Algorithms for Topology Discovery in Synchronous Optical Networks

Ali Muhammad

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

March 2019

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ali Muhammad**

Entitled: **Algorithms for Topology Discovery in Synchronous Optical Networks**

and submitted in partial fulfillment of the requirements for the degree of

Master of Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Gregory Butler

_____ Examiner
Dr. Denis Pankratov

_____ Examiner
Dr. Thomas Fevens

_____ Supervisor
Dr. Brigitte Jaumard

Approved by _____
Lata Narayanan, Chair
Department of Computer Science and Software Engineering

_____ 2019

_____
Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Algorithms for Topology Discovery in Synchronous Optical Networks

Ali Muhammad

Telecommunication networks are comprised of interconnected network elements which provide communication services to end users. The map of port-to-port connectivity of these network elements is referred to as the network topology. These networks undergo frequent changes in their topology as new fiber optic links, nodes and circuits are regularly provisioned and removed. One of the hurdles for network operators is to obtain complete network connectivity maps or topology. Topology discovery of legacy optical networks namely Synchronous Optical NETworks (SONET), though currently a challenge for network operators, has not been studied much in the literature. We have investigated two problems namely topology discovery and circuit stitching considering the missing and incorrect network provisioning information. We modelled the problem as a weighted matching graph problem and significantly improved the topology discovery computation time. The proposed algorithms have been implemented and evaluated on data sets of customers of Ciena Corporation.

Acknowledgments

I wish to express my sincere gratitude to my thesis advisor, Prof. Brigitte Jaumard. Her wide knowledge on the subject and constructive comments have been a great value for me. I have profoundly benefited from her expert guidance and professional mentorship.

I deeply thank my mother, late father and siblings for their unconditional trust and love. This journey would not have been possible without their support.

I would like to thank my spouse Jabeen Zehra who has selflessly supported my academic commitment. Her constant encouragement motivated me to stick to my goal.

I cannot neglect to acknowledge my dear friend Hamed Abdzadeh Ziabari who always showed up whenever I needed a friend and reminded me to trust in the process.

My acknowledgement would be incomplete without thanking my sister Kanwal Masroor and brother-in-law Masroor Abbas for their continuous support, love and care throughout my studies here in Canada.

I would like to dedicate this thesis to my younger sister late Saira Naqvi who left us at a very young age. She had a strong desire for Master's degree. Saira, you will always be missed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Telecommunication industry is currently going through a big revolution as the demands for services, such as big data, cloud services, video traffic, etc., have risen excessively. In order to satisfy these ever-growing traffic demands and to catch up with technological advances, network operators are either planning to migrate to new technologies or are in the process of optimizing their existing networks. One of the biggest challenges faced by network operators is to acquire the up-to-date information about their network's connectivity that is mandatory for the network migration and optimization tasks.

Communication networks consist of multiple elements to transport user traffic (voice, video or data) from one end point to another. The distance between two endpoints may range from few hundreds to thousands of kilometers. Therefore it needs multiple intermediate elements to make overall communication happen. Each Network Element (NE) is composed of physical termination points or ports that connect with adjacent network elements via an optical fiber cable. A map of port-to-port connectivity of these network elements is termed as network topology. This port level connectivity is achieved through

different connections within these network elements. Figure 1.1 represents a network topology with five network elements. The ports of different elements are connected through fiber optic cables that are shown as dark blue lines.

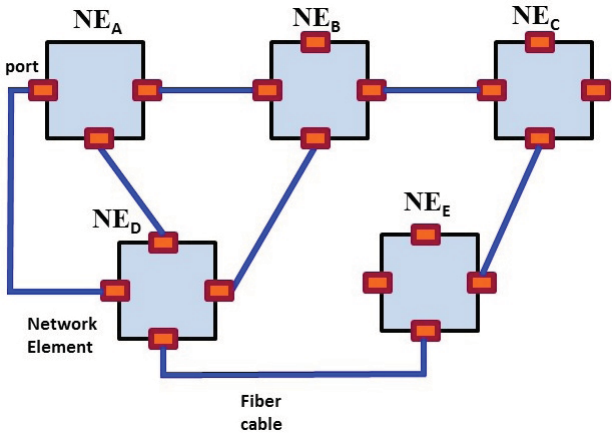

Figure 1.1: Network Topology

Topology discovery problem [31] is defined as acquiring and maintaining the existing information of network nodes and their connections, and also building a topological graph in order to have complete network connectivity.

The big question that arises is: why do the network operators not have topology of their own networks? The answer may be surprising: with continuous mergers and acquisitions, accompanying the trend of multi-vendor network environments, the connectivity maps of network get outdated. The only way to avoid this outdated information is by updating the connectivity information each time a change is made in the network. The traditional ways to handle these changes include manual tracing, using spreadsheets, manual updates to database [36], etc. However, all these methods are prone to errors due to human interventions and additionally are labor intensive. This brings a dire need for an automated solution to discover the network connectivity map. Changes in networks are frequent. Each time a network element is added or removed, a number of connections are affected. Keeping up-to-date knowledge of network connectivity is a big challenge for network operators. Generally, the topology discovery problem can be addressed in any network, but this problem

is of special interest for legacy optical networks since they have no means to automatically detect their network topology. The manual discovery of topology in these networks has been a tedious task since tracing each and every endpoint of fiber is not only labor intensive but also prone to errors. Therefore, means are required to provide automated discovery of network topology by using efficient tools. The main challenges within topology discovery are: the presence of multi-vendor equipment, the non-centralized network monitoring systems, the non-availability of network diagrams, and the existence of nodes that do not provide any information to be used for topology discovery.

1.2 Thesis Contributions

The objective of this thesis is to develop tools to discover the topology of SONETs (Synchronous Optical NETwork). We have investigated two different problems by proposing efficient algorithms. The data used in this study is obtained from customer networks of Ciena Corporation.

The first contribution consists of algorithms for *Network Connectivity Problem* which provide a solution to topology discovery problem of a SONET at the physical layer. It resulted in a Conference paper ”Topology Discovery of Synchronous Optical Networks” published in IEEE International Conference on Computing, Networking and Communications (ICNC) 2017 [22]. An extended version of this paper will be submitted to an international journal soon. The work on topology discovery resulted in a commercial tool used by Ciena Corporation (see Topology Discovery - Gaining Insight to Your Network [8])

The second contribution is to solve the *Circuit Stitching Problem* with an objective to develop efficient algorithms in order to stitch the circuits at different layers. Stitching circuits means identifying the two endpoints of a connection as well as its complete path.

To the best of our knowledge, there are no algorithms that were able to solve the topology discovery problem for SONETs when there is erroneous/missing information.

1.3 Statements of the Problems

Topology Discovery Problem: We modelled SONET as a graph $G = (V, E)$ where V is the set of ports and E is the set of edges between the ports. We define port signature as a unique ID of a port that consists of four parameters. The parameters may be either user provisioned or system generated. One or more of the parameters may be missing and may contain erroneous information. S_{ij} is a similarity measure between the signature of two ports. L_{ij} is the label that represents the number of signature elements used in calculating S_{ij}. We define topology discovery as finding the unique pairing of fiber end points or ports while considering the similarity value and labels of port signatures. The input to topology discovery problem is a set of ports and output is the ports pairings each with a confidence value. The confidence value is the measure of reliability of solution that ranges [0,1].

Circuit Stitching Problem: Circuit stitching problem consists of building circuits that identify the starting and ending point of a connection along with its complete path. A circuit is a set of alternating cross connection and topological links. Topological links are the connections between ports of two different network elements connected through optical fiber cable while cross connection links are the connections between ports of same network element. We define circuit stitching problem as tracing circuits given topological and cross connection links as an input. The output of circuit stitching problem is a set of alternating topological and cross connection links.

1.4 Organization of Thesis

This thesis is organized as follows. Chapter 2 presents a literature review on related subjects and describes the research on topology discovery for SONET networks. It continues with

studies of topology discovery in other networks to provide the reader with an overview of this problem in other domains. Afterwards, the literature on classical graph matching and string matching algorithms has been presented. Chapter 3 presents the work on topology discovery of SONET networks. In Chapter 4, we discussed the circuit stitching problem along with its applications to improve network topology. Chapter 5 presents the conclusions that were made in this thesis along with the future line of research.

Chapter 2

Literature Review

There have been several researches about topology discovery in different network domains. In this section, we give a brief account of state of the art for topology discovery. Our main focus is to identify parameters that have been proposed by researchers to discover topology in Synchronous Optical Network (SONET) and Internet Protocol (IP) network domains.

2.1 SONETs

While it poses significant challenges for legacy networks like SONET, the topology discovery problem has not been thoroughly studied in the literature till date. The available work, in form of patents, generally discusses the parameters which could be used to discover node adjacency. However, how to use the parameters in a multi-vendor network environment where one network element may not support the same parameters being used at other network element, and where the provisioning information is sometimes incorrect or even missing, is not discussed.

One of the most relevant papers is Voigt *et al.* [36]. The authors proposed a method for determining network topology by performing iterative validation tests on each network port. A list of network elements is known in advance to decide which validation tests are

required for a network element. The network topology is deduced based on the response received at other ports on other network elements as a result of the validation tests. The validation tests are of two types; network-affecting tests and non-network-affecting tests. The network-affecting tests are not a point of interest for this thesis since it is nearly impossible to perform those tests on live networks as they affect the user traffic in terms of network downtime. The non-network-affecting tests include section trace validation, synchronous status message testing and connect tests where a parity bit is changed that generates an alarm at far-end network element to determine the connectivity. The authors also talked about the Synchronous Transport Signal (STS) mapping, which is a technique that matches the STS mapping of two ports in order to reduce the number of validation tests as the two connected ports should ideally have the same STS mapping.

There are many differences in the proposal made by the authors and our strategy. Firstly, the proposed strategy takes a list of network elements and their types in advance to perform the relative validation tests which is not readily available in real life. Even if you know that several kinds of network elements may support one parameter, it is not guaranteed that the parameter will always be enabled. For example: Section Trace (ST) is a parameter in SONET frame that is transmitted through J0 bytes. We can determine if two ports are connected by comparing the transmit and receive ST value. However, it has been observed from data set that this parameter is not always enabled on each port of a network element. Therefore, we can not rely on any single test. Secondly, the authors did not take into account the case of provisioning errors which are very common. For example, two network ports may not have identical STS mapping only due to one bit difference which may simply be a provisioning error. Thirdly, in large networks with hundred thousands of ports, the connectivity test performed at each single port and observing the response on a single port out of the remaining thousands of ports would involve a lot of time. Finally, there is no discussion about the ports that have missing information.

Most of the existing research related to the topology discovery accounts for the use of SONET overhead bytes to transmit unique identifiers from one port and receiving them on the other end port to deduce the network connectivity. We will briefly discuss some of those studies.

Olivia *et al.* [30] proposed transmitting a unique identification of each node and port using the overhead data channels, also termed as topology trace channels. The authors proposed that the J0 and J1 bytes of SONET frame can be repeatedly transmitted from source to destination ports and that by matching the information contained in these bytes, the network management system would be able to form a topology map of the whole network.

Barker *et al.* [2] proposed a method of sharing two sets of information to discover the neighboring nodes. The first piece of information is the unique key (section trace identifier) associated with each port. The second set of information comprises the identity of a node along with the identity of all the ports associated with this node and their section trace values. This second information set is then shared between nodes via the communication link. Then, whenever a node receives a packet on any of its ports, it looks inside the section trace identifier received, and maps it to the information table to determine the adjacent node. The technique suggested by the authors requires each node to have some processing means to compare the received information with the already built information table to deduce the topology.

Wallace *et al.* [38] proposed transmission of a unique string between the nodes to infer the topology. In the proposed strategy, each node had to report the transmitted and received unique value (termed as signature) to a network manager. The network manager correlates the data received from each node to build the network connectivity diagram. The information that this unique signature holds includes the system ID of the node and the universal port identifier that contains information of shelf, sub-shelf, slot and sub-slot.

Das *et al.* [9] studied the topology discovery problem in SONET rings and has proposed

the parameters to be used inside the Link State Advertisements (LSA) messages in MPLS-enabled SONETs to discover the adjacent links. The information to be placed in link state tables include, the topology supported by link (i.e., UPSR, BLSR, 1+1 protection, etc.), the state of link (i.e., whether working or protection), the Ring-ID, and the value as East or West. This work is an extension to the study on UPSR presented by Sharma *et al.* [33] and extends the protocol to 2F BLSR and 4F BLSR.

Although all aforementioned papers are concerned with topology discovery, none of them approached the topology discovery algorithms when the input data has missing or incorrect provisioning information. Moreover, to the best of our knowledge, none of the research talked about the reliability of the solution for the legacy SONETs.

2.2 IP Networks

IP networks can be considered as the fastest growing networks in the recent past decades. They have grown from a small experimental research network to a complex network of routers, switches, and host nodes. Therefore, understanding the topology of large scale IP networks is very significant in order to carry out the architectural design decisions [37]. In this section we discuss the literature on topology discovery of IP networks.

Zhao *et al.* [41] classified the topology discovery methods for IP networks into proactive and passive methods. The proactive methods are defined as those where topology is achieved by sending probes to the network and watching the response to deduce the topology. The passive methods define the means of setting a monitor to record the network's data that is later used to analyze the topology. The authors have presented a comparison of the methods based on protocols such as Internet Control Message Protocol (ICMP), Address Resolution Protocol (ARP), Open Shortest Path First (OSPF) and Simple Network Management Protocol (SNMP) and concluded that SNMP outperforms the rest of the protocols based on the speed, accuracy and the burden it poses on the network.

Breitbart *et al.* [3] have presented the physical topology discovery in heterogeneous IP networks. The proposed algorithms utilize the information from address forwarding tables of the network elements that capture the Medium Access Control (MAC) addresses to discover the data link layer topology. This information is routinely collected in the SNMP MIBs of router and switches.

Zhang *et al.* [40] explained the utilization of SNMP protocol to discover the topology of passive optical networks. The authors presented the discovery process by sending SNMP-get messages from Network Management System (NMS) to each device that has already been configured by an IP address. The response of this message in adequate time determines the discovery of a device.

Siamwalla *et al.* [34] proposed various algorithms for IP layer topology discovery. The first proposed algorithm is based on SNMP. The neighboring routers are found through router's IP route table and the hosts are discovered through the table entries of ARP. ARP is a data link layer protocol used to map the IP addresses of the devices to their hardware addresses. The second algorithm is based on broadcast ping message and DNS (Domain Name System) transfer zone. A DNS server keeps a list of all the hosts in its domain along with their IP addresses. This algorithm deals with getting the DNS list of all hosts and then verifying their validity through broadcast ping message. The third algorithm validates the host nodes list obtained from DNS transfer zone through traceroute commands.

Kracht [24] proposed OSI layer 2 and layer 3 discovery. The author proposed a three step process. The first step is to determine the address range for devices (this address range is normally entered by the system administrator). The second step is to identify addresses that are linked with devices (this involves making sub-groups of addresses and assigning each subgroup to a thread or process which randomly pings each address that it contains in the list, to all the devices, until the device corresponding to that address responds). The third step is to identify the type of device (this is done by using SNMP protocol MIB's that

are requested by each device). The final step is to gather the configuration information of the attached devices. This is done through SNMP requests where the received information contains the IP address and the interfaces that are associated with a device. The author also discusses hidden devices termed as ”Black devices” that are the part of network but may not be supporting the discovery protocol, e.g., SNMP protocol that is used to query information and hence the devices remained unidentified or hidden. The proposed discovery mechanism can infer the location of some black devices by using the information reported by other neighboring devices. i.e., if three or more devices report multiple neighboring devices on the same port, the discovery mechanism will infer that the devices are actually linked through a black cloud device.

The above-mentioned references for IP networks discuss the different protocols for topology discovery and the last given reference [24] talked about the discovery of missing nodes location. In previous sections, we talked about the topology discovery at physical layer. However, the information at other layers, like IP layer, can be used to improve the solution obtained from the single layer topology discovery, e.g. using Data Communication Networks (DCN).

2.3 Classical Algorithms

In literature, there are many studies that are directly related to the topology discovery problem. The two most relevant tools are the classical graph algorithms and the string matching algorithms. The graph algorithms can solve a variety of network connectivity problems, e.g., the weighted matching problem, the shortest path problem, the network flow problem, and others. The string matching algorithms can be used to solve records de-duplication and data cleansing problem. In this section, we will give a brief introduction to these two types of classical algorithms with some literature review.

2.3.1 Graph Algorithms

A graph G is defined as a finite set of elements and is characterized as directed or undirected. A directed graph $G = (V, E)$ is defined as a set V whose elements are called vertices (or nodes) and a set E whose elements $e \in E$ are ordered pairs of vertices, called arcs [20]. An undirected graph is same as directed graph except that the arcs are undirected, and called edges. In subsequent section, we will briefly discuss the matching problem in undirected graph.

2.3.2 Graph Matching Problem

A matching M in an undirected graph $G = (V, E)$ is a set of pairwise non-adjacent edges. **Graph Theory Terminology:** In this thesis, we will focus on weighted matching problem in mesh graphs. Figure 2.1 depicts a graph having four vertices (A-D) and five edges. Figure 2.2 represents an example of maximal matching (edge between A-D belongs to M). A matching M is maximal, if addition of any new edge that is not in currently in M will render the M no longer a matching. Figure 2.3 shows a maximum matching (involving all vertices from A to D). A matching M is said to be maximum if that contains the largest possible number of edges. A matching M is said to be perfect if it involves all the vertices of the graph. The matching M shown in Figure 2.3 is also a perfect matching. Figure 2.4 represents a weighted matching. The weight of matching is the sum of the weights of edges in the matching. There can be more than one matching in a graph. There are two matching in this example. Matching 1 has a weight of 14 and Matching 2 has a weight of 11. If the objective is to find the matching with maximum weight, Matching 1 will be the required solution.

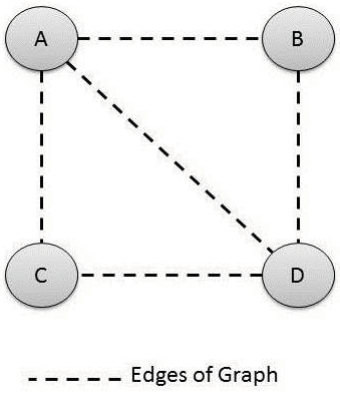

Figure 2.1: An Example of a Graph

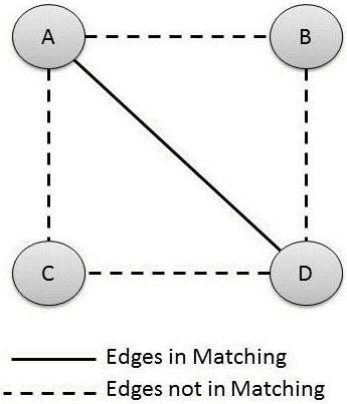

Figure 2.2: Maximal Matching

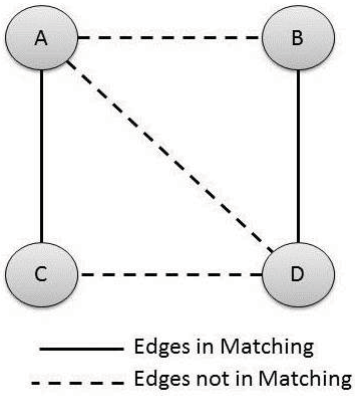

Figure 2.3: Maximum Matching

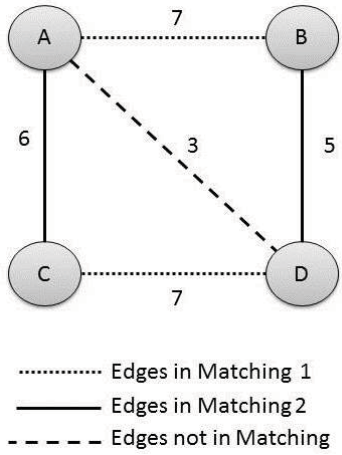

Figure 2.4: Weighted Matching

We aim to find maximum weighted matching in general graphs. The maximum weighted matching is constructed by iteratively adding edges to an initially empty matching M along augmenting paths in the graph. A path P is said to be *alternating* (with respect to M) such that every other edge on this path belongs to the matching. An *augmenting path* (with respect to M) is an alternating path that starts and ends at two distinct exposed vertices [18]. A vertex *v* of a graph is said to be *exposed* or *isolated* if no edge of M (matching) is incident with *v*. An edge is *matched* if it is in M and unmatched otherwise.

2.3.3 Literature on Matching Algorithms

Matching has been widely studied in the case of bipartite graphs (can be studied in references [11], [1], [13]), but less in the case of general graphs. In the context of this thesis, we are interested in the latter case, and therefore will only discuss the matching for general graphs. We will discuss some previous research on maximum matching and weighted matching problems.

Edmonds ([13],[12]) proposed the first polynomial time algorithm for maximum matching in weighted graphs. His proposed solution was based on using augmenting paths (see Section 2.3.2). Edmonds has presented a term "blossoms" which is defined as an odd-length cycle in a graph and proposed that shrinking a blossom to a single vertex helps to achieve the maximum matching. Each iteration of Edmond's algorithm either finds an augmenting path, or finds a blossom and recurses onto the corresponding contracted graph (the graph as a result of shrinking blossom to a single vertex), or concludes that there are no augmenting paths. Edmonds' algorithm has a runtime $O(n^2m)$ or $O(n^4)$ where n is the number of vertices in the graph and m is the number of edges. Later, many researchers worked on Edmond's algorithm and improved the worst-case complexity.

Gabow *et al.* [15] proposed an improvement to the Edmonds' maximum matching algorithm to achieve a complexity of $O(n^3)$ by eliminating the process of blossom expansion. The author proposed a system of labels to store the structure of alternating paths and explained that a time bound of $O(n^4)$ in Edmonds' algorithm results from n^2 blossom expansion operations since each operation requires time $O(n^2)$. However, the proposed solution avoids the shrinking and expansion of blossom by recording the pertinent structure, hence, results in a factor of n speedup.

Galil *et al.* [19] made a significant improvement to Edmonds' blossom in terms of the time by performing dual adjustment and achieve a time bound of $O(m\,n\log n)$. Gabow [16], later achieved a bound of $O(n(m + n\log n))$ which is till now the best improvement

of Edmonds' algorithm.

There has been a sequence of improvements in the implementation of Edmonds' algorithm. [6] proposed an efficient implementation known as Blossom IV that has a running time of $O(n^2m)$. The proposed method makes use of multiple search trees to improve the practical performance of Edmonds' algorithm on large-scale problem instances. [29] presented a variation of [19] which solves the weighted matching problem in general graphs in time $O(m\,n\,log\,n)$. This implementation outperforms Blossom IV by making use of concatenable priority queues. A recent implementation of minimum cost perfect matching algorithm as known as Blossom V is presented by [23] that claims to outperform the earlier best-known implementation, i.e., Blossom IV by maintaining an auxiliary graph whose nodes correspond to alternating trees in Edmonds' algorithm. Some open source libraries for Edmond's weighted matching algorithm can be found as LEDA [28], LEMON [10].

2.3.4 String Matching Problem

The string matching problem is defined as identifying all the occurrences of a pattern $x = x[0, 1, ...m - 1]$ of length m in a text $y = y[0, 1....n - 1]$ of length n. String matching is widely used in information retrieval and text-editing like applications, e.g., search for a particular pattern in DNA sequences, search for relevant web pages related to user queries by internet search engines, etc. An example is shown by Table 2.1 that represents the text y and a pattern x with an objective to find the occurrences of the pattern x inside the text y. It is shown by the example that the pattern x can be found in the given text with a character shift equal to three.

Table 2.1: An example of String Matching

Text y	a	b	c	a	b	d	c	e	f
Pattern x	a	b	d	c					
Shift s	s=1	s=2	s=3	**a**	**b**	**d**	**c**		

String Matching for Topology Discovery

The topology discovery problem comprises parameters that are composed of string values and are predisposed to typographical errors. The errors normally occur during the provisioning of information by the network operators. For example: the two end-points of a connection are supposed to have the same string ID. However, at the time of configuring the string ID, the network operator may perform some typographical errors and the two strings may not remain the same anymore. i.e., String1: Concordia, String2: Cxoncordiaa, it can be seen that string2 gets some additional characters (C**x**oncordia**a**) that made the two strings different. These errors could not be avoided since they are introduced by users and may result in topology mismatch. The string matching algorithms are therefore required to get the correct pairing of ports while tolerating a certain amount of error.

We will take the example of Connection ID (see Section 3.1.2) or CID for short, that is a string value provisioned for each time slot by a network operator. The CID may contain errors like misspellings or may have additional spaces or special characters introduced by mistake. Our proposed methodology for topology discovery involves comparing the two CID values. However, if there are errors in CID strings, this can lead to a mismatch. One straightforward solution is to perform the character-by-character matching of two strings in order to determine their equivalence. However, it is not very efficient and may be misleading as the insertion of an error-character at any point of the string will make the rest of the character comparisons leading to a mismatch. An example is shown by Table 2.2

where insertion of a space character inside a string leads to mismatch for all succeeding characters.

Table 2.2: An example of character-by-character String Matching

Text y	C	o	n	c	o	r	d	i	a		
Text z	C	o	#	n	c	o	r	d	i	a	
String Matching	✓	✓	x	x	x	x	x	x	x	x	

We therefore need efficient methods to perform the string matching that takes into account the errors that may be introduced in between the strings.

2.3.5 Literature on String Matching Algorithm

The problem that we are interested in for topology discovery is a variant of string matching problem and studied in literature as the duplicate record detection problem ([14] , [32]). The duplicate record detection is mostly needed for data cleansing inside databases and relies on string comparison techniques to identify the typographical variations. We will discuss some popular methods that have been used in literature for de-duplication of records.

Edit Distance

Levenshtein [27] has defined edit-distance between two strings as a measure to determine their similarity. The edit-distance between two strings m1 and m2 is the minimum number of edit operations of single character that are required to transform one string into another. The three types of edit operations are insertions, deletions, and substitutions. The edit-distance approach outperforms the character-by-character string matching operation by not taking into account the exact position of characters for matching. For example the edit-distance of two strings (String1: Concordia, String 2: Co#ncordia) is 1 as shown in

Table 2.2 since removing a single character, i.e., ”#” makes the two strings identical. However, the edit distance method has a problem when the numbers are used. For the specific case of topology discovery where the CID represents the address or location of a port, For example, String1: ”Concordia1-5-7” and String2: ”Concordia1-5-8”. The three numbers in the example in each string represents the shelf, slot and port location and therefore, should be considered a strong mismatch rather a match with a distance of 1. We therefore need to take care of this type of conditions in the topology discovery problem.

Affine Gap Distance

Waterman *et al.* [39] proposed a solution for strings that have been truncated or shortened as the edit-distance method does not work well in such cases. For example String 1: ”Computer Department UdeM” and String 2: ”Computer Department Universite de Montreal” are the same since University de Montreal in string2 has been abbreviated to UdeM in string1. The affine gap introduced two extra edit-operations called open-gap and extend-gap to tackle this type of problem. The affine-gap distance method is not relevant for the topology discovery string matching problem since the CID only represents the information regarding the name of a network element and the physical location of the port supporting this CID, i.e., the location of shelf, slot and the port itself.

Smith-Waterman Distance

Smith and Waterman [35] proposed an extension of edit distance and affine gap distance. The proposed method gives lower costs to mismatches at the beginning and the end of strings than the mismatches in the middle. For example, String1: ”Prof. David R. Cruise, Udem” and String2: ”David R. Cruise, Prof.” will have a very short distance using the Smith-Waterman distance.

The above discussed methods provide a solution for finding the similarity between the

two strings in terms of the distance between them. The topology discovery string matching problem is close in affinity with these problems however, as discussed earlier, the CID string values usually contain the physical address of the port (i.e., the numerical value) and none of the above given methods take care of such exceptions. Therefore we will use edit-distance (Levenshtein) method [26] with some modifications to achieve the objective for topology discovery string matching. Other research that has utilized some of the above discussed methods can be found in ([25], [5], [21]).

Chapter 3

Topology Discovery of Synchronous Optical Networks

This research work has been published as ”Topology Discovery of Synchronous Optical Networks” in IEEE International Conference on Computing, Networking and Communications (ICNC) 2017 [22].

3.1 Topology Discovery Problem and Port Signatures

The objective of this study is to develop tools in order to discover the topology or the port-level connectivity of optical networks, and for the experimental part of the study, the link connectivity of a SONET network, in view of being able to migrate it smoothly in a later stage. The proposed strategy will use the offline provisioning data and will provide a confidence level for each matched port pair.

We define the topology discovery problem as follows: given a set of fiber endpoints, or ports, how can we pair them in a unique way in order to re-establish the network connectivity. In order to do so, we will define a signature for each port, and then develop algorithms in order to uniquely pair these signatures. The port signature may not be unique since the

information related to ports might be incomplete, erroneous, or user provisioned.

3.1.1 Background

SONET is a transmission protocol that synchronously transfers the multiple digital bit streams over optical fiber cable by multiplexing the low rate signals into higher bandwidth signals. SONET defines its structure in Synchronous Transport Signal (STS) levels, in which STS-1 is the base level fundamental framing structure. Each STS-1 frame consists of overhead bytes in addition to payload or information bytes. Some of these different overhead bytes (Section overhead, Line overhead) will be used as signature parameters. In this thesis we will use SONET ports of different granularity. The standard unit of measure for a port is Optical Carrier (OC) that signifies the rate of transmission bandwidth as given by Table 3.1.

OC	**Rate (Mbps)**
3	155.52
12	622.08
48	2488.32
192	9953.28

Table 3.1: OC Vs Rate

3.1.2 Port Signature

We propose to define the port signature as a 4-tuple made of the four parameters: Auto Discovery (AD), Section Trace (ST), Time Slot Pattern (TSP) and Connection ID (CID), which are next described. Note that, in practice, one or more of them may be missing.

Auto Discovery. The Auto discovery is a system parameter that allows physical connectivity of nodes to be discovered on transport optics as well as tributaries (a structure to transport low rate signals [38]. Network element auto discovery involves transmitting a unique signature in spare bytes of SONET line overhead. By comparing the transmitted and received auto discovery signature between the two ports, the topology can be deduced. Auto discovery despite being a useful parameter has some limitations. It is not available on all legacy equipment, and therefore, is not a multi-vendor compatible parameter.
An example of auto discovery Transmit and Receive strings is: TX: (AB_1_01:54:33:77:55_001) and RX: (AB_1_01:52:63:33:55_010).

Section Trace. Section trace is a user provisioned field that is defined in SONET section overhead frame as J0 bytes. It is a 16-byte fixed length message string transported between end ports, so that a receiving terminal can verify its continued connection to the intended transmitter [30]. Matching a transmitted section trace value of an endpoint with the receiving section trace value at other endpoint allows the discovery of a connectivity between two endpoints provided the two unique section trace values within the network. An example of sample section trace values is:
Port P_i: Section trace _Tx: Concordia 15-9-1, Section trace _Rx: Mcgill 12-3-2.
Port P_j: Section trace _Tx: Mcgill 12-3-2, Section trace _Rx: Concordia 15-9-1.

Time Slot Pattern. Timeslot pattern parameter gives the information on the set of time slots for a given port. In SONET networks, the capacity of an optical carrier is divided into a number of subrate timeslots. The individual timeslots determine the minimum capacity subrate path available [4]. For example, if the underlying capacity of the carrier is 2,488 Mbps and the node elements support 48 subrate timeslots, each timeslot has a capacity of 52 Mbps. In order to transmit the traffic between two ports inside a node, a transparent path

is established which is termed as cross connection. The timeslot in which traffic leaves a port is identical to the timeslot in which traffic arrives in at the adjacent node. This is due to the optical fiber cable which does not manipulate the data passes trough it. An example of timeslot pattern for an OC-12 port is 100111100110: where 1 represents the occupied timeslot and 0 represents the empty slot (in this example time slots 1, 4, 5, 6, 7, 10 and 11 are being used).

Connection ID. In order to transmit the user traffic through the network, a path is created from source to destination node consisting of topological links and cross connection links (see Section 1.3) to switch traffic from one link to another through out the connection from the source node to the destination node. This path has an identifier - the Connection ID, which should be consistent across the network elements. When cross connects are provisioned on a node, they are tagged with this Connection ID. As each network element needs to be provisioned with its cross connect individually, typographical errors may occur during this manual provisioning. If two ports have the same set of connection IDs provisioned for same time slot pattern, it is highly likely that they belong to a pair connected by an optical fiber, or topological link. An example of connection ID parameter is given below for an OC-12 port. where "-" represents an empty slot that has not been configured for any cross connection.

CID: ($\underbrace{\text{UQAM1}}_{1}, \underbrace{-}_{2}, \underbrace{-}_{3}, \underbrace{\text{mcgill1}}_{4}, \underbrace{\text{UQAM2}}_{5}, \underbrace{\text{UQAM3}}_{6}, \underbrace{\text{mcgill1}}_{7}, \underbrace{-}_{8}, \underbrace{-}_{9}, \underbrace{\text{UQAM4}}_{10}, \underbrace{\text{UQAM5}}_{11}, \underbrace{-}_{12}$).

3.2 Similarity Graph and Confidence Coefficients

We now define a similarity graph for each possible port rate. Indeed, we can only pair ports with the same rate, i.e., we can not pair an OC-3 port with an OC-12 port. In this study, we consider 4 rates (OC-3, OC-12, OC-48 and OC-192), but the algorithm can be easily extended for more or different rates ports.

3.2.1 Definition of the Similarity Graph

We define the similarity graph as a weighted graph $G = (V, E)$, where V refers to the set of ports and E expresses the set of edges between the ports. An edge $e \in E$ if there is a potential for pairing ports P_i and P_j associated with v and v'. Potential pairing is expressed throughout two values: S_{ij}, a similarity value that measures the similarities between the signatures of ports P_i and P_j, and L_{ij}, a label that counts the number of signature elements taken into account in S_{ij}. As mentioned earlier, some signature elements may be missing in practice. For instance, user managed parameters have not been filled.

If for a given node pair, $S_{ij} < \tau$, where τ is a preset threshold value, then we do not set an edge between v and v' as it is unlikely that we can pair those two pairs based on the available information. Additional conditions may apply for the insertion of an edge, see for instance in Connection ID Similarity Coefficient calculation.

3.2.2 Similarity Coefficients

We establish a similarity coefficient for every pair of ports. It is defined as a measure of resemblance between two ports based on the four parameters of the port signatures (AD, ST, TSP, CID). It is a numerical value in $[0, 1]$ where 1 delineates complete similarity and 0 complete dissimilarity. For a set of two ports, P_i and P_j, the similarity coefficient S_{ij} is calculated as follows:

$$S_{ij} = \frac{1}{L_{ij}} \sum_{\bullet \in \{\text{AD,ST,TSP,CID}\}: S_{ij}^{\bullet} \in [0,1]} w_{\bullet}\, S_{ij}^{\bullet} \tag{1}$$

where $w_{\bullet}, \bullet \in \{\text{AD}, \text{ST}, \text{TSP}, \text{CID}\}$ defines the normalized weight (i.e., $w_{\text{AD}} + w_{\text{ST}} + w_{\text{TSP}} + w_{\text{CID}} = 1$) given to a signature parameter, and $S_{ij}^{\bullet}, \bullet \in \{\text{AD}, \text{ST}, \text{TSP}, \text{CID}\}$ are the similarity values associated with each signature element, assuming that $S_{ij}^{\bullet} \leq 1$. If some signature

elements are missing, corresponding similarities are set to 0 and consequently, the normalization is made according to the number of nonzero similarity coefficients as given by L_{ij}.

Auto Discovery Similarity Coefficient ($S_{ij}^{\mathbf{AD}}$)

In order to compare the AD values of two ports P_i and P_j, one needs to compare the transceiver AD of P_i with the receiver AD of port P_j and the transceiver AD of P_j with the receiver AD of port P_i. However, in practice, some values may be missing, and comparison could be limited to only one AD receiver with one AD transceiver. If some AD transceiver/receiver values are missing in such a way that we cannot compare the AD receiver of one port with the AD transceiver of the other port (e.g., only the AD receiver values are provided), then the contribution of AD parameter to overall similarity coefficient value is either significantly reduced (i.e., 0.6) or zero. The value is chosen slightly higher than half (i.e., 0.6) based on an observation made in real network where some port signatures had one pair of AD values matched but the other pair was not matched. This was reported as an anomaly and just to avoid having wrong pairings, we selected this value slightly larger than 0.5 which is the value given to user-provisioned parameters. All possible cases related to AD values are discussed in Table 3.2.

Section Trace Similarity Coefficient ($S_{ij}^{\mathbf{ST}}$)

ST is considered a reliable system parameter if correctly configured on each end port of a circuit, though it is provisioned manually by the operators. ST has to be seen as a strict two-way parameter. If ST has been defined for two ports and the first port receives the ST value from the second port, definitely the second port should also be able to receive the ST value from the first port and vice versa. This entails that, as for the AD strings, $S_{ij}^{\mathrm{ST}} = 1$ if we have identical ST values, i.e., if Tx-ST of port P_i matches with Rx-ST of port P_j, and

Table 3.2: AD similarity coefficient computation

Cases	Ports	Availability	Conditions	S_{ij}^{AD}
# 1	P_i	Tx_AD_i Rx_AD_i	Tx_AD_i = Rx_AD_j Tx_AD_j = Rx_AD_i	$S_{ij}^{\text{AD}} = 1$
	P_j	Tx_AD_j Rx_AD_j	All other cases	$S_{ij}^{\text{AD}} = 0$
#2	P_i	Tx_AD_i Rx_AD_i	Tx_AD_j = Rx_AD_i	$S_{ij}^{\text{AD}} = .6$
	P_j	Tx_AD_j	All other cases	$S_{ij}^{\text{AD}} = 0$
# 3	P_i P_j	Rx_AD_i Tx_AD_j	Rx_AD_i = Tx_AD_j All other cases	$S_{ij}^{\text{AD}} = .6$ $S_{ij}^{\text{AD}} = 0$
# 4	All other cases			$S_{ij}^{\text{AD}} = 0$

Tx-ST of port P_j matches with Rx-ST of port P_i as shown by Table 3.3. However, as soon as there is a mismatch, or missing data $S_{ij}^{\text{ST}} = 0$. We apply stricter rules than for AD, as AD is system generated while ST is user provisioned parameter.

Indeed, if ST values match for one direction, but not for the reverse one, due to mismatched values, the reason may be due to incorrect fibering on-site. In addition, we cannot be 100% sure for the port connectivity based on the first match if not confirmed by the second one. Moreover, since ST is a user-provisioned parameter, it can not be guaranteed that it will be unique and this may lead to wrong pairing too. So it is important to verify if the ST is unique when pairing is performed. Based on above mentioned comments, we only give a weight if both the ends of ports are matched for section trace.

Time Slot Pattern Similarity Coefficient (S_{ij}^{TSP})

It is computed as follows:

$$S_{ij}^{\text{TSP}} = \frac{\#\text{ matched slots}}{\#\text{ non-empty slots}} \tag{2}$$

Table 3.3: ST similarity coefficient computation

Cases	Ports	Availability	Conditions	S_{ij}^{ST}
# 1	P_i	Tx_ST_i Rx_ST_i	Tx_ST_i = Rx_ST_j Tx_ST_j = Rx_ST_i	$S_{ij}^{\text{ST}} = 1$
	P_j	Tx_ST_j Rx_ST_j	All other cases	$S_{ij}^{\text{AD}} = 0$
#2	P_i	Tx_ST_i Rx_ST_i	Tx_ST_j = Rx_ST_i	$S_{ij}^{\text{ST}} = 0$
	P_j	Tx_ST_j	All other cases	$S_{ij}^{\text{ST}} = 0$
# 3	P_i P_j	Rx_ST_i Tx_ST_j	Rx_ST_i = Tx_ST_j All other cases	$S_{ij}^{\text{ST}} = 0$ $S_{ij}^{\text{ST}} = 0$
# 4	All other cases			$S_{ij}^{\text{ST}} = 0$

where # matched slots corresponds to the number of slots that are occupied in both ports while non-empty slots refers to the slots that are occupied in any one port but neither occupied in both ports nor non-occupied. An example is given below in Table 3.4 and illustrates different cases. The denominator of Formula 2 is equal to the number of non-empty slots. This way, it helps to differentiate cases such as Cases 1 and 4 in Table 3.4. In case 1, the length of TSP is 3 and 1 out of 3 slots is matched hence S_{ij}^{TSP} = 0.33. In case 2, both the non-empty slots are matched hence the S_{ij}^{TSP} = 1. In case 4, the length of the TSP strings is 192. However, 189 slots are empty considering the common slots of TSP_i and TSP_j which are equal to "0". Looking at the remaining slots, only 1 out of 3 is matched. It follows that $S_{ij}^{\text{TSP}} = 0.33$, while it would have been equal to 0.989 if we had consider the overall number of slots. However, considering that 189 slots are empty, cases 1 and 4 are not different.

Connection ID Similarity Coefficient (S_{ij}^{CID})

Each port has usually several time slots associated with it, and each time slot is identified by a CID string. Let S be the set of non-empty time slots, indexed by s. Hence, for a given port p, CID is the concatenation CID strings CID_s^p for $s \in S$. We then proceed in two steps

Table 3.4: Illustration of S_{ij}^{TSP} computation

Cases	Ports	TSP strings	S_{ij}^{TSP}	Length
1	TSP_i TSP_j	110 011	1/3 = 0.33	3
2	TSP_i TSP_j	110 110	2/2 = 1	3
3	TSP_i TSP_j	100 000	0	3
4	TSP_i TSP_j	00..00110 00..00011	1/3 = 0.33	192

for computing the CID similarity coefficient.

Firstly, we compute dissimilarity distance between each pair of CID strings, for each time slot (s):

$$\text{L\_DIST}\big(\text{CID}_s^{P_i}, \text{CID}_s^{P_j}\big) \qquad\qquad s \in S, \tag{3}$$

where L_DIST is the Levenshtein distance, i.e., a classical string metric for measuring the difference between two strings. Informally, it is the minimum number of single-character edits required to change one string into the other (see Section 2.3.5).

If the normalized Levenshtein distance is less than threshold value τ^{match} for a given s, i.e., if

$$\text{L\_DIST}_s^{\text{CID}} = \frac{\text{L\_DIST}\big(\text{CID}_s^{P_i}, \text{CID}_s^{P_j}\big)}{\max\{|\text{CID}_s^{P_i}|, |(\text{CID}_s^{P_j}|\}} \leq \tau^{\text{match}}, \tag{4}$$

we consider that the two strings associated with s match ($\text{MATCH}_s = 1$), otherwise they are declared as a non matching string pair and we forget about those CID strings ($\text{MATCH}_s = 0$). In addition, if the largest substrings (and not only suffixes) associated with numbers in the two CID strings do not match, then $\text{MATCH}_s^{\text{NUM}} = 0$, otherwise $\text{MATCH}_s^{\text{NUM}} = 1$. The reason of such a rule comes from the fact that numbers are usually associated with the address or the physical location of a node or port. Hence, we cannot allow any difference,

as we did for the other characters. We next compute

$$\text{ratio}^{\text{CID}} = \frac{\sum_{s \in S} \text{MATCH}_s}{|S|}. \tag{5}$$

If the ratio$^{\text{CID}}$ ratio is smaller than τ, i.e., the similarity threshold for adding an edge in the similarity graph, then no edge is added in G between v and v'.

The second step is the computation of S_{ij}^{CID}:

$$S_{ij}^{\text{CID}} = \frac{\sum_{s \in S} \text{MATCH}_s^{\text{NUM}}(1 - \text{L\_DIST}_s^{\text{CID}})}{|S|} \tag{6}$$

We illustrate in Table 3.5 the computation of S_{ij}^{CID} on an example with 3 time slots. Using $\tau^{\text{match}} = .25$ leads to the value of S_{ij}^{CID} that is computed in Table 3.6. Note that $\text{MATCH}_s = 0$ for the second slot, as the substrings made of the numbers do not match. In addition, if ratio$^{\text{CID}} < \tau$, there will be no edge $e = \{v, v'\}$ in G, otherwise $e = \{v, v'\}$ will be added with a weight equal to $S_{ij}^{\text{CID}} = 0.37$ and $L_{ij} = 4$ if AD, ST and TSP values are also available.

Table 3.5: CID data example

Ports	Time slots		
	#1	#2	#3
P_i	Standard St. IPCS	USC East North 998	McGill
P_j	IPCS Tanglefoot Lane	USC East North 2312	XMcGill

Table 3.6: S_{ij}^{CID} calculation

Slots	L_DIST	L_DIST$_s^{\text{CID}}$	match	ratio$^{\text{CID}}$	S_{ij}^{CID}
1	15	0.75 = 15/20	0		$1/3 \times (0.25$
2	∞	-	0	1/3 = 0.33	+ 0.86)
3	1	0.14 = 1/7	1		= 0.37

3.2.3 Weighted Matching and Confidence Level of a Port Pairing Solution

Once the similarity values have been computed, we identify the connected components of the similarity graph $G = (V, E)$. Components with a single node correspond to isolated ports, for which no pairing can be identified and are removed from similarity graph. Components with two nodes define straightforward port pairings and will be taken out of similarity graph in the pre-processing step.

Components with more than 3 nodes need more attention in order to deduce the port pairings. For them, we use the exact weighted matching algorithm (called W_MATCH for short) of [17] in order to identify the most likely port pairings.

Figure 3.1 represents two disconnected graph components, where one component has 3 vertices (Ports P_1, P_2, P_3) and other component has 6 vertices (Ports P_4, P_5, P_6, P_7, P_8, P_9). The similarity coefficient values for each pair of ports is shown. Figure 3.2 represents the weighted matching output as shown by bold edges of the two disconnected graphs where the matching sum is 0.9 and 0.866 respectively.

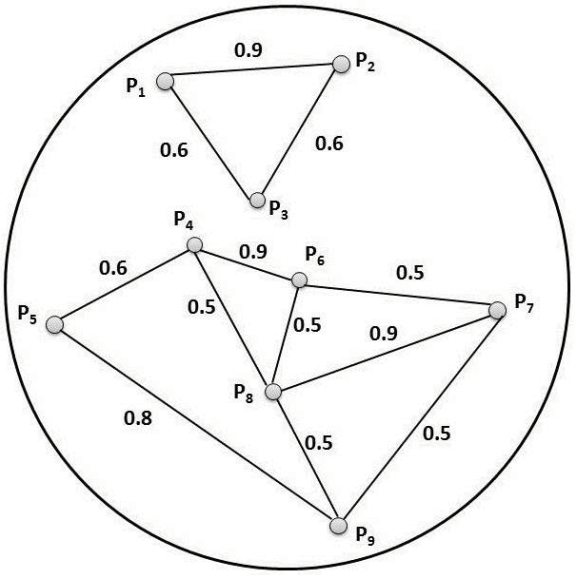

Figure 3.1: Port Pairing Input for Weighted Matching

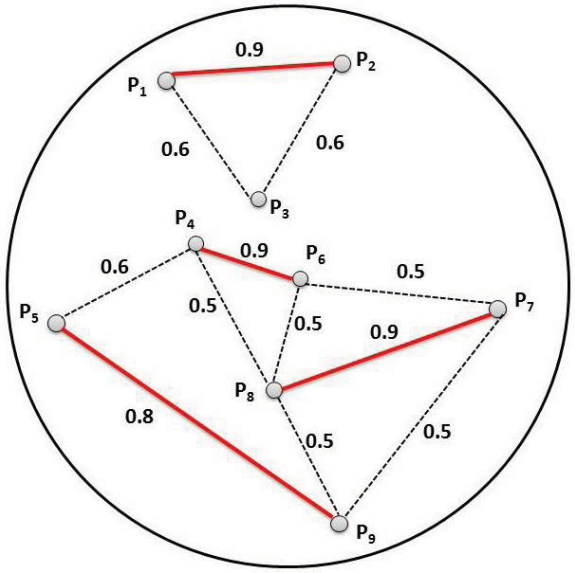

Figure 3.2: Port Pairing Weighted Matching Output

The outcome of the weighted matching algorithm leads to a set of potential port pairings where the confidence of each port pair can be estimated by the similarity value of the edge e connecting the two ports. We will denote it by CF_e. The confidence of a port pairing

solution, denoted by CF, can then be evaluated as follows:

$$\text{CF} = \frac{\sum_{e \in \text{MATCHING SOLUTION}} \text{CF}_e}{\text{\# matching elements}} \tag{7}$$

3.2.4 Algorithm PPWM

We identify the port pairing based on the outputs of a weighted matching algorithm, starting with the ports (nodes) with the highest label values, as described in the algorithm PPWM (Port Pairing Weighted Matching) that is described below.

Algorithm PPWM: Port pairing for a given OC$_x$

Preprocessing: Detecting isolated ports and obvious port pairing

Build G_{OC_x} with $E_{\text{OC}\_x} = \{\{v_i, v_j\} \in E : L_{ij} = 4\}$

Apply W_MATCH Algorithm on each connected component of G_{OC_x}

let $S_{\text{W\_MATCH}}$ be the union of the solutions on each component

Pair the ports associated with an edge in the solutions of $S_{\text{W\_MATCH}}$

For $\ell = 3$ **to** 1 **step** -1

$E_{\text{OC}_x} \leftarrow (E_{\text{OC}_x} \setminus S) \cup \{\{v_i, v_j\} \in E : L_{ij} = \ell\}$

Apply W_MATCH on each connected component of G_{OC_x}

Let $S_{\text{W\_MATCH}}$ be the union of the solutions on each component

Pair the ports associated with an edge in the solutions of $S_{\text{W\_MATCH}}$

3.3 Numerical Results

3.3.1 Data Instances

We tested the proposed algorithm on the data set of one customer of CIENA. It corresponds to a network with 870 ports, whose distribution is as follows:

Rate: #ports $\quad \text{OC}_3 : 92;\ \text{OC}_{12} : 80;\ \text{OC}_{48} : 281;\ \text{OC}_{192} : 417.$

Table 3.7 contains the completeness characteristics of the port signatures. Almost half (47.93%) of the port signatures contain all parameters, while the remaining half of them have from a very incomplete (3.10%) to an almost complete set of signatures.

Table 3.7: Signature completeness characteristics

# Available parameters	OC-3	OC-12	OC-48	OC-192	%
4	-	26	113	278	47.9
3	4	18	108	125	29.3
2	10	23	33	10	8.7
1	67	5	21	2	10.9
0	11	8	6	2	3.1

3.3.2 Port Pairings

We now report on the number of identified port pairs depending on the threshold values used for building the similarity graph. We have performed experiments based on different threshold values and results are presented by Tables 3.9 and 3.10. The threshold values used for the experiments are given below:

$$S_{ij}^{\text{ST}} = S_{ij}^{\text{AD}} = 1.0,\ S_{ij}^{\text{TSP}} = S_{ij}^{\text{CID}} = 0.8,\ (S_{ij}^{sig} = 0.5, 0.75).$$

Tables 3.9 and 3.10 provide the distribution of port pairings based on the number of parameters that were matched for the four different port granularities. The average confidence value obtained in each iteration is given by $z^{\star}_{\text{MATCH}}$. It can be observed that the number of port-pairing obtained for OC-192 is the largest in comparison to the other ports. The reason is the number of OC-192 ports that have all 4 signature parameters (i.e., 278 out of 417 ports).

Similarly, as we increase the threshold value (τ), the number of unmatched ports also increases. The reason being the less number of edges in the similarity graph since, only the

ports having a value of similarity coefficient greater than the threshold value can form an edge in the graph.

A large number of OC-3 ports had only a single parameter available which lead to a large fraction of unmatched ports.

When increasing τ, the number of port pairs decrease but the level of confidence increases as can be well observed in Table 3.8.

Table 3.8: Effect of τ on solution confidence

	$\tau = 0.5$	$\tau = 0.75$	% Increase in Confidence
Pre-Processing	0.749	0.817	9.078
Iteration # 1	0.907	0.937	3.307
Iteration # 2	0.602	0.743	23.421
Iteration # 3	0.472	0.492	4.237
Iteration # 4	0	0	-

3.4 Simulator Design and Testing

SONET network connections are manually provisioned by network operators. As the user demand grows, the connections are revised and upgraded. The network provisioning information is highly vulnerable to errors due to this human intervention. Based on the proposed topology discovery algorithms and working with one customer data set, several questions are raised:

- Given a first percentage of missing information and a second percentage of erroneous information, what is the fraction of network connectivity that can be rediscovered correctly? and what fraction of network connectivity is deduced wrongly?. By wrongly, we mean that depending on especially the erroneous information some wrong deductions could be made on the network connectivity.

- What is the maximum amount of erroneous and incorrect information that can be

permitted before the network topology cannot be fully discovered? and what is the relation of confidence of solution with the number of exact pairing obtained.

To answer these question we propose a simulation framework that will generate a data set resembling real network data. We will perform extensive testing with different levels of missing/erroneous information to validate our proposed algorithms.

3.4.1 Simulation Framework

Figure 3.3 presents a simulation framework design. The core modules of this framework are port-generator, signature-generator, noise Manager, topology discovery engine and accuracy monitor. We have used an online network map with 50 nodes (germany50: available on http://sndlib.zib.de/home.action).

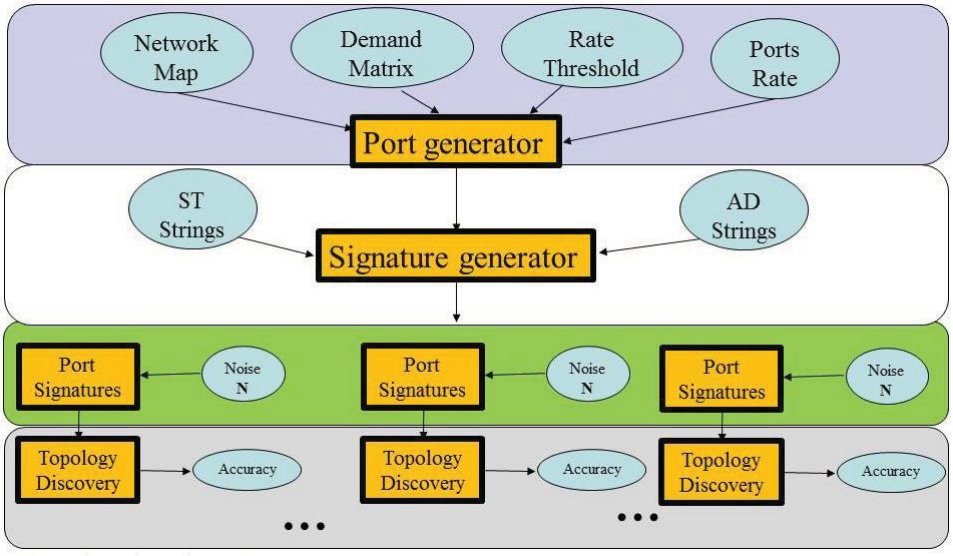

Figure 3.3: Simulation Framework Design

3.4.2 Design Steps

- Generate ports for NEs based on input parameters: network map, demand matrix, threshold value and port rate.

- Generate the port pairing for one granularity of ports, e.g., OC48.
- Generate exact signatures for each port pair.
- Generate a set of error vectors (noise).
- Introduce error/missing information (noise) in the generated port signatures.
- Run topology discovery and analyze the effect of errors on resultant topology by comparing the results with the reference port pairings (exact pairings).

Ports Generation

Algorithm 1 Ports generation

1: **Input:** Network Topology, Traffic demand matrix between nodes, Rate threshold (a constant value), Port Rate, max # Ports
2: Calculate the shortest path between the nodes
3: Aggregate the bandwidth demand on each link
4: **for** each link $\ell \in L$ **do**
5: Create (link rate / Rate threshold) Ports while # Ports for a node $\leq$ max # Ports.
6: **end for**

Signature Generation

After generating ports for each NE, port pairing is achieved by generating same signature for each port pair. Signatures are the collection of four parameters: Auto Discovery (AD), Section Trace (ST), Time Slot Pattern (TSP), and Connection ID (CID).

AD and ST contain two string values, i.e., Trans and Receive. We will use two string lists to build AD and ST parameters in each signature.

TSP is a sequence of 1's and 0's and will be generated randomly based on the port rate. However, there will be a threshold value for # used slots as shown in Table 3.11. For

example, if the port rate is 192, the length of TSP will be a random value between 49 and 192, that is to justify the use of specific port granularity. CID is composed of string values corresponding to the slots having 1 in TSP. We will take an input value for the length of CID string,i.e., the maximum length of CID. The CID strings will be randomly generated using the English alphabets [A-Z].

Table 3.11: Range for # Timeslots w.r.t Port rate

Port Rate	OC-3	OC-12	OC-48	OC-192
Range of slots	at least 1	[4,12]	[13,48]	[49,192]

Noise Generator

Introducing errors (noise) in to signatures is a challenging part of this simulation framework. Table 3.15 shows the attributes of missing/erroneous information $\mathbb{N}$ followed by the definition of each attribute.

(1) **# Missing nodes (NMN)**

In practice, it may happen that some nodes are hidden and that, in the process of collecting the circuit endpoint signatures, some nodes are forgotten. To keep the simulated network close in proximity with the real network, simulator randomly removes the given number of nodes completely along with all their ports.

(2) **# Missing parameters (NMP)**

Port parameters configuration is user-provisioned. Some parameters may therefore left unfilled. We provide a number of intended missing parameters in the network and the simulator randomly selects parameters to remove their values.

(3) **# Parameters in error**

In real network, it is likely to have parameters in error due to the poor configuration. The simulator randomly targets the signature parameters and introduces some errors

based on the nature of the parameter. This is to note that errors are not introduces in ST because if two ports have any single character mismatch in ST, the similarity coefficient weight for ST will be considered as 0. However for AD, if two ports have paired Tx AD with Rx AD but not the vice versa, it still gets a value of 0.6. These rules are a result of discussions with Ciena network engineers and based on the property of a parameter itself. For AD parameter to be in error, we simply remove the Tx or Rx AD value for the selected signature.

(4) **# TSP slots in error**

TSP errors are not frequent and do not involve errors in several slots. However, few slots may have errors. We consider 1-3 slots in error as shown by Table 3.15.

(5) **# CID slots in error**

CID is a collection of multiple string values, one for each time slot corresponding to TSP. It uses two inputs: a) First input value like TSP provides the number of signatures having 1-3 CID slots in error. b) Second input value demonstrates the number of errors to be introduced inside one CID value.

Table 3.17 shows four error values followed by the nature of their errors. For example, if the error value is +1, the program will randomly replace a single character inside the chosen CID slot. An example is to change ”Concordia” to ”Conco**A**dia” with a single character replacement.

Iterative process for introducing errors

Figure 3.4 presents the error generator diagram. We intend to introduce errors in the network and to investigate the number of discovered links and then repeating the process several times to obtain the overall trend of error effects on topology discovery. Algorithm 2 demonstrates how we plan to introduce error progressively on the generated signatures. A question here is important that how to calculate the percentage of error that we want to

introduce into the signatures. This means to translate Table 3.15 and Table 3.17 into a percentage value that we need for Algorithm 2. We begin with proposing error vectors based on error attributes and plan to iterate over 'N' such vectors to get a clear understanding of the behavior of error on the discovered topology discovery. There are rules associated with each vector that gives the input values and intended frequency of occurrence for each parameter as shown by Table 3.12. A value of p_1=1 and p_2=5 shows that a node will be removed in every 5 vectors. Same definition follows for the missing parameters. However, introducing errors inside signature parameters involve another criteria. We define a percentage value for each parameter i.e., AD,TSP,CID based on the nature of error occurrence in real network. For example, AD is given 5%, TSP is given 15%, and CID is given 80%.

3.4.3 Simulation Results

Figure 3.5 presents the simulation results showing the effects of incremental error on confidence value and on the number of exact matches. The X-axis presents number of iterations and Y-axis provides the number of exact matches obtained. The right Y-axis in red shows the confidence value in each iteration. It can be observed that at the beginning of simulation (at error vector 0) there is no error and all signature pairs are exact means a confidence value 1. As we go on increasing the error values in each iteration, the confidence value falls down as the number of exact matches decreases. However, the confidence value does not go down less than 65%.

Table 3.12: Rules for Error Vector (1)

S.No.	**Parameter**	**Value**	**Occurrence**
1	Missing Node	p_1	every p_2 vectors
2	Missing Parameter	p_3	every p_4 vectors
3	Errors	p_5	every p_6 vector

Table 3.13: Rules for Error Vector (2)

AD	**TSP**	**CID**
p_7	p_8	p_9
5%	15%	80%

Table 3.14: Values of Example Error Vector

p_2	p_4	p_6	p_7	p_8	p_9
5	4	1	5%	15%	80%

Algorithm 2 Iterative Error Generation

1: **Input:** Error vector, Set of Signatures, Set of Rules, # vectors to generate
2: **for** # Vectors times **do**
3: generate error values for each parameter
4: apply values to current Signatures
5: Run Topology discovery
6: Signatures ← current Signatures
7: **end for**

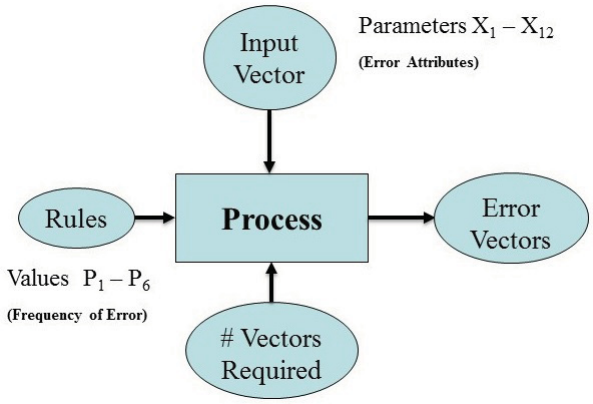

Figure 3.4: Iterative Error Generator

Table 3.9: Port pairings - $\tau = 0.50$

# labels	# Ports, i.e., # nodes in G OC-3	OC-12	OC-48	OC-192	Solution
Pre-Processing Results					
4	0	26	113	278	$z^\star_{\text{MATCH}} = 0.749$ $\rightsquigarrow$ 205 port pairs
3	4	18	108	125	
2	10	23	33	10	OC-3: 2 OC-12: 8 OC-48: 64 OC-192: 131
1	67	5	21	2	
Iteration #1					
4	0	24	73	115	$z^\star_{\text{MATCH}} = 0.907$ 81 port pairs OC-3:0 OC-12:2 OC-48:27 OC-192:52
Iteration #2					
4	0	20	19	11	$z^\star_{\text{MATCH}} = 0.602$ $\rightsquigarrow$ 57 port pairs OC-3:0 OC-12:6 OC-48:32 OC-192:19
3	2	12	58	37	
Iteration #3					
4	0	15	3	6	$z^\star_{\text{MATCH}} = 0.472$ $\rightsquigarrow$ 6 port pairs OC-3: 0 OC-12: 0 OC-48: 5 OC-192: 1
3	2	5	10	4	
2	8	15	13	1	
Iteration #4					
4	0	15	3	6	$z^\star_{\text{MATCH}} = 0$ $\rightsquigarrow$ 0 port pairs
3	2	5	7	3	
2	8	15	6	0	OC-3: 0 OC-12: 0 OC-48: 0 OC-192: 0
1	67	5	3	0	
Unmatched # Ports					
	77	40	19	9	overall: $\rightsquigarrow$ port pairs

Table 3.10: Port pairings - $\tau = 0.75$

# labels	# Ports, i.e., # nodes in G OC-3	OC-12	OC-48	OC-192	Solution
Pre-Processing Results					
4	0	26	113	278	$z^\star_{\text{MATCH}} = 0.817$ $\rightsquigarrow$ 262 port pairs
3	4	18	108	125	
2	10	23	33	10	OC-3: 1 OC-12: 10 OC-48: 75 OC-192: 176
1	67	5	21	2	
Iteration #1					
4	0	21	31	31	$z^\star_{\text{MATCH}} = 0.937$ 4 port pairs OC-3:0 OC-12:0 OC-48:3 OC-192:1
Iteration #2					
4	0	21	24	31	$z^\star_{\text{MATCH}} = 0.743$ $\rightsquigarrow$ 8 port pairs
3	4	14	64	29	OC-3:0 OC-12:0 OC-48:8 OC-192:0
Iteration #3					
4	0	21	22	31	$z^\star_{\text{MATCH}} = 0.492$ $\rightsquigarrow$ 8 port pairs
3	4	14	60	28	OC-3: 0 OC-12: 0 OC-48: 7 OC-192: 1
2	8	15	4	0	
Iteration #4					
4	0	21	22	31	$z^\star_{\text{MATCH}} = 0$ $\rightsquigarrow$ 0 port pairs
3	4	14	60	28	
2	8	15	4	0	OC-3: 0 OC-12: 0 OC-48: 0 OC-192: 0
1	67	2	3	0	
Unmatched # Ports					
	79	52	89	59	overall: $\rightsquigarrow$ port pairs

# Missing Nodes	# Missing Parameters	# Parameters with an Error			# TSP Slots in Error			# CID Slots in Error		
		AD	TSP	CID	1	2	3	1	2	3
5	25	20	69	90	18	48	3	18	48	24

Table 3.15: Attributes of Error Vector

# Vectors	# Missing Nodes	# Missing Parameters	# Parameters with an Error			# TSP Slots in Error			# CID Slots in Error		
			AD	TSP	CID	1	2	3	1	2	3
1	0	0	0	0	0	0	0	0	0	0	0
2	1	10	1	0	1	0	0	0	0	0	3
3	0	0	0	1	1	0	2	0	0	2	0
4	0	0	0	0	2	0	0	0	1	2	0
5	0	10	0	1	1	0	0	3	0	2	0
6	1	0	0	0	2	0	0	0	0	2	3
Total	**2**	**20**	**1**	**2**	**7**	**0**	**2**	**3**	**1**	**8**	**6**

Table 3.16: Error Vector Details

Error Value	+1	+2	+3	-1	-2	-3
Error Detail	Add 1 extra char	Add 2 extra chars	Add 3 extra chars	remove 1 extra char	remove 2 extra chars	remove 3 extra chars

Table 3.17: CID Error Vector

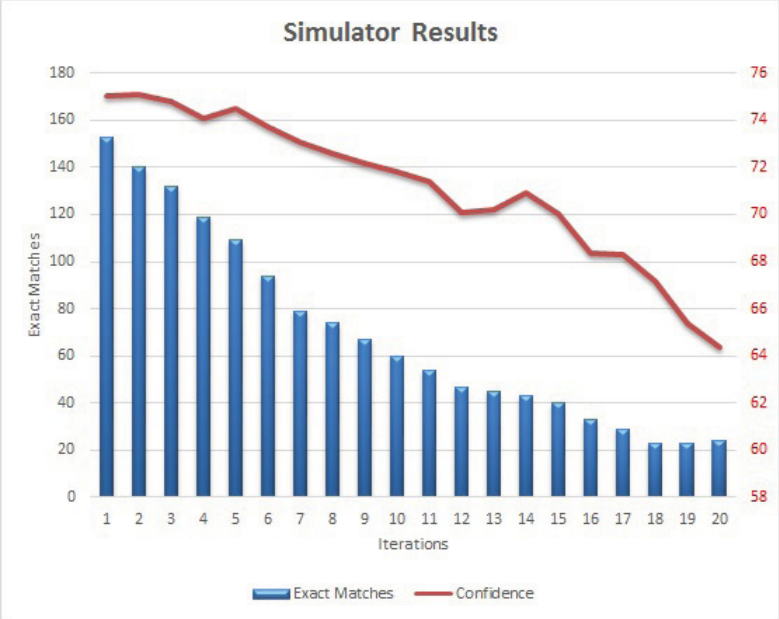

Figure 3.5: Simulator Result

Chapter 4

Circuit Stitching

4.1 Introduction

In the previous chapter, we discussed why network topology discovery is crucial for network operators and that despite its significance, topology discovery of legacy networks like SONET has not been studied much in the literature. There has been some work only on port-level connectivity of SONETs using overhead bandwidth [38] [30] [9] and different configuration parameters [36]. However, obtaining port-to-port connectivity is not sufficient in many cases. Network operators are always concerned about the routes that their customer traffic takes form a source node to its destination. The tasks like bandwidth optimization, priority traffic routing in case of network failures, and protection management largely depend on the knowledge of complete traffic paths.

After port-to-port connectivity discovery, the next step is to trace the complete path of users traffic between two endpoints. This is called circuit stitching. Circuit stitching is not only a solution for operators that are planning to migrate to new technologies but it is also of great interest for those optimizing legacy networks. In any case, the automated discovery helps in alleviating errors that otherwise would happen due to manual ways to update network information, and better assessing the confidence value of the discovered

circuits.

In this chapter, we devise an algorithm that performs circuit stitching, given an input that comprises of physical links, each associated with a confidence level, which measures its likeliness. This work can be seen as an advancement to SONET topology discovery where not only the two ports at fiber layer are traced but the two end points of a connection along with its complete path is retrieved. To the best of our knowledge, there has been no work in literature realizing circuit stitching for SONETs.

This chapter is organized as follows. Section 4.2 gives a brief introduction on SONET network and states formally the circuit stitching problem. The proposed algorithm is described in Section 4.3, together with its complexity. Section 4.4 talks about how circuit stitching can help to identify provisioning bugs and to trace protection. A discussion on circuit completeness characteristics is also given in Section 4.4. Results are presented in Section 4.5, together with their validation on a real data set coming from a customer of Ciena corporation.

4.2 Circuit Stitching

4.2.1 Single hop/ Multiple hop Circuits

Circuit stitching refers to identifying two end-points (start and end point) of a circuit as well as the complete path between the endpoints. A circuit is a set of alternating cross connection and topological links carrying user information having a bandwidth that is reserved from a given source NE to a destination NE in order to transmit user traffic. A circuit between two SONET nodes can be a single-hop or a multi-hop circuit. Figure 4.1 presents a SONETs of five NEs. A single-hop circuit (shown in black dotted line) is demonstrated between Nodes NE_A and NE_B and two multi-hop circuits (shown by green and red dotted

lines) between Nodes NE_A- NE_D- NE_B- NE_C- NE_E and NE_A- NE_B- NE_D- NE_E respectively.

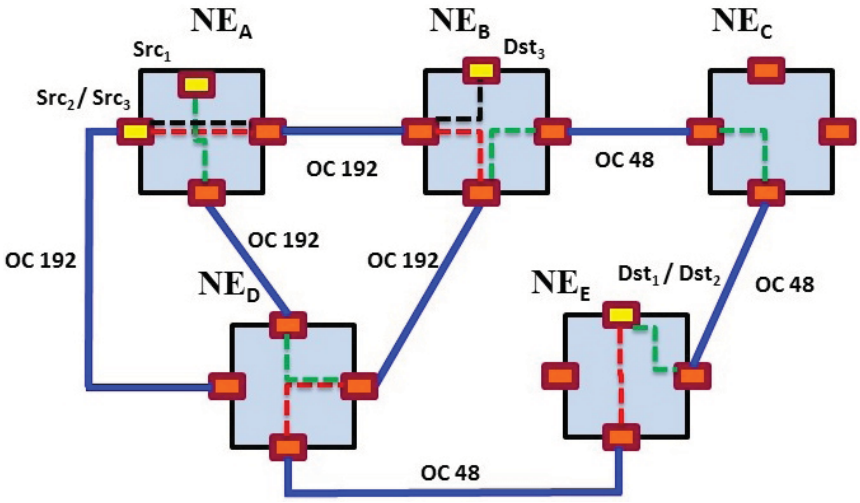

Figure 4.1: An Example of Single-hop and Multi-hop Circuits

Topological links and cross connections links are the two types of connections that contribute to traffic flow in SONET. Topological links are inter-NE connections as they connect ports of two different SONET network elements. Cross connections are intra-NE connections as built inside a network element. For example, as shown in Figure 4.1, user data is provided at ports of NE_A, shown as Src_1, Src_2 and Src_3 to be transported to NE_B and NE_E respectively. In order for the data to reach the destination NE, it must follow alternating cross connection (dotted lines) and topological links (solid lines). We will now discuss the two types of network connections in detail:

Topological Links

Topological links are the connections formed between the ports of two different network elements at fiber level. These connections are realized as transparent connections, as the fiber optic cable between the two NEs does not manipulate the data passing through it and instead provide a path to the traffic. The optical connections can be seen at granularity of

OC-3, OC-12, OC-48, OC-192, etc. Table 4.1 provides potential parameters for a topological link. Each topological link, since it provides connectivity between ports of two different NE's, has Aend and Zend NE name. Additionally, the ports that form the topological link are uniquely identified by their location, i.e., Physical Termination Point (PTP), that gives the shelf, slot and port location.

Table 4.1: Potential Parameters of Topological Link

Parameter	Value
aEnd Element Name	NE 1
zEnd Element Name	NE 2
aEnd PTP	/shelf=1/slot=6/port=1
zEnd PTP	/shelf=1/slot=1/port=7

Cross-connection Links

Cross connections provide the connectivity between two ports inside a network element using a cross connect circuit pack. The main parameters that are required for building a cross connection in SONET are presented by Table 4.2. Each cross connection is identified by an "A-end" and "Z-end" physical termination points, the exact timeslots in both end ports as Connection Termination Point (CTP) and a Connection ID (CID). The cross connections may have different granularity, that is STS-1, STS-3c, VT-1.5, etc. If the cross connection is built at STS-1 level, it means it can reserve only 1 timeslot in both the end-ports. If it is built at STS-3c, it means that three consecutive STS-1 timeslots are bunched together (concatenated) through out the connection. Table 4.3 represents the number of connections that a port may have for different rates of cross connections along with the potential slot numbering.

Table 4.2: Potential Parameters of a Cross Connection

Parameter	Value
Element Name	Network Element-1
aEnd PTP : aEnd CTP	/shelf=1/slot=6/port=1 : sts1 slot = 17
zEnd PTP : zEnd CTP	/shelf=1/slot=1/port=7 : sts1 slot = 17
Connection ID	University link # 7

Table 4.3: List of rate and number of connections for a Port [4]

Port	Rate	# Connections	Slot Numbering
OC192	sts192c	1	1
	sts48c	4	1-4
	sts12c	16	1-16
	sts3c	64	1-64
	sts1	192	1-192
	vt1.5	5,376	1-5376
OC48	sts48c	1	1
	sts12c	4	1-4
	sts3c	16	1-16
	sts1	48	1-48
	vt1.5	1,344	1-1344
OC12	sts12c	1	1
	sts3c	4	1-4
	sts1	12	1-12
	vt1.5	336	1-336
OC3	sts3c	1	1
	sts1	3	1-3
	vt1.5	84	1-84

4.2.2 Problem definition

We define the circuit stitching problem as identifying the two end-points of a circuit along with its complete path. The inputs to this problem are topological and cross connection links. We first select each endpoint of a topological link in turn and check whether we can

merge the set it belongs to with another set, in case the endpoints of the incomplete circuits in each set can be connected, i.e., they have the same label. A label is defined as follows: Label = {Shelf, Slot, STS Rate (high rate), STS TS, VT (low rate), VT TS}, where TS is the timeslot number and VT stands for Virtual Tributary. It should be noted that in this case we only match if the IDs are exactly the same. The reason being that if a connection is provisioned, it must have the values of shelf, slot and rate. We keep building the circuit this way until we reach to a point when we can not trace anymore. The circuits (set of topological links) are obtained as an output to this problem.

4.3 Circuit Stitching Algorithm (CISTAL)

The key to circuit stitching is to understand how the time slots are configured for different cross connections. Time slots allocated for a cross connection should be the same on both end ports for a successful connectivity. A variant of a union-find algorithm, (see e.g., Cormen et *al.* [7]) is proposed here to solve the circuit stitching problem. After a pre-processing step (Algorithm 2) in order to decouple the search of the primary and backup paths, we proceed with a union-find data structure (Algorithm 1). At the outset, every link defines a set. At any iteration, a set will contain either a full circuit, or a circuit fragment identified by the characteristics of its two endpoints. Note that a port may be configured for different rates (higher and lower rates). In that case, the time slots of all the available rates need to be matched. For example, some cross connections have VT1.5 connection on the top of STS1 connection. The union-find algorithm stops when we cannot merge any pair of sets, i.e., when we cannot stretch further any incomplete circuit. As soon as a set contains a complete circuit, it is eliminated from the list of sets to be considered. The complexity of the resulting algorithm is $O(n \log n)$, where n is the sum of the number of topological links and the number of cross-connect links, following the complexity of union-find algorithms [7].

Algorithm 3 CISTAL

$Input$: Set of links: $L = \{\ell_1, \ell_2, \ell_3 ..... \ell_n\}$ where $\ell \in L = \{(p, NE_p, Label_p), (p', NE_{p'}, Label_{p'}), \text{Type}\}$
Type $\in$ {topological, cross-connect}
p = source port, p' = destination port, Label = {Shelf, Slot, STS Rate, STS TS, VT, VT TS}, TS = Timeslot number
$Output : \mathcal{S} = \{S_1, S_2, ..., S_n\}$ (a set of sets)
for $\ell \in L$ **do**
    Initialize $\mathcal{S}$ with MakeSet(ℓ) : $\ell \in L$
end for
for p being the endpoint of a topological link **do**
    Let S be the set (incomplete circuit) with endpoint p
    Search for S' such that Edpt$(S') = p'$
    Pair S and S' if $label_p$= $label_{p'}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{S \cup S'\} \setminus \{S, S'\}$
    Keep track of the endpoints of $S \cup S'$ (new set) and their labels
end for

Algorithm 4 Pre Processing

$Input$: Set of links: $L = \{\ell_1, \ell_2, \ell_3 ..... \ell_n\}$ where $\ell \in L = \{(p, NE_p, Label_p), (p', NE_{p'}, Label_{p'}), \text{Type}\}$, $\ell^{\text{Type}} = \{CX, OPT\}$
$Output$: Updated L
for $\ell \in L^{CX}$ **do**
    Search for ℓ with one source ℓ_s and two destinations ℓ_{d1} and ℓ_{d2} or vice versa.
    Create link $\ell_i \leftarrow \{\ell_s, \ell_{d1}\}$
    Create link $\ell_j \leftarrow \{\ell_s, \ell_{d2}\}$
    $L \leftarrow L \setminus \ell$
end for

4.4 Learning Circuit Stitching to improve Network Topology

Circuit stitching originally intends to trace the two endpoints of a circuit along with its complete path. However, it can provide some more interesting results. In this section, we will discuss three applications of circuit stitching. *(i)* Typographical errors correction that occur during connection provisioning, *(ii)* Identifying circuit completeness characteristics, and *(iii)* Protection identification that helps to improve the discovered topology results.

4.4.1 Correction for Typographical Errors

Typographical errors are often introduced by human operators at the provisioning of circuits. Ideally, all the links belonging to a circuit should have the same CID (see Section 4.2). However, sometimes circuits are extended later by different operators, and sometimes due to typographical errors, the CID values do not remain the same. Table 4.4 shows an example of a circuit with 4 links. The confidence value (a measure of reliability, see [22]) for each link is much higher but the CID values are different for some links. It can be seen that all the CIDs refer to same connection, i.e., from Concordia to Mcgill, however, the naming convention is somehow different which can be easily corrected once the complete circuit is successfully traced. This would help the network operators in future to follow the corrected naming conventions.

Table 4.4: Correcting Typographical errors through Circuit Stitching

Circuit #	# Link	Connection ID	Confidence
1	1	Concordia to Mcgill1	0.9
	2	Concordia to Mcgil1	0.85
	3	Concord to Mcgill	0.8
	4	Concor to Mg1	0.9

4.4.2 Investigating Circuit Completeness

Circuit Stitching may help to identify the complete and incomplete circuits. *Complete circuits* are those circuits that have terminating ports at both ends. *Incomplete circuits* can be of two types. *(i)* circuits that have terminating port only at one end. *(ii)* circuits that end at a topological link (even-length circuits). Circuits ends at a topological link due to any of the two reasons: either the next link (cross connection) is not accessible or the stitched circuit has some wrong links. Table 4.7 provides a breakup of circuits based on their completeness.

4.4.3 Identifying Protection

Protection is a point of concern for almost every network including SONET that may suffer from equipment or link failures. As, the failure of a link or equipment may lead to traffic breakdown. We specifically look at path protection and define circuits as fully protected if there are two disjoint paths (i.e., there is no link shared by both paths as shown in Figure 4.2 (a)) connecting a source and destination. Protection comes at a cost for operators and hence all circuits are not fully protected normally. Circuit stitching may help us to identify if the protection is adequately provisioned in the network or if some links need to be protected further.

Based on the current data set, we have identified three circuit patterns as shown by Figure 4.2, where *(a)* presents the complete end to end protection, i.e., in case of failure of a link in first path (containing ports 1,2 and 3, 4) the traffic can flow through second path (containing ports 5,6 and 7,8). *(b)* presents under-protected links and *(c)* represents the broken circuits.

Table 4.5: Analysis of Protection

# Stitched Circuits	# Protected Circuits	# Retrieved E2E Protected Circuits	# Broken Circuits	% Retrieved Protected Circuit
3,634	989	704	40	71.18
3,674	981	714	14	72.78
3,711	1,085	880	14	81.10
4,016	1,300	1,022	14	78.61

4.5 Numerical Results

4.5.1 Data set

We use the data set of a customer of Ciena in order to validate the algorithms proposed in the previous section. Input to the circuit stitching problem consists of a set of links,

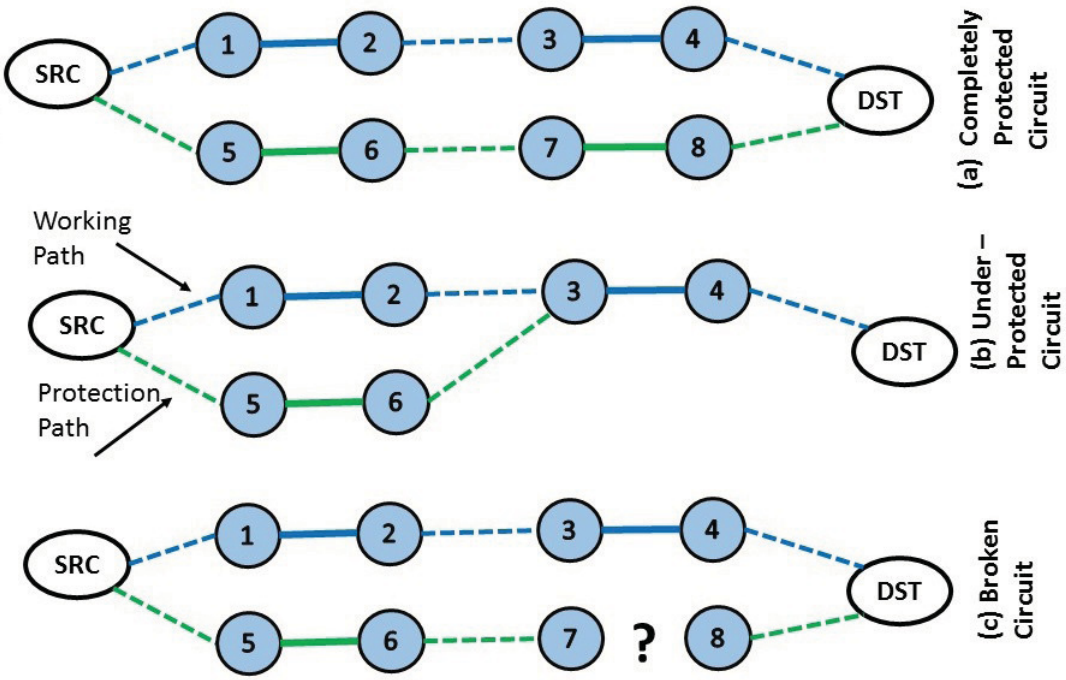

Figure 4.2: Protection Cases

partitioned into topological links and cross connection links. In the context of our study, the set of links comes as an output of a topology discovery algorithm, with a threshold value (τ) for the confidence level. Considering four different threshold values (see Table 4.6), we got four input instances to test the circuit stitching algorithms. It can be observed that pre-processing (i.e., Algorithm 2) increases the number of links significantly.

Table 4.6: Input to CISTAL

# Topological links	# Cross-connections	# Links before pre-processing	# Links after pre-processing
245 ($\tau = 0.75$)	12,580	12,825	15,169
271 ($\tau = 0.70$)	12,580	12,851	15,195
309 ($\tau = 0.65$)	12,580	12,889	15,233
324 ($\tau = 0.60$)	12,580	12,904	15,248

Table 4.7: Output of CISTAL

# Input Links	# Stitched Circuits	Complete Circuits		1-ECO Circuits		1-ECE Circuits		0-ECE Circuits		0-ECO Circuits	
		#	Avg length	#	Avg length	#	Avg length	#	Avg length	#	Avg length
15,169	3,634	71	4	118	5	11	3	1,126	2	2,308	5
15,195	3,674	83	4	128	6	13	4	1,152	2	2,298	5
15,233	3,711	134	5	73	8	20	7	1,223	2	2,261	6
15,248	4,016	140	5	69	8	30	7	1,553	2	2,224	6

1 ECE: 1-End-Circuit-Even (Circuit ending at an optical link and have one terminating end)
1 ECO: 1-End-Circuit-Odd (Circuit ending at cross connection and have one terminating end)
0 ECE: 1-End-Circuit-Even (Circuit ending at an optical link with no terminating end)
0 ECO: 1-End-Circuit-Odd (Circuit ending at cross connection with no terminating end)

Table 4.8: Validation of Results

	# Stitched circuits								Ciena Customer Input	
	$\tau = 0.75$		$\tau = 0.70$		$\tau = 0.65$		$\tau = 0.60$		# Circuits	Average length
% retrieved links	1 piece	2 pieces	1 piece	2 pieces	1 piece	2 pieces	1 piece	2 pieces	70	13.78
100%	3	0	9	1	11	2	13	26		
61-99%	6	22	16	32	16	32	7	17		
31-60%	4	34	2	10	2	7	2	4		
0-30%	0	1	0	0	0	0	0	1		
Total # circuits	13	57	27	43	29	41	22	48		
	70		70		70		70			

4.5.2 Analysis of Results

Table 4.7 presents the results of CISTAL algorithm for four input instances. A circuit is said to have one terminating end if only one endpoint is a topological link while it has even number of links if the endpoints of circuit are topological links and has odd number of links if the endpoints of circuit are cross-connection link.

The partially completed circuits are further categorized as: 1-ECE, i.e., circuits with 1 terminating end with even number of links, 1-ECO, i.e., circuits with 1 terminating end with odd number of links, 0-ECE, i.e., circuits with no terminating end with even number of links, and 0-ECO, i.e., circuits with no terminating end with odd number of links. It can be observed from Table 4.7 that as the number of input links increases, the number of complete circuits also increase.

We have validated results on a data set of Ciena customer. The validation is performed by Ciena engineers by comparing the results of our solution with the actual circuit traced. The results obtained by our algorithms sometimes gave better solution and sometimes gave worse regarding the count of traced circuits. We get better solution in terms of retrieving more circuits than that manually traced by Ciena partner. The reason being the limitations in manual tracing as discussed in Section 1. At times, our solution was worse due to the missing/erroneous values in provisioning parameters that turned out as getting circuits in two pieces (broken circuits) as shown by the Table 4.8. We have reported our observations on the basis of percentage of circuit (length) retrieved. One observation is: for the 70 input circuits, we have retrieved 39 out of 70 complete circuits by CISTAL with the fourth input case (with lowest threshold value). For the other cases (with a lower threshold value τ), the number of complete circuits increases as the threshold decreases. Table 4.5 presents the count of protected circuits, end-to-end protected circuits and broken circuits for the four different number of inputs links. It can be observed that as the number of stitched circuits increases, the number of complete end-to-end protected circuits also increase.

Chapter 5

Conclusions and Future Work

In this thesis, we proposed modelling of topology discovery problem as a weighted matching problem that appeared to be a good fit for maximizing the identification of proper port pairings, subject to missing or erroneous data. We also studied circuit stitching problem to determine paths that circuits follow from source endpoints to destinations.

This research work resulted in a tool (commercially being used in industry) [8] that helps to avoid tedious search and manual tracing for both topology discovery and circuit stitching. Unlike some previous works for topology discovery our algorithms take into account the effect of missing and incorrect network provisioning information which has never been considered before.

As a future work, we plan to investigate our algorithms for topology discovery in other network domains, e.g., IP, OTN, and SDN networks by modifying the signature parameters usable for these networks.

Bibliography

[1] A. Azad, M. Halappanavar, F. Dobrian, and A. Pothen. Computing maximum matching in parallel on bipartite graphs: Worth the effort? In *Proceedings of the First Workshop on Irregular Applications: Architectures and Algorithm (IAAA)*, pages 11–14, 2011.

[2] A. J. Barker. Communications network for self-determining its own topology, Nov. 4 2008. US Patent 7,447,753.

[3] Y. Breitbart, M. Garofalakis, B. Jai, C. Martin, R. Rastogi, and A. Silberschatz. Topology discovery in heterogeneous IP networks: the netinventory system. *IEEE Transactions on Networking*, 12(3):401–414, June 2004.

[4] S. Chatterjee. Unique numbering for SONET/SDH timeslots in network management system applications, Mar. 16 2006. US Patent App. 10/939,427.

[5] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *SIGMOD*, pages 313–324, 2003.

[6] W. Cook and A. Rohe. Computing minimum-weight perfect matchings. *INFORMS Journal on Computing*, 11(2):138–148, Feb. 1999.

[7] T. H. Cormen. *Introduction to Algorithms*. MIT press, 2009.

[8] C. Corporation. Topology discovery gaining insight to your network. https://www.ciena.com/insights/videos/Topology-Discovery-Gaining-Insight-to-Your-Network-prx.html. Accessed: 2019-11-3.

[9] S. Das. Topology discovery and path provisioning in SONET rings using GMPLS. In *International Conference on Wireless and Optical Communications Networks*, 2006.

[10] B. Dezs, A. Jttner, and P. Kovcs. {LEMON} an open source c++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23 – 45, 2011. Proceedings of the Second Workshop on Generative Technologies (WGT) 2010.

[11] R. Duan and H. Su. A scaling algorithm for maximum weight matching in bipartite graphs. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1413–1424, 2012.

[12] J. Edmonds. Matching and a polyhedron with 0,l vertices. *Journal of Research of the National Bureau of Standards. – B. Mathematics and Mathematical Physics*, 69(1 & 2):125–130, 1965.

[13] J. Edmonds. Path, trees and flowers. *Can. J. Math.*, 17:449–467, 1965.

[14] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, Jan 2007.

[15] H. N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. ACM*, 23(2):221–234, Apr. 1976.

[16] H. N. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.

[17] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph matching problems. *J. ACM*, 38(4):815–853, Oct. 1991.

[18] Z. Galil. Efficient algorithms for finding maximum matching in graphs. *ACM Comput. Surv.*, 18(1):23–38, Mar. 1986.

[19] Z. Galil, S. Micali, and H. Gabow. An O(EV\logV) algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing*, 15(1), 1986.

[20] M. Gondran, M. Minoux, and S. Vajda. *Graphs and Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 1984.

[21] M. A. Hernández and S. J. Stolfo. The Merge/Purge problem for large databases. *SIGMOD Rec.*, 24(2):127–138, May 1995.

[22] B. Jaumard, A. Muhammad, and R. Fahim. Topology discovery of synchronous optical networks. In *International Conference on Computing, Networking and Communications (ICNC)*, pages 194–199. IEEE, 2017.

[23] V. Kolmogorov. Blossom V: a new implementation of a minimum cost perfect matching algorithm. *Mathematical Programming Computation*, 1(1), 2009.

[24] J. E. Kracht. Approaches for determining actual physical topology of network based on gathered configuration information representing true neighboring devices, Feb. 4 2003. US Patent 6,516,345.

[25] M. L. Lee, H. Lu, T. W. Ling, L. Tok, W. Ling, and Y. T. Ko. *Cleansing Data for Mining and Warehousing*. Springer, 1999.

[26] V. Levenshtein. Levenshtein distance. https://en.wikipedia.org/wiki/Levenshtein_distance. Accessed: 2019-10-1.

[27] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[28] K. Mehlhorn and S. Näher. Leda a library of efficient data types and algorithms. In *International Symposium on Mathematical Foundations of Computer Science*, pages 88–106. Springer, 1989.

[29] K. Mehlhorn and G. Schäfer. Implementation of O(Nmlogn) Weighted Matchings in General Graphs: The Power of Data Structures. *J. Exp. Algorithmics*, 7, Dec. 2002.

[30] S. A. Oliva and B. Crowe. Network system and method for automatic discovery of topology using overhead bandwidth, Nov. 25 2003. US Patent 6,654,802.

[31] Y. Qiuxiang and Z. Lihong. A research on the automatic discovery technology of network topology. In *2nd International Conference on Biomedical Engineering and Informatics*, pages 1–3, Oct 2009.

[32] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 269–278, 2002.

[33] V. Sharma, A. Das, and C. Chen. Leveraging IP signaling and routing to manage upsr-based transport networks. In *IEEE International Conference on Communications (ICC)*, volume 2, pages 1268–1272, May 2003.

[34] R. Siamwalla, R. Sharma, and S. Keshav. Discovering internet topology. https://www.cs.cornell.edu/skeshav/papers/discovery.pdf, 1998. Accessed: 2015-07-21.

[35] T. F. Smith and M. S. Waterman. Comparison of biosequences. *Advances in Applied Mathematics*, 2(4):482 – 489, 1981.

[36] J. D. Voigt, C. T. Coston, R. J. Feuerstein, D. Youngblood, D. Rosenstock, T. Kau, and G. Bernhardt. Systems and methods for discovering network topology, Jan. 8 2013. US Patent 8,352,632.

[37] D. G. Waddington, F. Chang, R. Viswanathan, and B. Yao. Topology discovery for public ipv6 networks. *SIGCOMM Comput. Commun. Rev.*, 33(3):59–68, July 2003.

[38] C. Wallace, G. McCloskey, K. Cherwenka, and H. Truong. Determining connectivity in communication networks, Sept. 18 2003. US Patent App. 10/051,930.

[39] M. S. Waterman, T. F. Smith, and W. A. Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367 – 387, 1976.

[40] Y. Zhang, Y. Zhao, and X. Chen. Design and implementation of topology automatic discovery algorithm in PON NMS. In *2nd International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pages 1490–1493, Dec 2012.

[41] Y. Zhao, J. Yan, and H. Zou. Study on network topology discovery in IP networks. In *3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT)*, pages 186–190, Oct 2010.