REGENERATOR PLACEMENT IN OPTICAL NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Onur Özkök

January, 2004

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Oya Ekin Karaşan (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Osman Oğuz

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Bahar Yetiş Kara

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet B. Baray
Director of the Institute Engineering and Science

ABSTRACT

REGENERATOR PLACEMENT IN OPTICAL NETWORKS

Onur Özkök
M.S. in Industrial Engineering
Supervisor: Assist. Prof. Dr. Oya Ekin Karaşan
January, 2004

Increase in the number of users and resources consumed by modern applications results in an explosive growth in the traffic on the Internet. Optical networks with higher bandwidths offer faster and more reliable transmission of data and allows transmission of more data. Fiber optical cables have these advantages over the traditional copper wires. So it is expected that optical networks will have a wide application area.

However, there are some physical impairments and optical layer constraints in optical networks. One of these is signal degradation which limits the range of optical signals. Signals are degraded during transmission and below a threshold the signals become useless. In order to prevent this, regenerators which are capable of re-amplifying optical signals are used. Since regeneration is a costly process, it is important to decrease the number of regenerators used in an optical network.

To increase the reliability of the network, two edge-disjoint paths between each terminal on the network are to be constructed. So the second path could be used in case of a failure in transmitting data on an edge of the first path. Considering these requirements, selecting the nodes on which regenerators are to be placed is an important decision.

In this thesis, we discuss the problem of placing signal regenerators on optical networks with restoration. An integer linear program is formulated for this problem. Due to the huge size and other problems of the formulation, it is impractical to use it on large networks. For this reason, a fast heuristic algorithm is proposed to solve this problem. Three methods are proposed to check the feasibility when a fixed set of regenerators are placed on specific nodes. Additionally, a branch and bound algorithm which employs the proposed heuristic is developed to find the optimal solution of our problem. Performance of both the heuristics and the branch and bound method are evaluated in terms of number of regenerators placed and solution times of the algorithms.

Keywords: optical networks, regenerator placement, branch and bound algorithm, working and restoration path design.

ÖZET

OPTİK AĞLARDA REJENERATÖR KONUMLANDIRILMASI

Onur Özkök
Endüstri Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Yard. Doç. Dr. Oya Ekin Karaşan
Ocak, 2004

Kullanıcı sayısındaki ve modern uygulamaların kullandığı kaynaklardaki artış, internet üzerindeki trafikte hızlı bir büyümeye yol açmıştır. Optik ağlar, daha çok veriyi, daha hızlı ve daha güvenli şekilde iletme imkanı sunmaktadır. Fiber optik kabloların, geleneksel bakır tellere göre bu avantajları vardır. Bu nedenle optik ağların geniş bir uygulama alanı bulması beklenmektedir.

Ancak, optik ağlarda bazı fiziksel uyumsuzluklar ve optik katman kısıtları vardır. Bunlardan biri, optik sinyallerin menzilini kısıtlayan sinyal zayıflamasıdır. Sinyaller iletim sırasında zayıflamakta ve belli bir eşik değerinin altında kullanılmaz hale gelmektedir. Bunu önlemek için optik sinyalleri yeniden güçlendirebilen rejeneratörler kullanılmaktadır. Sinyal rejenerasyonu maliyetli bir işlem olduğundan, optik ağlarda kullanılan rejeneratör sayısını azaltmak önemlidir.

Ağın güvenilirliğini artırmak için, ağ uzerindeki her düğüm çifti arasında iki ayrık yol bulunmalıdır. Böylece, çalışan yol üzerindeki bir ayrıt veri iletiminde başarısız olursa, onarım yolu kullanılarak veri hedefine iletilebilinir. Bütün bu gereklilikleri göz önüne alarak hangi düğümlere rejeneratör yerleştireleceğini belirlemek önemli bir karardır.

Bu tez çalşmasında, onarımlı optik ağlarda sinyal rejeneratörleri konumlandırılması problemi üzerinde çalışmaktayız. Bu problem için bir tamsayılı doğrusal karar modeli geliştirilmiştir. Ancak modelin boyutunun çok büyük olması ve modeldeki başka sorunlar nedeniyle, modelin büyük ağlarda kullanılması pratik değildir. Bu nedenle, problemin çözümü için hızlı çalışan sezgisel bir algoritma önerilmiştir. Belirli düğümlere rejeneratör yerleştirilmesini içeren bir çözüm kümesinin olurluğunu kontrol eden üç farklı yöntem önerilmiştir. İlaveten, önerilen sezgisel algoritmayı da kullanarak eniyi çözümü bulan bir dal-sınır algoritması geliştirilmiştir. Hem sezgisel algoritmaların, hem de dal-sınır algoritmasının performansları, konumlandırılan rejeneratör sayısı ve çözüm zamanları açısından değerlendirilmiştir.

Anahtar sözcükler: optik ağlar, rejeneratör konumlandırılması, dal-sınır algoritması, aktif ve onarım yolları tasarımı.

Acknowledgement

First and foremost, I wish to express my thanks to Dr. Oya Ekin Karaşan for introducing me to this interesting and challenging subject and her guidance throughout the development of this thesis. I would also like to thank her for her considerable encouragement and tolerance during my graduate study. Without her support it would not have been possible for me to complete my graduate study. I would also like to thank Dr. Osman Oğuz and Dr. Bahar Yetiş Kara for their comments and helpful suggestions for the preparation of this text.

I would like to express my gratitude to Dr. İmdat Kara for his support and encouragement during my graduate study. I am grateful to my roommates Demiralp Hatipoğlu for helping me to draw the figures and Tolga Bektaş for his comments. Thanks to Onur Aköz, Ergün Eraslan, Yazgı Tütüncü and Buğra Çamlıca for their patience, understanding and encouraging support.

I would also like to express my deepest gratitude to my family and my friend Sezen Altınova and appreciate their endless support throughout my education and life.

Contents

List of Figures

List of Tables

To My Family. . .

Chapter 1

Introduction

Since the initial deployment of the original ARPANET which was developed by Advanced Research Projects Agency (ARPA) and became the basis for the Internet, Internet architecture has evolved in response to technological progress and user needs. The explosive growth in traffic fed by the increase in the number of users and resources consumed by modern applications has put an ever increasing load on the Internet. A report from the U.S Department of Commerce [3] suggests that the rate at which the Internet has been adopted has surpassed all other technologies preceding it, including radio, television, and the personal computer. A common expectation is the convergence of voice, video, and data communications to happen over the Internet. With this rising of the Internet there have arisen corresponding requirements for network reliability, efficiency, and Quality of Service (QoS) [4]. There are different definitions about QoS. In a general context, QoS is a set of methods and processes which a service based organization implements to maintain a specific level of quality. It is also the ability for an application to obtain the network service it requires for successful operation. To respond to this challenge, Internet service providers (ISPs) examine their operational environment to scale their networks and optimize performance. ISPs employ three complementary technical instruments:

Network architecture It is the abstract structure of the network. It involves the components or object classes of the network, their functions and the relations between them.

Capacity expansion This is employed as a response to the traffic growth by the large ISPs.

Traffic engineering This is used to address the Internet growth, and has attracted significant attention at recent times.

Development of Optical Transport Networks is seen as an important step in the evolution of data transmission. Fiber optic cables are used in the optical networks and the paths through which data are transmitted are referred to as *lightpaths*. These optical networks have emerged as an alternative to traditional copper wire or wireless networks, because they can transfer more data at higher speeds than copper wire. Optical networks could handle the increasing traffic on the Internet since multiple signals can be sent simultaneously. Copper wires, on the other hand, can send only one signal at a much slower speed. It is expected that the amount of information carried on computer and telecommunication networks will grow very rapidly in the future, therefore optical networks are likely to play an important role.

However, optical networks have some disadvantages, in addition to their beneficial properties. Some transmission impairments make some signal routes unusable in optical networks. These impairments are discussed in [13]. One of these impairments is the degradation of the signal. An optical signal has a *signal-to-noise* (SNR) ratio, which may be considered as the quality of the signal. In optical networks, signals are degraded during emission from a node [13]. Below a threshold value, the signal cannot be used. So an acceptable SNR level, which depends on the properties of the optical network, needs to be maintained at the receiver [13].

The amount of the degradation depends on the length of the fiber optic cable. For this reason, with current technology at hand, the physical layer constraints limit the maximum transmission range for optical signals. Beyond this length, the optical signals are degraded, and the SNR becomes too low to be usable.

In such cases the optical signal must be regenerated to increase the transmission range of the optical network. This regeneration process is called *3R Regeneration*, which refers to re-amplification, re-shaping and re-timing of optical signals [23].

Since it is possible to transmit data along the fibers, optical networks are represented using undirected graphs. The vertices (nodes) of the graph represent the receivers and the edges of the graph are used to represent the fiber connections of the network.

The nodes on optical networks are called *optical cross connects* (OXC). There are two types of optical nodes in an optical network. The first one is transparent node and the second is opaque node. Transparent nodes receive and send the optical signal to another node, whereas the opaque nodes receive, regenerate and send the optical signal to its destination. How the opaque nodes regenerate the optical signals is beyond the scope of this work and will not be discussed.

Optical networks can be classified into three main classes according to the regeneration functions of their nodes [18].

Transparent A network is referred to as transparent if its nodes are optical cross connects without any regeneration function.

Opaque When regeneration is available at every node, the network is called an opaque network.

Translucent If regeneration is available on some nodes of the network, then it is called a translucent network.

Each type of network has its advantages and disadvantages. We will briefly mention them. In a transparent network, it is required that lightpaths can reach any destination with required signal-to-noise ratios (SNRs) because payload regeneration is not available en route. Consequently, in such a network there is a critical diameter above which it cannot extend with given optical transmission technology. Opaque networks do not have this drawback, on the other hand cost, space, power and reliability implications prevent using these networks widely [18].

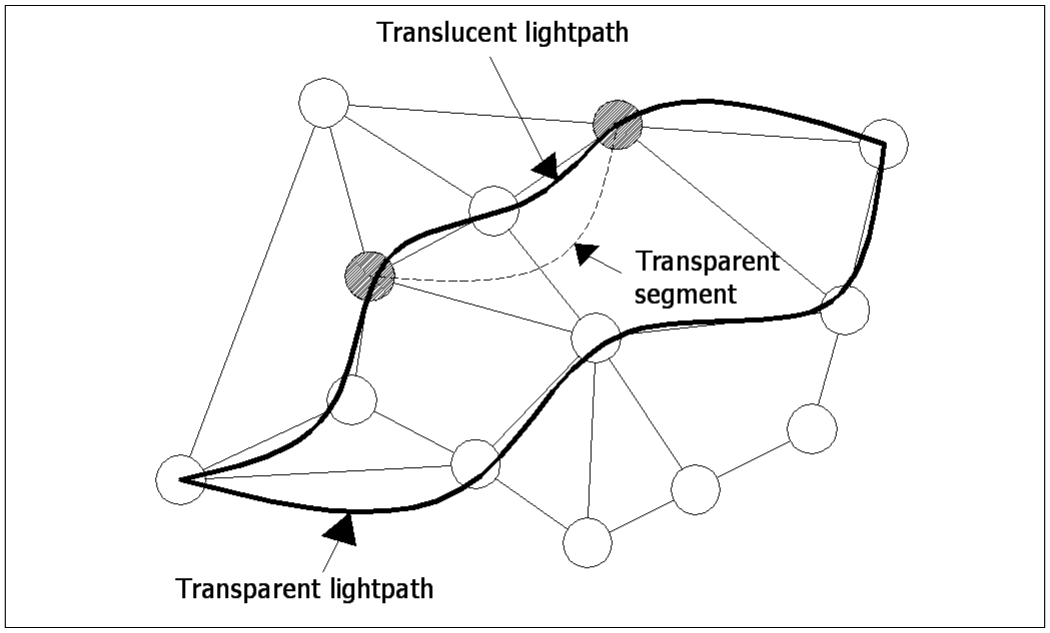

Figure 1.1: Transparent and Translucent Lightpaths

Translucent networks lie between these two extreme cases. To establish a connection under optical impairments, it is necessary to divide some lightpaths into two or more fragments using regenerators [15]. Therefore in a translucent network, there are two types of lightpaths. A lightpath is called *translucent lightpath* if there are some opaque nodes en route, otherwise it is called *transparent lightpath*. Note that the path segments between two opaque nodes are transparent lightpaths [18, 23]. Transparent and translucent lightpaths can be seen in Figure (1.1).

There is strong interest in implementing transparent all-optical networks due to significant cost saving in not regenerating optical signals in transit nodes. Unfortunately, those impairments mentioned above limit the transparent length of a transparent lightpath. At present and in the foreseeable future, regeneration is necessary to establish a lengthy end-to-end lightpath [23].

Using networks of *transparent islands* is another way of getting benefits of both transparent and opaque networks. In such networks, the optical network is divided into islands (domains) each of which is a transparent network itself.

Regenerators are placed on the boundaries of the islands. In an island, signals reach destinations without regeneration. However, for communication between different islands, signals are regenerated on the boundaries [18].

It is known that translucent networks are superior to opaque networks in the sense of cost structure. Performance, however, is another important aspect. Experiments showed that about 20% of regeneration nodes are needed to achieve a performance close to an opaque network [22]. Therefore, translucent networks could be used instead of opaque networks without a decrease in the performance of the network. However, placing the regenerators on the network such that we achieve the performance of an opaque network using the minimum number of regenerators is a challenging problem [15]. Although the necessity of signal regeneration is discussed, efficient regenerator placement problem is not dealt with widely.

Connectivity is the major concern of the network routing protocols deployed today. An important component of Quality of Service (QoS) is the ability of the network to transport data reliably and efficiently.

Connections of the network may be subject to failure [1]. In case of a failure, we need to respond to this failure and transport the data somehow to its destination. For this reason, some techniques are needed to recover the failure and these techniques are called *Recovery Methods*. When a lightpath is broken due to a failed connection of the network, another lightpath must be established in order to transport the signal to its destination. Recovering the failure in fact is constructing this new lightpath, rather than to repair the failed connection.

Recovery methods can be classified based on the scope of the recovery action. Some methods are explained in [18, 24]. *Local Repair*, also known as *Span-based Restoration*, aims to protect against single link or node failures. In this case, failure detection and restoration are carried out between the two terminating nodes of the failed connection. So, the recovery path shares overlapping portions with the working path. *Global Repair* which is also known as *Path Restoration* or *Path-based Restoration* aims to protect against any link or node failure on the working path. In many cases the restoration path may be selected to be link

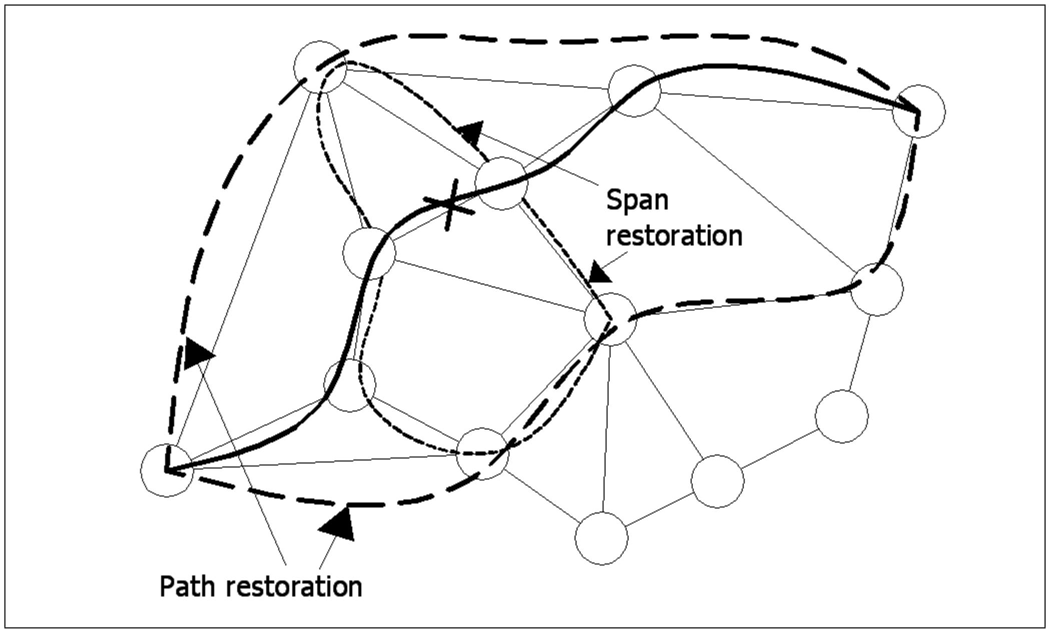

Figure 1.2: Restoration Methods

or node disjoint from the working path to protect against all link and/or node failures on the working path [18, 24]. In a span-restorable network, the failure detection and restoration are carried out between the two terminating nodes of a failed span [18]. Since the response is local, span-restoration is thought of as being fast but with relatively lower space capacity efficiency. In contrast, a path-restorable network is more efficient in spare capacity but often has a slower restoration speed [18]. Examples of span-based and path-based restoration can be seen in Figure (1.2).

The networks, in which path-based restoration is preferred, should be designed such that there is a restoration-path disjoint from the working-path in the network. *Diversity* is a relationship between ligthpaths. Two lightpaths are said to be diverse if they have no single point of failure. Diversity is often equated to routing two lightpaths between a single pair of points, or different pair of points, so that no single failure will disrupt them both [13].

For this reason, the networks must be designed such that in case of a failure the network should still remain active. This problem is called *survivable network* *design problem* (SDNP). This problem has applications to the design of survivable telecommunication networks [10]. With the introduction of optic technology in telecommunication, designing a minimum cost survivable network has become a major objective in the telecommunication industry [12]. In optical networks, 2-edge connected networks in which there are at least two edge disjoint paths between each pair of terminals have shown to be cost effective and provide an adequate level of survivability [10]. The 2-edge connected subgraph problem, a special case of SDNP, is known to be NP-hard [14].

Next step in network survivability is the *Two edge-disjoint Hop-constrained Paths Problem* (THPP). Regarding the reliability of a telecommunication network, having two edge-disjoint paths is often insufficient [12]. The alternative path (restoration path) could be too long to guarantee an effective routing, which may result in degradation of the signal. In such cases, *L-path* requirement guarantees exactly the needed quality for the paths [12]. L-path requirement means that there can be at most L nodes in a path.

In this thesis, an optical network which is assumed to transmit data between two terminal nodes will be designed. In the design problem, a given transparent network will be transformed into a survivable translucent network to gain the range of an opaque network with minimum cost. Since the cost of regenerator nodes is the dominant factor, deciding on where to place regenerator nodes is an important problem. In this thesis, an effective method is proposed for placing the regenerator nodes and routing the signals through two-edge disjoint paths.

1.1 Problem Definition

Having defined optical networks and their physical impairments, we want to state the problem for which we will propose solution methods in this study.

Let $G(V, E)$ be an undirected graph with n vertices (nodes) and m edges, and $c(E)$ be the vector representing the lengths of the edges in G. Assume that this graph represents an optical network and edge lengths stand for the amount of attenuation the signal degrades through the edge. Our aim is to send a signal from each vertex to all other vertices within the prespecified degradation limit.

Assume that, path-based restoration as explained before in this chapter, is to be incorporated in this optical network. Therefore, we need to find alternative ways to send the signal. The more number of disjoint paths we have, the more reliable the network is. Nevertheless, as the number of disjoint paths increases, it becomes harder to find these paths. Finding the number of disjoint paths to make the network reliable enough is beyond the scope of this work. Generally, two disjoints paths are assumed to be sufficient in the literature [10] and we adopt this assumption. By this way, the reliability of the network is increased. Therefore, we are going to find two edge-disjoint paths between all pairs of vertices of G.

The second point is the degradation limit. As explained before, a signal attenuates in time after it is sent from a vertex and after a period it will be completely lost. This amount of degradation (threshold value) after which the signal becomes useless is called the *degradation limit*. Therefore the signal must arrive at its destination within the degradation limit otherwise it would be lost. This means, the lengths of the paths we find through which the signal is sent must be within the degradation limit. There is no problem if we can find paths that have reasonable lengths. What can be done if there are not such paths? In such cases, a device called regenerator could be used to regenerate the signals. Regenerators are devices that receive and resend the signal in order to realize the regeneration process mentioned in this chapter. By this way, the signal will be recovered and will not be lost until degradation limit is reached. Think of a path that passes through a vertex on which there is a regenerator. We can divide this path into two, from the vertex with regenerator. First part starts from the source vertex and ends at the regenerator vertex, and the second part starts from regenerator vertex and ends at the terminal vertex. Each part of the path between neighboring opaque nodes will be referred to as *sub-paths*. Remember that these sub-paths are the same with the transparent path-segments described previously in this chapter. Having defined what a sub-path is, it can easily be seen that a path could have more sub-paths than two. The number of sub-paths depends on the number of regenerators that the path passes through. If there is one regenerator, then there are two sub-paths. With four regenerators, for example, there will be five sub-paths. Using regenerators, we can relax the degradation limit constraint. Now, it is not necessary for the paths as wholes to be within the degradation limit, it is sufficient if each sub-path is within the degradation limit.

It is assumed that there are two edge-disjoint paths between all pairs of the graph. This means that using sufficient number of regenerators, paths that satisfy degradation constraint could be constructed. The regenerator, however, is an expensive device so we want to minimize the total number of regenerators placed on the network. Therefore, our problem is to determine the minimum number of regenerators that would be sufficient to satisfy the degradation constraint and to decide where the regenerators should be placed.

1.2 Organization of the Thesis

In this thesis, we formulate the regenerator placement problem as an integer linear program. Since the solution of this problem, due to its size, is very hard, we develop and implement a heuristic approach. After this, we propose a branch and bound algorithm which employs the heuristic, to find the optimal solution. We obtain numerical results for these approaches on sample networks and compare their results and efficiencies with those of the solution methods which have been proposed in [24, 25].

The thesis is organized as follows: Chapter 2 summarizes the studies in the literature on network reliability and regenerator placement and on finding edge-disjoint paths. In Chapter 3, an integer linear formulation of this problem, the difficulties in modelling and solving the problem are discussed. Chapter 4 includes the methods to check the feasibility of a fixed set of regenerators placed at specific vertices and explains why even the feasibility check is a difficult problem. In Chapter 5, a heuristic algorithm is proposed and numerical results are obtained. Another method which finds the optimal solution of the problem is developed in Chapter 6. Numerical results are also provided in this chapter. Finally, Chapter 7 concludes the thesis.

Chapter 2

Related Work

As stated in Section (1.1) our problem has two main aspects. The first one is network reliability and the second one is respecting the degradation limit i.e. regenerator placement. Both subjects have been studied and research about them could be found in the literature. In this chapter, we will give the research about these subjects. These two subjects seem to be independent but in fact they are inter-related. Therefore we want to discuss the literature in which both problems are studied together first, and then we will look at the studies in which these subjects are examined individually.

2.1 Network Reliability and Regenerator Placement

Our problem consists of two main components, the first one is network reliability and the second one is deciding on the places of regenerators. In our problem, survivable networks are designed according to path-restoration technique, hence two edge-disjoint paths are to be found between each pair $(s, t) \in V$ of the graph. Related work about this subject is given in Section (2.3). The work done about the second part of the problem, regenerator placement, is given in Section (2.2).

In these works, only one part of our problem is studied separately, however, both parts are closely related in our analysis. Works that take both components into account are given in this section. Unfortunately this subject has not been widely studied. We have found only two published work on this problem.

Capacities of the fiber links are also important since data cannot be transmitted if the capacity of a link is exceeded. For this reason, the traffic flow on the network should be controlled to prevent exceeding the capacities in data transmission. This control process is called *traffic engineering*.

Yetginer, who mainly focused on traffic engineering, partially studied regenerator placement in [24]. In a closely related article [25], the same subject is investigated. In both studies, traffic engineering is the main focus. They have introduced a nonlinear integer program to solve the problem. However, due to its size, they could not solve the program. Two heuristic algorithms are proposed but they have not discussed the efficiency of their solutions. Besides, their nonlinear integer program may fail to find the optimal solution even if its size were reasonable. To overcome this, authors in [24, 25] bring in an additional assumption to restrict the solution space. This will be fully discussed in Chapter 3. Our analysis, however, will relax this assumption.

2.2 Regenerator Placement

In [18], there is a different approach to optical signal recovery, where regenerators (opaque nodes) are used to detect failures. Two basic restoration techniques, path-based and span-based restoration, were described in Chapter (1). The approach in [18] is an intermediate and more general option, called segment-based survivability. Segment-based survivability schemes carry out restoration of a failed path segment between the opaque nodes upstream and downstream of the actual span failure. Remember, this segment is the transparent fragment (sub-path) in other words signals are not regenerated except from the source and terminating nodes of the sub-path. Based on segment-based survivability, opaque node (regenerator) placement problem is solved using a heuristic algorithm in [18]. A method to find the optimal solution and alternative optima is described and implemented. However the optimal solution technique is nothing but complete enumeration. There is not any intelligent search technique employed to traverse the solution space. Therefore this method is not efficient. The optimal and heuristic solutions are compared. Although the heuristic solutions are worse than the optimal solution, the running times of the heuristic are very effective.

Routing and wavelength assignments are considered jointly in [23]. Although, this study considers the need for regeneration, it uses a different approach. Instead of adding regenerator nodes to the network, some nodes that have spare capacity, are used as regenerators. We are not interested in the mechanism of using ordinary nodes as regenerator nodes. The nodes which will be used as regenerators are determined using two different algorithms. These algorithms take the blocking probability of the signal due to wavelength unavailability into account. After deciding on the places of the regenerator nodes, routing algorithms are employed. Survivability of the network is not considered in this study [23].

Kim and Seo [15] studied regenerator problem which is similar to the problem in [23]. They did not consider network survivability, but included another factor which is the blocking of the lightpaths throughout the network. They used two approaches to the problem, the first one being a minimal-cost placement which minimizes the blocking of lightpaths using dynamic programming, the second being a heuristic for locating the signal regeneration nodes. A comparison between these two algorithms and other ones proposed in different studies is also given in [15].

In the research done by Yang and Ramamurthy [22], regenerator placement problem is examined. In this study, regenerator placement problem is solved offline, i.e. at the design phase of the network. Routing algorithms are used after regenerator places are determined and fixed. They have solved the problem using two heuristic methods and measured the performance of the system with simulation models. Again survivability of the network is not considered.

2.3 Network Reliability

Network reliability is an important concept in network design problems especially in communication networks. The problem of network reliability arises from the fact that the edges may be subject to failure [1]. A signal is sent from a vertex to another via some set of edges and vertices. A network reliability problem is to compute the probability that the specified node pair can communicate at a given time i.e. no arc on the path fails to transmit data. Given the probabilities of the edges, the aim is to design more reliable networks. Most network reliability problems are NP-hard [1]. A survey of this problem can be found in [1]. Since we are not interested in probability of communication, we will not go into details here.

The concept of network reliability or network survivability is a bit different in our problem. No matter how reliable a path between a node pair is, the path may fail to transport data from source to the terminal. If a failure occurs, there is a need for a responding strategy about transporting the data from the source to its destination. The responding strategies have been mentioned in Chapter (1). In our problem, path restoration strategy is preferred. Therefore, we need to establish a restoration path in the designing phase of the network. That is to say two paths are to be constructed between a node pair, one as the working path and the other as the restoration path. These two paths can share some edges or can be either edge-disjoint or vertex-disjoint. In our problem edge-disjoint paths are preferred. For this reason, two edge-disjoint paths, lengths of which should be within degradation limit, must be constructed.

Network reliability and survivability are improved if disjoint paths are used between source and terminal nodes in the network [9]. Assuming that the edges and nodes of a network have known reliability, the reliability of a path between two nodes can be computed. The number of edges in a path and the number of disjoint paths are given as an input. In [9], an algorithm is proposed to identify the set of K-most reliable paths. This algorithm calculates the reliability of the paths and outputs them in the decreasing order.

Survivable network design problem (SNDP) is an important and widely studied problem. In SNDP problems, a sub-graph of the main graph is formed such that there are two disjoint paths between each pair of the graph. Usually the aim is to minimize the total cost of the subgraph and this resulting problem is NP-hard [11]. Given a graph which is 2-edge connected, a spanning subgraph with minimum number of edges is found. Huh proposed an algorithm is to find the subgraph in [11]. Huygens et al. [12] has also studied such a problem. They try to find a subgraph where the number of the edges in the disjoint paths are constrained. An integer program is formulated and facets of the problem polyhedron are defined. Different versions of this problem in which length of the paths are also constrained could be found in the literature. Kerivin and Mahjoub [14] are also interested in the polyhedron of a special SNDP. Partition inequalities are used in this study and they showed that this problem can be solved in polynomial time in special type of graphs.

Without degradation limit, the problem of finding edge-disjoint paths is widely studied. There are two major kinds of disjoint paths in the literature. In the first type, disjoint paths are found between different pairs of nodes. Such a problem arises in the telecommunication area. Connections interfere with each other if the paths share some edges. The question of how many connections can be made simultaneously could be answered by solving this problem. Most of these problems are NP-complete and an example could be found in [16]. This problem, however is not closely related with our problem.

The second type of problem is to find disjoint paths between a single pair of nodes. Different studies on this problem could be found. Vygen discussed some edge-disjoint problems in his work and showed that they are NP-complete [21]. Brandes et al. developed a software package which includes algorithms for different disjoint problems in planar graphs in [6]. Coupry studied a maximum cardinality set of edge-disjoint paths problem. In this problem, the graph is non-weighted and the aim is to find maximum flow between two nodes where each edge has unit capacity. He proposed a linear algorithm for this problem in [8]. Conforti, Hassin and Ravi used a different approach to this problem. They studied the problem of constructing edge-disjoint paths using previously found edge-disjoint paths. Assume that, we look for edge-disjoint paths for the pair (s,t) and say we already know edge-disjoint paths between pairs (s,r) and (r,t). These known paths are combined to solve the problem which is equivalent to the stable matching problem [7].

The problems mentioned above often do not have objective functions. When an objective function, such as minimizing the cost or weighted cost of disjoint paths etc., is inserted to the problem, it gets even harder. It is shown in [17] that the problem of finding two edge-disjoint paths and minimizing the length of the longer one is strongly NP-complete. The underlying network of this problem can be directed or undirected. This problem is of great interest to us, since it is closely related to our problem. Remember the impairments we have discussed in Chapter 1. We have overcome the impairments using regenerator nodes in networks. Assume that we do not want to use regenerator nodes and want to know which pairs can communicate without regeneration. Since two paths are to be established, one working and one restoration path, we should solve the problem stated above. If the length of the longer path is within the degradation limit given, then we can conclude that the pair can communicate without regenerators.

A similar problem to this one, is to find a pair of disjoint paths, shortest pair, so as to minimize the total length of the two paths. Solution techniques to this problem could be found in the literature. For example, Suurballe proposed an algorithm to solve both node disjoint and edge-disjoint versions of the problem [19]. Suurballe and Tarjan improved the algorithm in [19]. They again tried to minimize the total length of the paths, but this time they found the paths between a single source and each possible sink nodes [20]. Banerjee et al. [5] has proposed a parallel algorithm for the two problems, which are single sink and multi sink versions. They used Dijsktra's shortest path algorithm [2] iteratively to find the shortest pair of disjoint paths. In all problems, the underlying graphs are directed.

Chapter 3

Difficulties in Modelling the Problem

Our problem, as defined in Section (1.1) is to establish two edge-disjoint paths, one working path and one restoration path, between each pair of vertices of the graph such that the lengths of the paths have to be within the given degradation limit. Remember that our optical network is represented by the undirected graph, $G(V, E)$. If paths with such lengths cannot be found, regenerators will be placed on some nodes of the graph. In this case, the paths will be divided into fragments and length of each fragment must be within the pre-specified degradation limit. Here we want to define the length of a path. Although length of a path is a well known term, it is a bit different here, since we are interested in not only the entire path but also the path fragments. Say $L(P)$ denotes the length of path P and l_i denotes the length of path fragment i. Then the length of path P, in which there are k path segments will be, $L(P) = \max\{l_1, l_2, \ldots l_k\}$

In order to find the optimal solution of our problem, we have attempted to formulate an integer linear program to solve it. Hereafter, this integer linear program will be denoted by Model1. Model1 in its simplest form may fail to find the optimal solution. We have developed a technique to overcome this problem. In this chapter we will introduce Model1, discuss its drawbacks and show how to use it iteratively to find the optimal solution. Although G is an undirected graph, directions are important in signal transportation. For this reason, we transform $G(V, E)$ into a directed network $\widehat{G}(N, A)$. To do this, each undirected edge $\{i, j\} \in E$ is replaced by two directed arcs (i, j) and (j, i), both with cost c($\{i, j\}$) [2]. Therefore, the directed network $\widehat{G}$ which will be used in formulating Model1 and other integer linear programs (ILPs) has n nodes and $2m$ directed arcs.

3.1 Integer Linear Program For the Entire Graph

To find the optimal solution to this problem a mathematical program is formulated. This model shows some similarities with the one introduced in [24, 25], however, the latter one is not linear. The model is to find the nodes where the regenerators should be placed and simultaneously decide on a working and a restoration path between any pair of vertices. Definitions of the decision variables and explanations of the constraints used in the model are given below:

Objective:

$$Minimize \sum_{i \in N} r_i \tag{3.1}$$

Subject to:

$$\sum_{j:(i,j)\in A} x_{ijk}^{st} - \sum_{j:(j,i)\in A} x_{jik}^{st} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}t; \\ 0, & i \neq s{,}t; \end{cases} \qquad \forall i \in N, \forall k, \forall (s,t) s < t \tag{3.2}$$

$$w_{jk}^{st-} \geq w_{ik}^{st+} - M(1 - x_{ijk}^{st}) + c_{ij}x_{ijk}^{st}, \qquad \forall (i,j) \in A, \forall k, \forall (s,t) s < t \tag{3.3}$$

$$w_{ik}^{st+} \geq w_{ik}^{st-} - Mr_i, \qquad \forall i \in N, \forall k, \forall (s,t) s < t \tag{3.4}$$

$$w_{ik}^{st-} \leq R_{max}, \qquad \forall i \in N, \forall k, \forall (s,t) s < t \tag{3.5}$$

$$\begin{aligned} \begin{matrix} x_{ij1}^{st} + x_{ij2}^{st} \leq 1 \\ x_{ij1}^{st} + x_{ji2}^{st} \leq 1 \end{matrix}, &\qquad \forall (i,j) \in A \quad (3.6) \\ w_{sk}^{st-} = w_{sk}^{st+} = 0, &\qquad \forall k, \forall (s,t) s < t \\ x_{ijk}^{st} \in \{0,1\},\ r_i \in \{0,1\} & \\ \text{all variables} \geq 0 & \end{aligned}$$

Here x_{ijk}^{st}, r_i, w_{ik}^{st+}, w_{ik}^{st-} are the decision variables and c_{ij} is a parameter of Model1. They are explained below:

In this formulation k takes values of 1 and 2, k equals to 1 represents the working path and k equals to 2 represents the restoration path.

$$x_{ijk}^{st} = \begin{cases} 1, & \text{if } k^{th} \text{ path for pair } (s,t) \text{ includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

$$r_i = \begin{cases} 1, & \text{if a regenerator is placed on node } i \\ 0, & \text{otherwise} \end{cases}$$

The length of arc (i,j) is given by c_{ij}. w_{ik}^{st-} and w_{ik}^{st+} denote the path lengths for k^{th} path from s to t into and out of node i, respectively. R_{max} is the maximum allowable length (attenuation) of a path segment. M is a large number and it is sufficient to set M equal to R_{max}.

Explanation of the constraints:

- Only the number of regenerators placed is included in the objective function (3.1) of Model1, since we want to make the communication possible with minimum number of regenerators.
- Constraints (3.2) are the flow conservation constraints from node s to node t for both paths.
- The distance (attenuation) travelled up to a node i on path k from s to t is denoted by variable w_{ik}^{st-} and called length of inflow. Length of the inflow into node i is equal to the sum of length of the outflow from node j which precedes node i in the path and length of arc (i,j). If arc (i,j) is on k^{th}

path, constraint (3.3) calculates the length of inflow into node i. Otherwise, no additional constraint is placed on the inflow value for that node.

- Variable w_{ik}^{st+} denotes the length of the outflow from node i, i.e. the amount of attenuation of the signal. Constraint (3.4) calculates the length of the outflow. If regeneration occurs at that node then the length is reset to zero and only the length of the path after that node will be taken into account.
- Constraint (3.5) enforces that the length travelled up to any node in a path is within the given limit. By this way, only the paths that satisfy our degradation limit or length constraint are taken into account.
- We need to find two edge-disjoint paths, so an edge cannot be on both paths. Constraint set (3.6) forces the paths to be edge-disjoint. These constraints are formed for both arc (i, j) and arc (j, i), resulting in four constraints. For example for arcs (1,2) and (2,1) we have

$$\begin{aligned} x_{121} + x_{122} &\leq 1 \\ x_{121} + x_{212} &\leq 1 \\ x_{211} + x_{122} &\leq 1 \\ x_{211} + x_{212} &\leq 1 \end{aligned}$$

  According to these constraints if arc (1,2) is used in the working path arc (1,2) or arc (2,1) cannot be used in the restoration path, but arc (2,1) can be used in the working path since when $x_{121} = 1$, $x_{122} = 0$ and $x_{212} = 0$ we have $x_{211} \leq 1$

However, size of Model1 is very large, growing in the order of $\mathcal{O}(mn^2)$, hence this approach is not applicable except for very small networks. For example, Model1 is formulated for a sample network topology (Figure (3.1)), obtained from [18], with 14 nodes and 21 edges. The optimal solution of this problem could not be found in 24 hours. Therefore we are going to examine if Model1 could be used for only one pair of vertices rather than for the entire graph.

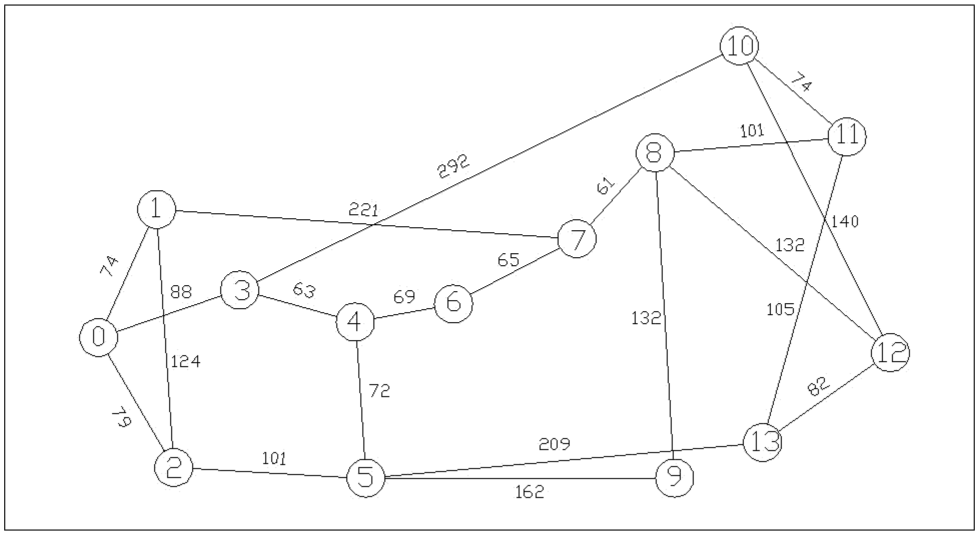

Figure 3.1: 14-node network

3.2 Integer Linear Program For a Specific Pair

In Section (3.1), we have given Model1 to solve the problem for the entire graph and have seen that the size of Model1 is too large to be of practical use. Our experimentation has also confirmed that we can't go beyond $n \geq 10$. Here, we are going to formulate another ILP for a specific starting node s and terminating node t and will try to use it to find the optimal solution. New ILP will be referred to as Model2.

Objective:

$$Minimize \sum_{i \in N} r_i$$

Subject to:

$$\sum_{j:(i,j)\in A} x_{ijk} - \sum_{j:(j,i)\in A} x_{jik} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}t; \\ 0, & i \neq s,t; \end{cases} \qquad \forall i \in N, \forall k$$

$$w^-_{jk} \geq w^+_{ik} - M(1 - x_{ijk}) + c_{ij}x_{ijk}, \qquad \forall (i,j) \in A, \forall k$$

$$w^+_{ik} \geq w^-_{ik} - Mr_i, \qquad \forall i \in N, \forall k$$

$$\begin{aligned}
w_{ik}^{-} &\leq R_{max}, && \forall i \in N, \forall k \\
\left.\begin{aligned} x_{ij1} + x_{ij2} &\leq 1 \\ x_{ij1} + x_{ji2} &\leq 1 \end{aligned}\right. &, && \forall (i,j) \in A \\
w_{sk}^{-} = w_{sk}^{+} &= 0, && \forall k \\
x_{ijk} \in \{0,1\},\ r_i &\in \{0,1\} \\
\text{all variables} &\geq 0
\end{aligned}$$

For a graph with n nodes and m edges, Model2, which includes only a specific pair, has $5n+4m$ variables and $6n+8m$ constraints. For example, in a graph with 32 nodes and 50 edges, Model2 would have 368 variables and 608 constraints. Although these numbers do not seem excessively large, our experimentation showed that Model2 is hard to solve and needs long solution times. Besides, restricting our attention to a pair of nodes at a time constitutes a major restraint for our problem. It is interesting to note that though the pairs are independent from each other, they are inter-related at the same time. Since an arc included in a path for pair (s_1, t_1) can also be used in a path for pair (s_2, t_2), the pairs are independent. However, if a regenerator is placed on node i for pair (s_1, t_1), it can be used for regeneration for pair (s_2, t_2), too. For this reason, the pairs cannot be considered separately, otherwise the solution for the entire graph may be sub-optimal, even though each solution for specific pairs is optimal.

We might still employ this formulation to find near optimal solutions nonetheless. However, there is another major drawback of this model apart from its size and long solution times. This drawback will be explained in Section (3.3).

3.3 Problem With the Integer Linear Program

Let us look at the graph given in Figure (3.2). Say this graph represents a communication network with a degradation limit of 7, and we want to solve our problem on this network. Since the graph is very small, we can use Model1 to solve the problem.

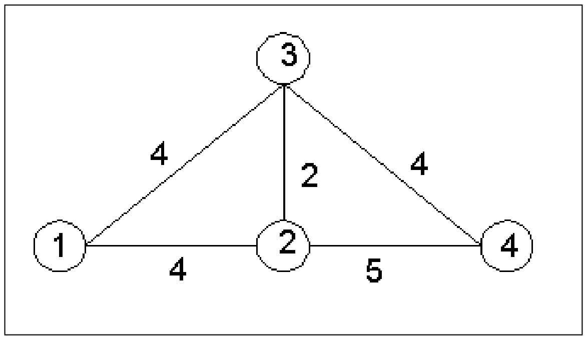

Figure 3.2: A small example

Solution of Model1 tells us to place three regenerators on nodes 1, 2 and 3. In addition, Model2 can be used to solve the problem for the pair (1,4) and in the solution two regenerators are placed on nodes 2 and 3. Here one can see that there is a problem with the solutions of both Model1 and Model2. One and two regenerators are sufficient for the pair and for the entire network, respectively. For example, for pair (1,4) $1 \rightarrow 3 \rightarrow 4$ could be selected as working path and $1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 4$ as the restoration path. With a regenerator placed on node 3, lengths of both paths will be within the degradation limit. In other words our models might end up in suboptimal solutions.

A critical point is that, both Model1 and Model2 will result in simple paths i.e. no vertices are repeated. Looking at constraints (3.3) and (3.4), it can be seen that if a path passes a node more than once, w_{ik}^{+} and w_{ik}^{-} variables would have different values to take and they will take the largest value. This seems normal at first. Since the signal attenuates at every edge it passes, it is not logical to use a vertex or edge more than once. However, things get more complicated once we start thinking about regenerators on the paths.

Notice that, the restoration path in the optimal solution is a non-simple path. Since the signal is regenerated at node 3, it is better to select this non-simple path, which is longer than the simple path, rather than to place an additional regenerator on node 2.

We want to discuss the reason of the problem in detail.

Table 3.1: Path lengths.

Path Name	Path	Length
P_1	1–2–4	9
P_2	1–2–3–2–4	$max\{6,7\} = 7$
P_3	1–2–3–4	$max\{6,4\} = 6$
P_4	1–3–2–4	$max\{4,7\} = 7$
P_5	1–3–4	$max\{4,4\} = 4$
P_6	1–3–2–3–4	$max\{4,8\} = 8$

In the graph given in Figure (3.2) there are six different paths between pair (1,4). Assuming there is a regenerator on node 3, lengths of the alternatives are calculated. The paths and their lengths are given in Table (3.1).

With a degradation limit of 7, it is seen from Table (3.1) that all paths but P_1 and P_6 are feasible paths. This shows us that the paths in the feasible solutions do not have to be simple paths. Assuming, P_2 is selected; some of the constraints of Model2 for this graph would be realized as follows.

$$\begin{array}{rr}
r_3 = 1 & w_3^+ = 0 \\
w_1^+ = 0 & w_2^- \geq 4 \\
w_2^+ \geq w_2^- & w_3^- \geq w_2^+ + 2 \\
w_3^+ = 0 & w_2^- \geq 2 \\
w_4^+ \geq w_4^- & w_4^- \geq w_2^+ + 5
\end{array}$$

These constraints imply that

$$\begin{aligned}
w_2^+ \geq w_2^- &\geq 4 \\
w_4^- &\geq 9
\end{aligned}$$

This means that, the length of P_2 is 9 units, which is equal to the length of P_1. In fact, it is smaller as seen from Table (3.1). But, Model2 has treated P_2 as P_1 and ignored the benefit we gained using the regenerator. This shows that Model2 is not appropriate to work with paths that are not simple. However, as just argued, there may be paths which are not simple but feasible. In addition, it may be

Table 3.2: Modified Graph with EC=2.

	Original Graph	Modified Graph
Vertices	$V = \{1, 2, ..., n\}$	$V' = \{1, 2, ..., n, n+1, ..., 2n\}$
Edges	$E = (i, j)$	$E' = (i, j) \cup (i, j+n) \cup (i+n, j) \cup (i+n, j+n)$

necessary to use those paths to find an optimal solution. Therefore, the model must be modified in order to handle non-simple paths.

3.4 Solution Approach to the Problem in Model2

Since the model can examine only simple paths, we could try to modify the graph so that the paths that are not simple could be treated as simple paths. Our proposed modification is in fact, forming replicas of nodes and edges. The number of replicas we form is called; *the expanding coefficient*, which will be denoted as *EC*. For example, if we have two replicas along with the original node in the modified graph, then the expanding coefficient will be three.

First we want to study the case in which the expanding coefficient is two, which means we duplicate each node. The solution idea is that, if we duplicate each node, some non-simple paths will become simple paths since the path will visit the original node and then its duplicate node. This definition will be used later. By this way, each node is visited only once in the modified graph, although some nodes on the original graph are visited twice. If the nodes are duplicated, the edges must also be multiplied. We will represent the modified graph as $G'(V', E')$ and the corresponding directed network as $\widehat{G}'(N', A')$. The vertices and edges of the original and the modified graph are shown in Table (3.2).

As it is seen from Table (3.2), we have an additional vertex for each vertex in V. Since there are n vertices in V, vertices i and $i + n$ represent the same vertex in the modified graph, $\widehat{G}'$. For simplicity, we will show the duplicate of vertex i

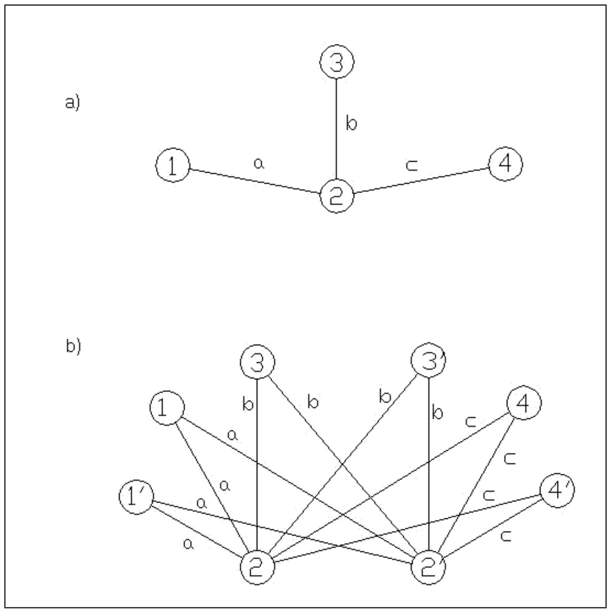

Figure 3.3: a)Original Graph b)Modified Graph (EC=2)

with i'. In addition, to simplify our representation, we assume that $(i')'$ is i, in fact. This assumption is used to define the domains of the constraints. Similarly, we have four edges in the modified graph, instead of one original edge. By this way, using the modified graph as the input, the model we constructed is able to evaluate the paths in which some vertices are visited twice. Unfortunately, the size of the graph grows very fast. Now we have $2n$ vertices and $4m$ edges instead of n and m, respectively. To show how this modification changes a graph, original and modified versions of a simple graph are provided in Figure (3.3). As it is seen, although the original graph is very small, it becomes very complex after the modification.

3.4.1 Modified Integer Linear Program

ILP of the modified graph, which will be referred to as Model3 is very similar to Model2. The domains of the constraints and objective function are changed according to the modified graph. In addition, new constraints are added to Model2. Model3 is given and the new constraints are explained below:

Objective:

$$Minimize \sum_{i \in N'} r_i$$

Subject to:

$$\sum_{j:(i,j)\in A'} x_{ijk} - \sum_{j:(j,i)\in A'} x_{jik} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}s; \\ 0, & i \neq s,t; \end{cases} \qquad \forall i \in N', \forall k$$

$$w^-_{ik} \geq w^+_{ik} - M(1 - x_{ij}) + c_{ij}x_{ij}, \qquad \forall (i,j) \in A', \forall k$$

$$w^+_{ik} \geq w^-_{ik} - Mr_i, \qquad \forall i \in N', \forall k$$

$$w^-_{ik} \leq R_{max}, \qquad \forall i \in N', \forall k$$

$$\begin{array}{c} x_{ij1} + x_{ji1} + x_{ij2} + x_{ji2} \leq 1 \\ x_{ij1} + x_{ji1} + x_{ij'2} + x_{j'i2} \leq 1 \\ x_{ij1} + x_{ji1} + x_{i'j2} + x_{ji'2} \leq 1 \\ x_{ij1} + x_{ji1} + x_{i'j'2} + x_{j'i'2} \leq 1 \end{array}, \qquad \forall (i,j) \in A' \qquad (3.7)$$

$$w^-_{sk} = w^+_{sk} = 0, \qquad \forall k$$

$$r_i = r_{i'}, \qquad \forall i \in N \qquad (3.8)$$

$$x_{ijk} \in \{0,1\},\ r_i \in \{0,1\}$$

$$\text{all variables} \geq 0$$

In this formulation:

- Constraint (3.7) is a bit different from constraint (3.6). Since we have expanded the graph, there is no need to use an edge of E' more than once in a path. Instead, we use the duplicate of that edge if necessary. Notice that the number of constraints used for disjoint paths gets bigger as EC increases.

- Constraint (3.8) is added to Model3. Since we have duplicated nodes, we have now two nodes representing the same node. Therefore, if we have placed a regenerator on a node, this means we also have a regenerator on its duplicate node.

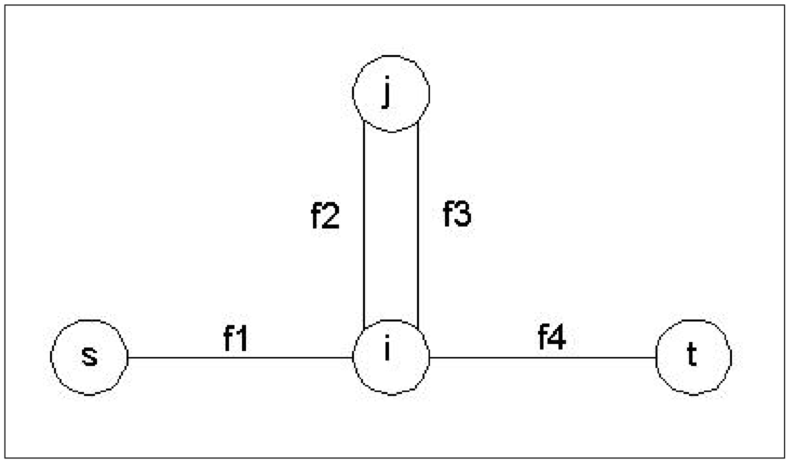

Figure 3.4: Rings in a path

Model3 has $10n + 16m$ variables and $17n + 48m$ constraints, in other words, its size is more than twice of the size of Model2. We have stated that, solving Model2 takes long time, naturally Model3 is harder to solve. For this reason, we have tried to find additional constraints to make the feasible area of the linear programming relaxation smaller without omitting integer feasible solutions. But before giving the constraints we will propose a theorem which will make the constraints valid.

Here, we want to make some definitions which will be used in the following theorems. The first one is the preference concept. Say there are two paths P_1 and P_2. If $L(P_1) \leq L(P_2)$ we say that P_1 is preferred to P_2 and show this with $P_1 \succ P_2$. The second one is the *ring* in a path. If each edge appears exactly once in the cycle portion of a non-simple path, then this portion is called a ring and the path is said to have a ring. For example, in the path $f_1 \cup f_2 \cup f_3 \cup f_4$ of Figure (3.4) the cycle portion $f_2 \cup f_3$ forms a ring. But the cycle portion $f_2 \cup f_2$ of the path $f_1 \cup f_2 \cup f_2 \cup f_4$ is not a ring.

It is obvious that a cycle in a path only occurs if there is a regenerator on the cycle. It seems it is possible to eliminate the rings of a path. In a ring, the path goes to the regenerator node through a sub-path, and comes back through another sub-path. It seems reasonable to use the same sub-path to go and come back. This will be shown formally in the following Lemma.

Lemma 1 *If there is a feasible $s-t$ path P that has a ring, then there is always another feasible $s-t$ path P' free of rings which is preferred to P.*

Proof Look at the graph given in Figure (3.4). In this graph, s and t are the terminal vertices, j is the regenerator vertex and i is the vertex which is visited more than once. f_1, f_2, f_3 and f_4 are the path fragments between the vertices. The possible paths between pair (s,t) and their lengths are given in Table (3.3).

There are four possible cases:

1. $\begin{array}{l} L(P_4) = l_{f_1} + l_{f_2} \\ L(P_2) = l_{f_1} + l_{f_2} \end{array} \Rightarrow L(P_2) \leq L(P_4) \Rightarrow P_2 \succ P_4$

2. $L(P_4) = l_{f_1} + l_{f_2} \Rightarrow l_{f_1} + l_{f_2} > l_{f_3} + l_{f_4}$
   $L(P_2) = l_{f_2} + l_{f_4} \Rightarrow l_{f_4} > l_{f_1} \Rightarrow L(P_3) = l_{f_3} + l_{f_4}$
   $L(P_3) \leq L(P_4) \Rightarrow P_3 \succ P_4$

3. $\begin{array}{l} L(P_4) = l_{f_3} + l_{f_4} \\ L(P_3) = l_{f_3} + l_{f_4} \end{array} \Rightarrow L(P_3) \leq L(P_4) \Rightarrow P_3 \succ P_4$

4. $L(P_4) = l_{f_3} + l_{f_4} \Rightarrow l_{f_4} + l_{f_4} > l_{f_1} + l_{f_2}$
   $L(P_3) = l_{f_1} + l_{f_3} \Rightarrow l_{f_1} > l_{f_4} \Rightarrow L(P_2) = l_{f_1} + l_{f_2}$
   $L(P_2) \leq L(P_4) \Rightarrow P_2 \succ P_4$

Enumerating all possibilities, we see that there is always a path free of rings (P_2 or P_3) which is preferred to the one with rings (P_4). With this Lemma we are able to eliminate paths with rings.$\square$

Hence the following list of constraints can be incorporated into our model without eliminating the optimal solution.

Table 3.3: Rings in a path.

Path Name	Path	Path Length
P_1	$f_1 \cup f_4$	$L(P_1) = l_{f_1} + l_{f_4}$
P_2	$f_1 \cup f_2 \cup f_2 \cup f_4$	$L(P_2) = \max\{l_{f_1} + l_{f_2}, l_{f_2} + l_{f_4}\}$
P_3	$f_1 \cup f_3 \cup f_3 \cup f_4$	$L(P_3) = \max\{l_{f_1} + l_{f_3}, l_{f_3} + l_{f_4}\}$
P_4	$f_1 \cup f_2 \cup f_3 \cup f_4$	$L(P_4) = \max\{l_{f_1} + l_{f_2}, l_{f_3} + l_{f_4}\}$

Additional Constraints:

$$x_{ijk} \geq x_{ji'k}, \qquad \forall (i,j) \in A \tag{3.9}$$

$$x_{ijk} + x_{ji'k} \leq r_j + 1, \qquad \forall (i,j) \in A \tag{3.10}$$

$$\sum_{a:(a,j)\in A' a\neq i'} x_{ajk} \geq x_{i'j'k}, \qquad \forall (i,j) \in A \tag{3.11}$$

There are many alternative feasible paths in the modified graph. Although they are different in the modified graph, they represent the same path in the original graph. For example, for the graph given in Figure (3.2) assume that the working path, P_1 is as follows, $1 \to 2 \to 3 \to 2 \to 4$. Some of the paths which represent P_1 in G' are given in Table (3.4). In Table (3.4) there are five different paths, each of which represent the same path, P_1 in fact. Besides, more paths, representing P_1, could be found. It is obvious that, there is no need to allow this multi-representation case. It would be fine to prevent these additional paths, because omitting these additional paths will not result in losing feasible solutions for the original graph.

The arcs which are duplicates of the original arc are listed below and they are examined to identify which arcs can be selected under which conditions. The arc selection rules are as follows:

$i \to j$ This is the original arc of the graph. Therefore we prefer to use this arc, if possible.

$i \to j'$ This arc is a duplicate, so we use the arc if only node j is used before.

Table 3.4: Some of the paths which represent P_1

P_1	Representation in modified graph
$1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 4$	$1 \rightarrow 2 \rightarrow 3 \rightarrow 2' \rightarrow 4$
	$1 \rightarrow 2 \rightarrow 3 \rightarrow 2' \rightarrow 4'$
	$1 \rightarrow 2 \rightarrow 3' \rightarrow 2' \rightarrow 4$
	$1 \rightarrow 2 \rightarrow 3' \rightarrow 2' \rightarrow 4'$
	$1 \rightarrow 2' \rightarrow 3 \rightarrow 2 \rightarrow 4$

$i' \rightarrow j$ According to the preferences given above, a path visits a duplicate node if the original node is visited before. Therefore, this arc can be used only if the path comes to node i'. In this case, we again prefer to go to the original node using this arc.

$i' \rightarrow j'$ This arc goes from a duplicate node to another duplicate node. Therefore, this arc is not used if arc (i', j) or (i, j') is available.

An edge can be on the simple part of the path which means this edge is used only once in the path or it can be on the cycle portion of the path meaning that the corresponding edge on the original graph is used twice. Edges used in a path on the modified graph can be divided into three types. If an edge is on the simple part then it is a type-I edge. Arc (i, j) is used if the edge is a type-I edge. If an edge is on the cycle portion of the path and if one of its terminating nodes is the regenerator node then this edge is a type-II edge. Type-II edges are used twice and used pair of arcs are (i, j) and (j, i'). If an arc is on the cycle portion and does not have an end on the regenerator node then this arc is a type-III arc. In type-III edges, arcs (i, j) and (j', i') are used. Having defined the edge types, the paths to be omitted are identified according to the following rules;

- Model3 is formulated for a specific pair of nodes. There is no need to send the initial signal from duplicate of the source node instead of original source. This statement is valid for terminating node, too. Therefore, it is not necessary to duplicate starting and terminating nodes and related edges.

- Constraint (3.9) is used for type-I edges. If an edge on the path is type-I edge then only arc (i, j) could be used to represent this edge.
- Constraint (3.10) is used for type-II edges. Both arc (i, j) and arc (j, i') can be used only if the edge is a type-II edge. This constraint uses variable r_i to identify whether the edge is a type-II edge or not.
- Constraint (3.11) is used for type-III edges. Arc (i', j') is used if edge $\{i', j'\}$ is a type-III edge. If there is another entrance to node j', which means the node is visited more than once resulting in the edge is a type-III edge.

Notice that we do not omit any feasible solutions from the original integer solution set using these additional constraints.

As it is said before, we are examining a special case where EC=2. Therefore solution of Model3 will be optimal only if all the feasible paths visit each node twice at the most. Otherwise, some feasible solutions and possibly the optimal solution might be ignored. This means, if there is a node which is visited three or more times in a feasible path, there would be a problem again. Now we want to discuss the optimality of solutions of Model3 for different EC values, first.

A node is visited more than once only if there are regenerators placed on the graph. To be more exact, it can be said that a signal visits a node twice if there is one regenerator, visits three times if there are two regenerators etc... Otherwise, there is no need to visit those nodes more than once. Therefore an optimality range can be defined here, and we can say that a solution for a given EC value is optimal if the solution is in related optimality range. Optimality ranges are given in Table (3.5).

Having defined the optimality range, we can explain the procedure to find optimal solution for a given pair using ILP formulation. First, Model2 is formulated for EC=1 and solved. If the result is in optimality range, it is optimal. Otherwise, increase EC by one, solve again. Continue this procedure until the solution is within the optimality range.

This is a demanding process, so we want to find a simpler way. If we can show

Table 3.5: Optimality Ranges.

Expanding Coefficient	Optimality Range
1	0–1
2	1–2
3	2–3
x	$(x-1)-x$

that, whenever a node is to be visited three or more times in a feasible path P, there is always another path P' that can be preferred to P and on which each node is visited at most twice, we can conclude that EC=2 case does not ignore the optimal solution.

Theorem 3.4.1 *If there is a feasible $s-t$ path P that visits a node more than twice in a graph G, then there is always another feasible $s-t$ path P' which visits a node at most twice and is preferable to P.*

Proof Look at the graph given in Figure (3.5). In this graph, s and t are the terminal vertices, j and k are the regenerator vertices and i is the vertex which is visited more than twice. f_1, f_2, f_3 and f_4 are the path fragments between the vertices. The possible paths between pair (s,t) and their lengths are given in Table (3.6). Notice that, using Lemma (1), some other possible paths with rings can be discarded.

There are three possible cases and in each case there are two more possibilities for the lengths of the path fragments:

1. $L(P_3) = l_{f_1} + l_{f_2} \Rightarrow \begin{array}{l} l_{f_1} > l_{f_3} \\ l_{f_1} + l_{f_2} > l_{f_3} + l_{f_4} \end{array}$
   (a) $L(P_1) = l_{f_1} + l_{f_2} \Rightarrow P_1 \succ P_3$
   (b) $L(P_1) = l_{f_2} + l_{f_4} \Rightarrow l_{f_4} > l_{f_1} \Rightarrow L(P_2) = l_{f_3} + l_{f_4} \Rightarrow P_2 \succ P_3$
2. $L(P_3) = l_{f_2} + l_{f_3} \Rightarrow \begin{array}{l} l_{f_3} > l_{f_1} \\ l_{f_2} > l_{f_4} \end{array}$

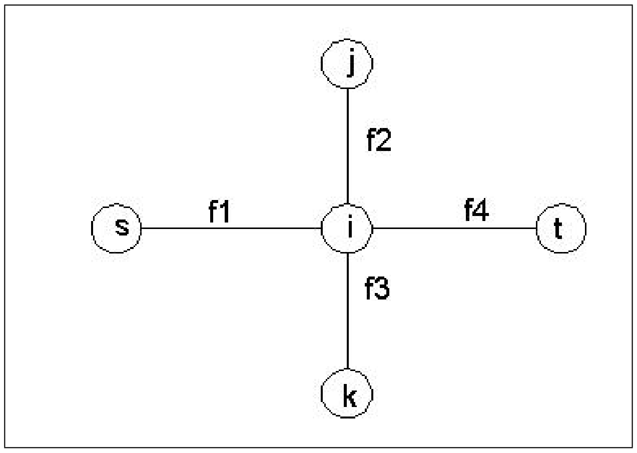

Figure 3.5: Visiting a vertex more than twice

Table 3.6: Multi visited vertices.

Path Name	Path	Path Length
P_1	$f_1 \cup f_2 \cup f_2 \cup f_4$	$L(P_1) = \max\{l_{f_1} + l_{f_2}, l_{f_2} + l_{f_4}\}$
P_2	$f_1 \cup f_3 \cup f_3 \cup f_4$	$L(P_1) = \max\{l_{f_2} + l_{f_3}, l_{f_3} + l_{f_4}\}$
P_3	$f_1 \cup f_2 \cup f_2 \cup f_3 \cup f_3 \cup f_4$	$L(P_3) = \max\{l_{f_1} + l_{f_2}, l_{f_2} + l_{f_3}, l_{f_3} + l_{f_4}\}$
P_4	$f_1 \cup f_3 \cup f_3 \cup f_2 \cup f_2 \cup f_4$	$L(P_4) = \max\{l_{f_1} + l_{f_3}, l_{f_3} + l_{f_2}, l_{f_2} + l_{f_4}\}$

(a) $L(P_1) = l_{f_1} + l_{f_2} \Rightarrow P_1 \succ P_3$

(b) $L(P_1) = l_{f_2} + l_{f_4} \Rightarrow l_{f_4} > l_{f_1} \Rightarrow L(P_2) = l_{f_3} + l_{f_4} \Rightarrow P_2 \succ P_3$

3. $L(P_3) = l_{f_3} + l_{f_4} \Rightarrow \begin{array}{l} l_{f_4} > l_{f_2} \\ l_{f_3} + l_{f_4} > l_{f_1} + l_{f_2} \end{array}$

(a) $L(P_1) = l_{f_1} + l_{f_2} \Rightarrow P_1 \succ P_3$

(b) $L(P_1) = l_{f_2} + l_{f_4} \Rightarrow l_{f_4} > l_{f_1} \Rightarrow L(P_2) = l_{f_3} + l_{f_4} \Rightarrow P_2 \succ P_3$

According to Theorem (3.4.1), we can conclude that modification of the graph with EC=2 is sufficient to solve the problem. □

3.5 Mathematical Program in Practice

The mathematical program can be used to find an optimal solution in theory. We will try to discuss whether the model is usable in practice or not. So we will make an assumption, which will be relaxed later. Our assumption is that, the original graph i.e. $EC = 1$ and Model2 are sufficient to find the optimal solution for a pair of vertices. In other words, we assume that restricting our attention to simple paths will not exclude optimal solutions from the feasible sets.

Model2 given above is constructed for only one pair of nodes. However, there are $n(n-1)$ pairs in an n node network. Because the graph is undirected, the number of pairs reduces to $\frac{n(n-1)}{2}$. If the model were solved for just one pair, we would probably get a sub-optimal solution. Therefore, the constraints for each pair must be included in the ILP as it is done in Section (3.1), since each pair is independent. However, the number of pairs that needs to be included in the model is too high and the size of Model1 (number of constraints and number of variables) grows too fast. For this reason, using Model1, to solve the problem and find a global optimum, is not suitable in practice. Smaller models, including less number of pairs, could be used. However, this may result in local optimal solutions, as well. The solution will be optimal for only pairs included in the model. When another pair is added to the model, the optimal solution may or may not change. So it can be said that the solution found using some pairs, can deviate from the global optimum.

Besides, the integer program is not useful even if it is formed for only one pair. Solution times could be too long since the size of the branch and bound tree grows too large very quickly. Another weakness of the mathematical program is its weak lp relaxation. When we solve the linear relaxation of the model we get zero as the solution value. This value does not give any idea about how many regenerators are needed in the network. In addition to this, we have another drawback that even if the model finds the optimal solution for the integer program, it fails to prove that this solution is optimal until all the branch and bound tree is evaluated. This causes long solution times. For example, for the graph given in Figure (3.6), we have formulated Model2 for the pair (1,32) and set the degradation limit to

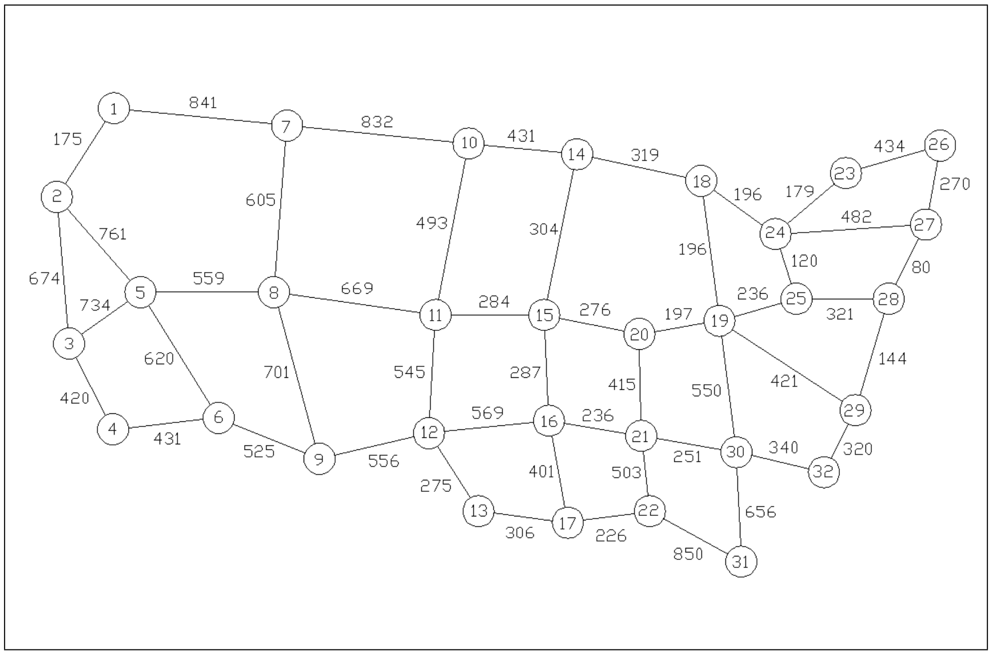

Figure 3.6: Network topology used in experiments

1500. This graph has 32 vertices and 50 edges. Model2 is solved using CPLEX 8.1.0 and it takes 10955 seconds which is approximately 3 hours. Thinking there are many other pairs to be solved, this methodology is way too long as a solution procedure.

It is obviously seen that the mathematical program could not be used to find a global optimum even with our assumption, that the solution set consists only of simple paths.

It may seem unnecessary to spend this much effort on formulating three integer programs, finding inequalities to shrink the feasible area of the lp relaxation since we argued that all three models are not useful in practice even for medium size graphs. However, remember the transparency islands mentioned in Chapter (1). Some partitions called transparency islands are formed in optical networks. In these islands, signals are transmitted without regeneration and signals are regenerated on the boundaries of the islands if they are transmitted outside the island. In a large optical network, if some transparency islands are formed they can be represented with just a node. This can be considered as a generalized version of this problem. Forming a few islands can reduce the size of the network significantly. With this new relatively small network the mathematical models described in this chapter could be used to find the optimal solution without the need for any other algorithms.

Chapter 4

Feasibility of an Instance

In the previous chapter it was concluded that the ILP models are not appropriate to solve this problem. This justifies us in employing some heuristic algorithms to find solutions to the problem. These heuristic ideas are usually based on checking whether a solution, a fixed set of regenerators, is feasible or not. By this way, the solution with the best objective function value can be selected. Therefore we need a method that could check feasibility of a given solution.

4.1 Using Model2

We could not use Model2 to find the optimal solution. However, our formulation would be useful if it can be incorporated in heuristic algorithms. A solution set is a fixed set of regenerators placed on specific vertices. After a solution set (instance) is determined, Model2 is formulated according to this solution set. Solving Model2, one can see if the instance is feasible. This technique was also used in [25, 24]. This method does not give satisfactory results, however. Solution times are not very short even for feasibility check. Running times of some solution instances are shown in Table (4.1). The graph used in these experiments, has 32 vertices and 52 edges and can be seen in Figure (3.6). Degradation limit is set to 2300. Model2 is solved first for pair (2,32) and some solution instances are

Table 4.1: Running Times For Feasibility Check.

Solution Instance	Solution Time (seconds)	Solution Status
{10,11}	372	optimal
{10}	552	infeasible
{11,12}	186	infeasible
{10,11,12}	42	feasible
{15,17,19,20}	32	infeasible

checked for feasibility using Model2.

It is interesting that, finding the optimal solution for the pair sometimes takes less time than finding if a solution instance is feasible. Here, do not forget the problem in Model2 discussed in Section (3.3). The solution {10,11} may not be optimal. Model3 should be used since the result is not in the optimality range for $EC = 1$. This is also valid for the other solution sets. Placing a regenerator only on vertices 10 and 11 may be a feasible solution. To understand that, Model3 must be used, however, in that case solution times will increase. Besides, feasibility of an instance is determined for only one pair, not for the entire graph. This means that the solution must be checked for feasibility for $\frac{n(n-1)}{2}$ pairs.

So the model is not very useful for a feasibility check. For this reason, a new technique is developed, and a model was formulated to check feasibility.

4.2 Exact Feasibility

We propose a method to find whether a solution is feasible or not. The term *exact feasibility* means that, using this method one can find if a solution is feasible. The *exact* differentiation is important because we will propose another method in Section (4.3) by which we can in most instances understand if a solution is feasible. Second method runs very quickly, however, it sometimes gives false results. Some feasible solutions can be treated as infeasible. Hence, we need to make a differentiation such as exact and approximate feasibility.

This technique is based on linear programming formulation of shortest path problem. The model given below is the well-known shortest path formulation between a given pair of vertices s and t.

Objective:

$$Minimize \sum_i \sum_j c_{ij} x_{ij}$$

Subject to:

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}t; \\ 0, & i \neq s,t; \end{cases} \qquad \forall i$$

$$x_{ij} \in \{0,1\}$$

We are going to modify this formulation in order to use it in checking feasibility. Modified formulation is given below.

Objective:

$$Minimize \quad a \tag{4.1}$$

Subject to:

$$\sum_{j:(i,j)\in A} x_{ijk} - \sum_{j:(j,i)\in A} x_{jik} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}t; \\ 0, & i \neq s,t; \end{cases} \qquad \forall i \in N, \forall k \tag{4.2}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ijk} \leq R_{max}, \forall k \tag{4.3}$$

$$\sum_k (x_{ijk} + x_{jik}) \leq 1, \forall (i,j) \in A \tag{4.4}$$

$$x_{ijk} \in \{0,1\} \tag{4.5}$$

There is one decision variable as follows:

$$x_{ijk} = \begin{cases} 1, & \text{if } k^{th} \text{ path includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

In this formulation, we want to find out if there are two feasible edge-disjoint paths between a given pair of nodes. In this solution instance there are not any regenerators.

- The objective function (4.1)is trivial, variable a is simply a constant, in fact. Since we want to know whether there is a feasible solution without any regenerators, we do not need an objective function. Objective function is used just to complete the linear program.
- (4.2) defines flow balance constraints.
- Left hand side of constraint (4.3) represents the length of a path. We make sure that both the path lengths are within the degradation limit by these constraints.
- We also want the two paths be disjoint, therefore we include constraints (4.4). Notice that, in (4.4) we force the paths be simple. This is logical since there are not any regenerators placed on the network.

If this model has a feasible solution, this means that there are two edge-disjoint paths that satisfy the degradation limit constraint without using any regenerators. Unfortunately, the model does not give any other information such as how many regenerators we need or where to place them. However, with a little modification, we can still employ this model in checking feasibility. Remember the sub-paths defined in the previous chapter. The paths can be divided into sub-paths when they pass through regenerators. Assume that there is a regenerator on node k and we search for paths between node s and node t. Hereafter, first path will be referred to as the working-path and the second one as the restoration-path. Let node k lie on the working-path and say the restoration-path does not include node k. Then working-path can be divided into two sub-paths. First one is from node s to node k and the second part is from node k to node t. Now we have two sub-paths for working-path and a simple restoration-path. According to these paths we can modify the above model. In this modified model, we use x_{ijk} for the working-path and y_{ij} for the restoration path.

$$x_{ijk} = \begin{cases} 1, & \text{if } k^{th} \text{ sub-path of working path includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1, & \text{if the restoration path includes arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$$

Objective:

$$Minimize \quad a$$

Subject to:

$$\sum_{j:(i,j)\in A} x_{ij1} - \sum_{j:(j,i)\in A} x_{ji1} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}k; \\ 0, & i \neq s{,}k; \end{cases} \qquad \forall i \in N$$

$$\sum_{j:(i,j)\in A} x_{ij2} - \sum_{j:(j,i)\in A} x_{ji2} = \begin{cases} 1, & i{=}k; \\ -1, & i{=}t; \\ 0, & i \neq k{,}t; \end{cases} \qquad \forall i \in N$$

$$\sum_{(i,j)\in A} c_{ij}x_{ij1} \leq R_{max}$$

$$\sum_{(i,j)\in A} c_{ij}x_{ij2} \leq R_{max}$$

$$\sum_{j:(i,j)\in A} y_{ij} - \sum_{j:(j,i)\in A} y_{ji} = \begin{cases} 1, & i{=}s; \\ -1, & i{=}t; \\ 0, & i \neq s{,}t; \end{cases} \qquad \forall i \in N$$

$$\sum_{(i,j)\in A} c_{ij}y_{ij} \leq R_{max}$$

$$\left.\begin{array}{l} x_{ij1} + x_{ji1} + y_{ij} + y_{ji} \leq 1 \\ x_{ij2} + x_{ji2} + y_{ij} + y_{ji} \leq 1 \end{array}\right. , \forall (i,j) \in A$$

$$x_{ijk} \in \{0,1\} \quad y_{ij} \in \{0,1\}$$

$$\text{all variables} \geq 0$$

This mathematical model is both compact and easy to solve. Starting and terminating nodes of the sub-paths are determined and the model is constructed using flow balance and path length constraints. These can be considered as

Table 4.2: Path Alternatives.

# of Regs in working-path	w-path	# of Regs in restoration path	r-path
2	s-k-l-t	0	s-t
2	s-k-l-t	1	s-k-t
2	s-k-l-t	1	s-l-t
2	s-k-l-t	2	s-k-l-t
2	s-k-l-t	2	s-l-k-t
2	s-l-k-t	0	s-t
2	s-l-k-t	1	s-k-t
2	s-l-k-t	1	s-l-t
2	s-l-k-t	2	s-k-l-t
2	s-l-k-t	2	s-l-k-t

parameters of the problem. What can we do if we do not know these parameters? Or how do we find these parameters? We will try to answer these questions step by step, going from simpler to more complex cases.

Assume that we have one regenerator placed on the network and are searching for paths satisfying the constraints. In this case, determining the sub-paths is quite simple. Both working-path and restoration-path can be simple, both paths may include the regenerator node, working-path can be simple and the restoration-path can include regenerator node or vice versa. We have four alternatives with one regenerator placed on the network. When there are two regenerators on nodes k and l and assuming that the working-path is passing through both regenerators, the alternatives for the paths are given in the Table (4.2). There are ten different possible ways to form feasible paths and it cannot be seen easily before solving the feasibility models whether they are feasible or not.

The question of how many alternatives there will be can be answered if finding the possible alternatives is systematically examined. When there are n regenerators on the graph and r of them are used, there are $P(n, r)$ alternatives. Here, $P(n, r)$ shows the permutation of n elements with r selections. We use permutations to calculate the number of alternatives since not only the selection of the

Table 4.3: Number of Path Alternatives.

# of Regs	# of Alternatives for one path	Total # of Alternatives
1	2	4
2	5	25
3	16	256
5	326	106276
10	9864101	—

regenerators is important but also the order of the selected regenerators is also important. How can one determine if a solution is feasible using these alternatives? One alternative is selected for the working-path and one for the restoration path. ILPs are formed according to these selections and are solved.

- The solution is infeasible if models formed according to all alternatives are infeasible
- The solution is feasible if at least one model formed using the alternatives is feasible

This permutation gives the number of alternatives for fixed n and r. The number of possibilities can be calculated by $\sum_{r=0}^{n} P(n,r)$. Since these alternatives are valid for both working-path and restoration-path, the total number of alternatives is $(\sum_{r=0}^{n} P(n,r))^2$. Total number of alternatives is calculated for different number of regenerators and is given in Table (4.3).

As it can be seen from the table, the number of alternatives grows exponentially fast. So it is too hard to understand if a solution is feasible if the number of regenerators is not small. Besides, all this work is done for only one pair, not for the entire graph. Since it is hard to find a solution by checking the feasibility of an instance by this method, we need to find an easier way to check the feasibility of a solution.

4.3 Approximate Feasibility

In section (4.2) we have proposed a method to understand whether a solution is feasible or not. However, that method can be very demanding if the size of the instance is not small. Therefore we will propose another method in this section. The technique explained here requires a little pre-processing. In this pre-processing, *shortest-2-paths* are computed and the arcs used in those paths are recorded for later use. We need to define here, what shortest-2-path means. We want to find two edge-disjoint paths, where, typically one of them will be longer. If the length of the longer path is smaller than delay limit, then there is no need for regenerators for this pair.

We use an integer linear program to find the paths and their lengths. Using this program, we minimize the length of the longer path. Paths found by this integer linear program are called shortest-2-paths.

Approximate feasibility technique is based on dividing the paths between the pairs into two — before and after regenerator. In this technique, we store the states of the pairs. State of a pair shows whether a solution is found for that pair or not. The philosophy of this method is quite different from that of exact feasibility algorithm. Approximate feasibility technique is very efficient and effective in identifying the feasibility of the solution not only for a pair but also for the entire graph, whereas exact feasibility can only deal with pairs separately. Assume that there is a regenerator on node r and we are searching for paths from node s to node t. This method has four main steps, which are described below.

1. It is determined, if it is possible to find paths satisfying delay limit constraint. The criterion used here is that, we look at the states of pairs (s, r) and (r, t). If states of both pairs are positive (there are two paths already found) then pair (s, t) may communicate, too, and carry on to step 2. Otherwise,the solution is infeasible for pair (s, t) and stop.

2. We form 4 sets using the data gathered from pre-processing. We call these sets as α, β, γ and θ. The arcs used in working-path of pair (s, r) are stored

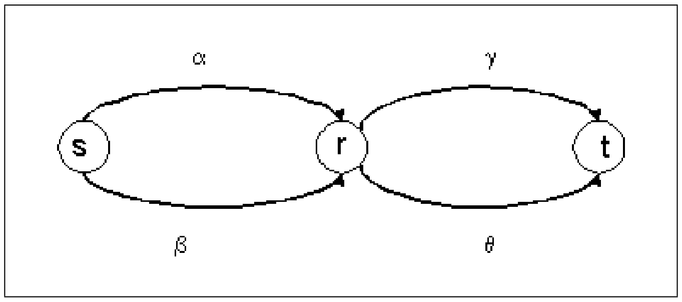

Figure 4.1: Path matchings

in set α, arcs of restoration-path are stored in set β. Similarly, arcs used in working-path of pair (r, t) are stored in set γ, arcs of restoration-path are stored in set θ. After forming the sets, two new sets A and B are formed such as

$A = \alpha \cup \gamma \quad B = \beta \cup \theta$

Sets A and B represent paths for pair (s, t). These two could be seen in Figure (4.1). Since there is a regenerator on node r they already satisfy the time constraint. If $A \cap B = \emptyset$ then they are edge-disjoint, too and solution is feasible for this pair. If $A \cap B \neq \emptyset$ then we change the sets A and B slightly.

$A = \alpha \cup \theta \quad B = \beta \cup \gamma$

If $A \cap B = \emptyset$ then solution is feasible for this pair.

If solution is feasible stop, else go to step 3.

3. From pre-processing we know how the signal came from node s to node r. We search for simple paths from node r to node t using an integer linear program. If we can find edge-disjoint paths, solution is feasible. Otherwise go to step 4.

4. From pre-processing we know how the signal goes from node r to node t. We search for simple paths from node s to node r using a an integer linear program. If we can find edge-disjoint paths, solution is feasible. Otherwise solution is infeasible.

This technique has a very important advantage. No matter how many regenerators are placed on the graph, this method is interested in only one regenerator placed on node r. Assume that there are 4 regenerator nodes on the network and we are placing a new one. In order to check the feasibility of this new solution set, other methods try to construct feasible paths using all of the regenerator nodes on the network. This requires trying a large number of possible alternatives if there are many regenerators on the network. However, the approximate feasibility method does not try to use all regenerators. It assumes previously placed regenerators are already used and it divides the problem into two smaller parts. The first part is the problem of pair (s, r) and the second part is the problem of pair (r, t). Since these two problems are already solved, problem turns into a matching problem. Hence the algorithm is very efficient. It does not deal with the alternative paths we have defined in section (4.2).

The algorithm has a disadvantage, however. It forces both paths to pass through node r which is not always necessary. For this reason, this algorithm may give wrong results sometimes. However, these wrong results are only pessimistic in nature, since the algorithm can declare some feasible solutions as infeasible but not vice versa. Besides, our experimentation shows that the algorithm gives correct results about %90 of the time. Therefore, it is effective at the same time.

Now we have proposed two methods to check feasibility of solution instances. One gives exact results but takes long time to find solutions if the size of the instance is not small. The other is efficient and fast, however, it can give false results. Both methods will be used and we will explain how they are incorporated in our solution methodology in the succeeding chapters.

Chapter 5

Heuristic Solutions

In the previous chapters, we have formulated ILPs to find the optimal solution to this problem. Unfortunately, that approach has failed and we wanted to employ some heuristics. So two methods have been proposed to check the feasibility of a candidate solution. In this chapter, a heuristic algorithm which solves the problem is proposed and evaluated.

Heuristic algorithms can be classified into two main types. One type is iterative algorithms. These algorithms start from an initial solution and search for better solutions by going to neighbor solutions of the initial solutions. Second type is constructive algorithms. These type algorithms directly aim at constructing good solutions. The algorithm proposed here is a constructive algorithm.

5.1 Proposed Heuristic Algorithm

As stated before, a constructive type algorithm is developed in this section. We begin with an empty solution set and add a node to this set at each iteration to make the solution set feasible for the entire graph.

We need two things for this algorithm to give good results. First we need to decide at each iteration which node is selected to place a regenerator on. We define *marginal utility* of a node to help us make this decision. Selecting a node to place a regenerator is very important, since if a node whose marginal utility to the system is low; this may result in worse solutions, i.e. high number of regenerators. We need to define what we mean with marginal utility here. Marginal utility of a node is the number of pairs that start communicating after placing a regenerator on that node. We say that a pair of nodes are communicating or reached if two edge-disjoint paths with lengths within the degradation limit can be constructed. But there is a serious problem here, how can we compute the marginal utilities of the nodes? Unfortunately, this is a demanding and time consuming problem. However, we can make an approximation to the marginal utilities. Take node s and node t, for example. Assume that they cannot communicate currently. Let r be the candidate node for placing a regenerator. If node s can communicate with node r and node t can communicate with node r then there is a possibility that node s can communicate with node t through node r. In this case, we increase the marginal utility of node r. This method gives us a rough idea about the marginal utility of the candidate node. Do not forget, there is a possibility that (s, t) pair cannot communicate although this rule says they can since we also have an additional requirement on edge disjointness.

Second, we need a technique to find if a solution set is feasible for pairs, consequently for the entire graph. Two methods have been developed and given in the previous chapter. Here we will incorporate them.

We want to give the main steps of the heuristic algorithm first and then explain it.

Algorithm:

1. Initialize the problem.
2. Determine the pairs which were not reached.
3. If there is a pair not solved go to step (4), else end.
4. Select a node to place a regenerator.
5. Examine the non-solved pairs and find out if they are feasible.

6. Go to step (2)

As it is seen the algorithm is quite simple and straightforward. Now we want to explain what is done in each step.

- In step (1) we examine each pair, and find the minimum lengths of the paths between them. The shortest path lengths and shortest-2-path (as explained before) lengths are recorded in a file to be used later. The arcs used in the paths are also recorded. This step can be considered as a pre-processing step.
- We also keep the status of each pair. This status tells us whether or not the pair can communicate. At the beginning of the algorithm, only the pairs whose shortest-2-path length is smaller than the degradation limit can communicate while others cannot. In step (2) we determine which pairs cannot communicate yet.
- In step (4), we want to find the node with the largest marginal utility and place a regenerator on it. The candidate node whose marginal utility is the largest is selected and a regenerator is placed on it.
- After adding a new regenerator to the system, we want to find which pairs become able to communicate. This analysis is performed in step (5). In Chapter (4) we have proposed three methods to check feasibility. Here one of those methods can be used. Each one has advantages and disadvantages. If *exact feasibility* method is used, it is expected to get better results as this method gives correct results, but the solution times get longer. Model2 can also be used, however the solution times are also long in this method and it might be necessary to use Model3 to get correct results. We prefer to use the last method, *approximate feasibility*, since it runs very fast and often gives correct results.

Table 5.1: Heuristic Solutions for 32 node network.

Degradation Limit	Number of Regenerators Placed	Solution Time (seconds)
2500	2	1
2200	4	2
2000	5	5
1800	5	3

Table 5.2: Heuristic Solutions for 50 node network.

Degradation Limit	Number of Regenerators Placed	Solution Time (seconds)
1500	3	11
1400	3	12
1300	3	16
1200	5	23
1100	6	13

5.2 Computational Analysis

In the previous section a heuristic algorithm is proposed to solve our problem. It is used to solve different problems with different network topologies and different degradation limits. Two different topologies are used for analysis. The first topology is obtained from [25] and has 32 nodes and 50 edges. This network is given in Figure (3.6). The second network is randomly generated with 50 nodes and 108 edges. The solutions we found using the heuristic algorithm and running times of the algorithm are tabulated in Table (5.1) and Table (5.2). The algorithm is implemented using C programming language and CPLEX 6.0 optimization software package is used to solve mathematical programs. The algorithm is run on a computer with 1 gigabyte memory and PIII 733 Mhz processor.

From the solutions we can say that as the range of the optical signals (degradation limit) decrease the number of regenerators increase, as expected. These solutions are expected to be good solutions but we cannot tell how good they are. A lower bound for our problem is needed to evaluate the heuristic solutions. Finding the lower bound of the problem is discussed in Section (6.1). The major advantage of the heuristic algorithm is the running times. From the tables it is seen that the solution times are very small which means that the heuristic algorithm is very fast. This property of the heuristic algorithm can be used to bound the optimal solution to our problem. A method to find the optimal solution is developed in the next chapter.

Chapter 6

Finding an Optimal Solution

In the previous chapter we have proposed an algorithm and with the algorithms of [24, 25] there are three proposed solution techniques at hand. All three are heuristic algorithms so we cannot guarantee that the solutions we get using them are optimal solutions. However, we need to show that the solutions are reasonable solutions. Since we do not know the optimal solutions of the problems there is only one way to show whether of not the solutions we found are good enough.

6.1 Lower Bound of the Problem

Comparing the solutions with a lower bound is a common and useful way to evaluate how good a solution is. By this way, one can compute the deviations from the lower bound and this would be a good measure. Therefore, we need to find a lower bound for the problem, so that we can see how good the proposed solution is. Naturally, it is better to have tight lower bounds. A good lower bound means a tight lower bound.

Generally, a lower bound can be found solving the linear relaxation of the integer program of the problem. Unfortunately, we could not find good lower bounds for the problem using this technique. In previous chapters, we have remarked that this approach gives us zero as relaxed value. This value does not tell us anything. So we have employed another method to find a lower bound.

This method is very straightforward. In the pre-processing (initialization) step of the heuristic algorithm, we have calculated shortest path lengths and shortest-2-path lengths between the pairs. We take the largest length and find a lower bound using this value. Think of an example, where the degradation limit is 10 and the largest shortest-2-path length is 25. It can be seen that length of each sub-path must be smaller or equal to 10. In this case, in the longest path there must be at least three sub-paths, which means that we need at least 2 regenerators to solve the problem.

Let L show the length of the longest path found in the pre-processing step. Then the lower bound can be found using the formula $lb \geq \lfloor \frac{L}{delaylimit} \rfloor$. Since the division may result in a fraction, the lower bound is rounded to the greatest integer lower or equal to this fraction.

It is obvious that this lower bound is also rough and does not tell us much. After some experiments it is seen that the gap between the lower bound and the solution found from the heuristic is relatively large.

So we have two alternatives, we should find either a tighter lower bound or a better solution to make this gap smaller.

6.2 Branch and Bound Algorithm

We have preferred to find a better solution, optimal if possible instead of finding a tighter lower bound. As it is said before, we do not have a mathematical program or another way to find a globally optimal solution for the problem. So we have to explicitly search for the global optimum and evaluate all possible solutions and find out if they are feasible. By this way, the best solution among the feasible solutions could be selected as the optimal solution. There are too many solutions, however, to evaluate and too many feasibility-checking operations are needed for each solution. So there is a need for an effective method, which can search the solution space efficiently and evaluate the solutions for feasibility easily. To achieve this, a branch and bound algorithm is employed.

6.2.1 Adaptation of the Branch and Bound Algorithm

Branch and bound is a well known algorithm that provides us with a method that creates sub-problems or solutions according to previously evaluated solutions (branching part) and evaluate newly created sub-problems by comparing with current upper-bound and lower-bound values (bounding part). Actually branch and bound is a routine used to solve integer programs. Using the idea of this technique, branch and bound based algorithms can be develeoped to solve different problems. Although it takes long time to solve, its major advantage is that branch and bound algorithm gives optimal solutions. Though, the philosophy is the same, the methods used to solve the sub-problems and to calculate the lower and upper bounds can be different in different branch and bound based algorithms which are applied to different areas. In order to apply branch and bound algorithm to our problem, we make some modifications and definitions. These are explained below. Firstly, we want to give our notation.

U_i shows the upper bound of the solution represented by node i.

L_i shows the lower bound of the solution represented by node i.

Lb represents a lower bound of the problem, found as described previously.

$cutoff$ shows the value of the current solution

$(r_1 \dots r_k)$ This representation means we have k regenerators placed on nodes $r_1, r_2 \dots r_k$ respectively in this order.

Each node of the branch and bound tree represents a possible solution set of the problem. For instance, node (1,2) means that nodes 1 and 2 are opaque and the other nodes are transparent nodes.

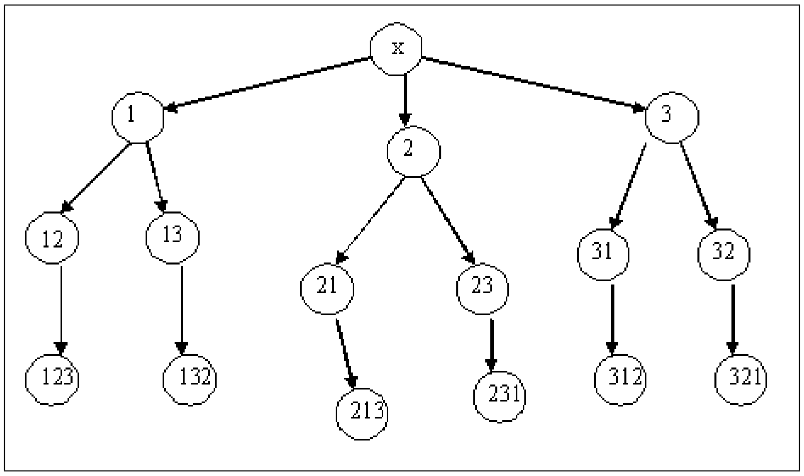

Figure 6.1: Branch and Bound Tree

Now suppose that we have a graph with 3 nodes. The branch and bound tree will be as follows:

There are 16 possible solutions for the problem and each solution is represented by a node in the branch and bound tree. The number of possible solutions is expected to be very large for larger problems and it would take too much time to evaluate all nodes. In fact branch and bound algorithm is a kind of implicit enumeration. The advantage of this algorithm is that it prevents evaluation of all nodes by cutting them using a methodology. Observing the tree, one can easily notice that some of the nodes represent the same solutions. For example, $(1, 2)$ and $(2, 1)$ or $(3, 1, 2)$ and $(1, 2, 3)$ are the same. We have said that the order of the regenerators used is as important as the selected regenerators. Is there a conflict between these two facts? Actually no, because we use feasibility approach to evaluate the nodes, so we indirectly take the order of the regenerators into account. Therefore, the order is not important in solutions and we can and should prevent the evaluation of a solution more than once. We have setup a rule to do this. In a solution, selected nodes may occur in ascending order i.e. we can have $(2, 3)$ but not $(3, 2)$ or we can have $(1, 2, 3)$ but not $(3, 1, 2)$ etc. Applying this rule to the branch and bound tree in Figure (6.1), we get a new tree given in Figure (6.2).

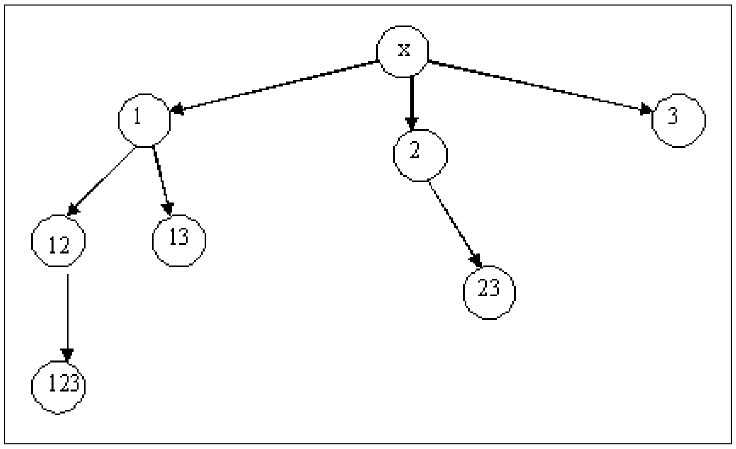

Figure 6.2: Branch and Bound Tree- Eliminated

Now we have 8 nodes in branch and bound tree and number of nodes reduced to half of the number of nodes in the original tree. This reduction is very beneficial but the number of possible solutions might still be too many in large problems. It would be beneficial if we can find other rules that would help us disregard some nodes without disregarding the optimal solution. These rules are our bounding rules.

Suppose that we have a feasible solution for the problem. The objective value of this solution is our cutoff value. Cutoff means that we already have a solution and we do not need to search the nodes whose lower bound is greater or equal than the cutoff value. The lower bound of nodes will be explained later.

Before applying the branch and bound algorithm, we can easily find an initial solution using the heuristics explained in Section (5.1). Having this solution we can fathom some nodes.

By this way, we can reduce the number of the nodes that will be evaluated. Suppose that we used the heuristic and found $(1, 2)$ as the solution. This means we have a solution with two regenerators. Therefore, we do not have to search $(1, 2, 3)$ or $(2, 3)$ or any other solutions which use at least 2 regenerators, since we cannot find a better solution from such nodes. In our example, if we have $(1, 2)$ as the solution, we can prune $(1, 3)$, $(2, 3)$ and $(1, 2, 3)$.

Three lower bound values are calculated for each node using different techniques and the largest one is used as the lower bound of the node. First one is the number of regenerators in the solution. If there are 2 regenerators in a solution we cannot find a solution with better objective value than 2, from that node or branch. Second one is slightly different; from preprocessing step of the heuristic algorithm we know how many regenerators are required to make communication possible between a pair. Suppose that we are on the node, $(1, 2)$, and we know that we need 2 regenerators for pair $(1, i)$ for example. This means that we need at least 2 regenerators other the regenerator placed on node 1, because the regenerator on node 1 does not have any effect on the communication between node 1 and node i. Therefore, we can say that we need at least 3 regenerators in this solution. Starting with this combination we cannot find a better solution. To find a lower bound according to this rule, we need to look at all the pairs for which one of the nodes is present in the solution and we take the largest of them. The way we compute the third lower bound is very similar to the second one. Here we take two nodes with regenerators instead of one, and look at the number of regenerator requirements. We take the largest of them again. Remember our example, we are at node $(1, 2)$, and we need 3 regenerators for pair $(1, 2)$, then our lower bound is 5. After calculating these, we have three numbers, maximum of these is the lower bound of the node. If the lower bound of the node is greater or equal to the cutoff value, we can prune that node.

Another property of a node is its upper bound. This is related with our definition of the branch and bound tree. Since we setup a rule to prevent creating same solutions on different nodes we will not have a node such as $(3, 2)$ or $(2, 1)$ etc... Remember our example with three nodes. Let one node be (1), branching from this node we can create other nodes. The nodes we will have are $(1, 2)$, $(1, 3)$ and $(1, 2, 3)$. The node with largest number of regenerators has 3 regenerators. If we branch from node (2), we will get nodes $(2, 3)$ only. From here we can say that the upper bound of node (1) is 3, where the upper bound of node (2) is 2. Formally, the upper bound of a node $(r_1, \ldots r_k)$ is calculated as $U_i = n - r_k + k$. Here n shows the number of nodes in the graph, k is the number of regenerators in the node and r_k shows the largest element of the node.

Using the upper bound we can cut additional nodes. If the upper bound of a node is smaller than its lower bound then we can prune it, because in this case we cannot create a node with enough regenerators branching from that node. Also if the upper bound of the node is smaller than the lower bound of the problem we can prune that node, too.

The conditions in which we can fathom a node are listed below:

- If the lower bound of a node is greater or equal to the cutoff value
- If the upper bound of a node is smaller than its lower bound
- If the upper bound of a node is smaller than the lower bound of the problem

There is yet another condition. If the size of the node (number of regenerators in the solution) is smaller than the lower bound of the problem or lower bound of itself, then we do not need to evaluate this node for feasibility since it is obvious that that node (solution will not be feasible). However, it would not be pruned, otherwise a candidate optimal solution may be ignored. Although, it is not necessary to evaluate such nodes, we prefer to evaluate them to use the information attained from this evaluation in evaluating the succeeding nodes.

6.3 Computational Analysis

In Section (5.2), different problems were solved using the proposed heuristic algorithm. In this section, the same problems are solved using the branch and bound algorithm to find the optimal solutions. By this way, we can compare both heuristic and optimal solutions and consequently we can tell how good the heuristic solutions are. Heuristic and optimal solutions and the running times of the branch and bound algorithm are given in Table (6.1) and in Table (6.2). The branch and bound algorithm is coded in C programming language and run on the same computer with configuration described in Section (5.2).

Table 6.1: Optimal and Heuristic Solutions for the 32 node network.

Degradation Limit	Heuristic Solution	Optimal Solution	Solution Time (seconds)
2500	2	2	3
2200	4	3	186
2000	5	4	1524

Table 6.2: Optimal and Heuristic Solutions for the 50 node network.

Degradation Limit	Heuristic Solution	Optimal Solution	Solution Time (seconds)
1500	3	2	44
1400	3	2	51
1300	3	3	890
1200	5	4	3200

Branch and bound algorithm finds the optimal solutions in reasonable times, if the number of regenerators in the optimal solution is small. Besides it is observed that the branch and bound algorithm runs very fast for medium size networks with about 50 nodes and relatively large degradation limits. However, running time of the algorithm increases as the number of regenerator nodes placed on the network increases since the number of nodes in the branch and bound tree gets larger.

The important point here is the difference between the heuristic and optimal solutions. The number of regenerators placed on the network do not differ much. The heuristic algorithm finds near optimal solutions. From here, we can conclude that for hard problems, which have large number of nodes and small degradation limits, proposed heuristic algorithm could be used to find good solutions.

Chapter 7

Conclusion

The explosive growth in the number of users and the volume of traffic carried in the Internet put an ever increasing load on the networks. Since Internet is used for carrying real-time and high-priority data, QoS and reliability have become crucial issues.

Optical Transport Networks are seen as an important step in the evolution of data transmission. Optical networks have emerged as an alternative to traditional networks as they can transfer more data at higher speeds than copper wire. In this context, optical networks offer solutions to respond to the explosive growth in the traffic on the Internet, which makes the capacity expansion inevitable. For this reason, it is expected that the optical networks will be used widely as the optical data transmission technology develops.

However, the optical layer constraints such as optical signal degradation limit the range of optical networks necessitating optical signal regeneration which is a costly process. Therefore, the problem of placing regenerator nodes on optical networks arises. Besides, there is a requirement for restoration in the networks in case of transmission failures. For path-based restoration, two edge disjoint paths should be established between source and destination pairs of the networks. Although regenerator placement problem is studied, the problem of regenerator placement on a network with path-restoration has not been widely studied in the literature.

Our contribution in this thesis is the development of the method which finds the optimal solution of our problem. An integer linear program is formulated for this problem. But the huge size of the program necessitates the need for other solution techniques. For this reason, two heuristic algorithms are developed and numerical results are obtained for different degradation limits and different topologies. It is observed that the two heuristics often give the same solutions but one is faster and computationally more efficient than the other. Three methods to check the feasibility of a solution instance, when a fixed set of regenerators are placed on specific nodes, are proposed. It is also observed that, it gets harder to check the feasibility of a solution instance, as the size of the solution set, the number of regenerators placed, increases. This increase is also observed in the running times of the heuristic algorithms. To understand how good the results of the heuristic algorithms are, a branch and bound algorithm is developed to find optimal solutions of the problem. It is observed that the branch and bound algorithm runs very fast for medium size networks and relatively large degradation limits. The running time of the algorithm increases as the number of regenerator nodes placed on the network increases since the number of nodes in the branch and bound tree gets larger. However, optimal solutions could be found for medium sized networks, having less than 50 nodes, in reasonable amount of time. As this is a design problem, the same problem will not be solved frequently. For this reason, longer solution times are reasonable to find the optimal solution since regeneration is a costly process. Besides the deviations of the results of the heuristic algorithms are small when compared with the optimal solutions. Therefore, it expected that using heuristic algorithms to solve the problem in large networks gives satisfactory solutions.

Bibliography

[1] A. Agrawal and R. E. Barlow. A survey of network reliability and domination theory. *Operations Research*, 32(3):478–492, May-June 1984.

[2] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

[3] D.O. Awduche. Mpls and traffic engineering in ip networks. *IEEE Communications Magazine*, 37:42–47, December 1999.

[4] D.O. Awduche and B. Jabbari. Internet traffic engineering using multi-protocol label switching (mpls). *Computer Networks*, 40:111–129, 2002.

[5] S. Banerjee and A.P.K. Reddy. Parallel algorithm for shortest pairs of edge-disjoint paths. *Journal Of Parallel and Distributed Computing*, 33:165–171, 1996.

[6] U. Brandes, W. Schlickenrieder, G. Neyer, D. Wagner, and K. Weihe. A software packagwe of algorithms and heuristics for disjoint paths in planar networks. *Discrete Applied Mathematics*, 92:91–110, 1999.

[7] M. Conforti, R. Hassin, and R. Ravi. Reconstructing edge-disjoint paths. *Operations Research Letters*, 31:273–276, 2003.

[8] L. Coupry. A simple linear algorithm for the edge-disjoint (s,t)-path problem in undirected planar graphs. *Information Processing Letters*, 64:83–86, 1997.

[9] T. Gomes, J. Craveirinha, L. Martins, E. Martins, and M. Pascoal. An algorithm for calculating the k most reliable disjoint paths with a maximum

number of arcs. In *Proceedings of the European Conference on Safety and Reliability*, pages 1659–1666, 2001.

[10] M. Grtschel, C.L. Monma, and M. Stoer. Design of survivable networks. In M.O. Ball et al., editor, *Handbooks in OR & MS*, volume 7, pages 617–671. 1995.

[11] W. T. Huh. Finding 2-edge connected spanning subgraphs. *Operations Research Letters*, 32:212–216, 2004.

[12] D. Huygens, A. R. Mahjoub, and P. Pesneau. Two edge-disjoint hop-constrained paths and polyhedra.

[13] J.Strand, A.L. Chiu, and Tkach R. Issues for routing in the optical layer. *IEEE Communications Magazine*, 39:81–87, February 2001.

[14] H. Kerivin and A.R. Mahjoub. Separation of partition inequalities for the (1,2)-survivable network design problem. *Operations Research Letters*, 30:265–268, 2002.

[15] S.W. Kim and S.W. Seo. Regenerator placement algorithms for connection establishment in all-optical networks. *IEE Proceedings Commun.*, 148(1):25–30, February 2001.

[16] J. Kleinberg and E. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. *Journal of Computer and System Sciences*, 57:61–73, 1998.

[17] C. LI, S. T. McCormick, and D. Simchi-Levi. The complexity of finding two disjoint paths with min-max objective function. *Discrete Applied Mathematics*, 26:105–115, 1990.

[18] G. Shen and W. D. Grover. Segment-based approaches to survivable translucent network design under vaious ultra-long-haul system reach capabilities. *Journal of Optical Networking*, 3(1):1–24, January 2004.

[19] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[20] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.

[21] J. Vygen. Np-completeness of some edge-disjoint path problems. *Discrete Applied Mathematics*, 61:83–90, 1995.

[22] X. Yang and B. Ramamurthy. Dynamic routing in translucent wdm optical networks. In *Proceedings of IEEE ICC'2002*, New York,NY, 2002.

[23] Y. Ye, T.H. Cheng, and C. Lu. Routing and wavelength assignment algorithms for translucent optical networks. *Optics Communications*, 229:233–239, 2004.

[24] E. Yetginer. Traffic engineering and regenerator placement in mpls and gmpls networks with restoration. Master's thesis, Bilkent University, 2002.

[25] E. Yetginer and E. Karaṣan. Regenerator placement and traffic engineering with restoration in gmpls networks. *Photonic Network Communications*, 6(2):139–149, 2003.