the $\boldsymbol{(h-1)^{th}}$ level vertices to the source vertex ($\boldsymbol{s}$) which is the only vertex in level $\boldsymbol{h}$, and $\boldsymbol{E_2}$ members connect pairs of vertices together in a way that for every $\boldsymbol{i} = \boldsymbol{0}, \boldsymbol{1}, \ldots, \boldsymbol{h}-\boldsymbol{2}$ and $\boldsymbol{v} \in \boldsymbol{I_i}$, the parent[7] of

[6] In tree $\boldsymbol{T} = (\boldsymbol{V}, \boldsymbol{E})$ of root $\boldsymbol{r} \in \boldsymbol{V}$, a number is mapped to each vertex $\boldsymbol{v} \in \boldsymbol{V}$ as its level, represented by $\boldsymbol{level_T(v)}$. The level of root $\boldsymbol{r}$ is defined as $\max_{\boldsymbol{v} \in \boldsymbol{V}} \boldsymbol{d_T}(\boldsymbol{r}, \boldsymbol{v})$. The level of every other vertex $\boldsymbol{v} \neq \boldsymbol{r}$ is then equal to $\boldsymbol{level_T(r)} - \boldsymbol{d_T}(\boldsymbol{r}, \boldsymbol{v})$. The tree height is defined as $\boldsymbol{level_T(r)}$.

[7] In tree $\boldsymbol{T} = (\boldsymbol{V}, \boldsymbol{E})$, for every edge $\{\boldsymbol{u}, \boldsymbol{v}\} \in \boldsymbol{E}$, if $\boldsymbol{level_T(u)} > \boldsymbol{level_T(v)}$, $\boldsymbol{v}$ is called child of $\boldsymbol{u}$, and $\boldsymbol{u}$ is parent of $\boldsymbol{v}$.

vertex (v, i) will be $(u, i+1)$ such that u is the closest vertex to s in G that holds these two conditions:

- $u \in I_{i+1}$; and
- $d_G(u, v) \leq 2^{i+1} - 1$ or $u \in B_G(v, 2^{i+1} - 1)$ (the distance high threshold between child and parent).

To satisfy these conditions, the parent of every vertex at level $i \leq h - 2$ has to be chosen among the members of the output set of function Par_1. You will see in Lemma B.2.3 that Par_1 output is not empty for every level $i \leq h - 2$ vertex; however, its output may contain more than one vertex which leads to different HITs corresponding to one hierarchical independent sequence.

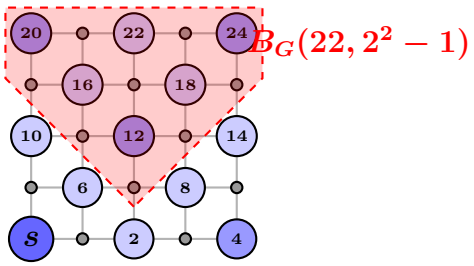


Figure B.7: The way of computing $Par_1((22, 1))$ has been depicted. Note that set $B_G(22, 2^2 - 1) \cap I_2$ is equal to $\{20, 12, 24\}$.

Figure B.8 briefly represents an HIT corresponding to the hierarchical independent sequence of graph $G_{5\times5}$ shown in Figure B.6. In this example, as an illustration of function Par_1, we compute $Par_1((22, 1))$. Among all the five members of I_2, vertices **12**, **20**, and **24** are close enough to vertex **22** (their distance from vertex **22** is not greater than the aforementioned threshold $2^2 - 1 = 3$). Additionally, vertices **12** and **20** are closer to s than vertex **24** (see Figure B.7); however, as both of them are equally far away of vertex s, set $Par_1((22, 1))$ contains both of them:

$$Par_1((22, 1)) = \{(20, 2), (12, 2)\}$$

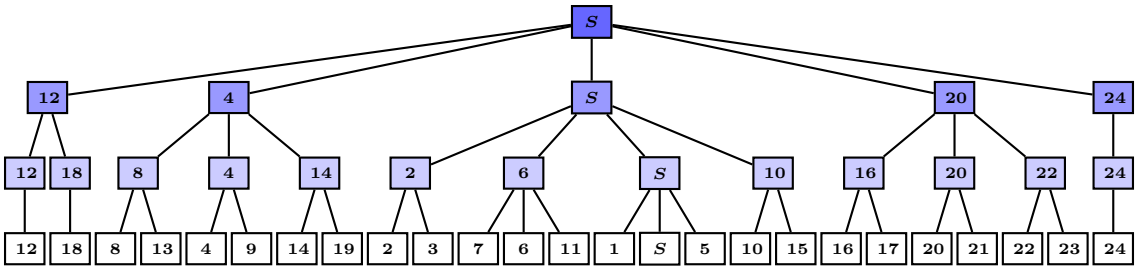


Figure B.8: Brief representation of the HIT type-1 corresponding to the hierarchical independent sequence of graph $\boldsymbol{G_{5\times5}}$ shown in Figure B.6. Each vertex of the HIT is an ordered pair represented by a rectangle of the figure in a way that its first element is denoted by the rectangle label and the second one is specified as the level of the rectangle in the shown tree.

Note that in the depicted HIT, for every $\boldsymbol{i = 1, 2, \ldots, h}$, vertex $\boldsymbol{(v, i)}$ has a child in the form $\boldsymbol{(v, i-1)}$. Generally, this property is true for an HIT type-1 of every HIS $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ of the connected graphs.

Lemma B.2.2. *Assume HIS* $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ *of source vertex* $\boldsymbol{s \in V}$ *and connected graph* $\boldsymbol{G = (V, E, w)}$. *Also consider* $\boldsymbol{T^s}$ *as an HIT type-1 corresponding to* $\boldsymbol{\bar{I}_s}$. *Every vertex* $\boldsymbol{(v, i)}$ *in tree* $\boldsymbol{T^s}$ *has a child in the form* $\boldsymbol{(v, i-1)}$ *where* $\boldsymbol{i = 1, 2, \ldots, h}$.

Proof. To prove this claim, we show that for an arbitrary tree vertex $\boldsymbol{(v, i)}$, vertex $\boldsymbol{(v, i-1)}$ exists and $\boldsymbol{\{(v, i), (v, i-1)\}}$ specifies a tree edge for every $\boldsymbol{i = 1, 2, \ldots, h}$. As $\boldsymbol{(v, i)}$ is a vertex of the HIT, regarding Equation B.2, $\boldsymbol{v}$ must belong to $\boldsymbol{I_i}$. Furthermore, since $\boldsymbol{I_i \subseteq I_{i-1}}$, $\boldsymbol{v}$ also belongs to $\boldsymbol{I_{i-1}}$. As the result, $\boldsymbol{(v, i-1)}$ is an HIT vertex (proof of existence).

Now, we show that vertex $\boldsymbol{(v, i-1)}$ is connected to $\boldsymbol{(v, i), \forall i = 1, 2, \ldots, h}$. In the case that $\boldsymbol{i = h}$, as $\boldsymbol{I_i}$ is a subset of $\boldsymbol{I_{i-1}}$, tree $\boldsymbol{T^s}$ has vertex $\boldsymbol{(s, h-1)}$ at the $\boldsymbol{(h-1)^{th}}$ level; on the other hand, regarding Equation B.3, all the vertices of tree $\boldsymbol{T^s}$ at level $\boldsymbol{(h-1)}$ are connected to the root ($\boldsymbol{(s, h)}$). This implies that the proof is complete for this case.

If $i = 1, 2, \ldots, h-1$, assume that vertex (u, i) is the parent of $(v, i-1)$ in tree T^s; hence this is the case that $(u, i) \in Par_1((v, i-1))$. Concerning Equation B.4, vertex (u, i) must belong to set $B_G(v, 2^i - 1)$, or:

$$d_G(u, v) \leq 2^i - 1 \tag{B.5}$$

In addition, since $u \in I_i$, $v \in I_i$, and I_i is a 2^i-independent set of G, $d_G(u, v) \geq 2^i$ unless $u = v$. Subsequently, regarding Equation B.5, u and v are the same and vertex (v, i) is the parent of $(v, i-1)$ in HIT T^s.

□

Before presenting a lemma which guarantees the existence of the hierarchical independence tree associated with an arbitrary HIS of a connected graph, some notation is contracted here. Considering tree $T = (V, E)$ of root $s \in V$, the k^{th} parent of node $v \in V$ is represented as $parent_k(v)$ and inductively defined below:

- $\mathcal{P}_0(v)$ is v;
- and $\mathcal{P}_{k+1}(v)$ is parent of $\mathcal{P}_k(v)$ in tree T.

Lemma B.2.3. *Consider a connected unweighted graph $G = (V, E)$ and HIS $\bar{I}_s$ of source $s \in V$ in G. There exists an HIT type-1 corresponding to sequence $\bar{I}_s$.*

Proof. Consider the definition of hierarchical independence tree $T^s = (V_{T^s}, E_{T^s})$. We first prove that for every $i = 0, 1, \ldots, h-2$, if vertex (v, i) belongs to set V_{T^s}, $Par_1((v, i))$ will not be empty (assuming $h = \lceil \log_2(diam(G)) \rceil \geq 2$). Then, we use this result to prove the existence of tree T^s.

Since $(v, i) \in V_{T^s}$, v is a vertex in I_i. On the other hand, as $I_{i+1} \subseteq I_i$, two possibilities exist: either $v \in I_{i+1}$ or $v \in I_i - I_{i+1}$.

In the earlier case, concerning Lemma B.2.2, vertex $(v, i+1)$ is the parent of (v, i). Consequently, according to the definition of the HIT type-1, $(v, i+1) \in Par_1((v, i))$ and $Par_1((v, i)) \neq \emptyset$.

In the later case, $v \in I_i$, but $v \notin I_{i+1}$. By contradiction, assume that $Par_1((v, i)) = \emptyset$. Henceforth, regarding Equation B.4, $I_{i+1} \cap B_G(v, 2^{i+1} - 1) = \emptyset$. This implies that:

$$\begin{aligned} \forall x \in I_{i+1} : \quad & x \notin B_G(v, 2^{i+1} - 1) \\ & \rightarrow d_G(v, x) > 2^{i+1} - 1 \\ & \xrightarrow{(*)} d_G(v, x) \geq 2^{i+1} \end{aligned} \tag{B.6}$$

Note that the last deduction (*) is made because of the assumption that graph G is unweighted (in an unweighted graph, distance function always outputs an integer number).

Since I_{i+1} is a maximal 2^{i+1}-independent set, according to Equation B.6, set $I_{i+1} \cup \{v\}$ is also a 2^{i+1}-independent set. As $v \notin I_{i+1}$, we obtain $|I_{i+1} \cup \{v\}| > |I_{i+1}|$, which contradicts the assumption that I_{i+1} is maximal 2^{i+1}-independent set.

Now, we need to prove that graph $T^s = (V_{T^s}, E_{T^s})$ is an acyclic connected graph where V_{T^s} and E_{T^s} are defined in Definition B.2.1. To do this, we should show that for every $u, v \in V_{T^s}$ $(u \neq v)$, there is a path between u and v in T^s (connectivity) and this path is unique (acyclic).

Consider (u, i) and (v, j) as two arbitrary vertices of T^s where $(u, i) \neq (v, j)$. Without loss of generality, we assume that $i \geq j$. We continue the proof in these two possible cases: $i = j$ and $i > j$.

First assume that $i = j$. In this case, regarding Definition B.2.1, $(h - i)^{th}$ parent of (u, i) and (v, j) is (s, h) (this is easy to show by induction); i.e.

$$\mathcal{P}_{h-i}((u, i)) = \mathcal{P}_{h-i}((v, j))$$

Consider the smallest integer $k \leq h-i$ that makes this equation true: $\mathcal{P}_k((u, i)) = \mathcal{P}_k((v, j))$. Subsequently, for every $c \in [0, k - 1]$, $\mathcal{P}_c((u, i)) \neq \mathcal{P}_c((v, j))$. As the result, set $p = p_1 \cup p_2$ specifies a walk of no repetitive edge[8] from (u, i) to (v, j) in graph T^s where:

$$p_1 = \{\{\mathcal{P}_c((u, i)), \mathcal{P}_{c+1}((u, i))\} | c \leq k - 1\}$$

and

$$p_2 = \{\{\mathcal{P}_c((v, i)), \mathcal{P}_{c+1}((v, i))\} | c \leq k - 1\}$$

Note that p_1 is a walk from (u, i) to vertex $\mathcal{P}_k((u, i))$ in graph G and p_2 is a walk from (v, i) to vertex $\mathcal{P}_k((v, i))$ which is equal to $\mathcal{P}_k((u, i))$.

Consequently, if $i = j$, there is a walk (and also a simple path) that connects (u, i) and (v, j). In the second case, we only need to connect vertex (v, j) to its $(i - j)^{th}$ parent using the following walk, and then use the result of the first case to connect vertices $\mathcal{P}_{i-j}(v, j)$ and (u, i) together.

$$p' = \{\{\mathcal{P}_c((v, j)), \mathcal{P}_{c+1}((v, j))\} | c \leq i - j - 1\}$$

Hence, in the case that $i > j$, walk $p \cup p'$ connects vertices (u, i) and (v, j) together. This implies that for every two vertices in graph T^s, there is a path between them (proof of connectivity).

Again, consider (u, i) and (v, j) as two arbitrary vertices of T^s that $(u, i) \neq (v, j)$ and $i \geq j$. Also, let A be an arbitrary path between these two vertices.

[8] A walk of no repetitive edge was defined in Section 1.1.

Here, we prove by induction on j that for every $j \in [1, i]$, if e denotes the edge connecting (v, j) to one of its children, $e \notin A$. In the case that $k = 1$, assume that $e \in A$. Regarding the path definition in chapter 1, $A = \{e\} \cup A'$ where A' is some path from a vertex in the form $(w, 0)$ to vertex (u, i) such that $e \notin A'$. As e is the only edge connecting $(w, 0)$ to other vertices of T^s, there is no path A' (contradiction). Assuming that children of (v, k) don't participate in path A for every $k \in [1, i-1]$ (induction step), we prove that the same statement is true for $k+1$. Again, by contradiction, assume that $A = \{e\} \cup A'$ where e connects $(v, k+1)$ to one of its children in the form (w, k), and A' denotes a path from (w, k) to (u, i) such that $e \notin A'$. In addition, according to induction hypothesis, none of (w, k)'s children are crossed by A'; so, there is no edge in A' that connects (w, k) to any other vertex in T^s (contradiction).

From the above discussion, we conclude that for every $j \in [0, i]$, the path from (v, j) to (u, i) contains the parent of (v, j), not its children (in the case that $j = 0$, vertex (v, j) has no children). Hence, regarding the path definition, we have:

$$A = \{\{\mathcal{P}_c((v, j)), \mathcal{P}_{c+1}((v, j))\} | c \leq i - j\} \cup B$$

where B is a path from $(x, i+1) = \mathcal{P}_{i-j+1}((v, j))$ to (u, i). Assume that m is the smallest integer that holds this equation: $\mathcal{P}_m((x, i+1)), \mathcal{P}_{m+1}((u, i))$ (we know for sure that $m \leq h - i$). Now, by induction on n, we prove that path B contains edge $e_n = \{\mathcal{P}_n((x, i+1)), \mathcal{P}_{n+1}((x, i+1))\}$ for every $n \leq m - 1$. In the case that $n = 0$, if $e_n \notin B$, some edge between $(x, i+1)$ and one of its children, say (y, i) belongs to B. As vertex $(x, i+1)$ is not the parent of (u, i), this is the case that $(y, i) \neq (u, i)$, and according to the previous results, $\{(y, i), (x, i+1)\} \notin B$ which means that vertex $(x, i+1)$ is not crossed by B (contradiction). This implies that $B = \{e_0\} \cup B_1$, where B_1 is a path from

$\boldsymbol{\mathcal{P}_1((x, i+1))}$ to $\boldsymbol{(u, i)}$. Also, we assume that $\boldsymbol{B = \{e_r | r \in [0, k]\} \cup B_{k+1}}$, for every $\boldsymbol{k < m}$, where $\boldsymbol{B_k}$ is a path from $\boldsymbol{\mathcal{P}_k((x, i+1))}$ to $\boldsymbol{(u, i)}$. By contradiction, assume that $\boldsymbol{e_{k+1}}$ doesn't belong to path $\boldsymbol{B_{k+1}}$. Subsequently, there exists an edge like $\boldsymbol{e'}$ which connects $\boldsymbol{\mathcal{P}_{k+1}((x, i+1))}$ to one of its children (say $\boldsymbol{(z, i+k+1)}$) such that $\boldsymbol{e' \in B_{k+1}}$; i.e.

$$\boldsymbol{B_{k+1} = \{e'\} \cup B'}$$

where $\boldsymbol{B'}$ is a path from $\boldsymbol{(z, i+k+1)}$ to $\boldsymbol{(u, i)}$. As $\boldsymbol{(z, i+k+1)}$ is not the parent of parents of $\boldsymbol{(u, i)}$, by similar inductive reasoning, we can show that $\boldsymbol{B'}$ doesn't contain any edge connecting $\boldsymbol{(z, i+k+1)}$ to its children. Consequently, there would be no edge in $\boldsymbol{B'}$ connecting $\boldsymbol{(z, i+k+1)}$ to any other vertex in $\boldsymbol{T^s}$ (contradiction). As the result,

$$\boldsymbol{B = \{e_r | r \in [0, m-1]\} \cup B_m}$$

which $\boldsymbol{B_m}$ is a path from $\boldsymbol{\mathcal{P}_m((x, i+1)) = \mathcal{P}_{m+1}((u, i))}$ to $\boldsymbol{(u, i)}$. With the similar reasoning, we will obtain the following equation:

$$\boldsymbol{B = \{e'_r | r \in [0, m]\} \cup B'_m}$$

where $\boldsymbol{e'_r = \{\mathcal{P}_r((u, i)), \mathcal{P}_{r+1}((u, i))\}}$ and $\boldsymbol{B'_m}$ is a path from $\boldsymbol{\mathcal{P}_{m+1}((u, i))}$ to $\boldsymbol{(x, i+1)}$.

Consequently, the path from $\boldsymbol{(v, j)}$ to $\boldsymbol{(u, i)}$ is unique and can be written in the following form:

$$\boldsymbol{A = \{\{\mathcal{P}_c((v, j)), \mathcal{P}_{c+1}((v, j))\} | c \leq i - j\} \cup \{e'_r | r \leq m\} \cup \{e_r | r \leq m-1\}}$$

□

B.2.2 Induced Partitions of HIT

In Section B.1, we addressed the hierarchical decomposition trees and the vertex set partitions of an arbitrary connected graph were created using the HDTs. Here, we also focus on another type of partitioning which is induced by the HIT type-1. At first, some contracted notations are presented.

Considering tree $\boldsymbol{T = (V, E)}$ of root $\boldsymbol{s \in V}$, the unique path between any two vertices of tree $\boldsymbol{u, v \in V}$ is shown as $\boldsymbol{p_T(u, v)}$. In addition, $\boldsymbol{T_v}$ represents the *subtree*[9] of tree $\boldsymbol{T}$ and is rooted at vertex $\boldsymbol{v \in V}$. Additionally, the set of all the *leaves*[10] of tree $\boldsymbol{T}$ is denoted by $\boldsymbol{T.leaves}$.

Lemma B.2.4. *Consider HIT type-1* $\boldsymbol{T^s = (V, E)}$ *and the arbitrary vertex* $\boldsymbol{(r, i) \in V}$*. Also, let* $\boldsymbol{T' = (V', E')}$ *denote subtree* $\boldsymbol{T^s{}_{(r,i)}}$*. Assuming that* $\boldsymbol{(u, j) \in V'}$*, this is the case that*

$$\boldsymbol{p_{T^s}((u, j), (r, i)) = p_{T'}((u, j), (r, i)) = \{\{\mathcal{P}_c((u, j)), \mathcal{P}_{c+1}((u, j))\} | c \leq i - j - 1\}} \tag{B.7}$$

In addition, assuming vertex $\boldsymbol{(u', j') \in V'}$*, there is a walk of no repetitive edge in* $\boldsymbol{T^s}$ *and* $\boldsymbol{T'}$ *in the following form:*

$$\boldsymbol{p_{T^s}((u, j), (u', j')) = p_{T^s}((u, j), (r, i)) \Delta p_{T^s}((r, i), (u', j'))} \tag{B.8}$$

Proof. This lemma is directly concluded from Lemma B.2.3 and the subtree definition.

□

[9]Tree $\boldsymbol{T' = (V', E')}$ is called a subtree of tree $\boldsymbol{T = (V, E)}$ rooted at $\boldsymbol{x \in V}$ if graph $\boldsymbol{T'}$ is a subgraph of $\boldsymbol{T}$ induced by $\boldsymbol{V' = \{v \in V | \exists i : v = \mathcal{P}_i(x)\}}$.

[10]Assuming $\boldsymbol{T = (V, E)}$ as a tree rooted at $\boldsymbol{s \in V}$, vertex $\boldsymbol{v \in V}$ is a tree leaf if and only if $\boldsymbol{\forall u \neq v : p_T(s, v) \not\subseteq p_T(s, u)}$, where $\boldsymbol{p_T(u, v)}$ denotes the unique path between every two vertices $\boldsymbol{u, v \in V}$ in tree $\boldsymbol{T}$.

Lemma B.2.5. *Let* $\boldsymbol{T^s = (V_{T^s}, E_{T^s})}$ *denote a hierarchical independence tree type-1 in graph* $\boldsymbol{G = (V, E)}$*. For every tree vertex* $\boldsymbol{(r, i) \in V_{T^s}}$*, an arbitrary tree leaf of subtree* $\boldsymbol{T^s{}_{(r,i)}}$ *is in the form* $\boldsymbol{(l, 0)}$ *and this is the case that:*

$$\boldsymbol{d_G(l, r) < 2^{i+1}} \tag{B.9}$$

Proof. Let $\boldsymbol{(l, j)}$ be a leaf of tree $\boldsymbol{T'}$. At first, we prove that $\boldsymbol{j = 0}$; then, we prove Inequality B.9.

By contradiction, assume that $\boldsymbol{j \neq 0}$. Subsequently, according to Lemma B.2.2, $\boldsymbol{(l, j)}$ has a child of the form $\boldsymbol{(l, j - 1)}$ (in both $\boldsymbol{T^s}$ and $\boldsymbol{T'}$). This implies that $\boldsymbol{p_{T'}((l, j - 1), (r, i)) = p_{T'}((l, j), (r, i)) \cup \{\{(l, j - 1), (l, j)\}\}}$. As the result, $\boldsymbol{p_{T'}((l, j), (r, i)) \subseteq p_{T'}((l, j - 1), (r, i))}$ which contradicts the assumption that $\boldsymbol{(l, j)}$ is a $\boldsymbol{T'}$ leaf. Hence, every leaf of $\boldsymbol{T'}$ is in the form $\boldsymbol{(l, 0)}$ where $\boldsymbol{l}$ is some vertex in $\boldsymbol{G}$.

Consider vertex $\boldsymbol{(l, 0) \in T'.leaves}$. According to Equation B.7, since $\boldsymbol{T'}$ is a subtree of $\boldsymbol{T^s}$ rooted at $\boldsymbol{(r, i)}$,

$$\boldsymbol{p_{T^s}((l, 0), (r, i)) = \{\{\mathcal{P}_c((l, 0)), \mathcal{P}_{c+1}((l, 0))\} | c \in [0, i - 1]\}}$$

For every $\boldsymbol{c \in [0, i - 1]}$, we map edge $\boldsymbol{\{\mathcal{P}_c((l, 0)), \mathcal{P}_{c+1}((l, 0))\}}$ to path $\boldsymbol{p_c}$ which is defined in the following way:

$\boldsymbol{p_c \subseteq V}$ is the shortest path in $\boldsymbol{G}$ from $\boldsymbol{\Pi_1(\mathcal{P}_c((l, 0))}$ to $\boldsymbol{\Pi_1(\mathcal{P}_{c+1}((l, 0))}$.

By this mapping, *the projection of path* $\boldsymbol{p_{T^s}((l, 0), (r, i))}$ *on graph* $\boldsymbol{G}$ is obtained by the following formula.

$$\boldsymbol{p = \bigcup_{c=0}^{i-1} p_c} \tag{B.10}$$

Concerning Equation B.10, $\boldsymbol{|p|}$ equals $\boldsymbol{\sum_{c=0}^{i-1} |p_c|}$. Since $\boldsymbol{p_c}$ is the shortest path in $\boldsymbol{G}$ from $\boldsymbol{\Pi_1(\mathcal{P}_c((l, 0))}$ to $\boldsymbol{\Pi_1(\mathcal{P}_{c+1}((l, 0))}$, regarding the definition of $\boldsymbol{Par_1}$ in

Equation B.4, this is the case that:

$$\begin{aligned} |\boldsymbol{p_c}| &= \boldsymbol{d_G}(\boldsymbol{\mathcal{P}_c}((\boldsymbol{l},\boldsymbol{0})), \boldsymbol{\mathcal{P}_{c+1}}((\boldsymbol{l},\boldsymbol{0}))) \\ &\leq \boldsymbol{2^{c+1}-1} \\ &< \boldsymbol{2^{c+1}} \end{aligned}$$

Subsequently,

$$\begin{aligned} |\boldsymbol{p}| &= \sum_{\boldsymbol{c=0}}^{\boldsymbol{i-1}} |\boldsymbol{p_c}| \\ &< \sum_{\boldsymbol{c=0}}^{\boldsymbol{i-1}} \boldsymbol{2^{c+1}} \\ &= \boldsymbol{2^{i+1}-2} \\ &< \boldsymbol{2^{i+1}} \end{aligned}$$

Path $\boldsymbol{p}$ may cross itself, which results in the creation of cycles. By removing those edges that are participating in $\boldsymbol{p}$ cycles, we get simple path $\boldsymbol{p'}$ from $\boldsymbol{v_0} = \boldsymbol{l}$ to $\boldsymbol{v_i} = \boldsymbol{r}$ in $\boldsymbol{G}$ with less cardinality than $\boldsymbol{p}$. Finally, we obtain the following inequality:

$$\boldsymbol{d_G}(\boldsymbol{l},\boldsymbol{r}) \leq |\boldsymbol{p'}| \leq |\boldsymbol{p}| < \boldsymbol{2^{i+1}}$$

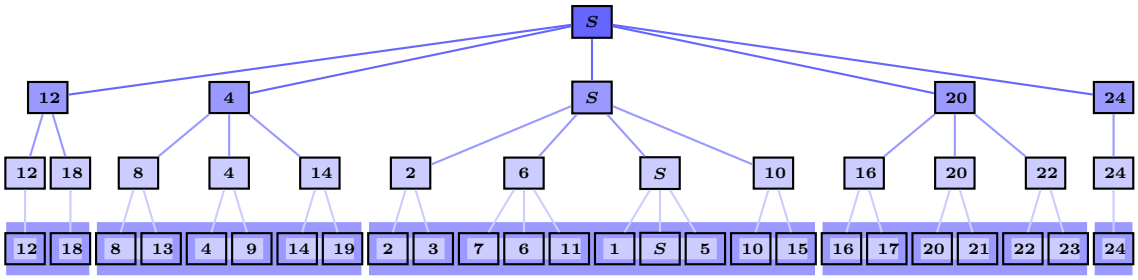
□

The hierarchical independence tree of height $\boldsymbol{h}$ in graph $\boldsymbol{G} = (\boldsymbol{V},\boldsymbol{E})$ induces $(\boldsymbol{h}+\boldsymbol{1})$ partitions of set $\boldsymbol{V}$. These partitions are obtained in the following way.

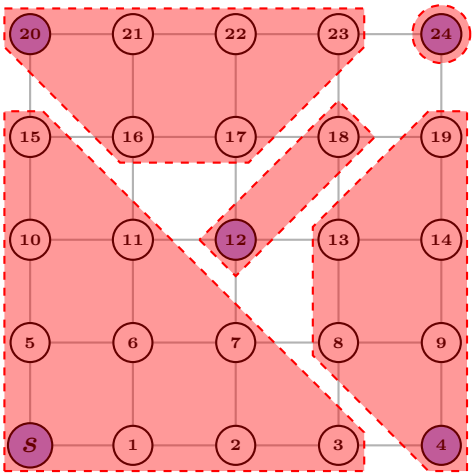
Definition B.2.6. *Let* $\boldsymbol{T^s} = (\boldsymbol{V_{T^s}}, \boldsymbol{E_{T^s}})$ *denote an HIT type-1 of HIS* $\boldsymbol{I_s} = (\boldsymbol{I_0}, \boldsymbol{I_1}, \ldots, \boldsymbol{I_h})$ *in graph* $\boldsymbol{G} = (\boldsymbol{V},\boldsymbol{E})$*. For every* $\boldsymbol{i} \in [\boldsymbol{0},\boldsymbol{h}]$*, the level-*$\boldsymbol{i}$ *induced partition of tree* $\boldsymbol{T^s}$ *is denoted by* $\boldsymbol{\mathcal{V}_i}$ *and defined in the following equation:*

$$\boldsymbol{\mathcal{V}_i} = \{\boldsymbol{\mathcal{L}_{T^s}}(\boldsymbol{v},\boldsymbol{i}) | \boldsymbol{v} \in \boldsymbol{I_i}\}$$

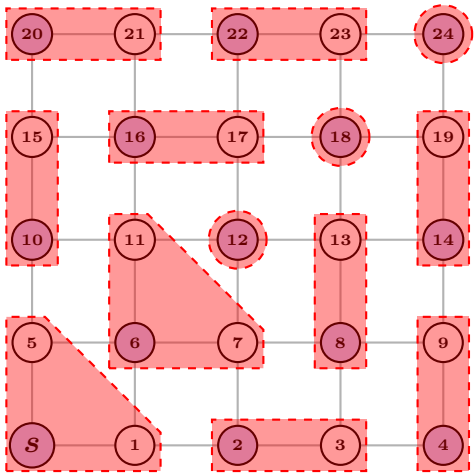
where $\boldsymbol{\mathcal{L}_{T^s}}(\boldsymbol{v},\boldsymbol{i}) = \{\boldsymbol{u} | (\boldsymbol{u},\boldsymbol{0}) \in \boldsymbol{T^s}_{(\boldsymbol{v},\boldsymbol{i})}.\boldsymbol{leaves}\}$.



(a) The way that level-1 and level-2 partitions are induced by HIT type-1 of $G_{5\times5}$



(b) Level-2 induced partition of HIT shown in (a)



(c) Level-1 induced partition of HIT shown in (a)

Figure B.9: Induced level-1 and level-2 partitions of HIT type-1 shown in Figure B.8

In Definition B.2.6, note that for any two members u_1 and u_2 of $\mathcal{L}_{T^s}(v, i)$, the following inequality holds (triangle inequality):

$$d_G(u_1, u_2) \leq d_G(u_1, v) + d_G(v, u_2)$$

Moreover, regarding Lemma B.2.5, we obtain that:

$$d_G(u_1, v) + d_G(v, u_2) < 2^{i+1} + 2^{i+1}$$

Consequently, for every vertex $v \in I_i$, this is the case that:

$$\forall u_1, u_2 \in \mathcal{L}_{T^s}(v, i) : d_G(u_1, u_2) < 2^{i+2} \tag{B.11}$$

Equation B.11 shows that in induced partition $\boldsymbol{\mathcal{V}_i}$, the vertices belonging to cluster $\boldsymbol{\mathcal{L}_{T^s}(v,i)}$ become exponentially closer to each other as we go to the lower levels (with smaller $\boldsymbol{i}$). This property implies that set $\boldsymbol{\mathcal{L}_{T^s}(v,i)}$ is a $\boldsymbol{2^{i+2}}$-*partition* of the vertex set of graph $\boldsymbol{G}$. In Section 2.7, you will be asked to prove that the sequence $\boldsymbol{(\mathcal{V}_0,\mathcal{V}_0,\mathcal{V}_0,\mathcal{V}_1,\mathcal{V}_2,\ldots,\mathcal{V}_{h-3},\{V\})}$ specifies an HDS of graph $\boldsymbol{G}$. Similar to the HDS, in the induced partitions, cluster $\boldsymbol{\mathcal{L}_{T^s}(v,i)}$ does not necessarily induce a connected subgraph in $\boldsymbol{G}$. Figure B.9 represents the level-1 and level-2 induced partitions of the HIT type-1 depicted in Figure B.8.

B.3 Hierarchical Independence Tree Type-2

The second type of the hierarchical independence trees is defined in the similar way that the first one was defined. The only difference is the parent function is defined in a different way than what was mentioned in Equation B.4. Here is the definition of function $\boldsymbol{Par_2 : V_{T^s} \rightarrow 2^{V_{T^s}}}$:

$$\boldsymbol{Par_2((v,i)) = \{(u,i+1) | u \in I_{i+1} \cap B_G(v, 2^{i+2}-2)} \\ \boldsymbol{\wedge\, d_G(u,s) = \min_{\substack{B_G(v,2^{i+2}-2) \\ x \in I_{i+1}}} d_G(x,s)\}} \qquad \boldsymbol{\forall i \leq h-1} \tag{B.12}$$

As you see, the distance high threshold between child and parent in the $\boldsymbol{i^{th}}$ level is $\boldsymbol{2^{i+2}-2}$ which is two times more than type-1's. Note that concerning Equation B.12, we obtain the following:

$$\boldsymbol{\forall v \in I_{h-1} : Par_2((v,h-1)) = \{(s,h)\}}$$

This means that every vertex at the $\boldsymbol{(h-1)^{th}}$ level is connected to the root as its child (like the HIT type-1).

Lemma B.3.1. *For every connected unweighted graph* $\boldsymbol{G = (V, E)}$ *and HIS* $\boldsymbol{\bar{I}_s}$ *of source* $\boldsymbol{s \in V}$ *in* $\boldsymbol{G}$*, there is an HIT type-2 corresponding to* $\boldsymbol{\bar{I}_s}$*.*

Proof. Just like the existence proof of the HIT type-1, here we first prove that:

$$\forall \boldsymbol{v \in I_i : Par_2((v, i)) \neq \emptyset}$$

And then we must show that $\boldsymbol{T^s}$ is an acyclic connected graph.

Let $\boldsymbol{v}$ be some vertex in set $\boldsymbol{I_i}$. If $\boldsymbol{d_G(v, s) \leq 2^{i+2} - 2}$, regarding Equation B.12, vertex $\boldsymbol{(s, i + 1)}$ belongs to $\boldsymbol{Par_2((v, i))}$ and the proof is complete.

Now, assume that $\boldsymbol{d_G(v, s) > 2^{i+2} - 2}$, and set $\boldsymbol{p}$ denotes the shortest path in $\boldsymbol{G}$ that connects graph vertex $\boldsymbol{v}$ to source vertex $\boldsymbol{s}$. Assume that path $\boldsymbol{p}$ is partitioned to two subpaths $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ such that $\boldsymbol{p_1}$ is a path of length $\boldsymbol{2^{i+1} - 1}$ from $\boldsymbol{v}$ to $\boldsymbol{x}$, and $\boldsymbol{p_2}$ is a path from $\boldsymbol{x}$ to $\boldsymbol{s}$. Since $\boldsymbol{p = p_1 \cup p_2}$ and $\boldsymbol{|p_1| = 2^{i+1} - 1}$, this is the case that:

$$\boldsymbol{|p_2| = d_G(v, s) - 2^{i+1} - 1} \tag{B.13}$$

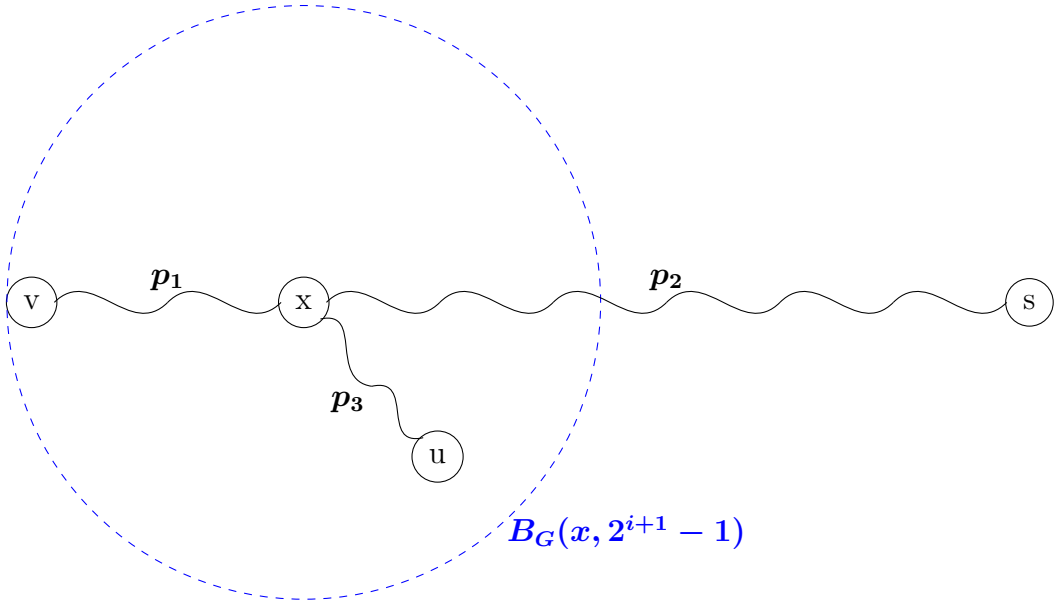
Since $\boldsymbol{d_G(v, s)}$ is greater than $\boldsymbol{2^{i+2} - 2}$, the value of $\boldsymbol{|p_2|}$ is positive. This implies that it is possible to partition $\boldsymbol{p}$ in this way. The following figure depicts vertices $\boldsymbol{s}$, $\boldsymbol{v}$, $\boldsymbol{x}$, and the paths between them.

Now, we continue the proof in two possible cases: $\boldsymbol{x \in I_{i+1}}$ and $\boldsymbol{x \notin I_{i+1}}$. In the first case, since $\boldsymbol{d_G(x, v) \leq |p_2| < 2^{i+2} - 2}$, $\boldsymbol{x}$ is a member of set $\boldsymbol{I_{i+1} \cap B_G(v, 2^{i+2} - 2)}$. Henceforth, according to Equation B.12, $\boldsymbol{Par_2((v, i)) \neq \emptyset}$.

In the second case, we show that there is some vertex $\boldsymbol{u}$ in $\boldsymbol{I_{i+1}}$ such that $\boldsymbol{u \in B_G(x, 2^{i+1} - 1)}$. By contradiction, assume that there is no such vertex. Hence, this is the case that:

$$\forall \boldsymbol{u \in I_{i+1} : d_G(x, u) \geq 2^{i+1}}$$

which means that $\boldsymbol{x}$ is so far from all of the $\boldsymbol{I_{i+1}}$ members that by adding $\boldsymbol{x}$ to set $\boldsymbol{I_{i+1}}$, it still remains a $\boldsymbol{2^{i+1}}$-independent set. In addition, since $\boldsymbol{x}$ doesn't belong



to I_{i+1}, $|I_{i+1} \cup \{x\}| > |I_{i+1}|$ which contradicts the assumption that I_{i+1} is a *maximal* 2^{i+1}-independent set. Henceforth, we have shown that:

$$\exists u \in I_{i+1} : u \in B_G(x, 2^{i+1} - 1)$$

Let p_3 denote the shortest path in G from x to u. Here, we find an upper-bound of $d_G(u, v)$.

$$d_G(u, v) \leq d_G(u, x) + d_G(x, v) \leq |p_1| + |p_2|$$

Since $|p_1| = 2^{i+1} - 1$, we obtain the following inequality:

$$d_G(u, v) \leq (2^{i+1} - 1) + (2^{i+1} - 1) = 2^{i+2} - 2 \tag{B.14}$$

Regarding Inequality B.14 and the definition of Par_2, we conclude that $Par_2((v, i)) \neq \emptyset$.

The rest of the proof is the same as what we did to prove that HIT type-1 is connected and acyclic.

□

After presenting the existence proof of the HIT type-2, we will see some of its properties.

Lemma B.3.2. *Let* $\boldsymbol{T^s = (V_{T^s}, E_{T^s})}$ *be a hierarchical independence tree type-2 of HIS* $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ *in graph* $\boldsymbol{G = (V, E)}$.

i. $\boldsymbol{\forall v \in I_i, \forall u \in \mathcal{L}_{T^s}(v, i) : d_G(u, v) < 2^{i+2}} \qquad \boldsymbol{\forall i \in [0, h]}$

ii. $\boldsymbol{\forall v \in I_i, \forall u_1, u_2 \in \mathcal{L}_{T^s}(v, i) : d_G(u_1, u_2) < 2^{i+3}} \qquad \boldsymbol{\forall i \in [0, h]}$

iii. $\boldsymbol{B_G(s, 2^i - 1) \subseteq \mathcal{L}_{T^s}(s, i)} \qquad \boldsymbol{\forall i \in [0, h]}$

where $\boldsymbol{\mathcal{L}_{T^s}(v, i) = \{u | (u, 0) \in T^s{}_{(v,i)}.leaves\}}$.

Proof. The proof of the first property is similar to what you previously saw in the proof of Lemma B.2.5 for the HIT type-1. The only difference is that for the HIT type-2:

$$\boldsymbol{|p_c| \leq 2^{i+2} - 2 < 2^{i+2}}$$

Subsequently,

$$\boldsymbol{|p| = \sum_{c=0}^{i-1} 2^{c+2} < 2^{i+2}}$$

Since $\boldsymbol{d_G(u, v) \leq |p|}$, the proof is complete.

In the second property, this is the case that:

$$\boldsymbol{\forall u_1, u_2 \in \mathcal{L}_{T^s}(v, i) : d_G(u_1, u_2) \leq d_G(u_1, v) + d_G(v, u_2)}$$

Consequently, according to the first property, we obtain the following inequality:

$$\begin{aligned} \boldsymbol{d_G(u_1, u_2)} &\boldsymbol{< 2^{i+2} + 2^{i+2}} \\ &\boldsymbol{\leq 2^{i+3}} \end{aligned}$$

To prove the last property, we will show that for every $\boldsymbol{j \in [0, i]}$, if $\boldsymbol{v}$ belongs to $\boldsymbol{I_j \cap B_G(s, 2^i - 1)}$, ordered pair $\boldsymbol{(v, j)}$ is a vertex in $\boldsymbol{T^s{}_{(s,i)}}$. We do this by induction on $\boldsymbol{j}$.

Let v denote an arbitrary vertex in $I_i \cap B_G(s, 2^i - 1)$. If $v \neq s$, since v is in set I_i, $d_G(v, s)$ will not be less than 2^i which contradicts the assumption that $v \in B_G(s, 2^i - 1)$. As the result, $v = s$, and $(v, i) = (s, i)$ is a vertex in tree $T^s{}_{(s,i)}$ (the first base case of the induction).

Additionally, assume that $v \in I_{i-1} \cap B_G(s, 2^i - 1)$. Regarding the definition of Par_2, $Par_2((v, i-1)) = \{(s, i)\}$ (because $s \in I_i$, $s \in B_G(v, 2^{i+1} - 2)$, and s is the closest vertex to itself that belongs to set $I_i \cap B_G(v, 2^{i+1} - 2)$). As (s, i) is the only member of $Par_2((v, i - 1))$, it is chosen as the parent of $(v, i - 1)$. Consequently, $(v, i - 1)$ is a vertex in subtree $T^s{}_{(s,i)}$ (the second base case of the induction).

Now, assume that for some $j \in [1, i - 1]$, if $y \in I_j \cap B_G(s, 2^i - 1)$, (y, j) will be a vertex of $T^s{}_{(s,i)}$ (Induction Hypothesis). Additionally, consider that $v \in I_{j-1} \cap B_G(s, 2^i - 1)$. Here, we continue the proof by considering two possible cases: $d_G(v, s) \leq 2^{j+1} - 2$ and $d_G(v, s) > 2^{j+1} - 2$. In the earlier one, as vertex s belongs to $I_j \cap B_G(v, 2^{j+1} - 2)$, it is the closest vertex to itself that holds this condition; i.e. $Par_2((v, j-1)) = \{(s, j)\}$. Subsequently, tree vertex (s, j) is the parent of $(v, j - 1)$. On the other hand, regarding the induction hypothesis, since $s \in I_j \cap B_G(s, 2^i - 1)$, vertex (s, j) is a vertex in subtree $T^s{}_{(s,i)}$. Consequently, $(v, j - 1)$ is also a vertex in $T^s{}_{(s,i)}$.

In the later case, let p denote the shortest path from v to s in G. We divide p to two subpaths p_1 and p_2 such that p_1 is a path from v to x such that $|p_1| = 2^j - 1$; consequently, p_2 is a path from x to s of length $|p| - (2^j - 1)$. Now, there are two possibilities: either $x \notin I_j$ or $x \in I_j$.

$\underline{x \notin I_j}$**:** Let's assume that there is no vertex $u \in B_G(x, 2^j - 1)$ such that $u \in I_j$. Henceforth, for every $g \in I_j : d_G(g, x) \geq 2^j$. This implies that $I_j \cup \{x\}$ is a 2^j-independent set. As far as $x \notin I_j$, $|I_j \cup \{x\}|$ is greater than $|I_j|$

which contradicts the assumption that $\boldsymbol{I_j}$ is a maximal $\boldsymbol{2^j}$-independent set. Subsequently, there is some graph vertex $\boldsymbol{u \in B_G(x, 2^j - 1) \cap I_j}$. Let $\boldsymbol{p_3}$ denote the shortest path from $\boldsymbol{u}$ to $\boldsymbol{x}$ in $\boldsymbol{G}$ ($\boldsymbol{|p_3| \leq 2^j - 1}$). This implies that $\boldsymbol{p_1 \cup p_3}$ is a path from $\boldsymbol{u}$ to $\boldsymbol{v}$ of length $\boldsymbol{|p_1| + |p_3|}$ which is not greater than $\boldsymbol{(2^j - 1) + (2^j - 1) = 2^{j+1} - 2}$. Hence, $\boldsymbol{u \in B_G(v, 2^{j+1} - 2) \cap I_j}$. Moreover, $\boldsymbol{p_2 \cup p_3}$ is a path from $\boldsymbol{u}$ to $\boldsymbol{s}$ of the following length $\boldsymbol{|p_2| + |p_3|}$. Consider the following inequalities:

$$\begin{aligned} |p_2| + |p_3| &\leq (|p| - 2^j + 1) + (2^j - 1) \\ &\leq |p| \\ &\leq 2^i - 1 \end{aligned}$$

This implies that $\boldsymbol{d_G(u, s) \leq 2^i - 1}$. Regarding the definition of $\boldsymbol{Par_2}$, since $\boldsymbol{u \in B_G(v, 2^{j+1} - 2) \cap I_j}$, and $\boldsymbol{d_G(u, s) \leq 2^i - 1}$, this is the case that:

$$\begin{aligned} \forall q \in Par_2((v, j-1)) : d_G(q, s) &\leq d(u, s) \\ &\leq 2^i - 1 \end{aligned}$$

As the result, the parent of $\boldsymbol{(v, j-1)}$ belongs to $\boldsymbol{B_G(s, 2^i - 1)}$. Consequently, according to the induction hypothesis, since the parent of $\boldsymbol{(v, j-1)}$ is a vertex in tree $\boldsymbol{T^s{}_{(s,i)}}$, vertex $\boldsymbol{(v, j-1)}$ also belongs to the vertex set of $\boldsymbol{T^s{}_{(s,i)}}$.

$\underline{\boldsymbol{x \in I_j}}$**:** As far as $\boldsymbol{d_G(v, x) \leq |p_1| = 2^j - 1}$, vertex $\boldsymbol{x}$ satisfies the following condition.

$$\boldsymbol{x \in I_j \cap B_G(v, 2^{j+1} - 2)} \tag{B.15}$$

Here, there are two possible cases. In the first one, $\boldsymbol{x}$ is the closest vertex to $\boldsymbol{s}$ that satisfies the above condition. Concerning the definition of $\boldsymbol{Par_2}$, $\boldsymbol{(x, j) \in Par_2((v, j-1))}$, and for every $\boldsymbol{(h, j) \in Par_2((v, j-1))}$, this is

the case that:

$$\begin{aligned} \boldsymbol{d_G(h,s)} &\leq \boldsymbol{d_G(x,s)} \\ &\leq \boldsymbol{2^j - 1} \\ &\leq \boldsymbol{2^i - 1.} \end{aligned}$$

As the result, we obtain the following proposition:

$$\forall (\boldsymbol{q,j}) \in \boldsymbol{Par_2((v,j-1))} : \boldsymbol{d_G(q,s)} \leq \boldsymbol{2^i - 1} \tag{B.16}$$

Hence, if $(\boldsymbol{z,j})$ denotes the parent of $(\boldsymbol{v,j-1})$ in tree $\boldsymbol{T^s}$, $\boldsymbol{d_G(z,s)}$ is not greater than $\boldsymbol{2^i - 1}$ which implies that $\boldsymbol{z} \in \boldsymbol{I_j} \cap \boldsymbol{B_G(s,2^i-1)}$. Consequently, regarding the induction hypothesis, $(\boldsymbol{z,j})$ is a vertex in $\boldsymbol{T^s_{(s,i)}}$; so, $(\boldsymbol{v,j-1})$ also belongs to vertex set of $\boldsymbol{T^s_{(s,i)}}$.

In the second case, $\boldsymbol{x}$ is not the closest vertex to $\boldsymbol{s}$ that satisfies the Proposition B.15. Again, according to the definition of $\boldsymbol{Par_2}$, Proposition B.16 is true. Using the same reasoning as we did for the first case, we conclude that $(\boldsymbol{v,j-1})$ is a vertex of $\boldsymbol{T^s_{(s,i)}}$.

Up to now, we have shown that $\forall \boldsymbol{j} \in \boldsymbol{[0,h]}$, if $\boldsymbol{v} \in \boldsymbol{I_j} \cap \boldsymbol{B_G(s,2^i-1)}$, $(\boldsymbol{v,j})$ will belong to the vertex set of $\boldsymbol{T^s_{(s,i)}}$. In the case that $\boldsymbol{j = 0}$, taking into consideration that $\boldsymbol{I_0 = V}$, we conclude that:

$$\begin{aligned} \boldsymbol{v} \in \boldsymbol{B_G(s,2^i-1)} &\rightarrow (\boldsymbol{v,0}) \text{ is a vertex in } \boldsymbol{T^s_{(s,i)}} \\ &\rightarrow \boldsymbol{v} \in \boldsymbol{\mathcal{L}_{T^s}(s,i)} \end{aligned}$$

Consequently, $\boldsymbol{B_G(s,2^i-1)} \subseteq \boldsymbol{\mathcal{L}_{T^s}(s,i)}$, and the proof is complete.

□

B.4 Hierarchical Independence Tree Type-0

In the previous sections, we have become familiar with two types of the hierarchical independence trees. They have some common properties and also some differences. In this section, we introduce the third type of HITs which is more efficient than the previous ones to be used in the versatile routing schemes that will be presented in the next chapter.

Previously, the hierarchical independent sequence of some source vertex in a connected graph was defined. Here, we define a special class of the HIS that will be used in defining the HIT type-0.

Definition B.4.1. *Let $\boldsymbol{G = (V, E)}$ denote a connected unweighted graph. Assuming distinguished vertex $\boldsymbol{s \in V}$ as the source vertex, sequence $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ is the "source-oriented hierarchical independent sequence" (source-oriented HIS), if:*

- $\boldsymbol{I_h = \{s\}}$
- $\boldsymbol{\forall i \in [0, h-1]\ \forall r \geq 2^i}$*:* $\boldsymbol{I_i \cap A_r}$ *is a maximal* $\boldsymbol{2^i}$*-independent set of subgraph* $\boldsymbol{G_{A_r}}$ *where* $\boldsymbol{A_r = B_G(s, r)}$.

Note that we will refer to the previous version of the hierarchical independence sequence as the *basic* HIS. In Definition B.4.1, set $\boldsymbol{I_{i+1}}$ is not necessarily a subset of $\boldsymbol{I_i}$ (for every $\boldsymbol{i < h - 1}$). This implies that every source-oriented HIS is not necessarily a basic one.

In the next chapter, it will be shown that for every connected graph and given source vertex, there exists a source-oriented HIS; however, like the basic HIS, there may exist more than one source-oriented HIS for a graph and a source.

The HIT type-0 is different from that of type-1 in two ways: it is defined based on the source-oriented HIS and it uses the parent function type-0 defined in the

following equation:

$$\boldsymbol{Par_0((v,i)) = \{(u,i+1)|u \in I_{i+1} \cap B_G(v,2^{i+1}-1) \wedge d_G(u,s) \leq d_G(v,s)\}} \tag{B.17}$$

for every $\boldsymbol{v \in I_i}$ and $\boldsymbol{i = 0,1,\ldots,h-2}$.

As you see, in the HIT type-0, instead of forcing the parent vertex be the closest one to the source, it only needs *not to be further* than its child from vertex $\boldsymbol{s}$.

Let $\boldsymbol{T^s}$ be an HIT type-0 of source-oreinted HIS $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ in graph $\boldsymbol{G = (V,E)}$. If $\boldsymbol{v}$ denotes an arbitrary member of $\boldsymbol{I_i}$, this is the case that:

i. For every vertex $\boldsymbol{v \in I_{i+1} \cap I_i}$, $\boldsymbol{(v,i+1) \in Par_0((v,i))}$.

ii. For every vertex $\boldsymbol{u \in \mathcal{L}_{T^s}(v,i)}$, $\boldsymbol{d_G(u,v) < 2^{i+1}}$.

iii. For every pair of vertices $\boldsymbol{u_1, u_2 \in \mathcal{L}_{T^s}(v,i)}$, $\boldsymbol{d_G(u_1,u_2) < 2^{i+2}}$.

iv. For every $\boldsymbol{i \in [0,h]}$, $\boldsymbol{B_G(s,2^i-1) \subseteq \mathcal{L}_{T^s}(s,i)}$.

where $\boldsymbol{\mathcal{L}_{T^s}(v,i) = \{l|(l,0) \in T^s{}_{(v,i)}.leaves\}}$. The first three properties of the HIT type-0 are common with the HIT type-1 and can be proved in the same way. Additionally, the last one is common with the HIT type-2 and will be proved in Lemma B.4.3.

The following lemma guarantees the existence of the HIT type-0 for any source-oriented HIS.

Lemma B.4.2. *For any source-oreinted HIS* $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ *of source vertex* $\boldsymbol{s}$ *in graph* $\boldsymbol{G = (V,E)}$*, there is a hierarchical independence tree type-0.*

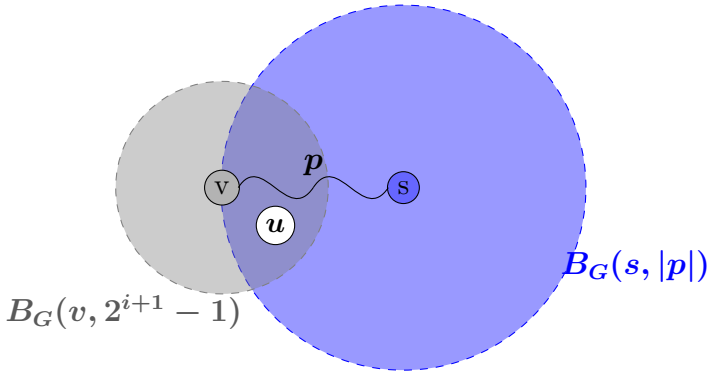
Proof. Like the HIT type-2, we only need to show that: $\boldsymbol{\forall v \in I_i : Par_0((v,i)) \neq \emptyset}$. To do this, we consider the following three cases:

- $v \in I_{i+1}$
- $v \in B_G(s, 2^{i+1} - 1)$
- $v \notin I_{i+1} \wedge v \notin B_G(s, 2^{i+1} - 1)$

If $v \in I_{i+1}$, regarding the first property of the HIT type-0, $(v, i+1) \in Par_0(v, i)$. Moreover, in the case that $v \in B_G(s, 2^{i+1} - 1)$, according to the definition of function Par_0, $s \in Par_0((v, i))$.

Now consider the case that $v \notin I_{i+1}$ and $v \notin B_G(s, 2^{i+1} - 1)$. If we prove the following proposition, we will conclude that $u \in Par_0((v, i))$ and the proof will be complete.

$$\exists u \in I_{i+1} : u \in B_G(v, 2^{i+1} - 1) \cap B_G(s, d_G(v, s)), \tag{B.18}$$



By contradiction, assume that Proposition B.18 is not true. Subsequently,

$$\forall u \in I_{i+1} : u \notin B_G(v, 2^{i+1} - 1) \cap B_G(s, d_G(v, s))$$

or equivalently,

$$\forall u \in I_{i+1} \cap B_G(s, d_G(v, s)) : u \notin B_G(v, 2^{i+1} - 1) \tag{B.19}$$

In addition, according to Definition B.4.1, $I_{i+1} \cap B_G(s, d_G(v, s))$ is a maximal 2^{i+1}-independent set in the subgraph of G induced by $B_G(s, d_G(v, s))$. We will refer to this subgraph as G'.

Proposition B.19 implies that all of the members of $\boldsymbol{I_{i+1}}$ in $\boldsymbol{G'}$ are so far from vertex $\boldsymbol{v}$ that if we add $\boldsymbol{v}$ to set $\boldsymbol{I_{i+1} \cap B_G(s, d_G(v, s))}$, the resulted set will remain a $\boldsymbol{2^{i+1}}$-independent set in $\boldsymbol{G'}$. Since vertex $\boldsymbol{v}$ doesn't belong to $\boldsymbol{I_{i+1} \cap B_G(s, d_G(v, s))}$, this is the case that:

$$\begin{aligned} |\{v\} \cup I_{i+1} \cap B_G(s, d_G(v, s))| &= |I_{i+1} \cap B_G(s, d_G(v, s))| + 1 \\ &> |I_{i+1} \cap B_G(s, d_G(v, s))| \end{aligned}$$

which contradicts our previous inference that $\boldsymbol{I_{i+1} \cap B_G(s, d_G(v, s))}$ is a maximal $\boldsymbol{2^{i+1}}$-independent set in $\boldsymbol{G'}$.

□

Finally, we show an important property of the HIT type-0 which is common with the HIT type-2.

Lemma B.4.3. *Let $\boldsymbol{T^s}$ denote an HIT type-0 of source-oriented HIS $\boldsymbol{\bar{I}_s = (I_0, I_1, \ldots, I_h)}$ in graph $\boldsymbol{G = (V, E)}$. For every $\boldsymbol{i \in [0, h]}$, this is the case that:*

$$\boldsymbol{B_G(s, 2^i - 1) \subseteq \mathcal{L}_{T^s}(s, i)}$$

Proof. To prove this property, like the proof of Lemma B.3.2, we first use the induction on $\boldsymbol{j}$ to show that:

$$\boldsymbol{\forall j \in [0, i]\ \forall v \in I_j \cap B_G(s, 2^i - 1) : (v, j)} \text{ is a vertex in subtree } \boldsymbol{T^s{}_{(s,i)}} \quad \text{(B.20)}$$

As the base case, since $\boldsymbol{I_i \cap B_G(s, 2^i - 1) = \{s\}}$, and $\boldsymbol{(s, i)}$ is the root of $\boldsymbol{T^s{}_{(s,i)}}$, we are done (note that for every vertex $\boldsymbol{q}$ in $\boldsymbol{I_i \cap B_G(s, 2^i - 1)}$, since $\boldsymbol{d_G(s, q) \leq 2^i - 1}$, graph vertices $\boldsymbol{q}$ and $\boldsymbol{s}$ cannot be in the same $\boldsymbol{2^i}$-independent set ($\boldsymbol{I_i}$) simultaneously unless $\boldsymbol{q = s}$).

Now, assume that for some $\boldsymbol{k \in [1, i]}$, for every vertex $\boldsymbol{y \in I_k \cap B_G(s, 2^i - 1)}$, $\boldsymbol{(y, k)}$ will be a vertex in $\boldsymbol{T^s{}_{(s,i)}}$ (induction hypothesis). Additionally, consider the arbitrary vertex $\boldsymbol{v}$ such that:

$$\boldsymbol{v \in I_{k-1} \cap B_G(s, 2^i - 1)}$$

As vertex $\boldsymbol{v \in I_{k-1}}$, there exists a tree vertex in the form $\boldsymbol{(v, k-1)}$. According to the definition of function $\boldsymbol{Par_0}$, if $\boldsymbol{(u, k)}$ denotes the parent of $\boldsymbol{(v, k-1)}$ in tree $\boldsymbol{T^s}$, this is the case that:

$$\boldsymbol{(u, k) \in Par_0((v, k-1)) \rightarrow}$$
$$\boldsymbol{u \in I_k \wedge u \in B_G(v, 2^k - 1) \wedge d_G(u, s) \leq d_G(v, s)}$$

In addition, since vertex $\boldsymbol{v \in B_G(s, 2^i - 1)}$, we obtain the following inequality:

$$\boldsymbol{d_G(v, s) \leq 2^i - 1}$$

Moreover, as far as $\boldsymbol{d_G(u, s) \leq d_G(v, s)}$, this is the case that:

$$\boldsymbol{d_G(u, s) \leq 2^i - 1 \rightarrow}$$
$$\boldsymbol{u \in B_G(s, 2^i - 1)}$$

This implies that $\boldsymbol{u \in I_k \cap B_G(s, 2^i - 1}$. As the result, regarding the induction hypothesis, $\boldsymbol{(u, k)}$ is a vertex in subtree $\boldsymbol{T^s{}_{(s,i)}}$. Since $\boldsymbol{(u, k)}$ is parent of $\boldsymbol{(v, k-1)}$, vertex $\boldsymbol{(v, k-1)}$ is also belongs to the vertex set of subtree $\boldsymbol{T^s{}_{(s,i)}}$.

Up to now, we have proved the Proposition B.20 for every $\boldsymbol{j \leq i}$. Consider the case that $\boldsymbol{j = 0}$, we obtain the following proposition:

$$\boldsymbol{\forall v \in I_0 \cap B_G(s, 2^i - 1) : (v, 0)} \text{ is a vertex in subtree } \boldsymbol{T^s{}_{(s,i)}}$$

or equivalently ($\boldsymbol{I_0 = V}$),

$$\boldsymbol{\forall v \in B_G(s, 2^i - 1) : (v, 0)} \text{ is a vertex in subtree } \boldsymbol{T^s{}_{(s,i)}}$$

Tree	Problem Type	Approach	Hierarchical Sequence	Partition	Child-Par. Distance	Par. Func.
HDT[a]	Mul. Source	Top-Down	HDS	$\mathbf{2^i}$-Partition	N/A	N/A
HIT Type-0[b]	Single Source	Bottom-Up	Source -Oriented HIS	$\mathbf{2^{i+2}}$-Partition	$\leq \mathbf{2^{i+1}-1}$	$\boldsymbol{Par_0}$
HIT Type-1	Single Source	Bottom-Up	Basic HIS	$\mathbf{2^{i+2}}$-Partition	$\leq \mathbf{2^{i+1}-1}$	$\boldsymbol{Par_1}$
HIT Type-2[c]	Single Source	Bottom-Up	Basic HIS	$\mathbf{2^{i+3}}$-Partition	$\leq \mathbf{2^{i+2}-2}$	$\boldsymbol{Par_2}$

[a]Presented by Gupta et. al. in 2006
[b]Our tree
[c]Presented by Srini, Busch, and Iyengar in 2012.

Table B.1: Comparison of different types of hierarchical routing trees.

which implies that:

$$\boldsymbol{B_G(s, 2^i - 1) \subseteq \mathcal{L}_{T^s}(s, i)}$$

□

Table B.1 shows a brief comparison of the three types of HITs and the HDT presented in this Appendix.

APPENDIX C

HOURLY METERED LOAD DATA OF PJM

PJM interconnection is a regional transmission organization (RTO) that manages the electricity in the following regions in the United States: Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, New Jersey, North Carolina, Ohio, Pennsylvania, Tennessee, Virginia, West Virginia and the District of Columbia [1]. In some of the mentioned regions, PJM is coordinating the wholesale electricity for the whole region. However, for other areas, it only serves a part of the region. Its main task is to provide reliable electricity for more than 61 million customers. Fig. C.1 represents the regions covered by PJM interconnection. Tables V and VI represent the historical daily load for the first and second six months of 2015 in the regions covered by PJM interconnection, respectively.

[1]Available Online: http://pjm.com/about-pjm/who-we-are.aspx

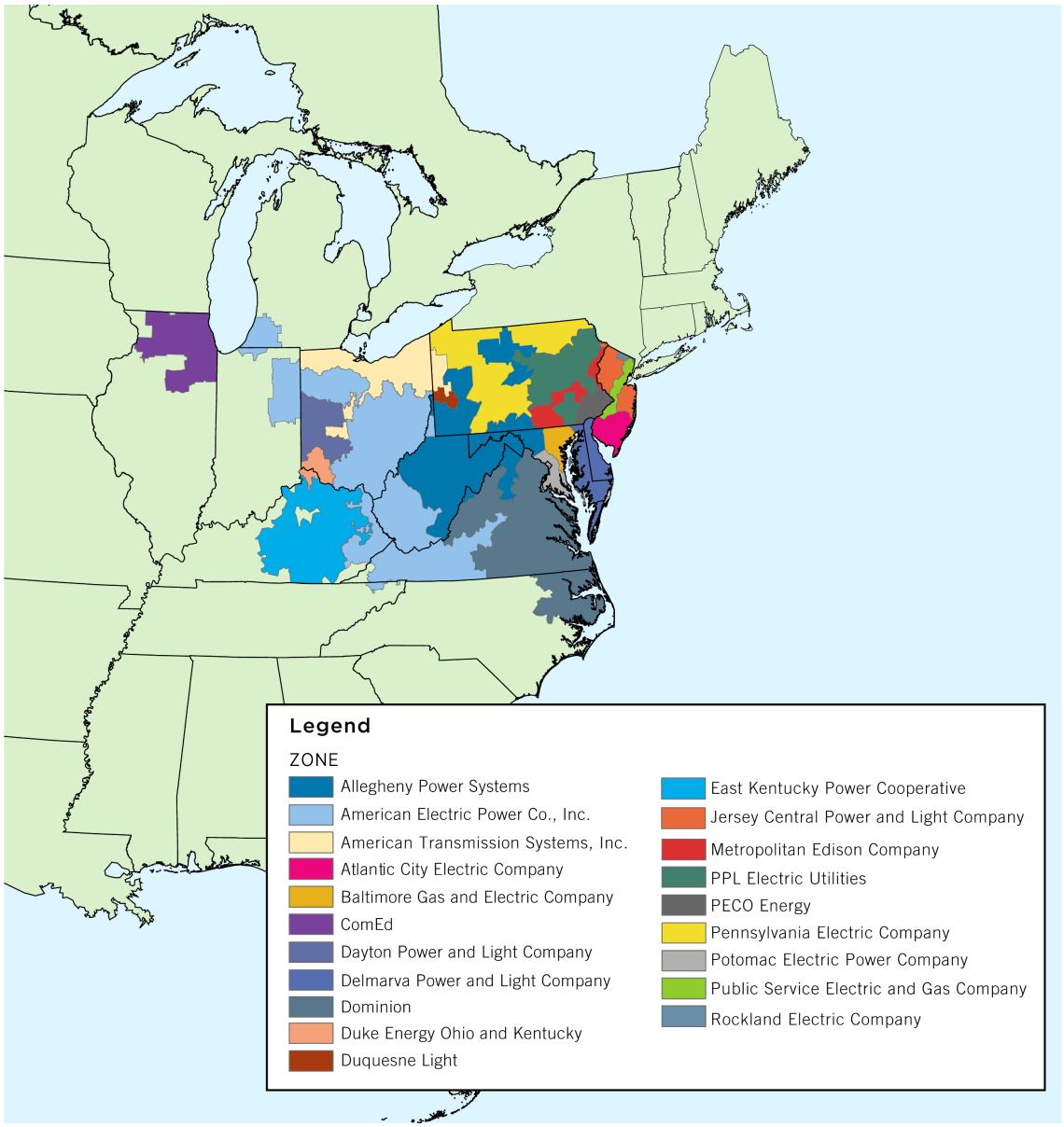


Figure C.1: Geographical regions covered by PJM interconnection.

Table C.1: The daily load in the first six months of 2015 in PJM. The numbers are shown in kW.

Day	January	February	March	April	May	June
1	2,224,247.0	2,287,549.2	2,442,177.8	2,024,486.3	1,847,056.1	2,101,982.5
2	2,214,133.4	2,467,346.6	2,437,075.9	1,974,545.2	1,681,862.1	1,943,814.3
3	2,146,834.0	2,619,797.2	2,529,970.7	1,807,215.0	1,657,509.5	1,922,430.1
4	1,974,231.4	2,445,748.3	2,351,650.9	1,730,857.6	1,944,146.7	2,008,393.8
5	2,445,435.6	2,571,644.5	2,564,235.5	1,692,526.7	2,016,051.7	2,033,350.8
6	2,652,063.8	2,681,963.0	2,705,979.7	1,898,950.8	2,030,419.7	1,871,852.8
7	2,843,836.9	2,243,782.4	2,369,589.7	1,922,330.2	2,067,433.1	1,868,774.4
8	3,023,713.9	2,025,259.8	1,954,528.4	1,970,665.4	2,131,769.8	2,238,112.9
9	2,736,306.6	2,347,400.7	2,139,678.0	2,028,948.3	1,927,307.7	2,270,840.2
10	2,679,408.0	2,504,609.6	2,165,532.0	1,943,293.5	1,931,900.8	2,382,590.1
11	2,463,715.8	2,459,178.1	2,055,011.6	1,718,914.8	2,239,332.3	2,542,460.0
12	2,461,286.0	2,547,267.4	2,090,851.4	1,688,730.1	2,169,028.7	2,578,623.9
13	2,561,561.2	2,755,521.6	2,127,633.3	1,886,606.5	1,922,147.6	2,352,410.3
14	2,690,702.6	2,543,273.5	1,932,114.2	1,879,084.5	1,890,764.5	2,353,527.6
15	2,547,062.4	2,809,745.1	1,884,176.9	1,882,273.2	1,928,452.4	2,621,394.3
16	2,453,213.0	3,010,474.6	2,060,271.8	1,884,624.7	1,904,483.3	2,590,952.4
17	2,304,689.7	2,802,988.6	2,017,453.2	1,876,043.3	2,001,731.2	2,332,716.6
18	2,159,352.6	2,803,212.0	2,184,042.3	1,705,664.2	2,281,909.0	2,393,511.2
19	2,297,892.3	3,052,242.2	2,190,832.1	1,629,643.8	2,160,130.8	2,338,444.6
20	2,345,274.7	3,104,390.6	2,226,302.0	1,901,434.6	1,952,688.8	2,161,401.8
21	2,412,450.4	2,720,441.3	1,958,744.1	1,879,815.0	1,881,833.7	2,312,095.6
22	2,416,012.7	2,306,819.3	1,896,000.7	1,926,903.3	1,847,969.6	2,638,997.9
23	2,399,030.1	2,670,971.2	2,203,770.5	1,974,713.6	1,665,668.7	2,694,198.8
24	2,247,946.0	2,868,672.0	2,220,515.3	1,947,838.4	1,701,091.1	2,399,544.4
25	2,152,295.5	2,609,601.0	2,179,801.4	1,804,096.3	1,910,615.1	2,307,415.4
26	2,492,879.1	2,609,484.5	2,066,330.6	1,707,445.2	2,285,864.7	2,234,547.0
27	2,571,906.4	2,640,380.4	2,137,934.4	1,919,708.2	2,350,697.4	1,929,125.4
28	2,607,071.1	2,539,066.3	2,161,144.5	1,902,106.7	2,359,577.4	1,851,272.4
29	2,562,531.5	–	2,101,593.8	1,882,956.0	2,336,808.4	2,087,940.5
30	2,481,083.7	–	2,093,609.3	1,867,538.4	2,178,553.2	2,237,095.2
31	2,484,570.9	–	2,048,286.0	–	2,056,848.5	–

Table C.2: The daily load in the last six months of 2015 in PJM. The numbers are shown in kW.

Day	July	August	September	October	November	December
1	2,235,981.1	2,326,506.7	2,675,402.2	1,871,454.0	1,625,687.1	2,089,866.7
2	2,098,594.8	2,326,394.3	2,686,777.1	1,848,997.8	1,900,526.7	2,076,625.0
3	1,947,908.6	2,560,275.1	2,719,161.1	1,756,196.2	1,932,687.6	2,161,429.0
4	1,897,211.1	2,574,999.1	2,599,482.7	1,674,459.5	1,925,894.8	2,175,708.5
5	2,000,731.8	2,498,275.3	2,280,105.6	1,869,228.5	1,949,722.0	2,055,484.6
6	2,428,637.2	2,325,697.5	2,215,356.6	1,897,800.1	1,931,240.4	2,013,747.7
7	2,523,706.6	2,284,944.6	2,375,704.9	1,914,453.2	1,739,918.6	2,173,588.0
8	2,399,394.5	2,172,007.5	2,679,940.3	1,930,268.7	1,746,607.9	2,164,163.3
9	2,428,976.7	2,169,595.7	2,630,928.8	1,918,961.5	2,026,658.3	2,168,256.1
10	2,366,755.8	2,391,373.9	2,366,200.6	1,667,591.6	1,969,193.8	2,065,124.2
11	2,169,871.9	2,437,264.5	2,193,611.3	1,641,818.7	1,945,546.9	1,995,847.2
12	2,157,110.2	2,334,476.4	1,888,958.6	1,874,666.6	1,970,728.4	1,797,150.0
13	2,442,000.9	2,317,973.4	1,750,362.8	1,892,341.3	1,951,856.9	1,746,418.1
14	2,495,718.8	2,400,619.0	1,954,813.6	1,866,502.2	1,858,388.4	1,963,965.5
15	2,338,597.1	2,336,837.2	2,081,851.1	1,873,318.1	1,800,851.9	2,004,113.1
16	2,221,536.9	2,409,186.7	2,178,468.4	1,844,691.9	1,964,457.0	2,048,928.6
17	2,404,239.5	2,698,831.6	2,240,706.4	1,750,889.6	1,961,147.6	2,087,605.0
18	2,478,797.6	2,636,636.9	2,230,933.6	1,791,688.1	1,941,140.4	2,187,556.3
19	2,589,602.2	2,634,215.7	2,039,646.9	2,015,386.1	1,928,128.6	2,176,180.8
20	2,824,010.1	2,497,295.3	1,847,147.5	1,952,358.9	1,975,289.0	2,107,844.3
21	2,675,592.0	2,347,128.5	1,964,309.2	1,917,727.2	1,927,928.7	2,155,731.4
22	2,412,955.5	2,094,643.9	2,006,255.5	1,903,754.3	1,952,532.7	2,011,890.0
23	2,379,320.3	2,079,893.8	2,036,401.2	1,849,517.5	2,237,398.5	1,958,759.2
24	2,408,843.6	2,363,988.9	2,043,572.7	1,718,982.9	2,181,561.0	1,767,741.6
25	2,352,382.1	2,274,723.0	1,980,467.2	1,657,375.6	2,097,915.4	1,669,885.1
26	2,361,044.4	2,128,817.6	1,767,919.0	1,895,559.4	1,745,687.3	1,726,500.0
27	2,638,585.8	2,121,712.1	1,752,623.6	1,963,221.4	1,745,832.1	1,743,846.6
28	2,770,611.5	2,141,847.0	2,119,168.7	1,953,809.3	1,758,048.5	2,052,214.7
29	2,813,593.6	2,086,999.4	2,172,743.8	1,913,205.3	1,820,186.8	2,028,147.0
30	2,725,597.2	2,190,522.9	2,073,200.0	1,902,573.2	2,115,151.5	2,011,894.9
31	2,562,081.4	2,561,310.6	–	1,781,476.8	–	1,968,810.2

VITA

KIANOUSH GHOLAMIBOROUJENI

September 23, 1989	Born, Boroojen, Iran
2012	B.S., Computer Engineering University of Tehran Tehran, Iran
2016	M.S., Computer Science Florida International University Miami, Florida

- 2016: Dissertation Year Fellowship Award, Florida International University.
- 2016: Best Graduate Student Research Award, School of Computing and Information Sciences, Florida International University.
- 2014: Best Paper Award for the paper entitled "A Hybrid Model for Forecasting Power and Demand in Smart Grids" in Eighth International Conference on Computer Communication Networks, Elsevier, July 2014.
- 2012: Graduate Assistantship, School of Computing and Information Sciences, Florida International University
- 2011: 1st place in RoboCup competition among all the students of Department of ECE, University of Tehran, Tehran, IR.

Kianoush is a PhD candidate in computer science, seasoned data scientist with 8 years of experience in creating statistical and learning models for use in analytics solutions. Special project work in industries of energy, networking, cloud networks, banking & finance, insurance software, entertainment and hospitality. He is an award winning author with multiple books & journal articles published on analytics. Kianoush is a pioneer of new methods for performing analytics on previously untested applications resulting in significant gains over existing solutions. This includes identifying areas for improvement, process re-engineering, and corporate growth. Expert in R, MATLAB, Hadoop, SAS, IMPLAN, Octave, Stata, SQL, and languages C, C++, Java, Python, MATLAB, C#.

PUBLICATIONS AND PRESENTATIONS

Three authored/co-authored books published by MIT Press and Springer, 17 peer-reviewed articles in the form of journal papers or book chapters, six paper presentations in major IEEE conferences and seven articles in conference proceedings. Here is the list of selected publications:

- Iyengar, Sundararaja S., and Kianoosh G. Boroojeni. Oblivious Network Routing: Algorithms and Applications. MIT Press, 2015.
- Iyengar, Sundararaja S., Kianoosh G. Boroojeni, N. Balakrishnan. Mathematical Theory of Distributed Sensor Networks. Springer, 2014.
- Boroojeni, Kianoosh G., M. Hadi Amini, S. S. Iyengar. Smart Grids: Security and Privacy Issues. Springer, 2017.
- Boroojeni, Kianoosh G., M. H. Amini, S. Bahrami, S. S. Iyengar, A. I. Sarwat, O. Karabasoglu. "A Novel Multi-Time-Scale Modeling for Electric Power Demand Forecasting: from Short-Term to Medium-Term Horizon." Electric Power System Research 142(2017): 58-73.
- Boroojeni, Kianoosh G., M. H. Amini, A. Nejadpak, T. Dragicevic, S. S. Iyengar, F. Blaabjerg. "A Novel Cloud-based Platform for Implementation of Oblivious Power Routing for Clusters of Microgrids." IEEE Acess 5(2016): 607-619.
- Boroojeni, Kianoosh G., M. H. Amini, S. S. Iyengar, M. Rahmani, P. M. Pardalos. "An Economic Dispatch Algorithm for Congestion Management of Smart Power Networks: An Oblivious Routing Approach." Energy Systems 7(2016): 1-25.
- Boroojeni Kianoosh G., S. Mokhtari, M. H. Amini, S. S. Iyengar. "Optimal Two-Tier Forecasting Power Generation Model in Smart Grids," International Journal of Information Processing 8(2014): 79-88.