ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

SURVIVABLE VIRTUAL TOPOLOGY DESIGN IN OPTICAL WDM NETWORKS USING NATURE-INSPIRED ALGORITHMS

Ph.D. THESIS

Fatma CORUT ERGİN

Department of Computer Science

Computer Science Programme

MAY 2012

ISTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

SURVIVABLE VIRTUAL TOPOLOGY DESIGN IN OPTICAL WDM NETWORKS USING NATURE-INSPIRED ALGORITHMS

Ph.D. THESIS

Fatma CORUT ERGİN
(704052006)

Department of Computer Science

Computer Science Programme

Thesis Advisor: Assoc. Prof. Dr. Ayşegül YAYIMLI

MAY 2012

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

DOĞA ESİNLİ ALGORİTMALAR KULLANARAK OPTİK WDM AĞLARDA HATAYA BAĞIŞIK SANAL TOPOLOJİ TASARLAMA

DOKTORA TEZİ

Fatma CORUT ERGİN
(704052006)

Bilgisayar Bilimleri Anabilim Dalı

Bilgisayar Bilimleri Programı

Tez Danışmanı: Assoc. Prof. Dr. Ayşegül YAYIMLI

MAYIS 2012

Fatma CORUT ERGİN, a Ph.D. student of ITU Informatics Institute 704052006 successfully defended the thesis entitled **"SURVIVABLE VIRTUAL TOPOLOGY DESIGN IN OPTICAL WDM NETWORKS USING NATURE-INSPIRED ALGORITHMS"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Ayşegül YAYIMLI** Istanbul Technical University	.....
Co-advisor :	**Assoc. Prof. A. Şima UYAR** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Emre HARMANCI** Istanbul Technical University	.....
	Prof. Dr. Sema OKTUĞ Istanbul Technical University	.....
	Prof. Dr. Haluk TOPCUOĞLU Marmara University	.....
	Prof. Dr. Cem ERSOY Boğaziçi University	.....
	Asst. Prof. Feza BUZLUCA Istanbul Technical University	.....

Date of Submission : **22 February 2012**
Date of Defense : **2 May 2012**

FOREWORD

Completing my PhD degree has been probably the most challenging activity of my life so far. Of course, this would not have been possible without the help of many people. I wish to thank them all for their valuable support.

My first debt of gratitude must go to my advisors, Assoc. Prof. Dr. Ayşegül Yayımlı and Assoc. Prof. Dr. A. Şima Uyar. I have been very lucky to have such excellent advisors, who complemented each other wonderfully well. They patiently provided the vision, encouragement and advise necessary for me to proceed through my PhD study and complete my PhD thesis. I want to thank them for their endless support and serving as a role model to me as a junior member of academia. The joy and enthusiasm they have for their research was very motivational for me, even during tough times in the PhD study. They were always cheerful, and this gave our weekly meetings a happy atmosphere. I will certainly miss these meetings.

I would warmly thank to my jury members, Prof. Dr. Emre Harmancı, Prof. Dr. Sema Oktuğ and Prof. Dr. Haluk Topcuoğlu for their support, guidance and helpful suggestions. Their feedbacks have served me well throughout my thesis. Special thanks to Prof. Dr. Cem Ersoy and Prof. Dr. Feza Buzluca for kindly accepting to be a member of my defense jury. I am deeply grateful to all members of the jury for agreeing to read the manuscript and to participate in the defense of this thesis.

I cannot skip the contributions of Elif Kaldırım in my thesis. She was a very hardworking fellow and a great friend.

This list is incomplete without acknowledging my colleagues, Berna Kiraz, Betül Demiröz Boz, and Sanem Arslan. They have always provided a motivating and fun-filled environment, always supported me and above all they have been great friends all the time.

I kindly acknowledge the Scientific and Technological Research Council of Turkey (TÜBİTAK) for providing scholarship during my PhD education.

Last but not the least, my husband, Lemi Orhan Ergin, whose love and encouragement allowed me to finish this journey. He already has my heart, so I will just give him a heartfelt “thanks”. I also wish to thank my whole family. Their love provided my inspiration and was my driving force.

May 2012 Fatma CORUT ERGİN

TABLE OF CONTENTS

Page

ABBREVIATIONS

ACO	:	Ant Colony Optimization
ACF	:	Autocorrelation Function
ACS	:	Ant Colony System
AICS	:	Adaptive Iterated Constructive Search
AIS	:	Artificial Immune System
ANTS	:	Approximate Nondeterministic Tree Search
AS	:	Ant System
BWAS	:	Best-Worst Ant System
DEA	:	Differential Evolutionary Algorithms
EA	:	Evolutionary Algorithms
EAS	:	Elitist Ant System
ES	:	Evolutionary Strategies
GA	:	Genetic Algorithms
GP	:	Genetic Programming
GRASP	:	Greedy Randomized Adaptive Search Procedure
ILP	:	Integer Linear Programming
IP	:	Internet Protocol
HH	:	Hyper Heuristics
LLH	:	Low Level Heuristics
MILP	:	Mixed Integer Linear Programming
MMAS	:	Max-Min Ant System
NIH	:	Nature Inspired Heuristics
NSFNET	:	NSF Network
OXC	:	Optical Cross Connect
PSO	:	Particle Swarm Optimization
RAS	:	Rank-based Ant System
RWA	:	Routing and Wavelength Assignment
SA	:	Simulated Annealing
TS	:	Tabu Search
VT	:	Virtual Topology
WDM	:	Wavelength Division Multiplexing

LIST OF TABLES

Page

LIST OF FIGURES

Page

SURVIVABLE VIRTUAL TOPOLOGY DESIGN IN OPTICAL WDM NETWORKS USING NATURE-INSPIRED ALGORITHMS

SUMMARY

Today, computer networking has become an integral part of our daily life. The steady increase in user demands of high speed and high bandwidth networks causes researchers to seek out new methods and algorithms to meet these demands. The transmission speed in the network is directly affected by the transmission medium. The most effective medium to transmit data is the fiber. Optical networks are designed for the best usage of the superior properties of the fiber, e.g. high speed, high bandwidth, low bit error rate, low attenuation, physical strength, cheapness, etc. The world's communication network infrastructure, from backbone networks to access networks, is consistently turning into optical networks.

One of the most important properties of the optical networks is the data transmission rate (up to 50 Tb/s on a single fiber). Today, with the help of the wavelength division multiplexing (WDM) technology, hundreds of channels can be built on a single fiber. WDM is a technology in which the optical transmission is split into a number of non-overlapping wavelength bands, with each wavelength supporting a single communication channel operating at the desired rate. Since multiple WDM channels, also called lightpaths, can coexist on a single fiber, the huge fiber bandwidth can be utilized.

Any damage to a physical link (fiber) on the network causes all the lightpaths routed through this link to be broken. Since huge data transmission (40 Gb/s) over each of these lightpaths is possible, such a damage results in a serious amount of data loss. Two different approaches can be used in order to avoid this situation: 1. Survivability on the physical layer, 2. Survivability on the virtual layer. The first approach is the problem of designing a backup link/path for each link/path of the optical layer. The second approach is the problem of designing the optical layer such that the optical layer remains connected in the event of a single or multiple link failure. While the first approach provides faster protection for time-critical applications (such as, IP phone, telemedicine) by reserving more resources, the second approach, i.e. the survivable virtual topology design, which has attracted a lot of attention in recent years, aims to protect connections using less resources. The problem that will be studied in this project is to develop methods for survivable virtual topology design, that enables effective usage of the resources.

Survivable virtual topology design consists of four subproblems: determining a set of lightpaths (forming the virtual topology), routing these lightpaths on the physical topology (routing and wavelength assignment (RWA) problem), so that any single fiber cut does not disconnect the virtual topology (survivable virtual topology mapping), assigning wavelengths, and routing the packet traffic. Each of these subproblems can

be solved separately. However, they are not independent problems and solving them one by one may degrade the quality of the final result considerably. Furthermore, the survivable virtual topology design is known to be NP-complete. Because of its complexity, it is not possible to solve the problem optimally in an acceptable amount of time using classical optimization techniques, for real-life sized networks. In this thesis, we solve the survivable virtual topology design problem as a whole, where the physical topology and the packet traffic intensities between nodes are given.

In the first phase, we propose two different nature inspired heuristics to find a survivable mapping of a given virtual topology with minimum resource usage. Evolutionary algorithms and ant colony optimization algorithms are applied to the problem. To assess the performance of the proposed algorithms, we compare the experimental results with those obtained through integer linear programming. The results show that both of our algorithms can solve the problem even for large-scale network topologies for which a feasible solution cannot be found using integer linear programming. Moreover, the CPU time and the memory used by the nature inspired heuristics is much lower.

In the second phase, we propose four different hyper-heuristic approaches to solve the survivable virtual topology design problem as a whole. Each hyper-heuristic approach is based on a different category of nature inspired heuristics: evolutionary algorithms, ant colony optimization, simulated annealing, and adaptive iterated constructive search. Experimental results show that, all proposed hyper-heuristic approaches are successful in designing survivable virtual topologies. Furthermore, the ant colony optimization based hyper-heuristic outperforms the others.

To balance the traffic flow over lightpaths, we adapt a flow-deviation method to the ant colony optimization based hyper-heuristic approach. We explore the performance of our hyper-heuristic approach for both single and double-link failures. The proposed approach can be applied to the multiple-link failure problem instances by only changing the survivability control routine. The experimental results show that our approach can solve the problem for both single-link and double-link failures in a reasonable amount of time. To evaluate the quality of the HH approach solutions, we compare these results with the results obtained using tabu search approach. The results show that HH approach outperforms tabu search approach both in solution quality and CPU time.

DOĞA ESİNLİ ALGORİTMALAR KULLANARAK OPTİK WDM AĞLARDA HATAYA BAĞIŞIK SANAL TOPOLOJİ TASARLAMA

ÖZET

Günümüzde bilgisayar ağları hayatımızın önemli bir parçası ve ihtiyaç haline gelmiştir. İstediğimiz veriye, istediğimiz anda, daha hızlı, daha güvenli ve kesintisiz olarak erişme isteğimiz aslında ağ altyapısının nasıl tasarlanacağını belirlemektedir. Kullanıcıların istekleri sürekli artarken, teknolojik gelişmelerle birlikte yeni yöntem ve algoritmalarla bu istekleri karşılamanın yolları aranmaktadır. Ağdaki aktarım hızı, aktarım ortamından doğrudan etkilenmektedir; bugün uzak mesafelere en yüksek kapasiteli ve hızlı aktarımın yapılabileceği ortam ise fiberdir. Fiber optik ağlar, fiberin üstün özelliklerini (hız, düşük bit hata oranı, elektromanyetik ortamlardan etkilenmeme, düşük işaret zayıflaması, fiziksel dayanıklılık, ucuzluk, güvenlilik, vs.) en iyi kullanacak şekilde tasarlanan ağlardır. Günümüzde dünyadaki iletişim ağ altyapısı, omurga ağlardan erişim ağlarına kadar, hızla fiber optik ağlara dönüşmektedir.

Optik ağların en önemli özelliklerinden biri veri aktarım hızıdır, tek bir fiberden teorik olarak 50 Tb/s veri aktarımı yapılabileceği hesaplanmaktadır. Bugün, lider iletişim firmaları 100 Gb/s ya da 1 Tb/s hızda veri aktarımı yapacak kanalllardan bahsedebiliyorsa, bu, fiziksel altyapı optik bir omurgadan oluştuğu içindir. Dalgaboyu bölmeli çoğullama (WDM) teknolojisi sayesinde bir fiber üzerinde aynı anda kurulabilecek kanal sayısı, günümüz teknolojisiyle yüzler mertebesine çıkabilmektedir. Dalgaboyu bölmeli çoğullama teknolojisi ile, optik aktarım birbiriyle çakışmayan dalgaboyu bantlarına bölünür ve her bir dalgaboyu istenen hızda çalışan, ışıkyolu olarak adlandırılan, bir iletişim kanalını destekler. Böylece, yakın gelecek için öngörülen çok yüksek hızlara çıkmadan bile, bir fiberden herbiri birkaç on Gb/s hızda çalışan yüz dolayında ışıkyolu geçebilmektedir.

Bu kadar yüksek hızlarda veri aktarımı, özellikle her bir fiberinde çok sayıda kanalın taşındığı omurga ağlarda bir konuya büyük önem kazandırmaktadır: Hataya bağışıklık. En sık rastlanan hata olan, bir fiberin, herhangi bir nedenle kesilmesi (çoğunlukla inşaat makineleri tarafından, ya da doğal afetlerce), fiber tamir edilene kadar, her saniyede birkaç terabitlik veri kaybı anlamına gelecektir. Örnek olarak 10 km uzunlukta bir fiberin kopma sıklığı 11 yılda birdir. Omurga ağlarda yüzlerce, bazen binlerce, kilometrelik fiberler döşendiği gözönüne alındığında, böyle bir hata durumu için tedbir alınmaması düşünülemez.

Optik ağ üzerindeki herhangi bir fibere zarar gelmesi demek bu fiber üzerinden yönlendirilmiş olan tüm ışıkyollarının kopması demektir. Her bir ışıkyolu üzerinden yüksek miktarda (40 Gb/s) veri aktarımı yapıldığından, böyle bir zarar ciddi veri kayıplarına neden olabilir. Temel olarak fiber kopmasına karşı geliştirilen iki yaklaşım vardır. Birinci yaklaşımda fiber üzerinden geçen her bir bağlantının, yani

ışıkyolunun, yedek yollarla korunmasıdır. İkinci yaklaşım ise, özellikle birçok internet uygulamasına da uygun ve yeterli olacak şekilde, ışıkyollarının oluşturduğu sanal topolojinin bağlı kalmasının sağlanmasıdır. Bu ikinci yaklaşımda herbir ışıkyoluna ayrı ayrı yedek koruma yollarının atanması yerine, sanal topolojinin korunması dikkate alınarak, üst katmanların (paket katmanları) koruma mekanizmalarının devreye girebilmesi için gereken minimum koşulların sağlanması amaçlanmaktadır.

Birinci yaklaşım belirli düzeylerde garantili bir koruma sağlarken yüksek miktarda ağ kaynağının atıl durmasına neden olmakta, dolayısıyla bu kadar üst düzey koruma gerektirmeyen uygulamalar için pahalı bir çözüm sunmaktadır. Son yıllarda özellikle dikkat çeken ikinci yaklaşım ise, daha ekonomik bir yöntemle iletişimin kopmaması garantisini vermekte, ancak daha yavaş bir düzeltme sağlamaktadır. Günümüzde birçok uygulama bağlantı kopmadığı sürece paket katmanının, yeni yol bulma gibi hata düzeltme mekanizmalarının devreye girmesi için gerekli olan, dakikalar mertebesindeki gecikmelere toleranslıdır (web dolaşımı, dosya aktarımı, mesajlaşma, uzaktan erişim gibi). Bu yaklaşım ilkine göre daha az ağ kaynağının atıl kalmasına neden olarak kullanıcıya daha ekonomik hizmet verilmesini sağlayacaktır. Bu çalışmada üzerinde durduğumuz hataya bağışık sanal topoloji tasarımı problemi de bu ikinci yaklaşımı benimsemektedir.

Hataya bağışık sanal topoloji tasarımı problemi kendi içinde dört alt probleme ayrılmaktadır: ışıkyollarının belirlenmesi (sanal topolojiyi oluşturma), bu ışıkyollarının herhangi bir fiber kopması durumunda bile sanal topolojinin bağlı kalmasını sağlayacak sekilde fiziksel topoloji üzerinde yönlendirilmesi, dalgaboyu atanması, ve paket trafiğinin yönlendirilmesi. Bu alt problemler ayrı ayrı çözülebilir. Ancak, bunlar bağımsız problemler değildir ve bunları tek tek çözmek elde edilen çözümün kalitesinin çok düşük olmasına neden olabilir. Bununla birlikte, hataya bağışık sanal topoloji tasarımı problemi NP-karmaşıktır. Karmaşıklığı nedeniyle bu problemin, gerçek boyutlu ağlar için, klasik optimizasyon teknikleriyle kabul edilebilir zamanda çözülmesi mümkün değildir. Bu çalışmada, fiziksel topolojinin ve düğümler arası paket trafiği yoğunluğunun bilindiği durumlar için, hataya bağışık sanal topoloji tasarımı problemi bütün halinde ele alınmaktadır.

Tezin ilk aşamasında, hataya bağışık sanal topoloji tasarımı probleminin alt problemi olan hataya bağışık sanal topoloji yönlendirmesi problemi ele alınmıştır. Verilen bir sanal topoloji için en az kaynak kullanarak hataya bağışık yönlendirme yapmak için iki farklı doğa-esinli algoritma önerilmektedir: evrimsel algoritmalar ve karınca kolonisi optimizasyonu. Öncelikle önerilen algoritmaların problem için uygun parametre kümesi belirlenmiş, daha sonra, algoritmaların başarımını ölçmek için, deneysel sonuçlar tamsayı doğrusal programlama (ILP) ile elde edilen sonuçlarla karşılaştırılmışır. Sonuçlar göstermektedir ki; önerdiğimiz iki algoritma da, tamsayı doğrusal programlama ile uygun bir çözüm bulunamayan büyük ölçekli ağlar için dahi, problemi çözebilmektedir. Bunun yanında, doğa-esinli algoritmalar çok daha az CPU zamanı ve hafıza kullanmaktadır. Elde edilen çözüm kalitesi ve çözüm için kullanılan CPU zamanının kabul edilebilir düzeyde olması, her iki doğa-esinli algoritmanın da gerçek boyutlu ağlar için kullanılabileceğini doğrulamaktadır.

İkinci aşamada, hataya bağışık sanal topoloji tasarımı problemini bir bütün halinde çözmek için dört farklı üst-sezgisel yöntem önerilmektedir. Önerilen

üst-sezgisel yöntemler alt seviyedeki sezgiselleri seçme asamasında dört farklı yöntem kullanmaktadır: evrimsel algoritmalar, benzetimli tavlama, karınca kolonisi optimizasyonu ve uyarlamalı yinelenen yapıcı arama. Deneysel sonuçlar tüm üst-sezgisel yöntemlerin hataya bağışık sanal topoloji tasarımı problemini çözmede başarılı olduğunu göstermektedir. Ancak, karınca kolonisi optimizasyonu tabanlı üst-sezgisel diğerlerine göre daha üstün sonuçlar vermektedir. Işıkyolları üzerindeki trafik akışını dengelemek için, karınca kolonisi optimizasyonu tabanlı üst-sezgisele akış deviasyonu yöntemi de eklenmiştir.

Literatürde hataya bağışık sanal topoloji tasarımı problemini ele alan tüm çalışmalar çift fiber kopması durumunu gözardı etmektedir. Bu çalışmada, önerdiğimiz üst-sezgisel yöntemin başarımını hem tek hem de çift fiber kopması durumları için değerlendirdik. Önerdiğimiz yöntem çoklu fiber kopması durumları için çok kolay şekilde adapte edilebilmektedir. Tek yapılması gereken hataya bağışıklık kontrolünü yapan yordamın değiştirilmesidir. Deneysel sonuçlar göstermiştir ki, önerdiğimiz karınca kolonisi optimizasyonu tabanlı üst-sezgisel hataya bağışık sanal topoloji tasarımı problemini hem tek hem de çift fiber kopması durumları için kabul edilebilir bir sürede çözebilmektedir. Üst-sezgisel yöntemlerin hataya bağışık sanal topoloji tasarımı çözmedeki başarımını değerlendirebilmek amacıyla, karınca kolonisi optimizasyonu tabanlı üst-sezgiselle elde edilen sonuçlar, literatürde bu problem için önerilmiş başka bir yöntemle karşılaştırılmıştır. Sonuçlar üst-sezgisel yöntemlerin, çok daha az CPU zamanı kullanarak, problem için daha kaliteli çözümler verdiğini göstermektedir.

1. INTRODUCTION

In today's world, the steady increase in user demands of high speed and high bandwidth networks causes researchers to seek out new methods and algorithms to meet these demands. The most effective medium to transmit data is the fiber. Optical networks [1] are designed for the best utilization of the superior properties of the fiber, e.g. high speed, high bandwidth, low bit error rate, low attenuation, physical strength, cheapness, etc. Today, with the help of the wavelength division multiplexing (WDM) technology, hundreds of channels can be set up on a single fiber. WDM is a technology in which the optical transmission is split into a number of non-overlapping wavelength bands, with each wavelength supporting a single optical communication channel operating at the desired rate. The upper layers (IP, Ethernet, etc.) can transmit data using these optical channels.

A wavelength-routed WDM network provides end-to-end optical channels between two nodes in the network that are not necessarily connected directly by a fiber in the physical layer. These optical channels are called lightpaths. Two nodes become virtually neighbors when a lightpath is set up between them. More than one lightpath, each operating on different wavelengths, can be routed on the same fiber. All the lightpaths set up on the network form the virtual topology (VT). Given the physical parameters of the network (physical topology, optical transceivers on the nodes, number of wavelengths that can be carried on the fibers, etc.) and the mean traffic intensities between the nodes, the problem of determining the lightpaths to be set up on the physical topology is known as the VT design problem.

In a WDM network, failure of a physical link (fiber) may result in the failure of several lightpaths routed through this link. Since huge amount of data (40 Gb/s, 100 Gb/s, or more) can be transmitted over each of these lightpaths, a fiber damage may result in a serious amount of data loss. Two different approaches can be used to avoid data loss [1]:

1. Survivable design of the physical layer

2. Survivable design of the virtual layer

The first approach is the problem of designing a backup link/path for each link/path of the virtual layer. The second approach is the problem of designing the virtual layer such that the virtual layer remains connected in the event of a single or multiple-link failure. While the first approach provides faster recovery for time-critical applications (such as, IP telephony, telemedicine) by reserving more resources; the second approach, i.e., the survivable VT design, aims to provide a continuous connectivity, using less resources. The continuous connectivity is ensured by designing the VT such that the VT remains connected in the event of a single or multiple-link failure.

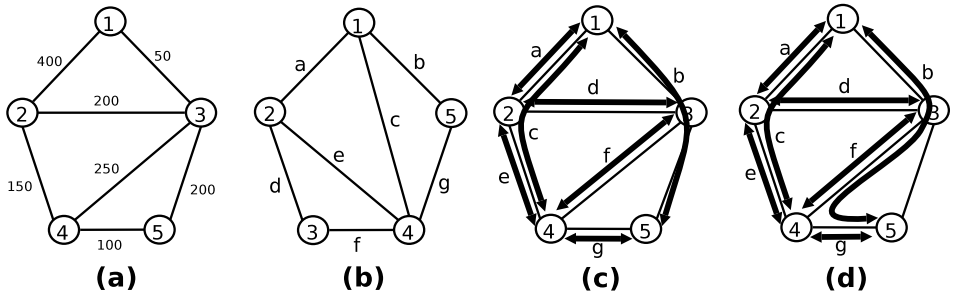

Figure 1.1: a.Physical Topology, b.Virtual Topology, c.Survivable Mapping, d.Unsurvivable Mapping.

To illustrate the survivable VT design problem, assume that we have a physical network topology as in Figure 1.1.a and the virtual network topology to be routed on this physical topology is designed as in Figure 1.1.b. To obtain a survivable design of this VT, the mapping may be as in Figure 1.1.c. In this survivable mapping, a single failure on any physical link does not disconnect the VT. However, if the routing of only one lightpath is changed, e.g., as in Figure 1.1.d, we end up with an unsurvivable mapping. In this case, if a failure occurs on the physical link between nodes 4 and 5, the nodes connected with lightpaths *b* and *g* will not be able to communicate and node 5 will be disconnected from the rest of the network.

Survivable VT design consists of four subproblems:

1. Determining a set of lightpaths (forming the VT)

2. Routing these lightpaths on the physical topology (routing and wavelength assignment (RWA)), so that any single fiber cut does not disconnect the VT (survivable VT mapping problem)

3. Assigning wavelengths to the lightpaths

4. Routing the packet traffic on the VT

Each of these subproblems can be solved separately. However, they are not independent problems and solving them one by one may degrade the quality of the final result considerably. Any solution to these subproblems affects the solution of other subproblems, therefore, the result obtained by solving the subproblems one by one and iteratively, may not be the optimum.

The survivable VT mapping subproblem is known to be NP-complete [2], thus, the survivable VT design problem is also NP-complete [3]. Therefore, in order to move to a harder problem, in the first phase of this thesis, we solve the second subproblem. Because of its complexity, it is not possible to solve the survivable VT mapping subproblem optimally in an acceptable amount of time using classical optimization techniques, for real-life sized networks. Therefore, heuristic approaches are used.

Nature inspired computing is an umbrella term that covers computing techniques which are inspired from nature, or are modeled on natural processes, or are constructed by using biological materials. Some of these techniques are relatively new, while some are quite old and have been extensively researched in literature. Today, most of them have become state-of-the-art techniques for many hard to solve real-world problems. Evolutionary algorithms (EA) [4], ant colony optimization algorithms (ACO) [5], particle swarm intelligence techniques [6], artificial immune systems [7], neural networks [8], quantum computing [9] and DNA computing [10] are some of the approaches in the literature that fall into this category. These techniques are commonly referred to as nature inspired heuristics (NIH).

In the first phase of this thesis, we solve the survivable VT mapping problem, i.e., the second subproblem of the survivable VT design problem. Since this problem is NP-complete, we propose to use NIHs, and compare their performance. We choose

EAs due to their successful applications on NP-complete problems and ACO due to its successful performance on constrained combinatorial optimization problems. For both the EA and the ACO algorithms, we first perform a series of parameter tuning tests. Then, for the performance comparison experiments, we use the best settings we determined. Also, we compare their performance to the results of the basic ILP formulation and to an ILP relaxation proposed in [2]. As a result of the experiments, we show that, for the cases studied, both ACO and EA can solve the survivable VT mapping problem with 100% success and in a reasonable amount of time.

After solving an NP-complete subproblem of the survivable VT design problem using nature inspired algorithms, we propose methods to solve the design problem as a whole. In the second phase of this thesis, we solve the survivable VT design problem in optical WDM networks as a whole, where the physical topology and the packet traffic intensities between nodes are given. We determine a set of lightpaths (forming the VT), route these lightpaths on the physical topology, so that any fiber failure does not disconnect the VT, assign wavelengths, and route the packet traffic. We assume that the number of wavelengths and transceivers per node are limited. Both the single-link and double-link failure scenarios are considered. We use different hyper-heuristic (HH) [11] approaches to solve the problem.

A HH is a method used to manage low-level heuristics (LLH) at each step of an optimization process. This way, the best features of different simple greedy heuristics can be combined.

We propose four HH approaches to design a survivable VT for a given physical topology, while minimizing resource usage. The proposed HHs use different methods for heuristic selection: EA, simulated annealing (SA) [12], ACO, and adaptive iterated constructive search (AICS) [12].

From these methods, SA and EA are perturbative search methods, while AICS and ACO belong to the group of constructive search algorithms. Furthermore, SA and AICS are single point search methods, whereas, EA and ACO work on a population of solution candidates. The experiments show that the ACO-based HH approach solves the survivable VT design problem for large size networks with a 100% success

rate. To balance the traffic flow on lightpaths, i.e., improve the scale-up, we adapt a flow-deviation method to the ACO-based HH approach.

Since optical networks span a huge geographical area, it is possible to have link failures on different parts of the network at the same instant. In this study, we explore the performance of our HH approach for both single and double-link failures. For double-link failures, we design the VT considering the failure of each pair of links through the network. We should stress that, the proposed approach can be applied to the multiple-link failure problem instances without any modifications. The only change is the survivability control routine which has a complexity of $O(n^2)$ for double-link failure cases, whereas it is $O(n)$ for cases with single-link failures. However, since the search algorithm is not changed, run times will not be affected considerably. The experimental results show that our approach can solve the problem for both single-link and double-link failures in a reasonable amount of time, i.e., 20 minutes for single-link failures and 30 minutes for double-link failures, respectively.

1.1 Contribution

As the first contribution of this thesis, we propose using nature inspired heuristics (NIH) for the survivable VT mapping problem and demonstrate their efficiency. Evolutionary algorithms (EA) and ant colony optimization (ACO) algorithms are applied to the problem after a set of parameter tuning tests. To assess the performance of the proposed algorithms, we compare the experimental results with those obtained through integer linear programming (ILP), i.e., the basic ILP formulation and an ILP relaxation proposed in [2]. The results show that both of our algorithms can solve the problem even for large-scale network topologies for which a feasible solution cannot be found using ILP. Moreover, the CPU time and the memory used by the NIHs is much lower. The solution quality and the CPU time usage results prove that both of our NIHs can easily be applied to real-world applications.

The second contribution is solving the survivable VT design problem as a whole. Most studies in the literature consider only the survivable VT mapping subproblem. The ILP solution [3] of the design problem constrains the problem to small-sized networks. The tabu search based approach considering the VT design problem [13] as a whole

constrains the nodal degrees of the VT. In this thesis, to solve the survivable VT design problem as a whole for large-sized networks, we propose four different HH methods based on EA, ACO, AICS, and SA. The experimental results show that, for the cases we studied, all the HH approaches can find a feasible solution for the problem with a 100% success rate. To evaluate the performance of the HH approach solutions, we compare these results with the results obtained using the tabu search approach. The results show that HH approach outperforms the tabu search approach both in solution quality (i.e., the resource usage of the solutions) and CPU time usage.

The third contribution is that, this is the first study considering double-link failure situations for the survivable VT design problem. The experimental results show that for double-link failure situations, our approach can be used to design survivable virtual topologies in a reasonable amount of time without any change in the algorithm.

1.2 Outline of the Thesis

The rest of the thesis is organized as follows. The definition of the problem is given in Chapter 2, including the related literature. In Chapter 3, NIHs are explained briefly, and, in Chapter 4, the details of the application of the NIHs to the survivable VT mapping problem are given. The experimental study for the survivable VT mapping problem is discussed thoroughly in Chapter 5. Next, hyper-heuristics are explained in Chapter 6.1 and the details of the HH designs for the survivable VT design problem are given in Chapter 6. Then, in Chapter 7 the experimental results for the survivable VT design problem are given and these results are discussed thoroughly. Finally, a conclusion followed by the future work can be found in the last chapter.

1.3 Academic Publications

As a result of the PhD research reported in this thesis, the following publications have been produced:

Journal publications

- Fatma Corut Ergin, Elif Kaldırım, Ayşegül Yayımlı, A. Şima Uyar, “Ensuring Resilience in Optical WDM Networks With Nature-Inspired Heuristics”, IEEE/OSA

Journal of Optical Communications and Networking, Vol. 2 Issue 8, pp.642-652, 2010. (SCI-E)

International conference publications

- Fatma Corut Ergin, Ayşegül Yayımlı, A. Şima Uyar, “Survivable Cross-layer Virtual Topology Design using a Hyper-heuristic Approach”, ICTON 2011, Stockholm, Sweden, Jun. 26-30, 2011, (Invited Paper).
- Fatma Corut Ergin, A. Şima Uyar, Ayşegül Yayımlı, “Investigation of Hyper-heuristics for Designing Survivable Virtual Topologies in Optical WDM Networks”, EvoStar - EvoWorkshops 2011, Torino, Italy, April 27-29, 2011.
- Fatma Corut Ergin, Elif Kaldırım, Ayşegül Yayımlı, A. Şima Uyar, “Performance Analysis of Nature Inspired Heuristics for Survivable Virtual Topology Mapping”, IEEE GLOBECOM 2009, Hawaii, USA, Nov. 30-Dec.4, 2009.
- Elif Kaldırım, Fatma Corut Ergin, A. Şima Uyar, Ayşegül Yayımlı, “Ant Colony Optimization for Survivable Virtual Topology Mapping in Optical WDM Networks”, ISCIS 2009, Turkish Republic of Northern Cyprus, Sept. 14-16, 2009.
- Fatma Corut Ergin, A. Şima Uyar, Ayşegül Yayımlı, “An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks” EvoStar - EvoWorkshops 2009, Tübingen, Germany, April 15-17, 2009.

2. PROBLEM DEFINITION AND RELATED LITERATURE

2.1 Optical WDM Networks

Today the most effective medium to transmit data is the fiber, and optical networks [1] are designed for the best usage of the superior properties of the fiber (high speed, low signal degradation, high bandwidth, physical strength, cheapness, reliability, etc.).

In optical networking terminology, the graph consisting of all the nodes in the network and the fiber links connecting these nodes is called the physical topology. The fiber links are also called the physical links. These physical links are assumed to be bidirectional and they may be associated with a weight which is the length of the fiber link.

WDM is a technology in which the optical transmission is split into a number of non-overlapping wavelength bands, with each wavelength supporting a single communication channel operating at the desired rate. With the help of WDM technology, it is possible to build hundreds of optical channels on a single fiber. These optical communication channels built on the fiber are called lightpaths.

A lightpath may span multiple physical links, and if there is no wavelength converter in the network, the lightpath is required to have the same wavelength throughout its path (the wavelength-continuity constraint). The intermediate nodes the lightpath traverses, are equipped with optical cross connects (OXC). These OXC enable switching the incoming wavelength to the outgoing wavelength. A typical OXC with n input and n output fibres, and m wavelengths is given in Figure 2.1. An OXC with n input and n output fibers capable of handling m wavelengths can be thought of as m independent nxn optical switches. These optical switches are preceded by a wavelength demultiplexer and followed by a wavelength multiplexer to implement an OXC.

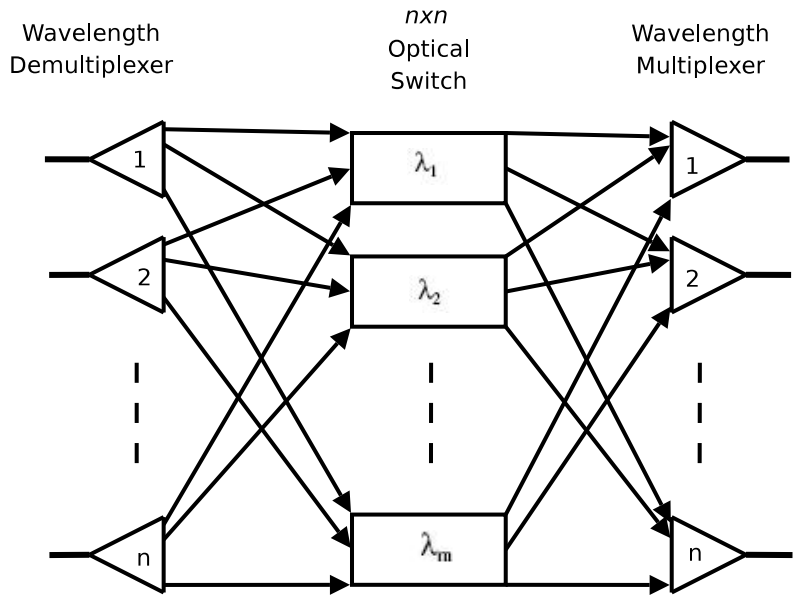

Figure 2.1: An Optical Cross Connect (OXC) with *n* input and *n* output fibres, and *m* wavelengths.

Establishing a lightpath between two nodes in the physical topology makes them optically neighbors, even if they are not physically neighbors, providing a circuit-switched interconnection between these nodes. All the lightpaths established on the physical topology form the virtual topology (VT) (or logical topology). The nodes in the VT usually correspond to the nodes in the physical topology. In Figure 2.2, a physical network is given, in which lightpaths are set up to allow communication between the nodes which are not physically connected by a fiber. In the figure, the lightpaths are indicated with dotted lines.

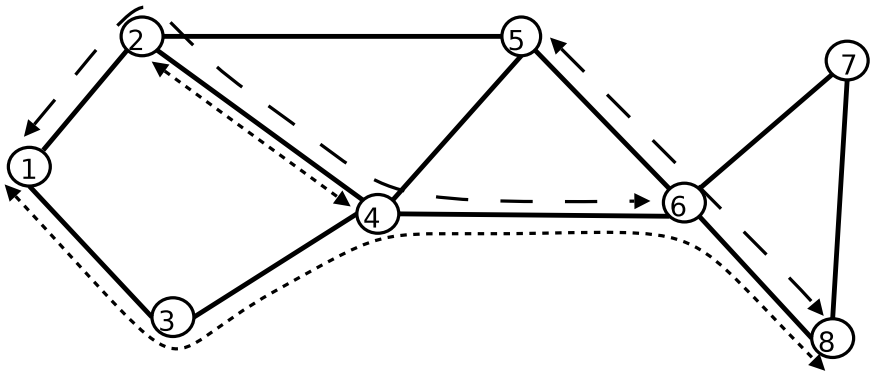

Figure 2.2: An optical WDM network. The dotted lines show lightpaths.

In an optical WDM network, each node is equipped with a set of transmitters and receivers (transceivers). The transmitter at a node sends data to the other nodes in the network and the receiver receives data from the other nodes in the network. If each node in an n-node network has n-1 transceivers, and each physical link has enough number of wavelengths, then a lightpath can be established between each node of the network, leading to a problem-free network. However, these transceivers are expensive equipments, and in a large-sized network, only a few of them can be used at each node. Similarly, the number of wavelengths on each physical link is limited. As a result, the number of lightpaths that can be established on a network is limited.

Once the lightpaths forming the VT are determined, each lightpath should be routed in the network and a wavelength should be assigned to it. This problem is known as routing and wavelength assignment (RWA) problem. RWA problem is also referred to as VT mapping problem. After mapping the VT onto the physical topology, the next problem to solve is to route the packet traffic on the VT.

As a result, while designing a VT for a given physical topology and packet traffic demands between the nodes of this physical topology, the problems to be solved can be summarized as:

- Determining a proper VT, i.e., a good set of lightpaths, considering the mean packet traffic rates between nodes,
- Routing the lightpaths of the VT over the links in the physical topology (RWA problem),
- Assigning wavelengths optimally to the lightpaths,
- Routing packet traffic over the VT

These subproblems can be solved separately. However, any solution to each of these subproblems affects the others. Therefore, solving them separately may lead to sub-optimal solutions. The whole problem is referred to as VT design problem [14–16].

In optical networks, any damage to a physical link (fiber) on the network causes all the lightpaths routed through this link to be broken. Since huge amounts of data (e.g.

40 Gb/s) can be transmitted over each of these lightpaths, a fiber damage may result in a serious amount of data loss. Therefore, survivability is an important issue in optical WDM networks. Survivable VT design problem deals with designing a VT on a given physical topology, such that in case of any single or multiple physical link failure the VT remains connected.

2.2 Virtual Topology Mapping (Routing and Wavelength Assignment (RWA)) Problem

While designing a VT for a given physical topology and packet traffic demands between nodes of this physical topology, the first subproblem to solve is determining the lightpaths for a proper VT. Once the lightpaths forming the VT are determined, each lightpath should be routed in the network and a wavelength should be assigned to it. The VT mapping problem, also known as the RWA problem, is defined as follows:

Given:

- Physical topology, i.e., the nodes and the physical links that connect the nodes,
- Virtual topology, i.e., the lightpaths established between the nodes,
- Number of wavelengths per physical link,

Find:

- A route for each lightpath over the physical topology,
- A proper wavelength for each lightpath.

Objectives:

- Minimize the number of physical links used in the whole physical topology.
- Minimize the total number of wavelength-links used in the whole physical topology, i.e., the sum of wavelengths used on each physical link.

2.2.1 Related literature

RWA problem is proved to be NP-complete problem that cannot be solved exactly in polynomial time [17, 18]. There are a number of studies devoted to the RWA problem proposing both exact and heuristic approaches [19, 20].

The RWA problem can be partitioned into two subproblems: 1) routing and 2) wavelength assignment. In the literature three basic approaches are proposed for the routing subproblem: 1) fixed routing, 2) fixed-alternate routing, and 3) adaptive routing [21–24]. Fixed routing is the simplest approach, while adaptive routing is proved to offer the best performance. Fixed-alternate routing offers a trade-off between complexity and performance.

Several heuristics have been proposed for the wavelength-assignment subproblem, [25–31]. These heuristics are random wavelength assignment, first fit, least used, most used, min product, least loaded, max sum, relative capacity loss, wavelength reservation, and protecting threshold, distributed relative capacity loss. Among these algorithms, distributed relative capacity loss is proved to yield the best performance.

An MILP (Mixed Integer Linear Programming) formulation to the RWA problem is given in [32]. In this formulation, given a set of connection requests and fixed number of wavelengths, the objective is to maximize the number of established connections.

In [33], advanced Boolean satisfiability techniques are proposed to solve the RWA problem. They show how to formulate the RWA problem as a SAT instance and evaluate several advanced SAT techniques in solving the problem. They experiment with different network topologies.

Ant colony optimization (ACO) is quite successful in solving routing problem in packet and circuit switched networks [34, 35]. It is applied both to the static [36] and the dynamic [37, 38] RWA problem. Particle swarm optimization is applied to the RWA problem in only [39]. There are a few studies on VT mapping [40,41] using EAs. Tabu search is also used to solve the RWA problem [42].

In the first study that applies ACO to this problem [36], the static RWA problem is considered. In this work, the objective is to minimize the number of wavelengths used in the given network. The authors use a simple greedy heuristic for wavelength

assignment. According to this approach, ants select their routes according to the weight of attraction of each physical link. Ants use a tabu list of previously visited nodes in order to avoid loops and backtracking. Various methods are tested for pheromone updating.

The next studies that use ant colony optimization approach handle the dynamic RWA problem [37,38], where Garlick et al. [37] is the first group that used ACO on dynamic RWA problem. In this approach, whenever a new connection request arrives, some of the ants are launched from source to destination. While deciding which path to use, ants use the length of the path and the number of available wavelengths along the path. After an ant reaches its destination, global pheromone updating takes place. The best path is decided when all paths complete their exploitation tasks. It is shown that, this algorithm results in less connection rejection than an exhaustive search over all available wavelengths for the shortest path [43]. In [43] adaptive routing in optical WDM networks is studied.

Ngo et al. propose an approach for the dynamic survivable RWA problem [38]. First, they design a new routing table structure to solve this problem. They use ants to observe the state changes in the network and to update these tables regularly. The results show that this algorithm outperforms the other alternative methods in terms of blocking probability.

The work that use particle swarm optimization for RWA problem in WDM networks uses a hybrid algorithm inspired from the ant system [39]. In this work, for the routing part of the problem, particles are used to determine the path together with the ant system. For the wavelength assignment part, they use first-fit algorithm. In [44], the bee colony optimization heuristic is applied to solve the RWA problem.

2.3 Survivable Virtual Topology Mapping Problem

Survivable mapping of VT is an NP-complete problem [2] and defined as the problem of finding a route for each lightpath, such that in case of a single physical link failure, the VT remains connected.

Consider a physical topology, which is composed of a set of nodes $N = 1..n$ and a set of edges E, where (i, j) is in E if there is a link between nodes i and j. Each link has a capacity of W wavelengths. The VT, on the other hand, has a set of virtual nodes N_L, which is a subset of N, and virtual edges (lightpaths) E_L, where an edge (s,t) exists in E_L if both node s and node t are in N_L and there is a lightpath between them.

An Integer Linear Programming (ILP) formulation of survivable lightpath routing of a VT on top of a given physical topology is given in [2]. Based on this formulation, a number of different objective functions can be considered for the problem of survivable mapping. The simplest objective is to minimize the number of physical links used. Another objective is to minimize the total number of wavelength-links used in the whole physical topology. A wavelength-link is defined as a wavelength used on a physical link.

The aim of lightpath routing is to find a set of physical links that connect the nodes of the lightpaths. Let $f_{ij}^{st} = 1$ if virtual link (s,t) is routed on physical link (i, j) and 0, otherwise. Since our objective is to minimize the total number of wavelength-links used in the whole physical topology, we can formulate the objective as in (**2.1**):

$$\texttt{Minimize} \quad \sum_{\substack{(i,j)\varepsilon E \\ (s,t)\varepsilon E_L}} f_{ij}^{st} \tag{2.1}$$

As we mentioned above, there are a number of constraints in survivable lightpath routing problem. These constraints are given in the following equations.

a)Connectivity constraint: for each pair (s,t) in E_L:

$$\sum_{js.t.(i,j)\varepsilon E} f_{ij}^{st} - \sum_{js.t.(j,i)\varepsilon E} f_{ji}^{st} = \begin{cases} 1, & \texttt{if } s = i \\ -1, & \texttt{if } t = i \\ 0, & \texttt{otherwise} \end{cases} \quad \forall i \varepsilon N. \tag{2.2}$$

The connectivity constraints are also called flow conservation constraints for routing one unit of traffic from node s to node t. (**2.2**) requires that equal amounts of flow due to lightpath (s,t) enter and leave each node that is not the source or destination of (s,t).

b)Survivability constraint:

$$\begin{matrix}\forall(i,j)\varepsilon E \\ \forall S \subset N_L\end{matrix} \text{ , } \sum_{(s,t)\varepsilon CS(S,N_L-S)} f_{ij}^{st} + f_{ji}^{st} <\mid CS(S,N_L-S) \mid . \qquad \textbf{(2.3)}$$

The survivability constraint states that for all proper cuts of the VT, the number of cut-set links flowing on any given physical link is less than the size of the cut-set. This means that all the lightpaths of a cut-set cannot be routed using the same physical link. A cut is a partition of the set of nodes N into two parts S and $N-S$. Each cut defines a set of edges consisting of edges in E with one endpoint in S and the other endpoint in $N-S$. The $CS(S,N_L-S)$ in the equation means the set of edges as the cut-set associated with the cut (S,N_L-S), and $\mid CS(S,N_L-S) \mid$ means the number of edges in the cut-set.

Consider the VT in Figure 1.1.b. In this VT, the lightpaths b and g forms one of the cut-sets, that is, they divide the VT into two parts. If these lightpaths are routed on the same physical link as in Figure 1.1.d, we end up with an unsurvivable mapping. Thus, the aim of survivability constraint is to prevent routing of all the lightpaths in a cut-set of the VT on the same physical link.

c)Wavelength capacity constraint:

$$\forall(i,j)\varepsilon E \text{ , } \sum_{(s,t)\varepsilon E_L} f_{ij}^{st} \leq W. \qquad \textbf{(2.4)}$$

The third constraint ensures that the number of wavelengths on a physical link is no more than its capacity W.

d)Integer flow constraints: $f_{ij}^{st} \varepsilon \{0,1\}$.

2.3.1 Related literature

The survivable VT mapping problem is first addressed as Design Protection [45, 46] in the literature. Crochat et al. use tabu search to find the minimum number of source-destination pairs that become disconnected in the event of a physical link failure. In their first study [45], no wavelength capacity is considered in the solution approach, while in the second study [46], they add the wavelength capacity constraint.

In other heuristic approaches to survivable VT mapping, Ducatelle et al. propose a local search based algorithm, FastSurv [47, 48]. The objective in their study is to minimize the number of unsurvivable nodes. To find the unsurvivable nodes, they remove the physical links from the physical topology one by one and search for alternative routes for the lightpaths routed on the corresponding link. If there is no alternative path, these physical and virtual links are considered as unsurvivable.

Kurant and Thiran [49–51] use a heuristic that divides the survivable mapping problem into subproblems. In [49], they propose a new algorithm (SMART) that divides the survivable VT mapping problem into subproblems. The algorithm in their next study [50] considers more than one link failure and in [51], they aim to solve the problem for larger networks and add heuristics to SMART algorithm. In their study, they use mesh topologies for both the physical and the virtual layers.

Simulated annealing is another heuristic used to solve the survivable VT mapping problem [52]. In [53], ACO is used considering back-up paths on the physical layer. In this study, Ngo et al. handle the survivable RWA problem on the physical layer, and use ACO to solve this problem.

Modiano and Narula-Tam use ILP to solve the VT mapping problem [2]. They add the survivability constraint in the problem formulation, such that, no physical link is shared by all virtual links belonging to a cut-set of the VT graph. However, they do not consider the capacity constraint. Their objective is to minimize the number of wavelengths used. For the cases when ILP cannot find an optimum solution in a reasonable amount of time due to the problem size, Modiano et al. propose two relaxations to ILP, which consider only small-sized cut-sets. These relaxations reduce the problem size; however, they may lead to suboptimal solutions.

In order to overcome the long execution time problem in ILP formulation, Todimala and Ramamurthy propose a new ILP formulation and they solved the problem for networks of up to 24 nodes [54]. In their work, besides the physical network and the virtual network topologies, the shared risk link groups should be known in advance. Similarly, because of the time-complexity of the problem different heuristics are tried in [55, 56].

2.4 Virtual Topology Design Problem

All the lightpaths set up on the network form the VT. Given the physical parameters of the network (physical topology, optical transceivers on the nodes, number of wavelengths that can be carried on the fibers, etc.) and the mean traffic rates between nodes, the problem of designing the lightpaths to be set up on the physical topology is known as the VT design problem. The objective of the VT design problem may be minimizing the network resources, minimizing the network-wide average packet delay (corresponding to a solution for present traffic demands), maximizing the packet traffic, or balancing the lightpath loads on the fibers, etc. The problem can be divided into four different subproblems [57], given in Section 2.1.

The VT design problem is defined as follows:

Given:

- Physical topology, i.e., the nodes and the physical links that connect the nodes,
- Average traffic rates between each node pair,
- Maximum number of lightpaths that can be established on a node, i.e., the number of transceivers per node,
- Number of wavelengths per physical link,
- Lightpath bandwidth capacity

Find:

- A collection of lightpaths to be established as a VT
- A route for each lightpath over the physical topology,
- A proper wavelength for each lightpath.
- A suitable routing of the packet traffic over the VT

Objectives:

- Minimize the resource usage of the network, i.e., the total number of wavelength-links used in the whole physical topology.

2.4.1 Related literature

There are only a few studies proposing solutions to the VT design problem as a whole. The MILP formulation of the VT design problem is given in [14, 58–60]. In [14] and [59], the number of wavelengths on fibers is not considered as a constraint and the wavelength assignment subproblem is omitted. Unlike these MILP formulations, in [58] and [60], number of wavelengths is introduced as a constraint to the problem. Different objectives are addressed in these formulations; minimize the network-wide average message delay [14, 58], minimize the network congestion [59], minimize the maximum offered load on any link (congestion) in the network [60].

an integer linear program (ILP) formulation for designing an optimal, stable virtual topology for time-varying demands.

The MILP formulation of the VT design problem grows quickly with the increase in the network size. This problem and also some of its subproblems are proved to be NP-complete. Therefore, heuristics are used to solve the problem for large-scale networks. However, the studies usually address only the RWA subproblem.

Saha et al. [16] use genetic algorithms (GA) to solve the VT design problem. They aim to find the best possible VT over a given wavelength-routed all-optical physical topology for wide area coverage. Their objective is to maximize the throughput as well as to minimize the delay for a given physical topology and traffic matrix. As in the VT design problem, they divide the problem into four categories: 1) determine good VT 2) route lightpaths over physical topology 3) wavelength assignment 4) route packet traffic on VT. For each of the sub-problems they use a different heuristic, and as a result of these four algorithms, a feasible VT is generated. For the initial population of the GA phase, a predefined number of VTs are created. After encoding these VTs to the defined string, genetic operators are applied to these individuals in the initial population. In the experimentation phase, queuing delay, propagation delay, average hop distance scalability with increase in throughput are studied.

2.5 Survivable Virtual Topology Design Problem

The survivability [61] is an important issue because of the high speed data transmission, especially for the backbone networks that carry several optical channels (lightpaths) on each fiber. In a WDM network, failure of a fiber (e.g. a fiber cut because of construction machines or natural disasters) may result in the failure of several optical channels and this leads to several terabits of data loss for every second until the problem is solved. For example, a fiber of 10 kilometers long is cut once every 11 years. Since there are hundreds, or sometimes thousands, of kilometers of fiber in a backbone network, it becomes a necessity to take precaution for this situation.

Different protection mechanisms have been proposed in the literature for the fiber and/or other network equipment failures [62]. Basically, there are two approaches for the fiber failure:

1. Survivability on the physical layer [63–65]

2. Survivability on the virtual layer [2, 3, 13, 15, 36–40, 45–47, 49, 53, 54, 66]

In the first approach, each connection passing through the fiber, i.e., the lightpath, is protected by assigning backup lightpaths that are disjointly routed from the connection's first lightpath. On the other hand, the second approach ensures the VT to be connected even in the failure of any single physical link. The first approach provides survivability by providing extra routes for each lightpath in the VT, however, with a cost of a high number of unemployed network resources. Thus, it offers an expensive solution for applications which may not need a high level of protection. The second approach, which has attracted attention especially in recent years, is a cost effective solution. Today, most applications are tolerant to latencies of several minutes of repair time needed by the packet layer (web search, file transfer, messaging, etc.), as long as the network connection is not terminated. This approach uses less network resources than the first one, thus, it enables service providers to offer a more economic service to their users.

In this study, our main aim is to design an efficient algorithm to find a survivable design of a given physical topology and mean traffic rates for this topology. The constraints for the problem are:

1. The transceiver capacity constraint.
2. The survivability constraint (given in inequality **2.3**).
3. The wavelength capacity constraint (given in inequality **2.4**).
4. The bandwidth capacity constraint.

The survivable virtual topology design problem is defined as follows [3]: Given:

- Physical topology, where P_{mn} is the number of physical links between nodes (m,n).
- Number of nodes in physical and virtual topologies, N and M, the number of wavelengths in fibers W, capacity of a lightpath C, and load factor α of a lightpath for maximum allowed loading.
- Traffic matrix λ_{sd} which denotes the average traffic flow between nodes (s,d).
- Set of nodes N_L in the virtual topology.

Find:

- Virtual topology connectivity V_{ij} which equals to 1 if there is a lightpath between nodes (i,j).
- Survivable routing of the lightpaths over the physical topology where p_{mn}^{ij} equals to 1 if lightpath (i,j) is routed through fiber link (m,n).
- Routing of the packet traffic over the virtual topology λ_{ij}^{sd} which equals to 1 if traffic (s,d) is routed through lightpath (i,j).

Two different objective functions can be considered.

- First objective is minimizing the total number of lightpaths in the virtual topology

$$\sum_{i,j \in N_L} V_{ij}, \tag{2.5}$$

which equals to minimizing total number of transceivers used in $N_L \subset N$ set of nodes.

- Second objective is minimizing the total number of wavelength-links used in physical topology,

$$\sum_{i,j \in N_L m,n \in N} p_{mn}^{ij} \tag{2.6}$$

The detailed ILP formulation for the survivable VT design problem can be found in [3]. Based on this formulation, the objective is to minimize the resource usage of the network, i.e., the total number of wavelength-links used in the physical topology. A wavelength-link is defined as a wavelength used on a physical link. For example, in Figure 1.1.c, 2 wavelength-links are used on the link between nodes 1 and 2, 1 wavelength-link is used on the link between nodes 1 and 3, ..., and a total of 9 wavelength-links are used in the physical topology.

2.5.1 Related literature

Solving the survivable VT design problem as a whole is NP-complete [3]. Therefore, most of the previous studies on this problem consider only the survivable VT mapping subproblem [15, 40, 45–47, 49]. There are only a few studies in the literature which try to solve all the subproblems together: a tabu search heuristic [13, 67], an ILP formulation [3], and an MILP formulation [68].

In [13, 67], Nucci et al. propose an approach that first uses a greedy heuristic to route the lightpaths on the physical topology. Then, for the routing of packet traffic on the VT, they use a tabu search mechanism. The routes for the lightpaths are designed to establish a survivable VT. The constraints in their study include transmitter and receiver constraints as well as wavelength capacity constraints. In their study, they address only three of the subproblems of VT mapping, the first subproblem, i.e., designing a proper VT is not handled. The lightpaths of randomly generated VTs are survivably routed, wavelengths are assigned and the given packet traffic is routed

on this random VT. The subproblems of the survivable VT design problem are solved sequentially.

An ILP formulation for the survivable VT design problem is proposed in [3]. Given the physical topology, the average packet traffic demand in the network, and the wavelength capacities, the formulation is designed to find the optimum survivable VT and the optimum routings on this VT. This ILP method can solve very small problem instances of up to 4 node physical topologies, optimally, because of the problem complexity.

In a recent work [68], Thulasiraman et al. propose solution for the problem with capacities on physical links and demands on virtual links. They present MILP formulations and heuristics to generate a survivable routing that maximizes the VT capacity and minimizes spare capacity requirements.

There are some other studies in the area of survivable VT design considering the node failures. The studies in [69] and [70] focuses on single-node failure. They propose algorithms satisfying that the VT remains connected after the failure of any node in the physical topology.

3. NATURE INSPIRED HEURISTICS

Nature inspired heuristics (NIH) are inspired from some mechanisms and processes in nature. Many optimization problems have become increasingly complex, so that, NIHs are gaining popularity in recent years. Various NIHs are proposed and many of them produce high quality solutions to complex real-world problems.

The most popular NIHs used in the literature can be collected under two main categories:

1. Search techniques starting from a single point
2. Search techniques starting from several points

Among the most common methods in the first group are simulated annealing (SA) [12], tabu search (TS) [71], and greedy randomized adaptive search procedure (GRASP) [72] algorithms. The second group includes evolutionary algorithms (EA) (genetic algorithms (GA), genetic programming (GP), evolutionary strategies (ES), etc.) [4], swarm intelligence algorithms [6] (ant colony optimization (ACO) [5], particle swarm optimization (PSO) [73], etc.), and artificial immune system (AIS) techniques [7].

In combinatorial optimization problems, a solution consists of a discrete set of subparts, which are called solution components. One method to classify search methods [12], relies on the underlying mechanism used to generate candidate solutions for a problem from the solution components:

1. A solution candidate can be transformed into another one by altering one or more of the components. A search algorithm which uses this approach is called a perturbative search algorithm. EA, PSO and SA are among the well known perturbative search methods.

2. On the other hand, it is also possible to construct a complete solution candidate by iteratively adding solution components to the partial solution candidates. A search algorithm which uses this approach is called a constructive search algorithm. GRASP and ACO algorithms fall within the category of constructive search methods.

Among these NIHs, in this thesis, to solve the survivable VT mapping problem, we chose EA due to its successful applications on NP-complete problems and ACO due to its successful performance on constrained combinatorial optimization problems. Therefore, only these nature inspired algorithms are explained in the following sections.

3.1 Evolutionary Algorithms

EAs are population-based stochastic search methods that have been applied successfully in many search, optimization, and machine learning problems [4]. EAs iteratively operate on a population of individuals (or chromosomes) that encode the possible solutions. EAs can be used to search for an individual yielding the optimum (i.e., minimum or maximum) numerical value of an evaluation function, called the fitness function. There are many variations of EAs in the literature, and the EA term provides a common basis for all of its variants.

There are two commonly used population dynamics in EAs: generational EAs and steady-state EAs. In steady-state EAs, the initial population is generated randomly or through some heuristics. Then, in each iteration a new individual is created and inserted into the population until the termination criteria are met. Commonly three genetic operators are used to generate new individuals: selection, recombination and mutation, based on modeling the natural evolution.

Individuals are selected as parents to produce offspring, based on their fitness values. The selection mechanism is usually probabilistic, where high quality solutions are more likely to become parents than low quality ones. There are various methods [4] for selecting the parents, such as, roulette-wheel selection, stochastic universal sampling, and tournament selection.

The recombination operator, is applied on the selected individuals with a predefined crossover probability. This operator takes two individuals and exchanges some genes between them to generate the offspring. The main idea in crossover is to combine good partial solutions of parents to form new solutions. There are many forms of crossover operators, such as, 1-point crossover, n-point crossover or uniform crossover [4]. Crossover occurs between two individuals with a predefined probability. This is called the crossover probability, which is usually a high value, close to 1.

After the crossover, offspring are mutated with a certain probability, called the mutation probability. Mutation operator is responsible for restoring diversity that may be lost from the repeated application of selection and crossover. This operator makes a small change in the solution, such as changing a bit from 1 to 0 in a bit string. Mutation probability is a small value close to 0.

Finally, the offspring is inserted into the population replacing one of the existing individuals based on some criteria, such as replacing the worst, replacing the oldest, replacing the most similar, etc. This cycle is performed until the termination criteria are reached, which may be defined as achieving a solution with a sufficient quality, reaching a predefined maximum number of generations or fitness evaluations, reaching convergence, etc.

The general view of an EA in pseudecode is given in Algorithm 1.

Algorithm 1 Steady-state Evolutionary Algorithm

```
1: generate initial population randomly
2: evaluate initial population
3: while not termination criteria met do
4:     select parents to mate
5:     recombine parents
6:     mutate the created offspring
7:     evaluate the created offspring
8: end while
```

3.2 Ant Colony Optimization Algorithms

ACO [5] is one of the most commonly used swarm intelligence techniques and is inspired from the behavior of real ants. ACO has been applied successfully to many

combinatorial optimization problems such as routing problems, assignment problems, scheduling and sequencing problems and subset problems [5].

One of the first successful implementations of ACO is the Ant System (AS). AS has been the basis for many ACO variants which have become the state-of-the-art for many applications. These variants include elitist AS (EAS), rank-based AS (RAS), MAX-MIN AS (MMAS), ant colony system (ACS), best-worst AS (BWAS), the approximate nondeterministic tree search (ANTS), and the hyper-cube framework. AS, ACS, EAS, RAS, MMAS and BWAS can be considered as direct variants of AS since they all use the basic AS framework. The main differences between AS and these variants are the pheromone update procedures and some additional details in the management of the pheromone trails.

The algorithmic flow of the basic ACO algorithm [5] is given in Algorithm 2. An iteration consists of the solution construction and pheromone update stages.

Algorithm 2 Basic Ant Colony Optimization Algorithm

```
1:  set ACO parameters
2:  initialize pheromone levels
3:  while stopping criteria not met do
4:      for each ant k do
5:          select random initial node
6:          repeat
7:              select next node based on decision policy
8:          until complete solution achieved
9:      end for
10:     update pheromone levels
11: end while
```

In each iteration, each ant in the colony constructs a complete solution. Ants start from random solution components and continue by adding the next component. For each component, the next component to be added is determined through a stochastic local decision policy based on the current pheromone levels and heuristic information between the current component and the others. An ant k determines its next component with a probability p_{ij}^k as calculated in Eq. **3.1**,

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k} \tau_{il}^\alpha \cdot \eta_{il}^\beta} & \text{if } j \in N_i^k \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(3.1)}$$

where τ_{ij} and η_{ij} are the pheromone level and heuristic information between components i and j respectively, α and β are parameters used to determine the effect of the pheromone level and heuristic information respectively, N_i^k is the allowed neighborhood of ant k when it is at node i.

Pheromone levels are modified when all ants have constructed a complete solution. First, the pheromone values are lowered (evaporated) by a constant factor between all component pairs. Then, pheromone values are increased between the components which the ants have used during their solution construction. Pheromone evaporation and pheromone update by the ants are implemented as given in Eq. **3.2** and Eq. **3.3** respectively,

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} \tag{3.2}$$

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k \tag{3.3}$$

where, m is the number of ants, $0 < \rho \leq 1$ is the pheromone evaporation rate and $\Delta\tau_{ij}^k$ is the amount of pheromone ant k deposits between the solution components it has used. $\Delta\tau_{ij}^k$ is defined as given in Eq. **3.4**, where C_k is the cost of the solution T_k built by the k-th ant.

$$\Delta\tau_{ij}^k = \begin{cases} 1/C_k & \text{if } edge(i,j) \in T_k \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

4. SURVIVABLE VT MAPPING USING NATURE INSPIRED HEURISTICS

In this thesis, to solve the survivable VT mapping problem, we chose evolutionary algorithms (EAs) due to their successful applications on NP-complete problems and ant colony optimization (ACO) due to their successful performance on constrained combinatorial optimization problems.

Designing a solution representation that is well-suited to the problem is crucial in EA and ACO performance. The survivable virtual topology (VT) mapping problem can be seen as a search for the best routing of lightpaths through physical links. Therefore, we use a solution encoding inspired from [40] for both heuristics. For this encoding, first, the k-shortest paths for each lightpath are determined. Then, a solution candidate is represented as an integer string of length l, where l is the number of lightpaths in the VT. Each location on the solution string gives the index of the shortest path for the corresponding lightpath, which can take on values between $[1..k]$, where k is the predefined number of shortest paths for each lightpath. For example, assume that $k = 3$ and there are 5 lightpaths (l) in the VT, a sample solution can be [2 1 1 3 1]. This means that, the first lightpath is routed using its 2^{nd} shortest path, the second lightpath is routed using its 1^{st} shortest path, etc.

Our objective is to minimize the total cost of resources (i.e., wavelength-links) used throughout the network. This resource cost is evaluated in two different ways:

- by considering the actual lengths of the physical links (link-cost)
- by counting the number of physical links used (hop-count)

There are survivability and capacity constraints (see Section 2.3) to be considered in the survivable VT mapping problem. Constraint violations are considered as penalties in the fitness evaluation stage of the EAs. In ACO, constraints are taken into consideration during solution construction, i.e., no constraint violations are possible.

4.1 Proposed Evolutionary Algorithms

We designed a steady-state EA with duplicate elimination, where a new individual is generated and inserted into the population at each iteration. After a random initial population generation, the EA operators are applied to the solution candidates until a predefined number of fitness evaluations are executed. The pseudocode for the EA used in this thesis is given in Algorithm 3.

Algorithm 3 Pseudocode for the Proposed Evolutionary Algorithm

```
1: generate initial population randomly
2: evaluate initial population
3: while not predefined number of fitness evaluations are done do
4:     select parents through binary tournament selection
5:     recombine selected parents uniformly
6:     mutate the created offspring according to the selected mutation type
7:     if offspring is not duplicate then
8:         evaluate the created offspring according to selected fitness evaluation type
9:         if fitness of offspring better than fitness of worst then
10:            offspring replaces worst in population
11:        end if
12:    else
13:        discard offspring
14:    end if
15: end while
```

Mating pairs are selected through binary tournament selection. Binary tournament selection considers randomly chosen two individuals; the fitter of the two is selected as a parent. After selecting two parents, these selected individuals undergo reproduction. Reproduction consists of uniform crossover and problem specific mutation operators. In uniform crossover, the offspring takes the genes from either parent with equal probability.

We define two different mutation operators. The first one is a simple random-reset mutation, called gene mutation (*gene*). In this type of mutation, the value of a gene is randomly reset to another value within the allowed range, i.e. between 1 and k, where k is the number of shortest paths.

The second is a problem-specific mutation operator, called least similar path mutation (*ls_pm*). This mutation operator considers the physical link similarities between the shortest paths of each lightpath, where similarity is defined as the number of common

physical links. In *ls_pm*, if mutation occurs on a gene, its current value is replaced by the index of the least similar shortest path for the corresponding lightpath. The pseudecode for *ls_pm* is given in Algorithm 4.

Algorithm 4 Pseudocode for *ls_pm*

```
1: for i = 1 to number of genes do
2:    if mutation takes place for this gene then
3:       find the index with the smallest similarity value
4:       if there are more than one with same smallest similarity then
5:          randomly select one of them
6:       end if
7:       change the value of the gene as the selected index
8:    end if
9: end for
```

Violations of the constraints for the problem, i.e., the survivability and the capacity constraints, are included as penalties in the fitness function. In order to determine if the solution is survivable or not, each physical link is deleted from the physical network one by one. If the VT becomes disconnected in the event of a broken physical link, the solution is taken as unsurvivable. The connectivity constraint is ensured within the algorithm, therefore, no explicit verification is needed. The penalty for an unsurvivable solution is determined in three different ways:

1. The total number of physical links, whose failure results in the network unsurvivability (us_1).

2. The sum of the total number of lightpaths that become disconnected in the event of each physical link failure [45, 47] (us_2).

3. The maximum of the total number of lightpaths that become disconnected in the event of each physical link failure [45] (us_3).

Each of the fitness evaluation methods we define, f_1, f_2, and f_3, use the penalty calculation methods, us_1, us_2, and us_3, respectively. In the first fitness evaluation method (f_1), the connectivity of the graph is checked for the failure of each physical link, which needs an algorithmic complexity of $O(e.n^3)$. On the other hand, for the second and the third fitness evaluation methods (f_2 and f_3), a shortest path algorithm is

applied for the failure of each physical link, which means $O(l.(e+n).logn)$ algorithmic complexity. Here, *e* is the number of physical links, *n* is the number of nodes, and *l* is the number of lightpaths.

A capacity constraint violation adds a penalty value which is proportional to the total number of physical links which are assigned more lightpaths than the predetermined wavelength capacity (wavelength capacity violation). Both penalties, i.e. the survivability and the capacity penalty, are multiplied with penalty factors and added to the fitness of the solution.

As a result, if we use the link cost evaluation method for the total cost of resources used throughout the network, the three fitness values are calculated as in the following:

1. f_1 = total length of the physical links used throughout the network + penalty factor*wavelength capacity violation + penalty factor*(us_1)
2. f_2 = total length of the physical links used throughout the network + penalty factor*wavelength capacity violation + penalty factor*(us_2)
3. f_3 = total length of the physical links used throughout the network + penalty factor*wavelength capacity violation + penalty factor*(us_3)

Similarly, if the hop count evaluation method is used for the total cost of resources used throughout the network, the fitness values are calculated as in the following:

1. f_1 = total number of wavelength-links used throughout the network + penalty factor*wavelength capacity violation + penalty factor*(us_1)
2. f_2 = total number of wavelength-links used throughout the network + penalty factor*wavelength capacity violation + penalty factor*(us_2)
3. f_3 = total number of wavelength-links used throughout the network + penalty factor*wavelength capacity violation + penalty factor*(us_3)

At the final stage of the EA loop, the fitness value of the offspring is calculated and compared to the worst individual in the current population. If the offspring is different

than this worst individual and has a better fitness, it replaces the worst individual, otherwise it is discarded.

4.2 Proposed Ant Colony Optimization Algorithms

ACO can be applied to the survivable VT mapping problem in a straightforward way. As the algorithm may lead to different solutions, ants route the lightpaths in a random order. For example, an ant may route the fifth lightpath and then the second lightpath and so on. The flexibility of selecting lightpaths in a random order may increase the number of feasible solutions because for a survivable solution if a lightpath can only be routed using a few shortest paths, routing this lightpath earlier may result in better solutions.

The physical topology is used as a construction graph on which ants travel and construct their solutions. Ants try to route lightpaths on the graph one-by-one. Each ant starts to route a random lightpath. The shortest paths between the end points of the lightpaths are provided to ants at the very beginning of the algorithm. Ants determine one of the shortest paths of the selected lightpath while visiting the nodes of the shortest paths on the construction graph and checking if the chosen shortest path violates the constraints (i.e. the capacity and the survivability constraints) or not. If the solution becomes infeasible for a shortest path selected for the lightpath, another shortest path is examined. If none of the shortest paths leads to a feasible solution, this ant is removed from the colony.

Ants decide their move on the construction graph based on heuristic and pheromone information. Two pheromone matrices are implemented: the lightpath and the shortest path pheromone matrix. According to the lightpath pheromone matrix value, τ^l_{ij}, lightpath j is chosen directly after lightpath i for mapping. The lightpath pheromone matrix has lightpaths in its columns and rows and gives the information which lightpath is more valuable to choose after the current lightpath. The shortest path pheromone matrix value, τ^s_{ij}, on the other hand, refer to the desirability of selecting j^{th} shortest path of lightpath i. Thus, the rows of the shortest path pheromone matrix have lightpaths and the columns have corresponding shortest paths and gives the information about which shortest path leads to better results when selected for the current lightpath.

The pheromone matrices are initialized in the beginning of the algorithm with the same value for each possible choice of the ant. The difference is created by the heuristic information η_{ij} that is inversely proportional to the length of the j^{th} shortest path of lightpath i, i.e. $\eta_{ij} = 1/d_{ij}$. The next lightpath to route is decided according to the lightpath pheromone matrix. To decide the proper shortest path of the chosen lightpath, the heuristic information is used together with the shortest path pheromone matrix. For the heuristic information, a combined pheromone matrix called the total pheromone matrix is used, which is computed as $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$. After the solution construction of each ant, the pheromone matrices are updated. The amount of accumulated pheromone is proportional to the quality of the solutions used to update the pheromone levels.

In a collaborative study [74, 75], 6 direct variants of Ant System (AS) (AS, elitist AS (EAS), rank-based AS, ant colony system (ACS), max-min AS, best-worst AS) for survivable VT mapping problem are designed. In the experimental study, we ended up with the result that, ACS and AS are the worst and EAS is the best of all. Since EAS outperforms the others for the survivable VT mapping problem, in this thesis, we use EAS as the AS variant, and solution approach using EAS is explained in this section. For the implementation details of other AS variants, you can refer to [74].

Algorithm 5 EAS Pheromone Update

```
1: for each ant do
2:     Global Pheromone Update
3: end for
4: for best-so-far ant do
5:     Weighted Global Pheromone Update
6: end for
```

The main differences between AS and its variants are the pheromone update procedures and some additional details in the management of the pheromone trails. There are three different pheromone update procedures: *local pheromone update*, *global pheromone update* and *weighted global pheromone update*. In EAS pheromone update procedure (see Algorithm 5), the global pheromone update (see Algorithm 6) and the weighted global pheromone update (see Algorithm 7) are used.

Algorithm 6 Global Pheromone Update

1: **for** each ant k **do**
2: **for** each lightpath i chosen j^{th} shortest path **do**
3: $\tau^s_{ij} + \frac{1}{resource\ usage}$ -> τ^s_{ij}
4: **end for**
5: **for** each lightpath i chosen before lightpath t **do**
6: $\tau^l_{it} + \frac{1}{resource\ usage}$ -> τ^l_{it}
7: **end for**
8: **end for**

Algorithm 7 Weighted Global Pheromone Update

1: **for** each ant k **do**
2: **for** each lightpath i chosen j^{th} shortest path **do**
3: $\tau^s_{ij} + \frac{weight}{resource\ usage}$ -> τ^s_{ij}
4: **end for**
5: **for** each lightpath i chosen before lightpath t **do**
6: $\tau^l_{it} + \frac{weight}{resource\ usage}$ -> τ^l_{it}
7: **end for**
8: **end for**

In global pheromone update, after each ant constructs its solution, both shortest path and lightpath pheromone matrices are updated on the edges the ants visited. The pheromone levels are updated according to the solution quality of each ant. In this thesis, $\frac{1}{resource\ usage}$ is accumulated on pheromones where *resource usage* is the number of wavelength-links used by the corresponding ant.

The weighted global pheromone update differs from the global update in the amount that is added to pheromones. The pheromone levels are increased with the amount $\frac{weight}{resource\ usage}$ where weight is used to allow the selected ants to deposit more or less pheromone. In EAS, global pheromone update is used only for the ant with the best performance at that instant (best-so-far ant). The weight is determined with the parameter *e*.

The algorithm flow of pheromone update is shown in Algorithm 8. Each ant, after constructing a solution, first evaporates pheromone levels on the visited edges and then calls the pheromone update procedure. In the end, the values of the total pheromone are updated as $\tau^{\alpha}_{ij} \cdot \eta^{\beta}_{ij}$.

Algorithm 8 Pheromone Trail Update

1: evaporate pheromones
2: call pheromone update procedure
3: compute total pheromone as $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$

While constructing a solution, each ant is initially placed on a randomly chosen start lightpath and one of its shortest paths is selected randomly. At each step, the ant iteratively adds an unvisited lightpath to its partial solution and determines the shortest path to route the selected lightpath. The solution construction terminates once all lightpaths have been routed.

Algorithm 9 Solution Construction

1: **for** each ant **do**
2: place ant on randomly selected lightpath
3: choose random shortest path for the selected lightpath
4: **end for**
5: **while** step $< n - 1$ **do**
6: step ++
7: **for** each ant **do**
8: move to next step
9: **end for**
10: **end while**
11: **for** each ant **do**
12: Pheromone Trail Update
13: **end for**

Solutions are constructed by applying the following simple constructive procedure to each ant:

(1) choose a start lightpath and one of its shortest paths,

(2) use lightpath pheromone information to select the next lightpath to route,

(3) use shortest path pheromone information together with the heuristic values to probabilistically determine the path between the nodes of the corresponding lightpath, until all lightpaths have been visited. If the ant cannot select a shortest path that makes the solution feasible, this ant is removed from the current iteration.

The algorithm flow of solution construction is shown in Algorithm 9.

Selecting the next lightpath to route is implemented using the pseudo-random proportional action choice rule. According to this rule, each lightpath is assigned a

probability proportional to the corresponding value in the lightpath pheromone matrix. A cumulative probability is calculated and a random point is selected in this probability array. The corresponding lightpath is selected. The shortest path for the selected lightpath is chosen using the same way but total pheromone matrix is used instead of the lightpath pheromone matrix. Algorithm 10 shows the pseudo-random proportional action choice rule.

Algorithm 10 Pseudo-random proportional action choice rule

```
1:  sum_prob = 0
2:  for each lightpath i do
3:     if visited then
4:        prop_ptr[i]=0
5:     else
6:        prop_ptr[i]= lightpath_pheromone[current lightpath][i]
7:        sum_prob += prop_ptr[i]
8:     end if
9:     select randomly a point in sum_prob
10:    calculate the associated lightpath l
11: end for
12: sum_prob = 0
13: for each shortest paths i of lightpath l do
14:    if not feasible then
15:       prop_ptr[i]=0
16:    else
17:       prop_ptr[i]= total_pheromone[l][i]
18:       sum_prob += prop_ptr[i]
19:    end if
20:    if sum_prob = 0 then
21:       remove ant from colony
22:    else
23:       select randomly a point in sum_prob
24:       calculate the associated shortest path
25:    end if
26: end for
```

4.3 An Illustrative Example

Consider the physical and virtual topologies given in Figure 4.1.a and b. The first 4 shortest paths calculated based on hop counts and link costs can be seen in Table 4.1. Here, the first column shows the lightpaths as source-destination node pairs. Four shortest paths found using hop counts are given in the next four columns, and 4 shortest paths found using link costs are given in the last four columns.

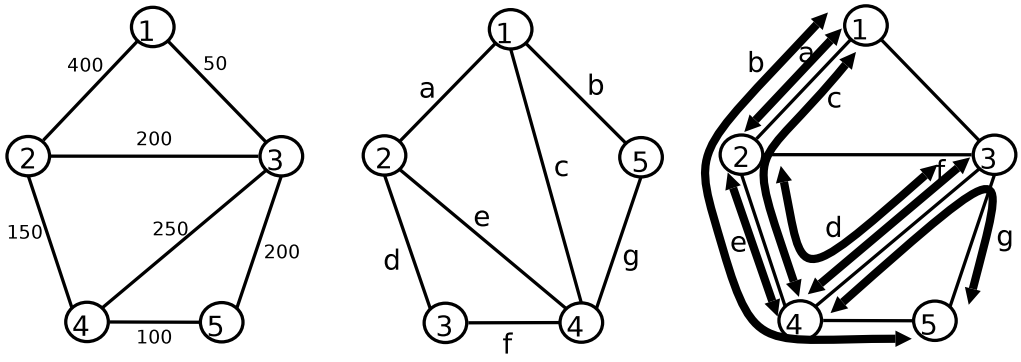

Figure 4.1: a.Physical Topology, b.Virtual Topology, c.Illustration of mapping for individual [1 1 2 3 1 1 2].

Table 4.1: Four different shortest paths for the lightpaths of the example VT given in Figure 4.1.b.

	hop count				**link cost**			
lightpath	sp_1	sp_2	sp_3	sp_4	sp_1	sp_2	sp_3	sp_4
1-2 (a)	1-2	1-3-2	1-3-4-2	1-3-5-4-2	1-3-2	1-2	1-3-4-2	1-3-5-4-2
1-4 (c)	1-2-4	1-3-4	1-3-2-4	1-3-5-4	1-3-4	1-3-5-4	1-3-2-4	1-2-4
1-5 (b)	1-3-5	1-2-4-5	1-2-3-5	1-3-4-5	1-3-5	1-3-4-5	1-3-2-4-5	1-2-4-5
2-3 (d)	2-3	2-1-3	2-4-3	2-4-5-3	2-3	2-4-3	2-1-3	2-4-5-3
2-4 (e)	2-4	2-3-4	2-1-3-4	2-3-5-4	2-4	2-3-4	2-3-5-4	2-1-3-4
3-4 (f)	3-4	3-2-4	3-5-4	3-1-2-4	3-4	3-5-4	3-2-4	3-1-2-4
4-5 (g)	4-5	4-3-5	4-2-3-5	4-2-1-3-5	4-5	4-3-5	4-2-3-5	4-2-1-3-5

Assume we have a solution encoded as [1 1 2 3 1 1 2], for which the physical topology, the VT, and the mappings are shown in Figure 4.1. According to the hop count evaluation method, this encoding means that the first lightpath uses the 1^{st} shortest path (1-2), the second one uses the 1^{st} shortest path (1-2-4), and the third one uses 2^{nd} shortest path (1-2-4-5), etc. If we sum up the number of wavelength-links used in this solution, we have a total of 12 wavelength-links for hop count evaluation. On the other hand, if the link cost evaluation method is used for the same solution encoding, the 1^{st} shortest path of the first lightpath is (1-3-2), which has a total cost of 250, the 1^{st} shortest path of the second lightpath is (1-3-4), with a cost of 300, and the 2^{nd} shortest path of the third lightpath is (1-3-4-5), with a cost of 400, etc. Summing up the cost of paths used for each lightpath gives a total cost of 2250 for the link cost evaluation.

For this sample solution, if EA is used, considering f_2, if a failure occurs on the physical links connecting nodes 1-2, 2-4, or 3-4, the VT becomes disconnected. In the EA, assume that, a penalty is applied to this solution based on the sum of the total number of lightpaths that become disconnected in the event of each physical link failure. If 1-2 link is broken, 3 lightpaths (a,b, and c) routed on this link will get disconnected, if 2-4 link is broken, 4 lightpaths (b,c,d, and e) will be disconnected, and if 3-4 link is broken, 2 lightpaths (d and f) would not find an alternative path to communicate. As a result, a penalty of $9 * p$ is added to the fitness, where p is the penalty factor. On the other hand, since ACO checks the constraints during the solution construction, such a solution would not be generated.

Six different fitness values for EA, calculated using three different evaluation functions and two different cost metrics are given in Table 4.2. When calculating the fitness, the link costs given in Figure 1.1.a are normalized dividing by 100. On the other hand, since ACO checks the constraints during solution construction, such a solution would not be generated.

Table 4.2: Fitness values for individual [1 1 2 3 1 1 2] calculated using three different evaluation functions and two different cost metrics. penalty factor=100.

	hop count	link cost
f_1	12+100*3=312	22.5+100*2=222.5
f_2	12+100*9=912	22.5+100*5=522.5
f_3	12+100*4=412	22.5+100*3=322.5

In the EA, if the crossover operator is applied to two individuals with the encodings [1 1 2 3 1 1 2] and [2 2 4 1 3 2 4], an offspring with encoding [2 1 2 1 3 2 2] would be created, if the second, third, and the last genes are taken from the first parent, and the other genes from the second parent. For the mutation operator of EA, assume a *ls_pm* mutation occurs on the second gene of this sample individual ([1 1 2 3 1 1 2]), the new individual becomes [1 2 2 3 1 1 2] or [1 4 2 3 1 1 2] with equal probability, if shortest paths are calculated according to hop count, since the first shortest path of the second lightpath has no common link with the second and the fourth shortest paths (least similar paths). However, if link costs are used instead, the individual becomes exactly [1 4 2 3 1 1 2], since this time the first shortest path of the second gene has one

common link with the second and the third shortest paths, and none with the fourth shortest path.

5. EXPERIMENTAL STUDY FOR SURVIVABLE VT MAPPING

To evaluate the performance of our newly designed solution approaches for the survivable virtual topology (VT) mapping problem, three different physical topologies are used throughout the thesis:

- The 14-node 21-link NSF-network (NSFNET) (see Figure 5.1)
- A 24-node 43-link network (see Figure 5.2)
- A 48-node 89-link network (A combination of two of the same topologies in Figure 5.2 joined at the nodes 1, 9 and 24) (see Figure 5.3)

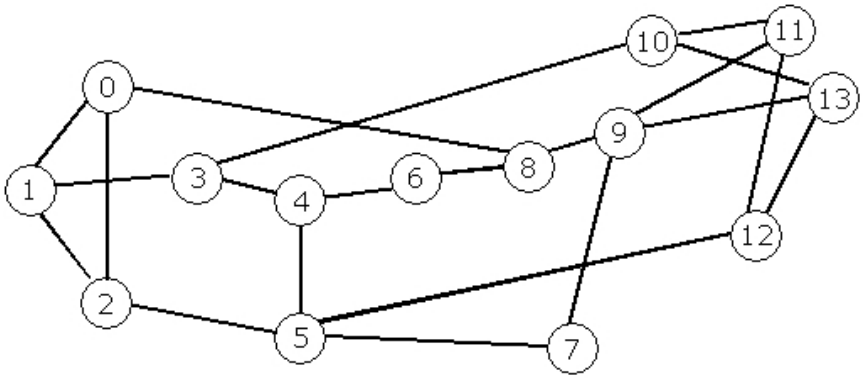

Figure 5.1: 14-node 21-link NSF-network topology.

For the experimental study of the survivable VT mapping problem, we used VTs of different average connectivity degrees: 3, 4, and 5. For each connectivity degree, we created random VTs.

The experimental study in this thesis has proceeded in four phases:

- Design an efficient evolutionary algorithm (EA) for the survivable VT mapping problem
- Fine-tune the parameters used in the EA for the survivable VT mapping problem
- Evaluate the EA performance for different sized physical networks

- Compare the EA and ACO performance designed for the survivable VT mapping problem and an ILP solution of the problem

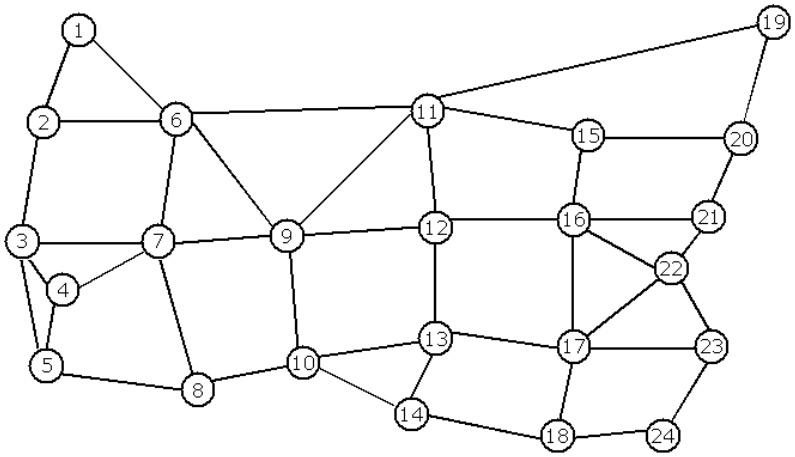

Figure 5.2: 24-node 43-link physical topology.

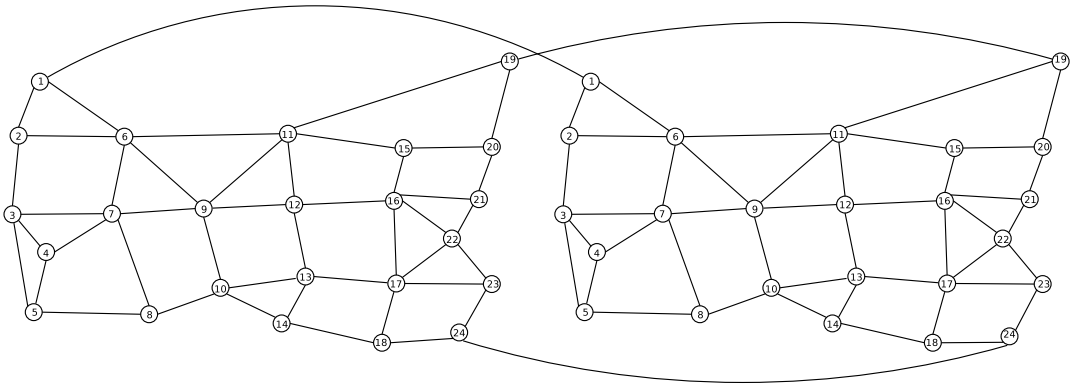

Figure 5.3: 48-node 89-link physical topology.

The experimental results for each phase are given in the following sections.

5.1 Designing an Efficient Evolutionary Algorithm for the Survivable Virtual Topology Mapping Problem

To find the most efficient EA design for the survivable VT mapping problem, we proposed two different mutation operators and three different fitness evaluation methods given in Chapter 4.1. To determine a good operator and evaluation method combination, we performed a series of experiments on our newly designed mutation operators and fitness evaluation methods. In these experiments, we used three metrics for performance comparisons, namely success rate (sr), first hit time (fht), and

correlation length. Success rate is defined as the percentage of program runs in which a survivable mapping that does not violate the capacity constraint is found. First hit time is the first iteration during which the best solution is encountered.

Ruggedness [76] is commonly used to analyze the structure of fitness landscapes and algorithm behavior. The autocorrelation function (ACF) is one of the simplest and the most commonly used techniques for analyzing the ruggedness of a landscape. The ACF looks at the amount of fitness correlation between the points in the search space and takes on values between $[-1,1]$. Values close to 1 denote a high positive correlation and values close to 0 show low correlation. The ACF value ρ_s is calculated as in Eq.(**5.1**).

$$\rho_s = \frac{\sum_{t=1}^{T-s}(f_t - \bar{f})(f_{t+s} - \bar{f})}{\sum_{t=1}^{T}(f_t - \bar{f})^2}. \tag{5.1}$$

where, s is the step size, T is the total number of sample points, f_t is the fitness of the t. solution, and $\bar{f}$ is the average fitness of all the points.

To compare the ruggedness of different landscapes, usually the correlation length, λ, as defined in Eq. (**5.2**) is used. A high correlation length means a smoother landscape, while a low correlation length shows a more rugged landscape.

$$\lambda = -\frac{1}{ln(\mid \rho_1 \mid)}. \tag{5.2}$$

A high correlation length means a smoother landscape, while a low correlation length shows a more rugged landscape.

A brief explanation of fitness landscape analysis is given in Appendix A.

For the experiments, we used two different physical topologies: the 14-node 21-link NSF network (see Figure 5.1) and a 24-node 43-link network (see Figure 5.2). For each physical topology we created 10 random VTs with average connectivity degrees of 3, 4, and 5. We assumed 10 wavelengths per physical link.

In the EA performance tests, we considered a maximum fitness evaluation count of 5000, mutation probability of $1/l$, where l is the number of lightpaths, crossover probability of 1.0, and population size of 100, and we ran the program 20 times for each parameter set[1]. These parameters values are determined according to the common

[1]On average a feasible solution is obtained for this problem in less than half a minute.

values of the corresponding parameter used in the literature. In the landscape analysis tests, we used a random walk consisting of 1000 steps and 50 runs.

We propose two different ways to evaluate the objective, i.e., minimizing the total cost of resources used throughout the network:

- considering the actual lengths of the physical links (link cost)
- counting the number of physical links used (hop count)

Since these two fitness evaluation methods result in different orders of fitness values, we apply different penalty factors. A penalty factor of 200 is used in the tests using hop count for shortest path calculation, and 300 in the tests using link cost. These values are determined experimentally.

The detailed test suit is given in Table 5.1 for the EA performance tests, and in Table 5.2 for the landscape analysis tests.

Table 5.1: Test suit for EA performance.

Number of physical topologies	2 (NSFNET and 24-node 43-link topology)
Number of virtual topologies	10 (3 connected)
	10 (4 connected)
	10 (5 connected)
Fitness function types	3 (f_1, f_2, and f_3 in Chapter 4.1)
Mutation types	2 (*gene* and *ls_pm* in Chapter 4.1)
Link evaluation methods	2 (hop count and link cost)
Shortest path numbers (k)	3 (2,3,4 for NSFNET; 5,10,15 for 24-node 43-link topology)
Number of runs	20
Total	2*30*3*2*2*3*20 = 43200 runs

5.1.1 Experimental results

All EA component combinations we tested were able to solve the problem for the NSF network for all 30 VTs, i.e., they were able to find survivable solutions of equal quality for all VTs. Since, the NSF network is a fairly simple and sparse graph, the results do not show meaningful differences between the tested combinations. Therefore, in this thesis we only report the results for the topology in Figure 5.2.

Table 5.2: Test suit for landscape analysis.

Number of physical topologies	2 (NSFNET and 24-node 43-link topology)
Number of virtual topologies	10 * 3 connected
	10 * 4 connected
	10 * 5 connected
Fitness function types	3 (f_1, f_2, and f_3 in Chapter 4.1)
Neighborhood types	2 (*gene* and *ls_pm* in Chapter 4.1)
Link evaluation methods	2 (hop count and link cost)
Shortest path numbers (k)	3 (2,3,4 for NSFNET; 5,10,15 for 24-node 43-link topology)
Number of runs	50
Total	2*30*3*2*2*3*50 = 108000 runs

The results of the experiments are given in Tables 5.3, 5.4, and 5.5. Table 5.3 shows the success rates, Table 5.4 shows the correlation lengths, and Table 5.5 shows the first hit times. In all the tables, f_i denotes the corresponding results for the i^{th} fitness evaluation method. In the top half of the tables, the results obtained using the *gene* mutation are given, whereas in the bottom half, the *ls_pm* results are given. Also, the results for three connectivity degrees, 3, 4, and 5, can be seen in the tables. For the correlation length and the EA first hit time results, we also showed the standard errors of the means (e) in the tables.

From the tables, we can see that there is a difference in all the combinations for smaller shortest path counts and low connectivity degrees. These are relatively difficult problems because the probability of finding potential mappings increases with the node degrees and the number of alternative shortest paths.

As can be seen in Table 5.3 the performance of the third fitness evaluation method (f_3) is the worst of all. This is confirmed by Table 5.4, where f_3 has lower correlation lengths than f_1 and f_2, showing a more rugged landscape.

A difference between f_1 and f_2 can be seen for the 3 connected virtual topology tests. We can say f_1 performs better than the others on relatively difficult problems. In order to confirm this result, we created 100 different virtual topologies of connectivity degree 3 and ran the program 100 times for each of these topologies, for 5 shortest paths, *gene* mutation, and hop count. We applied a 2 sample 2-tailed t-test with a significance level of 0.05 and saw a statistically significant difference between these

two fitness evaluation methods. However, if the algorithmic complexity of fitness evaluation methods, as explained in Chapter 4.1, are considered, we can say that f_2 is better. Therefore, f_2 should be preferred for virtual topologies having larger degrees of connectivity.

A difference can be seen between hop count and link cost results for f_1 and f_2 in Table 5.3. However, as a result of a t-test applied similarly as in the previous paragraph, we cannot say that there is a statistically significant difference between them. Similarly, there is no difference in their landscapes as given in Table 5.4. If we consider the first hit counts, we can prefer hop count to link cost.

A difference between *gene* mutation and *ls_pm* can be seen in Table 5.3. However, again as a result of the same type of t-test as in the previous paragraphs, we cannot say that there is a statistically significant difference between them. However, if we look at Table 5.5, we can see that the first hit times of *ls_pm* is lower than *gene* mutation. In Table 5.4, it can be seen that the correlation lengths for *ls_pm* is less than the *gene* mutation. This is an expected result, since the neighborhood definition for *ls_pm* means the most faraway results.

As a summary of the experiments, using *ls_pm*, hop count, f_2 (f_1 in sparse virtual topologies) can be recommended as components for an effective EA for the survivable VT mapping problem.

Table 5.3: Success rates for 24-node network.

		5 shortest paths						10 shortest paths						15 shortest paths					
		hop count			link cost			hop count			link cost			hop count			link cost		
		f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
gene	3	89	62	19.5	71	57	18.5	89.5	95	51.5	89	94	52.5	92	95.5	51	93.5	93.5	54.5
	4	90	90	88	90	90	78.5	100	100	97	100	100	92.5	98.5	99	95.5	99.5	100	93.5
	5	100	100	100	100	100	100	100	100	100	100	99.5	100	100	100	100	100	100	100
ls_pm	3	89	62.5	29	73	59.5	23	93	95	61	91.5	91	65.5	95	95.5	63.5	92	93.5	64
	4	90	90	90	90	89.5	88.5	100	100	100	99.5	100	96.5	98.5	99.5	100	99	100	96.5
	5	100	100	100	100	99.5	99	100	100	100	99.5	100	100	100	100	100	100	100	100

Table 5.4: Average and standard error of correlation lengths.

			5 shortest paths						10 shortest paths						15 shortest paths					
			hop count			link cost			hop count			link cost			hop count			link cost		
			f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3	f_1	f_2	f_3
gene	3	λ	15.3	16.21	9.37	15.49	16.52	11.44	15.22	16.53	10.11	15.47	16.8	11.54	14.77	16.41	10.24	14.74	16.41	11.22
		e	0.14	0.14	0.11	0.14	0.15	0.13	0.14	0.15	0.12	0.14	0.16	0.14	0.13	0.16	0.11	0.14	0.15	0.13
	4	λ	18.31	20.47	12.45	18.66	20.42	13.54	17.16	20.41	11.93	17.85	20.45	12.35	16.7	19.68	11.36	17.1	20.69	12.3
		e	0.2	0.22	0.16	0.2	0.22	0.17	0.19	0.23	0.14	0.2	0.22	0.14	0.17	0.21	0.11	0.19	0.23	0.14
	5	λ	16.97	22.82	13.86	18.74	23.82	14.83	16.38	22.81	13.31	17.08	23.23	14.25	15.49	22.42	12.34	15.96	22.19	13.32
		e	0.2	0.27	0.16	0.22	0.29	0.18	0.2	0.27	0.15	0.2	0.28	0.16	0.18	0.28	0.13	0.19	0.27	0.15
ls_pm	3	λ	10.81	11.09	6.59	10.55	10.78	7.02	10.44	10.83	6.73	10.2	10.56	7.13	10.51	11.18	6.91	10.18	10.84	7.12
		e	0.09	0.09	0.06	0.1	0.09	0.07	0.08	0.08	0.06	0.08	0.08	0.08	0.08	0.09	0.07	0.08	0.1	0.07
	4	λ	12.86	13.43	7.98	12.02	12.72	8.58	11.79	12.41	7.85	11.73	12.5	7.94	11.68	13.02	7.91	11.97	12.48	7.96
		e	0.14	0.13	0.07	0.11	0.11	0.08	0.11	0.11	0.08	0.1	0.12	0.07	0.1	0.12	0.08	0.12	0.11	0.07
	5	λ	12.09	14.76	9.53	12.41	15.26	9.77	11.77	14.27	9.15	11.96	14.26	9.37	11.96	14.2	9.36	11.88	13.53	9.21
		e	0.13	0.14	0.09	0.13	0.16	0.09	0.13	0.15	0.08	0.13	0.15	0.08	0.14	0.14	0.09	0.13	0.14	0.09

Table 5.5: Average and standard error of EA first hit times.

			5 shortest paths				10 shortest paths				15 shortest paths			
			hop count		link cost		hop count		link cost		hop count		link cost	
			f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2	f_1	f_2
gene	3	fht	2735	2715	4012	3530	4176	4256	4815	4834	4670	4666	4901	4895
		e	53.56	72.75	42.58	84.23	38.55	34.4	14.56	13.08	20.71	18.69	9.97	8.17
	4	fht	2721	2889	4436	4176	4343	4464	4900	4897	4751	4765	4922	4912
		e	48.15	55	31.35	61.08	33.39	27.9	7.35	8.8	13.94	13.53	5.47	8.29
	5	fht	3250	3382	4828	4573	4691	4764	4916	4921	4850	4871	4940	4923
		e	47.1	50.04	14.04	53.59	19.32	14.77	5.76	6.56	9.1	8.62	5.24	16.92
ls_pm	3	fht	2722	2875	3653	3334	3912	3993	4499	4624	4322	4442	4750	4814
		e	56.44	71.42	51.19	74.09	46.68	45.81	31.44	28.04	39.27	33.44	22.85	16.04
	4	fht	2621	2945	3964	3861	3849	4003	4731	4758	4469	4492	4867	4884
		e	55.42	59.4	45.1	62.98	45.65	45.03	19.75	20.03	30.44	25.22	9.37	8.72
	5	fht	2937	3146	4516	4367	4321	4415	4881	4878	4738	4768	4892	4915
		e	48.31	49.07	29.35	53.73	34.82	31	10.35	10.76	16.15	15.61	7.87	6.71

5.2 Evolutionary Algorithms Parameter Setting

In the first set of experiments given in Chapter 5.1, we found that the problem can be solved using EAs. In order to fine-tune the EA algorithm, we used the 24-node 43-link network topology in Figure 5.2, and 50 randomly created 5 connected virtual topologies for this network.

The hop count evaluation method is used for shortest path calculation because, unlike the link-cost method, it gives the exact number of used physical links, so when we compare two solutions based on hop count we can easily calculate how much better one solution is than the other. However, in the link-cost method, one solution may give a much higher resource usage value by using only one more physical link.

As the fitness evaluation method, we used f_2, i.e., the unsurvivability penalty calculated as the sum of the total number of lightpaths that become disconnected in the event of each physical link failure is added to the resource usage. To determine the penalty factor, we experimented with different values and as a result, we selected 200.

The number of shortest paths are selected as 15, which gives the largest search space in the experiments. The largest search space is selected because the maximum time allowed to the programs should be determined according to the problem that requires the longest time. The program is run 20 times for each set of parameters, and the results are the averages of these 20 runs.

Since *ls_pm* performs better than *gene* mutation according to the results in Chapter 5.1.1, we experimented with a variation of *ls_pm* based on the physical link costs. In this third mutation type (*ms_pm*), the current value of the *gene* is replaced by the index of the most similar shortest path for the corresponding lightpath. In the parameter setting tests, we include three mutation types, namely, *gene*, *ls_pm*, and *ms_pm*.

We performed a series of tests to fine-tune the parameters, including the crossover probability, the mutation probability, the population size, and the number of fitness

evaluations. In these experiments, we used three metrics for performance comparisons, namely success rate (sr), best fitness (bf), and first hit time (fht).

In Table 5.6, the commonly used range of values [4], the range of values we tested, and the range of values that can be selected are listed for the EA. The experimental results for the parameter setting tests are given in Tables 5.7, 5.8, 5.9,and in Figure 5.4.

Table 5.6: The Default and Tested Range of Values(RoV) for EA Parameters to be Fine-tuned.

PARAMETER	DEFAULT RoV	TESTED RoV	ELIGIBLE RoV
crossover probability	[0.5,1.0]	0.7, 0.8, 0.9, 1.0	1.0
mutation probability	$1/l$	$0.5/l$, $1/l$, $2/l$	$0.5/l$, $1/l$
population size	NA	5, 10, 20, 50, 100, 200	10, 20, 50

5.2.1 Experimental results

To find the most appropriate crossover probability, we used $1/l$ as mutation probability, where l is the number of lightpaths in the VT, 100 as population size, and 5000 as the maximum number of fitness evaluations allowed. The tests are performed for each mutation type we designed (see Chapter 4.1) and we tested 4 different crossover probabilities, i.e., 0.7, 0.8, 0.9, and 1.0. The results are given in Table 5.7. In the table *mt* means mutation type. From the table, we can see that there is always (except the case of success rate for *gene* mutation) an increase in performance for each mutation type in terms of all three criteria, namely, first hit time, best fitness, and success rate, with the increase in crossover probability. As a result, since the best performance is achieved when the crossover probability is 1.0. we can say that, the most proper selection for crossover probability is 1.0.

For the next parameter, i.e., the mutation probability, we tested 3 different mutation probabilities, $0.5/l$, $1/l$, and $2/l$, where l is the number of lightpaths in the VT. The results, given in Table 5.8 show that, the most proper mutation probability is $1/l$.

The third parameter that we fine-tuned is the population size. We tested 6 different population sizes, namely, 5, 10, 20, 50, 100, and 200. The results for this experimentation can be seen in Table 5.9. The results show that the first hit time increases with the increase in population size, which would lead to a result that

selecting a small size for the population, i.e., 5, is a good choice. On the other hand, the success rate increases with the increase in population size, which may be interpreted as 200 would be the best choice as the population size. If we consider the resource usages, we can see that the performance increases with the increase of the population size up until it is 50. As a result, selecting the population size as 50 is a good choice.

The last test for parameter setting is testing when to stop evolving, i.e., the maximum number of fitness calculations allowed until stopping the EA. The program is run for 5 minutes, during which the best so far fitness value is recorded every 5 seconds. The change in the quality of the best solution can be seen in Figure 5.4. From this figure, we can see that there is not much change in solution quality after 2 minutes, which corresponds to approximately 5000 fitness evaluations for this network.

As a result of parameter setting tests, we decided on a crossover probability of 1.0, a mutation probability of $1/l$, where l is the number of lightpaths, a population size of 50, and a maximum fitness evaluation count of 5000, for the EA performance tests.

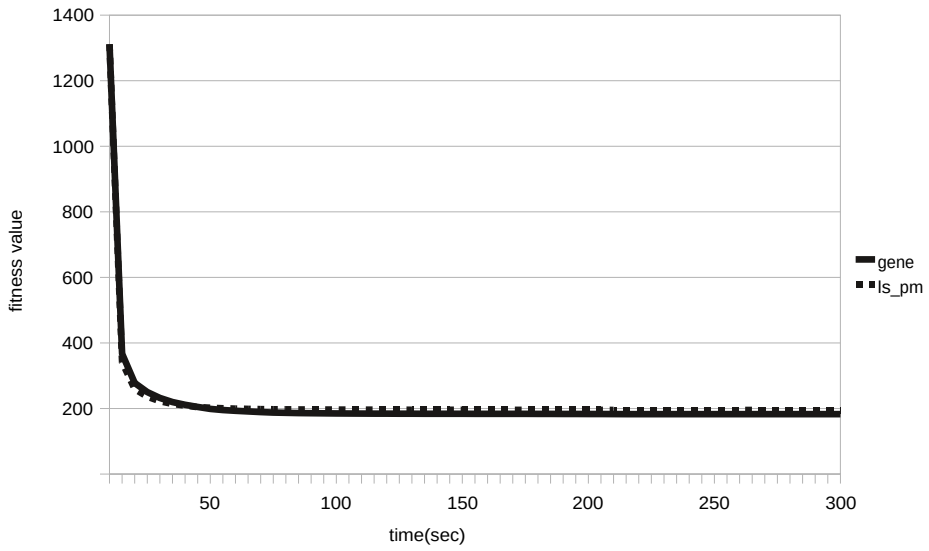

Figure 5.4: Change of best fitness over time (crossover probability=1.0, mutation probability=$1/l$, population size=50).

Table 5.7: Effect of crossover probability on first hit time, best fitness and success rate for three mutation types.

		first hit time				best fitness				success rate			
		0.7	0.8	0.9	1.0	0.7	0.8	0.9	1.0	0.7	0.8	0.9	1.0
mutation type	*gene*	4884	4879	4870	4861	199.5	197.25	195.78	193.8	99	99	100	99
	ls_pm	4815	4803	4787	4733	202.55	200.79	199.29	197.97	100	100	100	100
	ms_pm	4078	4014	3949	3877	236.9	232.03	226.71	220.96	84	91	95	97

Table 5.8: Effect of mutation probability on first hit time, best fitness and success rate for three mutation types.

		first hit time			best fitness			success rate		
		$0.5/l$	$1/l$	$2/l$	$0.5/l$	$1/l$	$2/l$	$0.5/l$	$1/l$	$2/l$
mutation type	*gene*	4861	4861	4854	193.86	193.8	198.83	99	99	100
	ls_pm	4682	4733	4806	198.45	197.97	200.81	100	100	100
	ms_pm	3664	3877	4452	220.5	220.96	223.22	96	97	97

Table 5.9: Effect of population size on first hit time, resource usage and success rate for three mutation types.

		first hit time						resource usage						success rate					
		5	10	20	50	100	200	5	10	20	50	100	200	5	10	20	50	100	200
mutation type	*gene*	2290	2863	3704	4624	4861	4872	185	185	185	187	194	212	93	96	97	99	99	100
	ls_pm	1477	1992	2779	4163	4733	4859	202	201	199	198	198	211	97	97	99	100	100	100
	ms_pm	646	966	1430	2496	3877	4869	259.5	256	250.5	236.7	221	216.6	6	17	41	82	97	100

5.3 Evolutionary Algorithms Performance Tests

The third set of tests include evaluating the performance of EA using different sized physical topologies. For these tests, we use the parameter setting determined based on the test results given in Chapter 5.2.1.

Two different networks are used to evaluate the EA performance. Since scalability is an important issue in network problems, we use a 24-node 43-link network (Figure 5.2) and a 48-node 89-link network in the tests. To generate a 48-node 89-link network, we combined two of the same 24-node 43-link topologies in Figure 5.2 at the nodes 1, 9 and 24 (see Figure 5.3). The test results are given in the next two sections.

5.3.1 Evolutionary algorithms performance tests with the 24 node network

We performed tests for two different population sizes, namely 50 and 100. We ran the program 20 times for each parameter set. A penalty factor of 200 is used in the tests. The detailed test suit is given in Table 5.10 for EA performance tests. We eliminated f_3 and *ms_pm* since they performed very poorly for the problem as seen in Chapter 5.2.1.

In the first part of the performance tests, we used the same network topology given in Figure 5.2, that is used in parameter setting tests, and 100 randomly created 3, 4, 5 connected virtual topologies. The wavelength capacity is selected as 10.

Table 5.10: Test suit for EA performance tests with 24 node network.

Number of virtual topologies	100 (3 connected)
	100 (4 connected)
	100 (5 connected)
Fitness function types	2 (f_1 and f_2 in Chapter 4.1)
Mutation types	2 (*gene*, *ls_pm* in Chapter 4.1)
Population sizes	2 (50 and 100)
Shortest path numbers (k)	3 (5, 10, 15)
Number of runs	20
Total	300*2*2*2*3*20 = 144000 runs

5.3.1.1 Experimental results

The results of the experiments are given in Tables 5.11- 5.16. Tables 5.11 and 5.12 show the success rates, Tables 5.15 and 5.16 show the average of best fitnesses found, and Tables 5.13 and 5.14 show the average of first hit times. In all the tables, f_i denotes the corresponding results for the i^{th} fitness evaluation method. In the first part of the tables, the results obtained using the *gene* mutation are given, whereas in the second part, the *ls_pm* results are given. Also, the results for three connectivity degrees, 3,4, and 5, can be seen in the tables.

Table 5.11: Success rates for the 24-node network - population size=50.

		5 shortest paths		10 shortest paths		15 shortest paths	
		f_1	f_2	f_1	f_2	f_1	f_2
gene	3	97	49	98	49	97	48
	4	99	92	100	94	100	94
	5	100	99	100	99	99	99
ls_pm	3	96	53	96	55	95	54
	4	100	95	100	96	100	96
	5	100	99	100	99	100	99

Table 5.12: Success rates for 24-node network - population size=100.

		5 shortest paths		10 shortest paths		15 shortest paths	
		f_1	f_2	f_1	f_2	f_1	f_2
gene	3	98	59	99	61	99	60
	4	100	97	100	98	100	98
	5	100	100	100	100	100	100
ls_pm	3	98	65	98	67	98	68
	4	100	98	100	98	100	99
	5	100	100	100	100	100	100

Comparing Tables 5.11 and 5.12, we can see that, using 100 as the population size results in an increase in success rates. Both tables show that the probability to find a feasible solution increases with the increase in connectivity degree of the VT. Although there is not much difference between *gene* and *ls_pm* type mutations, we can say that

if we use f_2, *ls_pm* performs better for 3 and 4 connected VTs. On the other hand, f_1 performs well with always a success rate greater than 95%, for each test set. If we consider f_1, *gene* mutation performs better than *ls_pm*. Also, *gene* mutation should be preferred since it is much less complex than *ls_pm*, in which a similarity comparison is needed every time mutation occurs.

Table 5.13: Average of EA first hit times - population size=50.

		5 shortest paths		10 shortest paths		15 shortest paths	
		f_1	f_2	f_1	f_2	f_1	f_2
gene	3	1855	1670	3049	2803	3856	3606
	4	1945	1944	3428	3419	4187	4209
	5	2358	2401	4067	4110	4612	4646
ls_pm	3	1890	1632	2790	2511	3302	3044
	4	1886	1902	2905	2894	3511	3527
	5	2195	2236	3376	3424	4088	4120

Table 5.14: Average of EA first hit times - population size=100.

		5 shortest paths		10 shortest paths		15 shortest paths	
		f_1	f_2	f_1	f_2	f_1	f_2
gene	3	2750	2385	4119	3838	4635	4457
	4	2718	2772	4362	4375	4737	4749
	5	3262	3339	4707	4729	4856	4869
ls_pm	3	2749	2386	3851	3555	4315	4102
	4	2638	2649	3910	3929	4400	4423
	5	3017	3124	4322	4373	4705	4732

Tables 5.13 and 5.14 show that, first hit times increase with the increase in the connectivity degree of VT. This is an expected result, since the search space is larger for larger connected VTs. We can see from both of the tables, first hit times for f_1 are almost always smaller than f_2 for 4 and 5 connected VTs. If we compare *gene* and *ls_pm*, we can say that first hit times of *ls_pm* are better. The comparison of two tables, i.e., the effect of population size, suggests that if the population size is smaller, the first hit time will be earlier. This may be a result of getting stuck to a local minimum.

Table 5.15: Average of EA resource usages - population size=50.

		5 shortest paths		10 shortest paths		15 shortest paths	
		f_1	f_2	f_1	f_2	f_1	f_2
gene	3	111	111	112	112	113	112
	4	144	144	145	145	146	146
	5	182	182	183	183	186	186
ls_pm	3	114	112	118	117	121	118
	4	148	147	153	153	156	155
	5	186	186	193	194	196	197

Table 5.16: Average of EA resource usages - population size=100.

		5 shortest paths		10 shortest paths		15 shortest paths	
		f_1	f_2	f_1	f_2	f_1	f_2
gene	3	111	110	112	111	114	113
	4	144	144	145	145	148	149
	5	182	182	185	185	192	193
ls_pm	3	112	111	116	115	119	117
	4	146	146	151	151	154	154
	5	184	184	191	191	196	197

The resource usages given in Tables 5.15 and 5.16, confirm the previous conclusion, that is the smaller population size gets stuck at a local minimum. If we look at the tables, we can say that, *gene* mutation always gives the best performance. The resource usage increases if we increase the number of shortest paths used for the tests. This is because of the increase in probability of getting stuck at a local minimum while the search space gets larger. From this table, we can say that we do not need to use more than 5 shortest paths in order to find a good solution. If we compare the performance of fitness evaluation methods, we can say that f_1 and f_2 almost always perform similarly. For 3 connected VTs, f_2 is slightly better than f_1. However, f_1 is recommended, since its success rate is better.

As a result of the experiments with 24-node 43-link topology, using *gene* mutation and f_1 can be recommended as components for an effective EA for the survivable VT mapping problem.

5.3.2 Evolutionary algorithms performance tests with the 48 node network

In order to see the performance of our EA design in larger sized networks, we formed a new topology by merging two previously used 24-node 43-link topologies. We combined two of this same topology from the nodes 1, 9 and 24. As a result we obtained a 48-node 89-link topology 5.3. Again, we created virtual topologies of different average connectivity degree. The time needed to create 100 different VTs for 48 nodes was very long, and was increasing with the increase in the connectivity degree, so that, we could create 100 different VTs for only 3 and 4 connectivity degrees, and 50 different VTs for the 5 connectivity degree.

In the tests, we considered a maximum fitness evaluation count of 5000, mutation probability of $1/l$, where l is the number of lightpaths, and crossover probability of 1.0 for the EA performance tests. We performed tests only for a population size of 50, since the size of this problem is much more larger and the first hit time results in Chapter 5.3.1 show that the increase in population size also increases the first hit time, with an acceptable amount of decrease in success rates. We ran the program 20 times for each parameter set. A penalty factor of 200 is used in the tests.

Since we decided that the most effective fitness is f_1 as explained in the previous section, we used f_1 in the tests. Again, we tested both mutation operators.

For this larger network, we increased the wavelength capacity. In order to find the suitable wavelength capacity, we tested 16 and 32 wavelength capacities. In the first set of tests that we used 16 as the wavelength capacity, we used 10 and 15 shortest paths. As a result of these tests, given in Table 5.17, we saw that the wavelength capacity of 16 is not enough for this size of network. From the table, we can see that the success rates are decreasing with the increase in connectivity degree, which should be just the opposite according to our previous experiences. The results show that the capacity constraints are almost always violated for 5 connected VTs.

Table 5.17: Success rates for EA in 48-node topology - 16 wavelength capacity.

		10 shortest paths	15 shortest paths
gene	3	52	66
	4	32	34
	5	2	2
ls_pm	3	54	72
	4	26	28
	5	2	2

As we realized that the wavelength capacity of 16 is not enough for this network, we increased the capacity to 32. This time, we tested 10, 15, and 20 shortest paths. The detailed test suit is given in Table 5.18 for EA performance tests for the 48-node network with a wavelength capacity of 32.

Table 5.18: Test suit for EA performance tests with 48 node network.

Number of virtual topologies	100 (3 connected)
	100 (4 connected)
	50 (5 connected)
Mutation types	2 (*gene*, *ls_pm* in Chapter 4.1)
Shortest path numbers (*k*)	3 (10, 15, 20)
Number of runs	20
Total	250*2*3*20 = 30000 runs

5.3.2.1 Experimental results

In Tables 5.19, 5.20, and 5.21, the success rate, the average resource usage, and the average first hit time results for the 48-node 89-link network topology with 32 wavelength capacity are given, respectively.

From Table 5.19, we can see that with each mutation type and shortest path pair a success rate of 100% is achieved for 5 connected VTs. For 3 and 4 connected VTs the success rates increase with the increase in the number of shortest paths. However, as can be seen from Table 5.20, the resource usage also increases with the number of

Table 5.19: Success rates for EA in 48-node topology - 32 wavelength capacity.

		10 shortest paths	15 shortest paths	20 shortest paths
gene	3	52	67	78
	4	82	86	89
	5	100	100	100
ls_pm	3	55	71	81
	4	70	74	89
	5	100	100	100

shortest paths. This is because of the search space getting larger and the probability of getting stuck at a local minimum is increasing.

Table 5.20: Average resource usages for EA in 48-node topology - 32 wavelength capacity.

		10 shortest paths	15 shortest paths	20 shortest paths
gene	3	311	314	317
	4	416	419	424
	5	519	525	532
ls_pm	3	330	337	342
	4	439	446	453
	5	547	557	565

If we compare two mutation types, we can say that using *gene* mutation results in better quality solutions, and on the other hand using *ls_pm* results in higher number of feasible solutions for the VTs with connectivity degree 3. However, for 4 connected VTs, *gene* mutation performs better both in means of success rate and resource usage. The first hit time results in Table 5.21 justifies the previous outcome, that is the *ls_pm* gets stuck at a local minimum.

5.4 Performance Comparison of Evolutionary Algorithms, Ant Colony Optimization and ILP

To evaluate the efficiency of the EA, we designed another NIH solution to the survivable VT mapping problem. We chose ACO due to its successful performance on

Table 5.21: Average first hit times for EA in 48-node topology - 32 wavelength capacity.

		10 shortest paths	15 shortest paths	20 shortest paths
gene	3	5295	6562	7207
	4	5889	7088	7523
	5	6628	7461	7694
ls_pm	3	4457	5557	6172
	4	4717	5811	6360
	5	5013	6081	6737

constrained combinatorial optimization problems [5]. The details of our ACO design can be found in Chapter 4.2.

To determine the best ACO variant, we experimented with 6 direct variants of the AS algorithm listed in [74]. For each connectivity degree of the VT, success rates of each variant are mostly greater than 90% when more than 5 shortest paths are provided to the algorithms. Since the algorithms have relatively equal performances, we applied ANOVA tests to the results. We ended up with the result that, ACS and AS are the worst and EAS is the best of all.

The first parameter setting test is to investigate the effect of the maximum allowed time on resource usage. For this test, the ACO algorithm is run for 60 seconds and the best-so-far solution is recorded every 5 seconds. The results show that the algorithm needs to run at most 15 seconds, because only very small contribution is provided after around the 15^{th} second.

To determine the number of ants used in ACO, we tested six different values, i.e., 1, 5, 10, 20, 50, and 100. According to the results, increasing the number of ants increases resource usage after 5 ants. Until a maximum number of solutions are generated, ants use pheromone matrices to construct solutions. Pheromone matrices are updated according to the solution quality after each ant constructs its solution. When the number of ants decrease, the use of pheromone increases. As a result, we set the number of ants as 10. ρ_0 is the parameter used while updating pheromone values. This parameter is determined after experimenting with ρ_0 values of 0, 0.1, 0.2, 0.3, 0.4, 0.5,

0.6, 0.7, 0.8, 0.9, and 1.0. As a result, we selected ρ_0 as 0.1, where the best solutions are retrieved. The parameters α and β represent the weights of the pheromone level and the heuristic information, respectively. We tested all combinations of four different values of α: 0, 1, 2, and 3, and five different values of β: 0, 1, 2, 3, and 4. The contribution of both parameters on resource usage are negligible for values greater than 2. α does not seem to have much effect on resource usage. The parameters are set as $\alpha = 3$ and $\beta = 4$, when the best solutions are retrieved. e is the weight of pheromone deposited for the best-so-far solution in EAS algorithm. For the fine-tuning of e, we tested with e values of 1, 2, 3, 4, 5, 10, 20, 50, and 100. According to the results, there is not much difference in resource usage when e is selected between 1 and 4. We chose e as 3, since it is a mid value.

The ILP formulation for the survivable VT mapping problem is given in Chapter 2.3. Due to the large number of constraints, solving the ILP for large networks can be very difficult. Hence, Modiano et al. [2] proposed two possible relaxations of the ILP formulation that yield survivable routings with reduced complexity. The first relaxation proposed is a simple relaxation that applies the survivability constraints only to cuts that include a single node, which prevents a single node from getting disconnected in the event of a fiber cut. With this relaxation, the number of survivability constraint equations is reduced to number of nodes. We implemented the basic and the relaxed ILP using the CPLEX suite [77].

It is reported in [2] that relaxations also perform well for this problem with a much more smaller run time. Since our problem size is much larger (24-node 43-link physical topology) than the one used in [2] (14-node 21-link physical topology), we decided first to compare our results with their first relaxation. Different from [2] we also included the capacity constraint in the ILP formulation.

We implemented this ILP relaxation using the CPLEX software package. CPLEX uses branch and bound techniques for solving ILPs and is capable of solving ILPs consisting of up to one million variables and constraints [77]. The results obtained are given in Tables 5.22 and 5.23.

Table 5.22: Number of feasible solutions obtained using EA, ACO, ILP-Relaxation, and Basic ILP.

	EA	ACO	ILP-Relaxation	Basic ILP
3 connected	100	100	52	58
4 connected	100	100	88	0
5 connected	100	100	100	0

5.4.1 Experimental results

For the comparison of basic and relaxed ILP solutions and our NIHs, the best EA combination, i.e., the gene mutation with f_1, and elitist AS for ACO with the most proper parameter set, are used. The results obtained are given in Table 5.22.

It is reported in [2] that relaxations also perform well for this problem with much smaller run times. For each VT, we also solved the problem using this ILP-Relaxation. Different from [2] we also included the capacity constraint in the ILP formulation.

From the tables we can see that, both of our NIHs perform well for the survivable VT mapping problem. From Table 5.22 we can see that, they both perform better than this ILP relaxation for all problem sets, especially for the case with 3 connected VTs.

For 3-connected VTs, we could solve the problem using the basic ILP for only 58% of the VTs. For the remaining 42%, we were out of memory although the physical memory of the machine was 3 Gb. Furthermore, if we increase the connectivity degree of the VT, solving the problem using the basic ILP was impossible. The time needed to create the CPLEX input file for the basic ILP solution was at least 2 hours and went up to more than 15 hours on a Pentium IV 1.6 GHz. Besides this long time to create the CPLEX input file, ILOG CPLEX solves the problem in at least 5 minutes, and this solution time goes up to more than 15 minutes. Moreover, the total size of the CPLEX input files for these 100 3-connected VTs is more than 30 Gb.

If we compare EA and ACO best solutions to these 58 basic ILP solutions, i.e., the optimum, 56 of the EA and 57 of the ACO solutions are the optimum. Since the basic ILP can not find a solution with less resource usage than the relaxation used here, we can say that the lower bound for the resource usage is the one found with the ILP

Table 5.23: Number of solutions (out of 100) having a resource usage equal to the feasible ILP-Relaxation solutions.

	EA	ACO
4 connected	97	98
5 connected	100	100

relaxation. Since we do not have any basic ILP solutions of 4 and 5 connected VTs, we compare EA and ACO solution qualities with the feasible ILP relaxation results. From Table 5.23 we can see that EA and ACO find the optimum solution for almost all test cases, with ACO performing slightly better.

The time needed to create the CPLEX input file for the basic ILP solution was 2 hours and for some cases it went up to more than 15 hours on a Pentium IV 1.6 GHz. ILOG CPLEX solved the problem in 5 minutes, and this solution time went up to more than 15 minutes for some cases. Moreover, the total size of CPLEX input files for the 100, 3 connected VTs is more than 30 Gbs. On the other hand, for this data set, EA does not need more than a minute to solve the problem and ACO can solve it in 10 seconds, on average. As a result, EA and ACO can find the optimum solutions for 3 connected VTs within 120 and 720 times less running times, respectively.

Table 5.24: Running Time Averages (in seconds) of EA and ACO for 24-node and 48-node Physical Topologies and VTs Having 3 Different Connectivity Degrees (3,4,5)

	24-node network		48-node network	
	EA	ACO	EA	ACO
3	30	15	300	350
4	50	15	400	350
5	60	15	480	350

The running times of EA and ACO for different sizes of networks and different sizes of connectivity degrees are given in Table 5.24. The running times in the table are the averages of all runs for all the VTs with corresponding connectivity degree. Since the termination condition for ACO is a predefined time, ACO has the same running time for all test cases of the same physical topology. From the table, it can be seen that the

increase in running time with the increase in network size is not as much as it is for the CPLEX.

As a conclusion, the results show that both heuristics are promising for the survivable VT mapping problem. Since the time needed to find a feasible solution is at most 8 minutes (for the routing of 5 connected VTs on the 48-node 89-link topology using EA), these heuristics can easily be applied to real world applications.

6. SURVIVABLE VT DESIGN USING HYPER-HEURISTICS

6.1 Overview in Hyper-Heuristics

The term hyper-heuristic (HH) [78, 79] was first used in 2000 to describe "heuristics to choose heuristics" in the combinatorial optimization context. HHs do not operate directly on the solutions, rather they operate on heuristics that operate on candidate solutions.

Classical optimization techniques are impractical in solving complex real-world problems, whereas meta-heuristics [80] provide better means by intelligently seeking optimal solutions within a search space [81]. To use meta-heuristics one needs to have expert level knowledge and experience on the problem. Furthermore, fine-tuning [80] in meta-heuristics is a time-consuming task which may sometimes require more time than the development time. However, HHs are general search methods that can be applied easily to a very wide range of complex real-world problems, where a solution is required in an acceptable amount of time [78].

For a problem in hand, many different simple greedy heuristics can be defined. However, each of these heuristics has its own weaknesses and strengths. The main purpose of HHs is to combine the best features of these heuristics. In HHs terminology, these heuristics are referred to as low-level heuristics (LLH), and a high-level heuristic is used to select between these LLHs at each step of an optimization process [80]. This way, the best features of different simple greedy heuristics which are used as LLHs, can be combined.

A HH is a high-level heuristic which receives as input the problem to be solved and a set of LLHs defined for the problem. At each step of the solution process, one of the LLHs is selected and applied to the problem, until a stopping condition has been met. The general framework of HHs is illustrated in Figure 6.1.

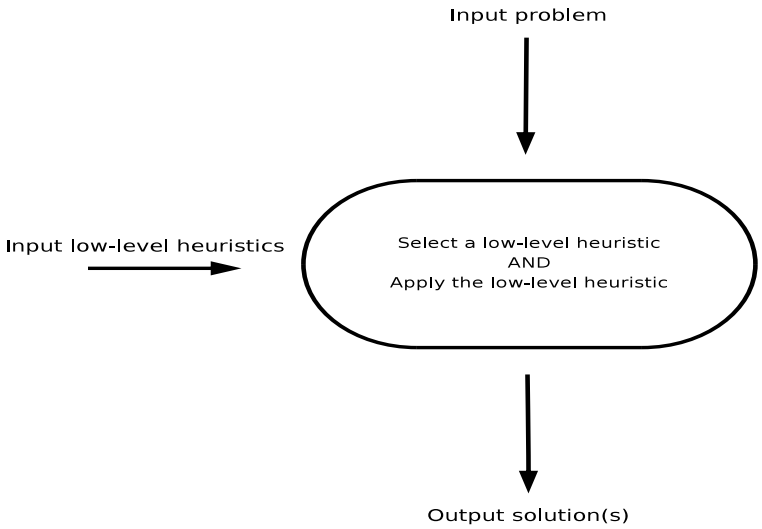

Figure 6.1: The general framework of a HH.

Unlike the metaheuristics, which are problem-specific methods, the HHs have no knowledge on the problem domain. The HHs only know whether the objective function is to be maximized or minimized. The main motivation behind HHs is to design problem-independent algorithms. To solve a problem, HHs communicate with the problem-specific LLHs. Thus, a HH can be applied to a wide range of problems by changing only the LLHs.

There are different classifications of hyper-heuristics. The first classification was (1) with learning, and (2) without learning [82]. The hyper-heuristics with learning include learning mechanisms based on the historical performance of the LLHs and change the preference of each LLH. On the other hand, hyper-heuristics without learning select the LLHs to call according to a predetermined sequence.

In 2005, Bai [83] classified HHs into two groups: constructive HHs and local search HHs. Constructive HHs start from an empty solution and build it gradually by selecting a LLH at each step of the construction process. On the other hand, local search HHs start from a complete initial solution and iteratively select LLHs to improve the solution quality. Based on the classification of Bai, in [11], Burke et al. used the terms constructive and perturbative to refer to these classes, respectively.

In [84], hyper-heuristics are classified into four categories according to the selection mechanism of LLHs: (1) random choice hyper-heuristics, (2) greedy and peckish

hyper-heuristics, (3) meta-heuristics based hyper-heuristics, and (4) hyper-heuristics employing learning mechanisms.

In hyper-heuristics with learning mechanism, the selection of heuristics can be in different ways: random gradient [78], random permutation gradient [78], choice function [78], reinforcement learning [85], and reinforcement learning with tabu search [86]. If no learning mechanism is applied in the hyper-heuristic, the heuristic selection methods are simple random [78], random permutation [78], greedy [78], and peckish [87].

HHs have been applied to many combinatorial optimization problems, such as, timetabling [88–90], bin packing [91], scheduling [92, 93], stock cutting [94], constraint satisfaction [95], vehicle routing [85], channel assignment [96], planning [97], and space allocation [98].

6.2 The Proposed Hyper-Heuristics Approach

To solve the survivable VT design problem, we use four HH approaches, each of which is based on a different type of nature inspired heuristic (NIH), used as the heuristic selection method. These NIHs are: evolutionary algorithms (EA), ant colony optimization (ACO), adaptive iterated constructive search (AICS), and simulated annealing (SA). Each method belongs to a different category of search approaches:

1. **EA:** population based, perturbative search
2. **ACO:** population based, constructive search
3. **SA:** single point perturbative search
4. **AICS:** single point, constructive search

Given the traffic matrix, the first step is to *determine a suitable VT*. For this subproblem, we selected five commonly used VT design heuristics as LLHs. These LLHs are:

1. Choose the nodes which have the maximum single direction traffic demand between them (**MAX_SNG**).

2. Choose the nodes which have the minimum single direction traffic demand between them (**MIN_SNG**).

3. Choose the nodes which have the maximum bidirectional total traffic demand between them (**MAX_TOT**).

4. Choose the nodes which have the minimum bidirectional total traffic demand between them (**MIN_TOT**).

5. Choose a random node pair (**RND**).

At each step of the solution construction, a LLH is used to choose the next set of node pairs to establish a lightpath in between. The lightpaths are established, such that, the in and out degrees of the nodes do not exceed the maximum number of transceivers on each node.

The *VT routing and wavelength assignment (mapping) subproblem* is solved as the first step in this thesis (see Chapter 5), with ACO which provides promising solutions in a relatively small amount of time. Therefore, we choose ACO to solve this subproblem.

The *traffic routing* is applied in a straightforward way. The shortest path routing method is used for routing the packet traffic.

In HHs, the quality of the solutions is determined through a fitness function. In this study, the fitness of a solution is measured as the total number of wavelength-links used throughout the network, which is referred to as resource usage. The objective of the survivable VT design problem is to minimize this resource usage while considering the survivability and the capacity constraints. Resource usage is calculated by counting the number of physical links that are used by the lightpaths. An infeasible solution can either be penalized by adding a value to the fitness function, or can be discarded. In our HH algorithms, if a solution is found to be infeasible during the phases of creating a VT and routing the lightpaths, it is discarded. In the next stage, if the traffic cannot be routed over the VT, a penalty value proportional to the amount of traffic that cannot be routed, is added to the fitness of the solution.

A solution candidate is represented as an array of integer values showing the order of LLHs to select lightpaths for the VT to be established. Since there

are 5 different LLHs, the integers can have values between 1 and 5. When a lightpath between two selected nodes is established, the lightpath is assumed to be bidirectional. Therefore, a transceiver at both ends is used. The length of the solution array is equal to the maximum number of lightpaths that can be established, i.e., $number\ of\ transceivers\ on\ each\ node * number\ of\ nodes/2$. For example, in a network with 6 nodes and 3 transceivers per node, each solution candidate is of length 6*3/2=9. If a solution candidate is represented with an array of [2 1 1 3 5 3 4 2 2], this means that, first a lightpath will be selected using the second LLH, then the next two using the first, continuing with the third, fifth, ... LLHs. While adding the lightpaths, the transceiver capacity constraint is handled. If the lightpath added according to the corresponding LLH results in using more than the existing number of transceivers in one or both ends, this lightpath is not added to the VT and the algorithm continues with the next LLH in the solution array. The lightpath capacity is assumed to be 40 Gb/s. After adding each lightpath to the VT, the traffic demand between the nodes of the corresponding lightpath is assumed to be met, and the traffic demand between these nodes is deleted from the traffic matrix. The algorithm continues to establish lightpaths until either the end of the solution array is reached or until no traffic remains in the traffic matrix.

For each solution candidate produced by the HH, the corresponding VT is determined using the method explained above. Then, if the generated VT is not at least 2-connected (at least 2 link-disjoint paths for packet traffic exist between each node pair), new lightpaths are added subject to the transceiver capacity until the VT becomes 2-connected. For the nodes that have a degree lower than two, a new lightpath is added between this node and the node with the highest traffic demand in between. Next, the best mapping for the VT is found using ACO [75]. Then, the packet traffic is routed through the shortest paths starting from the node pair with the largest traffic demand. Finally, the fitness is calculated as the total amount of resource usage, i.e., the number of wavelength-links used throughout the network. The general flow of the algorithm is given in Algorithm 11.

Algorithm 11 General flow for the survivable VT design algorithm

```
1: function DesignVT()
2:     use HHs to generate a VT
3:     use ACO to find a survivable mapping of lightpaths on the physical topology
4:     use shortest path heuristic to route traffic on the virtual topology
5:     calculate fitness of the solution candidate
6: end function
```

In the following section, we summarize each of the HH approaches we propose in this study, to solve the survivable VT design problem.

6.3 Evolutionary Algorithms as a Hyper-Heuristic

We use a steady-state EA (see Chapter 3) with duplicate elimination. After generating an initial set of random solution candidates, the EA operators, i.e., tournament selection, uniform crossover, and gene mutation [4], are applied and new solution candidates are generated. Gene mutation is defined as changing a LLH in the selected point of the string with another randomly determined LLH. Initial population of solution candidates (individuals) is generated randomly. After the fitness evaluation of each individual in the initial population, genetic operators are applied. The pseudocode for the whole process is given in Algorithm 12.

Algorithm 12 Pseudocode for the EA-based HH

```
FUNCTION EA()
generate random initial population
for each individual in the population do
   DesignVT()
end for
while not predefined # of individuals are created do
   select parents to mate
   recombine selected parents
   apply mutation to the offspring
   DesignVT()
   if offspring better than worst
      offspring replaces worst in population
   else discard offspring
end while
END FUNCTION
```

6.4 Ant Colony Optimization and Adaptive Iterated Constructive Search as Hyper-Heuristics

ACO and AICS are very similar in the way they solve the problem. We can say that AICS is a form of ACO which uses only one ant. We use the elitist ant system (EAS) as the ACO variation, based on the results of our previous study [75], which show that this variation performs better than the others on the survivable VT mapping subproblem. However, there is only one ant in AICS, hence, the ant system is applied as the ACO variation.

Algorithm 13 Pseudocode for the ACO-based and AICS-based HHs

```
FUNCTION ACO()
for each ant k do
  repeat
    select a random LLH
  until a complete solution is generated
  DesignVT()
end for
initialize pheromone levels
while not predefined # of individuals are created do
  for each ant k do
    repeat
      select a LLH based on decision policy
    until a complete solution is generated
    DesignVT()
  end for
  evaporate pheromone levels
  deposit new pheromones on the arcs the ants visited
  deposit pheromone on the arc the best-so-far ant visited (for ACO-based HH)
end while
END FUNCTION
```

Initially, each ant iteratively adds a random LLH to its partial solution. The solution construction terminates when a solution array with a length equal to the maximum number of lightpaths is generated. No constraint is applied to the solution in the construction phase. Since there is no heuristic information, the solution construction only depends on the pheromone trail. The pheromone trails τ_{ij} we use in this paper refer to the desirability of using the j^{th} LLH to add the i^{th} lightpath. Pheromone trails are initialized using the initial random solutions of the ants. Then, they are modified

each time that all ants have constructed a solution. The pseudocode for the whole process is given in Algorithm 13.

6.5 Simulated Annealing as a Hyper-Heuristic

Simulated annealing algorithm emulates the physical process in which the metal cools and freezes into a minimum energy crystalline structure (the annealing process). A random search is applied in the algorithm, if the problem is a minimization problem, the neighbor solutions with a lower fitness value are always accepted, and the neighbor solutions with a higher fitness value are accepted with some probability. This probability is defined as in the following formula:

$$p = e^{(\frac{-\Delta f}{T})} \quad \textbf{(6.1)}$$

In the formula, Δf is the increase in fitness value and T is a control parameter, which is known as the current system temperature.

Algorithm 14 Pseudocode for the SA-based HH

```
FUNCTION SA()
initialize temperature
generate a random initial solution
DesignVT()
while not predefined # of individuals are created do
  repeat
    select a neighbor solution using mutation
    DesignVT()
    if neighbor solution quality is better than current solution then
      go to the neighbor solution
    else
      go to the neighbor solution with some probability (Eq.6.1)
    end if
  until 5 solutions are generated
  decrease temperature by a constant factor
  decrease mutation probability by a constant factor
end while
END FUNCTION
```

In SA, the main elements of the algorithm are, the solution representation, the neighborhood operator, the fitness function, and the annealing schedule (initial temperature and rules for lowering it as the search progresses).

In this thesis, we use a non-standard SA where the neighborhood operator is modified over time. The neighborhood operator is defined similar to the mutation operator in the EA-based HH, where with a given mutation probability, a randomly chosen LLH on the solution candidate is replaced by another LLH. The difference is that, we define a larger mutation probability in the beginning of the SA. The mutation probability is decreased by a predefined factor each time after 5 solution candidates are generated. This allows us to have a high exploration rate in the beginning of the search while focusing on exploitation towards the end. The general flow of the algorithm is given in Algorithm 14. In our study, we use the formula given in [99] to calculate the initial temperature.

7. EXPERIMENTAL STUDY FOR SURVIVABLE VT DESIGN

To evaluate the performance of our newly designed solution approaches for the survivable virtual topology (VT) design problem, a 24-node 43-link network (see Figure 5.2) and 20 different randomly generated traffic matrices are used.

The experimental study in this thesis has proceeded in three phases:

- Design an efficient hyper-heuristic (HH) for the survivable VT design problem
- Evaluate the HH performance for double-link failures
- Compare the HH performance to another solution approach in the literature proposed for survivable VT design problem (tabu search proposed in [13])

The experimental results for each phase are given in the following sections.

7.1 Performance Evaluation of Hyper-Heuristics for Survivable Virtual Topology Design Problem

We present the experimental results for a 24-node 43-link telco network given in Figure 5.2, which is a fairly large-sized network for this problem. For the experiments, we use 20 different traffic matrices, randomly generated according to a frequently-used traffic generation method [1], where, 70% of the traffic is uniformly distributed over the range [0, 0.5 Gb/s] and 30% of the traffic is uniformly distributed over the range [0, 5 Gb/s]. The lightpath channel capacity is chosen as 40 Gb/s, which is typical in real-world networks.

We use five well-known VT design heuristics as LLH:

1. Choose the nodes which have the maximum single direction traffic demand between them (*MAX_SNG*).

2. Choose the nodes which have the minimum single direction traffic demand between them (*MIN_SNG*).

3. Choose the nodes which have the maximum bidirectional total traffic demand between them (*MAX_TOT*).

4. Choose the nodes which have the minimum bidirectional total traffic demand between them (*MIN_TOT*).

5. Choose a random node pair (*RND*).

First, we performed tests to see the performance of each LLH separately on the problem. For this test, only one LLH is used to generate the complete solution. Thus, there is no stochasticity and tests are run once for each LLH and traffic matrix pair.

To see the performance of LLHs with traffic demands showing different intensities, we generated 20 more traffic matrices for this test, giving a total of 40 different randomly generated traffic matrices. These 20 new traffic matrices have 25% less intensive traffic. Table 7.1 shows the number of traffic matrices for which the corresponding LLH obtains the best result. The results show that different heuristics are successful on different traffic matrices, while, the first and third heuristics perform better than the others. However, there are some cases in which these heuristics cannot find a feasible solution while another heuristic can. For two traffic matrices two heuristics found the same best solution, therefore, in the table the total number of the best results are 42.

Table 7.1: The number of problem instances that each single low-level heuristic outperforms others.

single LLH	LLH_{MAX_SNG}	LLH_{MIN_SNG}	LLH_{MAX_TOT}	LLH_{MIN_TOT}	LLH_{RND}
# of best results (out of 40)	21	6	11	3	1

For the rest of experimental study, we use the 20 traffic intensive matrices. We run the program 20 times for each approach and for each parameter set.

Next, for the second group of experiments, we perform tests to find a good parameter set for *EA*, using 5 LLHs. The population sizes of 5, 10, 15, 25, 50 and mutation

probabilities of $0.5/l$, $1/l$, and $2/l$, where l is the solution array length, are compared. The resource usages obtained using different population sizes and mutation probabilities are given in Table 7.2. From the results in the table, we can say that the *EA*-based HH performs similarly for each parameter value. Therefore, we applied a two-way ANOVA and a Tukey HSD post hoc test at a confidence level of 0.95. The results show that there is no significant difference between different parameters. As a result, we selected 10 as the population size and $1/l$ as the mutation probability. A typical value of 0.8 is selected as the crossover probability. The tests show that no significant improvement is obtained after a total of 100 individuals are created in the *EA*. Therefore, each run is terminated after creating 100 individuals.

Table 7.2: Effect of population size and mutation probability on resource usage.

	population size					**mutation probability**		
	5	**10**	**15**	**25**	**50**	$0.5/l$	$1/l$	$2/l$
resource usage	172	171	172	173	173	172	172	170

After the parameter tuning of *EA*, tests to explore the performance of single LLHs are conducted. In this set of experiments, we include a *Random* heuristic as a baseline for the performance comparisons. In *Random* heuristic, 100 random solution strings are generated, which are composed of randomly selected LLHs. For each traffic matrix, the solution candidate with the best fitness is selected as the solution. The tests are run 20 times for each traffic matrix. The results of the experiments are given in Table 7.3.

Table 7.3: Performance comparison of single LLHs, *EA*, and *Random* using 5 LLHs.

	LLH_{MAX_SNG}	LLH_{MIN_SNG}	LLH_{MAX_TOT}	LLH_{MIN_TOT}	LLH_{RND}	***EA***	***Random***
RU	171	190	175	225	222	171	176
SR	0.7	0.3	0.7	0.05	1.0	1.0	1.0

In Table 7.3, the performances of *EA* and *Random*, using all the 5 LLHs, are compared with the single LLHs, for the survivable VT design problem. In the table, the RU and SR stand for resource usage and success rate, respectively. The resource usage values are the average of the successful runs obtained for 20 different traffic matrices. Success rate is the percentage of feasible solutions found with these heuristics. The

table shows that the success rates for the random LLH (LLH_{RND}), *EA*, and *Random* are 100%, while, the *EA* achieves the best resource usage equal to that of the best LLH (LLH_{MAX_SNG}).

In Table 7.3, we see that the performances of LLH_{MIN_SNG} and LLH_{MIN_TOT} are not as good as the other LLHs, considering both resource usage and success rate. To see the performance change after omitting these LLHs, we conduct a set of experiments with *EA* and *Random*, using the remaining 3 LLHs, i.e., LLH_{MAX_SNG}, LLH_{MAX_TOT}, and LLH_{RND}. The success rate for both approaches are 100%, as it is in Table 7.3. The average resource usage values are given in Table 7.4. From the table, we see that the performances of both approaches increase when we use 3 LLHs. Therefore, the LLHs LLH_{MIN_SNG} and LLH_{MIN_TOT} are extracted from LLHs in the remaining experimental study.

Table 7.4: Performance comparison of *EA* and *Random*, using 5 LLHs and 3 LLHs.

	EA_{5LLH}	EA_{3LLH}	$Random_{5LLH}$	$Random_{3LLH}$
resource usage	171	163	176	167

Next, we perform tests to find good parameter settings for the other nature inspired approaches, i.e., *ACO*, *AICS*, and *SA*, used as heuristic selection methods in the HHs. We run the program 20 times for each parameter set.

In *SA* parameter setting tests, the initial temperature is calculated by using the following formula [100]:

$$\gamma = \frac{m_1 + m_2 \cdot e^{\frac{-\Delta f}{T^0}}}{m_1 + m_2}. \qquad \textbf{(7.1)}$$

In the formula, γ is the acceptance ratio, which is selected as 0.95, m_1 is the number of moves when the fitness cost is decreased, and m_2 is the number of moves when the cost increased relative to the previous step. Δf is the average increase for m_2 moves. After a full Markov chain of 1000 solutions is completed, the initial temperature T_0 is calculated using Equation **7.1**.

As a result of the tests, we use an initial temperature of 195. The termination condition is again the same, i.e., 100 solution generations. The cooling rate is selected as 0.85,

since this rate decreases the temperature to 5% of the initial temperature in 100 steps. The initial mutation probability is selected as $30/l$, where l is the solution array length, and is decreased with a factor of 0.85 in every 5 steps of solution generation. The large initial mutation probability is selected because of the large size of the solution array. If we start with a small mutation probability, the *SA* algorithm will search in a small neighborhood of the starting point, which may lead to getting stuck in local minima. The mutation rate is gradually decreased to avoid a random exploration in the search space. We decrease the mutation probability until it reaches a value of $1/l$.

Table 7.5: Resource usage results for different traffic matrices using different approaches.

	EA	***ACO***	***AICS***	***SA***	***Random***	LLH_{MAX_SNG}	LLH_{MAX_TOT}	LLH_{RND}
TM_1	**169**	177	181	180	176	186	200	222
TM_2	158	152	149	163	165	**148**	160	229
TM_3	163	**148**	149	168	166	155	*NA*	219
TM_4	160	**146**	152	160	162	172	172	226
TM_5	154	**149**	151	156	157	170	159	220
TM_6	172	**156**	**156**	173	170	183	188	212
TM_7	175	**169**	172	185	180	*NA*	**169**	203
TM_8	167	**159**	162	173	173	167	169	222
TM_9	157	157	**151**	166	163	*NA*	*NA*	235
TM_{10}	172	161	**159**	176	174	194	198	221
TM_{11}	156	**137**	145	165	161	*NA*	155	222
TM_{12}	158	146	**145**	164	164	149	*NA*	218
TM_{13}	157	158	158	156	**155**	182	*NA*	236
TM_{14}	150	**135**	**135**	156	154	152	170	229
TM_{15}	168	**148**	155	176	173	*NA*	161	236
TM_{16}	163	**145**	147	165	165	*NA*	*NA*	219
TM_{17}	178	**175**	182	177	182	195	188	215
TM_{18}	**152**	162	156	160	163	*NA*	*NA*	213
TM_{19}	154	**139**	140	156	157	178	180	223
TM_{20}	170	**163**	166	175	177	**163**	177	221
Average	163	154	156	168	167	171	175	222
SR	1.0	1.0	1.0	1.0	1.0	0.7	0.7	1.0

After the parameter tuning of each approach, tests to explore their performance are conducted. In these tests, 3 LLHs, i.e., LLH_{MAX_SNG}, LLH_{MAX_TOT}, and LLH_{RND}, are used. The tests are run 20 times for each traffic matrix. Each run uses a random initial seed. The results of the experiments are given in Table 7.5. To see the detailed results,

the table includes results for each traffic matrix, separately. In the table, SR stands for success rate. The TM_i values in the first column of the table indicate that, the results in the corresponding row are the results obtained using traffic matrix i. In the next four columns of Table 7.5, the results for the four HHs designed in this study are given. The fifth column lists the results of the *Random* heuristic. The last three columns contain the results of LLHs applied separately. In the table, the values in the first five columns are the averages of resource usages obtained after 20 runs for each traffic matrix using the corresponding method. The last three columns are the results of using only the corresponding single LLH in the solution. The *NA* values in the table mean that a feasible solution was not found in any of the runs. The corresponding box-whisker plot is given in Figure 7.1.

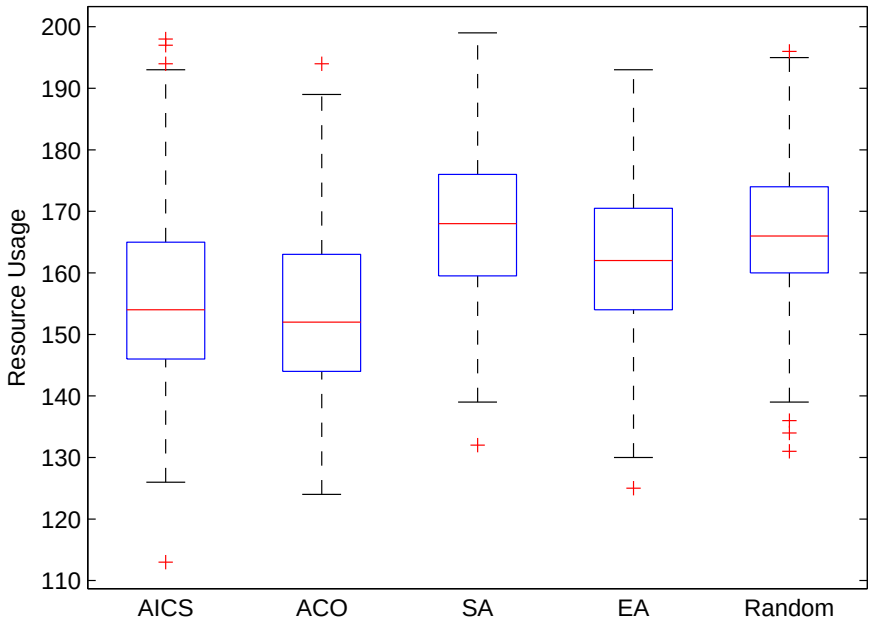

Figure 7.1: The box-whisker plot of the results obtained using HHs and *Random* heuristic.

From Table 7.5, we see that all the HHs perform better than the single LLHs. The success rates for the LLH_{RND}, *Random* and all the HHs are 100%, while, this rate is 70% for LLH_{MAX_SNG} and LLH_{MAX_TOT}. The best results obtained for each traffic matrix is marked in bold in the table. While in three of the traffic matrices, a single LLH produces the best result, it should be noted that the success rate for these LLHs is not 100%. The results show that, *ACO* finds the best result for 13 out of 20 traffic

matrices, while the next method is *AICS* with 5 best results. To test the statistical significance of the results of the HHs and *Random*, we applied a two-way ANOVA and a Tukey HSD post hoc test at a confidence level of 0.95. The results show that *ACO* and *AICS* produce results which are statistically significantly better than *SA*, *EA*, and *Random*, while a statistically significant difference cannot be observed between *ACO* and *AICS*. The box-whisker plot given in Figure 7.1 supports these results. Therefore, we can say that, constructive search techniques are more successful for this problem. Furthermore, based only on averages, we conclude that a population based scheme is preferable to a single point one.

7.2 Performance Evaluation of Hyper-Heuristics for Double-Link Failures

In Chapter 7.1, we show that the ACO-based HH approach outperforms other HH-based methods for the survivable VT design problem and has a 100% success rate. However, it did not include any mechanisms to improve scale-up.

To increase the scale-up of the solution, the flow-deviation method [101] is applied to the best solution obtained in the end. The aim in flow deviation is to balance the traffic load on lightpaths. After the traffic is routed through shortest paths, new weights are assigned to the lightpaths considering the traffic flow passing through them. The new weights are calculated using the equation, $W_{ij} = \frac{C_{ij}}{\gamma(C_{ij}-F_{ij})^2}$, where C_{ij} is the bandwidth capacity of the lightpath between nodes i and j, F_{ij} is the traffic flow routed through the lightpath between nodes i and j, and γ is the total traffic demand throughout the network. Then, the traffic amount of $\alpha * \gamma_{ij}$ for each node pair is routed again using shortest paths according to the new lightpath weights. γ_{ij} is the amount of traffic demand between nodes i and j, and α is the parameter to determine the proportion of traffic demand that will be rerouted. This reroute operation is repeated for a predefined number of steps. The flow-deviation method is given in Algorithm 15.

To see the effect of the flow deviation method, we analyze the traffic load on the lightpaths of a selected solution. In Figure 7.2, the traffic load on each lightpath is given both before and after applying the flow deviation method. The continuous line shows the load before and the dotted line shows the load after the flow deviation. The

Algorithm 15 General flow for flow-deviation method

1: route traffic through shortest paths using actual link costs
2: **for** a predefined number of steps **do**
3: assign new weights to the lightpaths considering the traffic flow passed through them
4: for each node pair, reroute some portion of traffic demand using new shortest paths
5: **end for**

figure clearly indicates that the traffic load on the lightpaths converge after applying the flow deviation method, so that, the traffic flow is balanced after flow deviation.

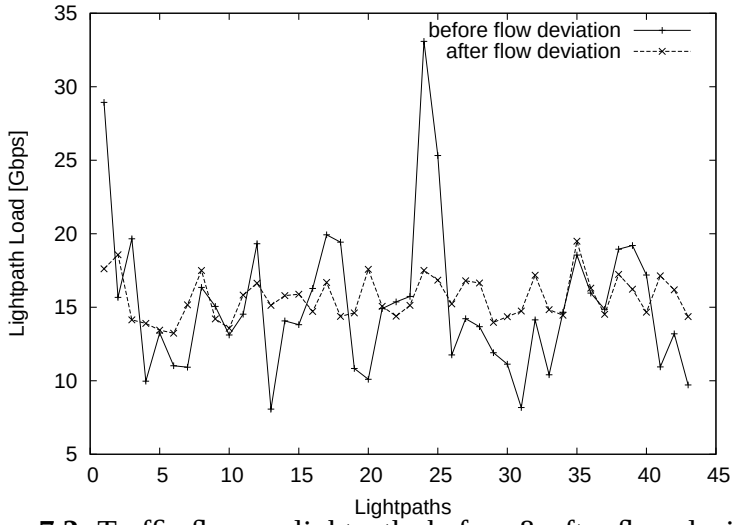

Figure 7.2: Traffic flow on lightpaths before & after flow deviation.

We tested different α values, i.e., 0.1, 0.2, and 0.3, used in flow-deviation applied for 100 steps to see its effect on the solution quality when there is a single-link failure. The results given in Table 7.6 are the averages of resource usages (RU) and scale-ups (SU) obtained after 20 runs for 20 different traffic matrices (a total of 400 results) using the corresponding α value. Table 7.6 shows no significant difference between different α values, and therefore, we chose 0.2.

Table 7.6: Effect of α for flow deviation (single-link failure case). Success rate is 100%.

$\alpha = 0.1$		$\alpha = 0.2$		$\alpha = 0.3$	
RU	scale-up	RU	scale-up	RU	scale-up
144	1.69	144	1.70	144	1.64

The proposed ACO-based HH solves the survivable VT design problem for single-link failures with a 100% success rate in 23 minutes on average. We also tested this method to see its performance when there is a double-link failure in the physical topology. The results of these experiments are shown in Table 7.7.

Table 7.7: Effect of number of shortest paths (sp) used for routing lightpaths and maximum search time (st) for an ant in ACO-M (double-link failure case).

	# of sp used for routing lightpaths (ant search time: 10sec)					**search time for an ant (# of sp: 15)**		
	10	15	20	25	50	10sec	15sec	20sec
success rate	14%	19%	28%	40%	48%	23%	25%	29%
resource usage	223	220	214	216	219	211	214	211
scale-up	2.83	2.85	2.69	2.75	2.71	2.74	2.85	2.77
run time (in mins)	24	27	28	34	55	24	33	47

The first observation is that, the success rate of our approach decreases when there is a double-link failure, as expected. However, it should be noted that, to solve the problem with double-link failures, our approach can still be used without any modification. The decrease in success rate led us to see how it is affected with the number of shortest paths used to route lightpaths (*sp-test*) and the maximum search time for an ant (*time-test*) in ACO-M[1]. First, we conducted the *sp-test*, in which, we tried different number of precalculated shortest paths. The values in columns between 2 and 6 of Table 7.7 are the averages of 20 algorithm runs for each of the 20 traffic matrices (total 400 runs). In order to isolate the effect of the maximum search time for an ant in the *time-test*, we omitted 4 traffic matrices for which no feasible solutions were found in the *sp-test*. Since the run time of the algorithm increases directly proportional with the increase in the maximum search time for an ant in ACO-M, in *time-test*, the algorithm is run 10 times for each traffic matrix. The last 3 columns in Table 7.7 show the results for these 16*10=160 tests.

Table 7.7 shows that the increase in both the number of shortest paths and the maximum search time, in turn, increases the success rate. Since there is a considerable increase in success rate, the increase in run time when we use larger number of shortest paths

[1]Since we use ACO in two phases for solution construction, i.e., mapping and heuristic selection phases, the ACO used for mapping will be referred to as ACO-M in this Chapter.

can be tolerable up to 25 shortest paths. We see that the run time increase is directly proportional with maximum search times for an ant in ACO-M, and the success rate performance does not increase considerably. As a result, using 25 shortest paths to route lightpaths and 10 seconds as a maximum search time for an ant in ACO-M are found to be a good set of parameters.

7.3 Performance Comparison of Hyper-Heuristics and Tabu Search for Survivable Virtual Topology Design Problem

In the literature, there are only two studies on survivable VT design problem [3, 13]. The ILP solution in [3] can solve the survivable VT design problem for only small (3-4 node) networks. Therefore, it will not be a reasonable comparison since our approach can also find the optimum solution for small networks in a fairly small amount of time.

The tabu search approach used in [13] can solve the survivable VT design problem for larger size networks. In their experimental study, they used 10, 14, 21, and 28 node networks. The proposed algorithm used in [13] first chooses an initial solution and then applies tabu search to this solution. At each tabu iteration, the whole neighborhood of the current solution is explored, and the best is selected as the current solution.

In [13], the tabu list size is selected as 7 and the stopping criterion is determined to be 300 iterations. However, they say that 100 iterations was not enough to find the optimum solution for only one case.

In this thesis, to compare the performance of our ACO-based HH approach and the tabu search approach we use a 24-node 43-link telco network [1]. The traffic matrices are the same matrices used in Chapter 7.1.

In [13], the VT degree is given as an input to the algorithm. The in-out degree of each node in the VT is same. In their experimental study, they used nodal degrees of 2, 3, 4, and 5. Their experimental results show that, the best performance is obtained when the nodal degree is 5. In our approach, the node degrees are not the same throughout the VT. We only give a maximum number of transceiver capacity (8 in the experiments of this thesis) and the minimum number of node degrees can be 2. As a result of

the experiments, the average of node degrees in our approach is 5.5. Therefore, we selected the nodal degree for tabu search approach as 5 in the experimental study.

Table 7.8: Resource usage results for different traffic matrices using different approaches.

	ACO-HH		**Tabu Search**	
	resource usage	**success rate**	**resource usage**	**success rate**
TM_1	177	100	174	80
TM_2	152	100	174	100
TM_3	148	100	171	100
TM_4	146	100	172	100
TM_5	149	100	177	100
TM_6	156	100	174	100
TM_7	169	100	179	100
TM_8	159	100	173	60
TM_9	157	100	175	100
TM_{10}	161	100	176	100
TM_{11}	137	100	170	100
TM_{12}	146	100	176	80
TM_{13}	158	100	168	100
TM_{14}	135	100	172	100
TM_{15}	148	100	175	100
TM_{16}	145	100	172	100
TM_{17}	175	100	178	80
TM_{18}	162	100	174	100
TM_{19}	139	100	171	100
TM_{20}	163	100	176	100

The experimental results are given in Table 7.8. In this table, the results of both approaches for each traffic matrix are given. Both the resource usage (RU in the table) and the success rate (SR in the table) results are discussed. The first column of the table indicates the traffic matrix for which the corresponding row has results. In the next two columns, the results of our approach and in the last two columns, the tabu search results are given. The values are the averages of resource usages obtained after 5 runs for each traffic matrix using the corresponding method.

The experimental results show that, our ACO-based HH approach can solve the problem with a 100% success rate for all the test cases. However, the success rate of tabu search approach is 100% for the 80% of the test cases. If we consider the quality of the solutions given in Table 7.8, we see that the solution quality of the HH approach is better than the tabu search approach for all the cases except 1. Moreover, the run time for our HH approach is at most 47 minutes, while it is at least 5 hours for tabu search[2] approach.

[2]In the paper, it is said that for a 28 node network, the run time of one tabu iteration is approximately 250 seconds. Similarly, with our implementation of their algorithm, the run time for one tabu iteration is approximately 180 seconds for a 24 node network.

8. CONCLUSION AND FUTURE WORK

In this thesis, we studied the survivable virtual topology design problem in optical WDM networks. First, we solve the survivable virtual topology mapping subproblem separately, and then, the survivable virtual topology design problem as a whole.

In the first phase of the thesis, for the survivable virtual topology mapping subproblem, we designed two different nature inspired heuristics, i.e. GA and ACO. After designing the proper operators for both heuristics, we tested these heuristics for different parameter values. In the parameter tuning tests, we did not notice any statistical difference between different values, which means that there is no need to struggle with the time consuming parameter setting tests while using these heuristics. Any parameter set will give a feasible solution for the survivable virtual topology mapping problem.

Moreover, we explored the effectiveness of these heuristics on network topologies of different sizes, i.e. 14-node, 24-node and 48-node physical topologies. Again, both heuristics performed well for the problem for any set of parameter values. Both heuristics can find a feasible solution in fairly short times, i.e. in seconds for 14-node topology, in less than a minute for 24-node topology, and in less than 8 minutes for 48-node topology. Furthermore, both heuristics can find feasible solutions for 100% of the studied cases of 14-node and 24-node topologies, and for 80-100% of the cases of 48-node topology.

To assess the performance of our nature inspired heuristics, we compared our experimental results with the ones obtained using basic ILP and an ILP relaxation given in the literature. Both the basic ILP and the ILP relaxation can be used only for the smaller size topologies of the problem due to the memory limitations. Test results show that, only 58 out of 300 cases for the problem with 24-node topology can be solved using basic ILP. Besides this, ILP-relaxation can find a lower-bound for 240 out of 300 cases for the problem with 24-node topology, whereas, this percentage is 300

out of 300 using both the GA and the ACO heuristics. The time requirements are even incomparable, which is at least 2 hours for basic ILP.

The results clearly demonstrate that both of our heuristics can solve the problem even for large-scale network topologies for which neither ILP can find the optimum solution nor the ILP relaxation can find a survivable solution. Additionally, the CPU time and the memory used by nature inspired heuristics is fairly low compared to the ILP method.

In the second phase of the thesis, to solve the survivable virtual topology design problem as a whole, we designed four different hyper-heuristic approaches. These hyper-heuristics are based on GA, ACO, AICS, and SA.

In hyper-heuristics, the first process is to determine some simple low-level heuristics that solve the problem. In this thesis, first, we designed five different low-level heuristics. Then, to assess their performance for the survivable virtual topology problem, we experimented with these low-level heuristics. As a result of this first phase of experiments, we eliminated two of these low-level heuristics which do not perform well for the problem.

After determining the low-level heuristics, we experimented with different parameter set for each hyper-heuristic to fine-tune the parameters. These experiments, as in the survivable virtual topology mapping problem, show that there is no need to deal with fine-tuning, any parameter set will give a feasible solution for the problem.

To compare the performance of our four hyper-heuristics designed for the problem, we carried out some experiments. In these perfomance comparison experiments, as a baseline for the performance comparisons, we generated 100 random solutions. As a result of the experiments, we can conclude that, each hyper-heuristic can find a feasible solution for 100% of the studied cases. However, ACO and AICS based hyper-heuristics outperfom other hyper-heuristics and the random solutions. The results show that, using hyper-heuristics can combine the best features of the low-level heuristics and give better results. The hyper-heuristic approaches proposed in this thesis for survivable virtual topology design are able to solve the problem for

large-sized networks having intensive traffic demands with a 100% success rate for all the test cases.

We also compared our results with the ones obtained using a tabu search approach proposed in the literature for the survivable virtual topology design problem. The success rate of tabu search approach is 100% for the 80% of the test cases. Furthermore, the solution quality of the hyper-heuristic approach is better than the tabu search approach for all the cases except 1. As a result, we can say that hyper-heuristic approaches can find better quality results using far less CPU time. The run time for our hyper-heuristic approach is at most 47 minutes, while it is at least 5 hours for tabu search.

In the literature, there are no studies considering double-link failure situations in survivable virtual topology design problem. However, our hyper-heuristic approach can be easily applied to the double-link failure situations without any modification in the algorithm. We also tested ant colony optimization based hyper-heuristic method to see its performance when there is a double-link failure in the physical topology. The results show that, our method can solve the survivable virtual topology design problem for double-link failures with more than 40% success rate.

There is no optimum solution method given in the literature for the survivable virtual topology design problem. The only study proposing an ILP method can find the optimum solutions for fairly small size networks (up to 4 nodes). As a future work, an ILP relaxation method can be derived to find the optimum solutions to the problem.

In this thesis, we studied the survivable virtual topology design problem considering only the resource usage objective. However, the survivable virtual topology design problem consists of two objectives: resource usage and scale-up. Another future work can be to design a multiobjective approach to solve the problem.

REFERENCES

[1] **Mukherjee, B.**, (2006). *Optical WDM Networks*., Springer, New York, USA.

[2] **Modiano, E. and Narula-Tam, A.**, (2002). Survivable Lightpath Routing: A New Approach to the Design of WDM-Based Networks., *IEEE Journal on Selected Areas in Communications*, **20(4)**, 800–809.

[3] **Cavdar, C.**, **Yayimli, A.G. and Mukherjee, B.**, (2008). Multilayer Resilient Design for Layer-1 VPNs., *Optical Fiber Communication Conference and Exposition (OFC)*, California, USA.

[4] **Eiben, A.E. and Smith, J.E.**, (2003). *Introduction to Evolutionary Computing*., Springer Verlag.

[5] **Dorigo, M. and Stutzle, T.**, (2004). *Ant Colony Optimization*., Massachusetts Institute of Technology, Cambridge Massachusetts.

[6] **Bonabeau, E.**, **Dorigo, M. and Theraulaz, G.**, (1999). *Swarm Intelligence: From Natural to Artificial Systems*., Oxford University Press, Oxford, England.

[7] **Dasgupta, D.**, (1999). *Artificial Immune Systems and Their Applications*., Springer-Verlag, Berlin.

[8] **Ripley, B.D.**, (2007). *Pattern Recognition and Neural Networks*., Cambridge University Press, New York, USA.

[9] **Pittenger, A.O.**, (2001). *An Introduction to Quantum Computing Algorithms*., Birkhauser, Boston.

[10] **Paun, G.**, **Rozenberg, G. and Salomaa, A.**, (1998). *DNA Computing: New Computing Paradigms*., Springer Verlag, Berlin.

[11] **Burke, E.**, **Hyde, M.**, **Kendall, G.**, **Ochoa, G.**, **Ozcan, E. and Woodward, J.**, (2010). **M. Gendreau and J. Potvin**, editors, *Handbook of Metaheuristics*, chapterA Classification of Hyper-heuristics Approaches, Springer.

[12] **Hoos, H.H. and Stutzle, T.**, (2005). *Stochastic Local Search: Foundations and Applications*., Morgan Kaufmann.

[13] **Nucci, A.**, **Sanso, B.**, **Crainic, T.**, **Leonardi, E. and Marsan, M.A.**, (2001). Design of Fault-Tolerant Virtual Topologies in Wavelength-Routed Optical IP Networks., *IEEE Global Communications Conference (GLOBECOM)*, Texas, USA.

[14] **Mukherjee, B.**, **Ramamurthy, S.**, **Banerjee, D. and Mukherjee, A.**, (1994). Some Principles for Designing a Wide-Area Optical Network., *IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, Canada, pp.110–118.

[15] **Saha, M. and Sengupta, I.**, (2005). A GA Based Approach for Static Virtual Topology Design in Optical Networks., *IEEE Indicon*, Chennai, India.

[16] **Saha, D.**, **Purkayastha, M.D. and Mukherjee, A.**, (1999). An Approach to Wide Area WDM Optical Network Design Using GA., *Computer Communications*, **22**, 156–172.

[17] **Chlamtac, I.**, **Ganz, A.**, **and Karmi, G.**, (1992). Lightpath Communications: An Approach to High Bandwidth Optical WAN's., *IEEE Transactions on Communications*, **40**, 1171–1182.

[18] **Phung, V.**, **Habibi, D. and Nguyen, H.**, (2004). An Efficient Approach to Optimal Wavelength Routing in WDM Optical Networks., *IEEE International Conference on Networks*, Singapore.

[19] **Dutta, R. and Rouskas, G.**, (2000). A Survey of Virtual Topology Design Algorithms for Wavelength Routed Optical Networks., *SPIE Optical Networks Magazine*, **1(1)**, 73–89.

[20] **Zang, H.**, **Jue, J. and Mukherjee, B.**, (2000). A Review of Routing and Wavelength Assignment Approaches for Wavelength-Routed Optical WDM Networks., *SPIE Optical Networks Magazine*, **1(1)**, 47–60.

[21] **Chan, K. and Yum, T.P.**, (1994). Analysis of Least Congested Path Routing in WDM Lightwave Networks., *IEEE International Conference on Computer Communications (INFOCOM)*, Toronto, Canada.

[22] **Li, L. and Somani, A.K.**, (1999). Dynamic Wavelength Routing Using Congestion and Neighborhood Information., *IEEE/ACM Transactions on Networking*, **7(5)**, 779–786.

[23] **Ramamurthy, S. and Mukherjee, B.**, (1998). Fixed-Alternate Routing and Wavelength Conversion in Wavelength-Routed Optical Networks., *IEEE Global Communications Conference (GLOBECOM)*.

[24] **Harai, H.**, **Murata, M. and Miyahara, H.**, (1997). Performance of Alternate Routing Methods in All-Optical Switching Networks., *IEEE International Conference on Computer Communications (INFOCOM)*, Kobe, Japan.

[25] **Chlamtac, I.**, **Ganz, A. and Karmi, G.**, (1989). Purely Optical Networks for Terabit Communication., *IEEE International Conference on Computer Communications (INFOCOM)*, Washington DC, USA.

[26] **Barry, R. and Subramaniam, S.**, (1997). The MAX-SUM Wavelength Assignment Algorithm for WDM Ring Networks., *Optical Fiber Communication Conference and Exposition (OFC)*.

[27] **Birman, A. and Kershenbaum, A.**, (1995). Routing and Wavelength Assignment Methods in Single-Hop All-Optical Networks with Blocking., *IEEE International Conference on Computer Communications (INFOCOM)*, Boston.

[28] **Jeong, G. and Ayanoglu, E.**, (1996). Comparison of Wavelength-Interchanging and Wavelength-Selective Cross-Connects in Multiwavelength All-Optical Networks., *IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco.

[29] **Subramaniam, S. and Barry, R.A.**, (1997). Wavelength Assignment in Fixed Routing WDM Networks., *IEEE International Communication Conference (ICC)*, Montreal, Canada.

[30] **Karasan, E. and Ayanoglu, E.**, (1998). Effects of Wavelength Routing and Selection Algorithms on Wavelength Conversion Gain in WDM Optical Networks., *IEEE/ACM Transactions on Networking*, **6(2)**, 186–196.

[31] **Zhang, X. and Qiao, C.**, (1998). Wavelength Assignment for Dynamic Traffic in Multi-fiber WDM Networks., *International Conference on Computer Communications and Networks*, Lafayette, LA.

[32] **Ramaswami, R. and Sivarajan, K.**, (1995). Routing and Wavelength Assignment in All-Optical Networks., *IEEE/ACM Transactions on Networking*, **3(5)**, 489–500.

[33] **Aloul, F.**, **Al-Rawi, B. and Aboelaze, M.**, (2008). Routing and Wavelength Assignment in Optical Networks Using Boolean Satisfiability., *IEEE Consumer Communications and Networking Conference*.

[34] **Di Caro, G. and Dorigo, M.**, (1998). AntNet: Distributed Stigmergetic Control for Communications Networks., *Journal of Artificial Intelligence Research*, **9**, 317–365.

[35] **R. Schoonderwoerd, O. Holland, J.B.**, (1996). Ant-like Agents for Load Balancing in Telecommunications Networks., *First International Conference on Autonomous Agents*, London, UK.

[36] **Varela, G.N. and Sinclair, M.C.**, (1999). Ant Colony Optimization for Virtual-Wavelength-Path Routing and Wavelength Allocation., *Congress on Evolutionary Computation*, **3**, 1809–1816.

[37] **Garlick, R.M. and Barr, R.S.**, (2002). Dynamic Wavelength Routing in WDM Networks via Ant Colony Optimization., *International Workshop on Ant Algorithms, LNCS 2463, Springer-Verlag*, **3**, 250–255.

[38] **Ngo, S.H.**, **Jiang, X. and Horiguchi, S.**, (2006). An Ant-Based Approach for Dynamic RWA in Optical WDM Networks., *Photonic Network Communications*, , 39–48.

[39] **Hassan, A.**, **Phillips, C. and Pitts, J.**, (2007). Dynamic Routing and Wavelength Assignment using Hybrid Particle Swarm Optimization., *EPSRC Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting*, Liverpool,UK.

[40] **Banerjee, N. and Sharan, S.**, (2004). An Evolutionary Algorithm for Solving the Single Objective Static Routing and Wavelength Assignment Problem in WDM Networks., *International Conference on Intelligent Sensing and Information Processing*, Chennai, India.

[41] **Banerjee, N.**, **Mehta, V. and Pandey, S.**, (2004). A Genetic Algorithm Approach for Solving the Routing and Wavelength Assignment Problem in WDM Networks., *IEEE/IEE International Conference on Networking*, Guadeloupe, France.

[42] **Dzongang, C.**, **Galinier, P. and Piere, S.**, (2005). A Tabu Search Heuristic for the Routing and Wavelength Assignment Problem in Optical Networks., *IEEE Communication Letters*, **9(5)**, 426–428.

[43] **Mokhtar, A. and Azizoglu, M.**, (1998). Adaptive Wavelength Routing in All-Optical Networks., *IEEE/ACM Transactions on Networking*, **6(2)**.

[44] **Markovic, G. and Raspopovic, V.A.**, (2010). Solving the RWA Problem in WDM Optical Networks Using the BCO Meta-Heuristic., *Telfor Journal*, **2(1)**.

[45] **Armitage, J.**, **Crochat, O. and Le Boudec, J.Y.**, (1997). Design of a Survivable WDM Photonic Network., *IEEE International Conference on Computer Communications (INFOCOM)*, Kobe, Japan.

[46] **Crochat, O. and Le Boudec, J.Y.**, (1998). Design Protection for WDM Optical Networks., *Journal on Selected Areas in Communications*, , 1158–1166.

[47] **Ducatelle, F. and Gambardella, L.M.**, (2004). A Scalable Algorithm for Survivable Routing in IP-Over-WDM Networks., *International Conference on Broadband Networks (BROADNETS)*, California, USA.

[48] **Ducatelle, F. and Gambardella, L.M.**, (2005). Survivable Routing in IP-over-WDM Networks: An Efficient and Scalable Local Search Algorithm., *Optical Switching and Networking*, **2(2)**, 86–99.

[49] **Kurant, M. and Thiran, P.**, (2004). Survivable Mapping Algorithm by Ring Trimming (SMART) for Large IP-over-WDM Networks., *International Conference on Broadband Networks (BROADNETS)*, California, USA.

[50] **Kurant, M. and Thiran, P.**, (2006). Survivable Routing in IP-over-WDM Networks in the Presence of Multiple Failures., *EuroNGI Workshop on Traffic Engineering, Protection and Restoration for NGI*, Poland.

[51] **Kurant, M. and Thiran, P.**, (2007). Survivable Routing of Mesh Topologies in IP-over-WDM Networks by Recursive Graph Contraction., *IEEE Journal on Selected Areas in Communications*, **25(5)**, 922–933.

[52] **Fumagalli, A. and Valcarenghi, L.**, (2000). IP Restoration vs. WDM Protection: Is There an Optimal Choice?., *IEEE Network*, .

[53] **Ngo, S.H.**, **Jiang, X.**, **Le, V. and Horiguchi, S.**, (2006). Ant-based Survivable Routing in Dynamic WDM Networks with Shared Backup Paths., *Journal of Supercomputing*, **36(3)**, 297–307.

[54] **Todimala, A. and Ramamurthy, B.**, (2007). A Scalable Approach for Survivable Virtual Topology Routing in Optical WDM Networks., *IEEE Journal on Selected Areas in Communications*, **25(6)**, 63–69.

[55] **Sahasrabuddhe, L.**, **Ramamurthy, S. and Mukherjee, B.**, (2002). Fault Management in IP-Over-WDM Networks: WDM Protection vs. IP Restoration., *IEEE Journal on Selected Areas in Communications*, **20**, 21–33.

[56] **Giroire, F.**, **Nucci, A.**, **Taft, N. and Diot, C.**, (2003). Increasing the Robustness of IP Backbones in the Absence of Optical Level Protection., *IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, USA.

[57] **Mukherjee, B.**, (1997). *Optical Communication Networks*., McGraw-Hill, New York, USA.

[58] **Mukherjee, B.**, **Ramamurthy, S.**, **Banerjee, D. and Mukherjee, A.**, (1996). Some Principles for Designing a Wide-Area Optical Network., *IEEE/ACM Transactions on Networking*, **4(5)**, 684–696.

[59] **Ramaswami, R. and Sivarajan, K.N.**, (1996). Design of Logical Topologies for Wavelength-Routed Optical Networks., *IEEE Journal on Selected Areas in Communications*, **14(5)**, 840–851.

[60] **Krishnaswamy, R. and SivaKajan, K.**, (1998). Design of Logical Topologies: A Linear Formulation for Wavelength-Routed Optical Networks with No Wavelength Changers., *IEEE International Conference on Computer Communications (INFOCOM)*, San Francisco, USA, pp.919–927.

[61] **Ou, C.S. and Mukherjee, B.**, (2005). *Survivable Optical WDM Networks*„ Springer.

[62] **Kurant, M.**, **Nguyen, H.X. and Thiran, P.**, (2006). Survey on Dependable IP over Fiber Networks., *Research Results of the DICS Program*, **4028**, 55–81.

[63] **Ramamurthy, S. and Mukherjee, B.**, (1999). Survivable WDM mesh networks, Part I-Protection., *IEEE International Conference on Computer Communications (INFOCOM)*, New York, USA, pp.744–751.

[64] **Ramamurthy, S. and Mukherjee, B.**, (1999). Survivable WDM Mesh Networks, Part II-Restoration., *IEEE International Communication Conference (ICC)*, Vancouver, BC, pp.2023–2030.

[65] **Mohan, G. and Somani, A.**, (2002). Routing Dependable Connections with Specified Failure Restoration Guarantees in WDM Networks., *IEEE International Conference on Computer Communications (INFOCOM)*, New York, USA.

[66] **Ergin, F.C.**, **Yayimli, A. and Uyar, S.**, (2009). An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks., *EvoWorkshops09, LNCS 5484, Springer-Verlag*, , 31–40.

[67] **Nucci, A.**, **Sanso, B.**, **Crainic, T.G.**, **Leonardi, E. and Marsan, M.A.**, (2004). On the Design of Fault-Tolerant Logical Topologies in Wavelength-Routed Packet Networks., *IEEE Journal on Selected Areas in Communications*, **2(9)**, 1884–1895.

[68] **Lin, T.**, **Zhou, Z. and Thulasiraman, K.**, (2011). Logical Topology Survivability in IP-over-WDM Networks: Survivable Lightpath Routing for Maximum Logical Topology Capacity and Minimum Spare Capacity Requirements., *Design of Reliable Communication Networks*, Krakow, Poland.

[69] **Zhang, L. and Zhang, N.**, (2011). Survivable Virtual Topology Design for Single-Node Failure ., *International Conference on Computer Research and Development (ICCRD)*, Shanghai, China.

[70] **Yuan, F.**, **Niu, X.**, **Li, X.**, **Huang, S. and Gu, W.**, (2011). Survivable Virtual Topology Mapping for Single-Node Failure in IP over WDM Network ., *Communications and Photonics Conference and Exhibition*, Shanghai, China.

[71] **Gendreau, M. and Potvin, J.**, (2010). **M. Gendreau and J. Potvin**, editors, *Handbook of Metaheuristics*, chapterTabu Search, Springer.

[72] **Resende, M. and Ribeiro, C.**, (2003). **F. Glover and G. Kochenberger**, editors, *Handbook of Metaheuristics*, chapterGreedy Randomized Adaptive Search Procedures, Kluwer Academic Publishers.

[73] **Clerc, M.**, (2006). *Particle Swarm Optimization*., ISTE.

[74] **Kaldirim, E.**, (2009), *Ant Colony Optimization for Survivable Virtual Topology Mapping in Optical WDM Networks*.

[75] **Ergin, F.C.**, **Kaldirim, E.**, **Yayimli, A. and Uyar, A.S.**, (2010). Ensuring Resilience in Optical WDM Networks with Nature Inspired Heuristics., *IEEE/OSA Journal of Optical Communications and Networking*, **2(8)**, 642–652.

[76] **Reeves, C.R.**, (2005). **E.K. Burke and G. Kendall**, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter19 - Fitness Landscapes, Springer.

[77] (2000). *ILOG CPLEX 6.5 User's Manual*.

[78] **Cowling, P.**, **Kendall, G. and Soubeiga, E.**, (2000). A Hyperheuristic Approach for Scheduling a Sales Summit., *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling*, **LNCS 2079**, 176–190.

[79] **Ross, P.**, (2005). **E. Burke and G. Kendall**, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapterHyperheuristics, Springer-Verlag.

[80] **Burke, E.**, **Hart, E.**, **Kendall, G.**, **Newall, J.**, **Ross, P. and Schulenburg, S.**, (2003). **F. Glover and G. Kochenberger**, editors, *Handbook of Metaheuristics*, chapterHyper-heuristics: An Emerging Direction in Modern Search Technology, Kluwer Academic Publishers.

[81] **Ozcan, E.**, **Bilgin, B. and Korkmaz, E.**, (2006). Hill Climbers and Mutational Heuristics in Hyperheuristics., *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature*, **LNCS 4193**, 202–211.

[82] **Soubeiga, E.**, (2003). *Development and Application of Hyperheuristics to Personnel Scheduling*, Ph.D. thesis, School of Computer Science and Information Technology, University of Nottingham.

[83] **Bai, R.**, (2005). *An Investigation of Novel Approaches for Optimizing Retail Shelf Space Allocation*, Ph.D. thesis, School of Computer Science and Information Technology, University of Nottingham.

[84] **Chakhlevitch, K. and Cowling, P.**, (2008). **C. Cotta**, editor, *Adaptive and Multilevel Metaheuristics*, chapterHyperheuristics: Recent developments, Springer.

[85] **Pisinger, D. and Ropke, S.**, (2007). A General Heuristic for Vehicle Routing Problems., *Computers & Operations Research*, **34(8)**, 2403–2435.

[86] **Dowsland, K.**, **Soubeiga, E. and Burke, E.**, (2007). A Simulated Annealing Hyper-heuristic for Determining Shipper Sizes., *European Journal of Operational Research*, **179(3)**, 759–774.

[87] **Cowling, P. and Chakhlevitch, K.**, (2003). Hyperheuristics for Managing a Large Collection of Low Level Heuristics to Schedule Personnel., *IEEE Congress on Evolutionary Computation(CEC)*.

[88] **Burke, E.**, **Kendall, G. and Soubeiga, E.**, (2003). A Tabu-Search Hyperheuristic for Timetabling and Rostering., *Journal of Heuristics*, **9(6)**, 451–470.

[89] **Burke, E.**, **McCollum, B.**, **Meisels, A.**, **Petrovic, S. and Qu, R.**, (2007). A Graph-based Hyper-heuristic for Educational Timetabling Problems., *European Journal of Operational Research*, **176**, 177–192.

[90] **Ross, P. and Marin-Blazquez, J.**, (2005). Constructive Hyper-heuristics in Class Timetabling., *IEEE Congress on Evolutionary Computation*, , 1493–1500.

[91] **Ross, P.**, **Schulenburg, S.**, **Marin-Blazquez, J. and Hart, E.**, (2002). Hyper-heuristics: Learning to Combine Simple Heuristics in Bin-packing Problem., *Genetic and Evolutionary Computation Conference (GECCO)*, New York, USA.

[92] **Vazquez-Rodriguez, J.**, **Petrovic, S. and Salhi, A.**, (2007). A Combined Meta-heuristic with Hyperheuristic Approach to the Scheduling of the Hybrid Flow Shop with Sequence Dependent Setup Times and Uniform Machines., *Multidisciplinary International Scheduling Conference (MISTA)*, Paris, France.

[93] **Hart, E.**, **Ross, P. and Nelson, J.**, (1998). Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule Builder., *Evolutionary Computation*, **6(1)**, 61–80.

[94] **Terashima-Marin, H.**, **Flores-Alvarez, E. and Ross, P.**, (2005). Hyper-heuristics and Classifier Systems for Solving 2D-regular Cutting Stock Problems., *Genetic and Evolutionary Computation Conference (GECCO)*, Washington DC, USA.

[95] **Terashima-Marin, H.**, **Ortiz-Bayliss, J.C.**, **Ross, P. and Valenzuela-Rendon, M.**, (2008). Hyper-heuristics for the Dynamic Variable Ordering in Constraint Satisfaction Problems., *Genetic and Evolutionary Computation Conference (GECCO)*, Atlanta, USA.

[96] **Kendall, G. and Mohamad, M.**, (2004). Channel Assignment In Cellular Communication Using A Great Deluge Hyper-heuristic., *IEEE International Conference on Network (ICON)*.

[97] **Nareyek, A.**, **Smith, S. and Ohler, C.**, (2003). Integrating Local-Search Advice into Refinement Search (Or Not)., *International Workshop on Cooperative Solvers in Constraint Programming*.

[98] **Bai, R.**, **Burke, E. and Kendall, G.**, (2008). Meta-heuristic and Hyper-heuristic Approaches for Fresh Produce Inventory Control and Shelf Space Allocation., *Journal of the Operational Research Society*, **59(10)**, 1387–1397.

[99] **Topcuoglu, H.**, **Ermis, M.**, **Corut, F. and Yilmaz, G.**, (2005). Solving the Uncapacitated Hub Location Problem Using Genetic Algorithms., *Computers & Operations Research*, **32(4)**, 967–984.

[100] **Aarts, E. and Korst, J.**, (1989). *Simulated Annealing and Boltzmann Machine*., Wiley, New York.

[101] **Fratta, L.**, **Gerla, M. and Kleinrock, L.**, (1973). The Flow Deviation Method: An Approach to Store-and-Forward Network Design., *Networks*, **3(2)**, 97–133.

[102] **Tavares, J.**, **Pereira, F.B. and Costa, E.**, (2008). Multidimensional Knapsack Problem: A Fitness Landscape Analysis., *IEEE Transactions on Systems, Man, and Cybernetics*, **38(3)**, 604–616.

[103] **Fonlupt, C.**, **Robilliard, D.**, **Preux, P. and Talbi, E.G.**, (1999). **S. Voss**, **S. Martello**, **I. Osman and C. Roucairol**, editors, *Metaheuristics – Advances and Trends in Local Search Paradigms for Optimization*, chapter18 - Fitness Landscapes and Performance of Metaheuristics, Kluwer Academic Press.

[104] **Jones, T.**, (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*, Ph.D. thesis, University of New Mexico, Albuquerque.

[105] **Weinberger, E.D.**, (1990). Correlated and Uncorrelated Landscapes and How to Tell the Difference., *Biological Cybernetics*, **63(5)**, 325–336.

[106] **Angel, E. and Zissimopoulos, V.**, (1998). Autocorrelation Coefficient for the Graph Bipartitioning Problem., *Theoretical Computer Science*, **191**, 229–243.

[107] **Stadler, P.F.**, (1992). Correlation in Landscapes of Combinatorial Optimization Problems., *Europhysics Letters*, **20(6)**, 479–482.

[108] **Brandt, H.**, (1999). Correlation Analysis of Fitness Landscapes., *International Institute for Applied Systems Analysis, Interim Report IR-01-058*, .

[109] **Angel, E. and Zissimopoulos, V.**, (2001). On the Landscape Ruggedness of the Quadratic Assignment Problem., *Theoretical Computer Science*, **263**, 159–172.

APPENDICES

APPENDIX A : Fitness Landscapes

APPENDIX A : Fitness Landscapes

The structure of the search space plays an important role on the performance and design of search algorithms. A search algorithm moves through the search space defined by the solution encoding and tries to locate high quality solutions which have high fitness values. Based on this, a fitness landscape is defined [102] as the relation between the points in the search space and the values in the fitness space. Many local search heuristics such as hillclimbing methods, simulated annealing, tabu search etc. and also evolutionary algorithms rely upon a neigborhood relation to navigate through the search space. At each move, the algorithm goes from one point in the search space to one of its neighbors. The encoding of the solutions and the chosen move operators determine the neigborhood structure. The ability of an algorithm to navigate through a neighborhood is closely related to its performance. Therefore the design of the neighborhood structure and the fitness landscape is crucial in designing successful search heuristics [102–104].

Ruggedness and the number of local optima can be considered among the most commonly utilized properties of fitness landscapes in analyzing the landscape structure and algorithm behavior. Usually a mathematical analysis of a fitness landscape [76] may not be possible. However there are some statistical measures used in analysing fitness landscapes. One of the earliest statistical techniques proposed to analyze the landscape ruggedness is the autocorrelation function [105] obtained through performing a random walk over the fitness landscape. The autocorrelation function (ACF) analysis has been performed on some well known combinatorial problem landscapes in the literature such as the graph bi-partitioning problem [106], the matching problem [107], the traveling salesman problem [108], the multi-dimensional knapsack problem [102], and the quadratic-assignment problem [109].

ACF looks at the amount of fitness correlation between the points in the search space. The ACF takes on values between $[-1,1]$ where values close to 1 denote a high correlation and values close to 0 show low correlation. The ACF value ρ_s is calculated as

$$\rho_s = \frac{\sum_{t=1}^{T-s}(f_t - \bar{f})(f_{t+s} - \bar{f})}{\sum_{t=1}^{T}(f_t - \bar{f})^2}, \qquad \textbf{(A.1)}$$

where s is the step size, T is the total number of sample points, f_t is the fitness of the t. solution, and $\bar{f}$ is the average fitness of all the points.

It is stated in [76] that approximately with 0.05 probability the $\mid \rho_s \mid$ values can be larger than given in Eq. (**A.2**) by chance where T is the length of the random walk. Therefore values less than this threshold may be taken as being equal to 0.

$$\tau = s : \mid \rho_{s+1} \mid < 2/\sqrt{T} \wedge \left\{ \mid \rho_k \mid > 2/\sqrt{T} \quad \forall k \leq s \right\} \qquad \textbf{(A.2)}$$

To compare the ruggedness of different landscapes, usually the correlation length which is obtained from the ACF, is used. The correlation length λ is defined as in Eq.(**A.3**) where

$$\lambda = -\frac{1}{ln(\mid \rho_1 \mid)} \tag{**A.3**}$$

A high correlation length (large λ), means a smoother landscape while low values (small λ) show that the landscape is more rugged. However, as stated in [76], even though the ACF and the correlation length values are very useful, sometimes they are hard to interpret and may not be sufficient alone to determine the true ruggedness of the landscape.

CURRICULUM VITAE

Name Surname: Fatma CORUT ERGİN

Place and Date of Birth: Polatlı - 1979

E-Mail: fatma.ergin@gmail.com

B.Sc.: June 2002

M.Sc.: October 2005

Professional Experience and Rewards:

- 2002 - Present: Teaching Assistant, Marmara University, Computer Engineering Department

List of Publications and Patents:

- Topcuoğlu H., **Corut F.**, Ermiş M., Yılmaz G., 2005: Solving the Uncapacitated Hub Location Problem Using Genetic Algorithms. *Computers and Operations Research*, Vol. 32, pp.967-984, 2005.

PUBLICATIONS/PRESENTATIONS ON THE THESIS

- **Corut Ergin F.**, Kaldırım E., Yayımlı A., Uyar A. Ş., 2010: Ensuring Resilience in Optical WDM Networks With Nature-Inspired Heuristics. *IEEE/OSA Journal of Optical Communications and Networking*, Vol. 2, Issue 8, pp.642-652, 2010.

- **Corut Ergin F.**, Yayımlı A., Uyar A. Ş., 2011: Survivable Cross-layer Virtual Topology Design using a Hyper-heuristic Approach. *ICTON*, Jun. 26-30, 2011 Stockholm, Sweden.

- **Corut Ergin F.**, Uyar A. Ş., Yayımlı A., 2011: Investigation of Hyper-heuristics for Designing Survivable Virtual Topologies in Optical WDM Networks. *EvoStar - EvoWorkshops*, April 27-29, 2011 Torino, Italy.

- **Corut Ergin F.**, Kaldırım E., Yayımlı A., Uyar A. Ş., 2009: Performance Analysis of Nature Inspired Heuristics for Survivable Virtual Topology Mapping. *IEEE GLOBECOM*, Nov. 30-Dec.4, 2009 Hawaii, USA.

- Kaldırım E., **Corut Ergin F.**, Uyar A. Ş, Yayımlı A., 2009: Ant Colony Optimization for Survivable Virtual Topology Mapping in Optical WDM Networks. *ISCIS*, Sept. 14-16, 2009, Turkish Republic of Northern Cyprus.

- **Corut Ergin F.**, Uyar A. Ş, Yayımlı A., 2009: An Evolutionary Algorithm for Survivable Virtual Topology Mapping in Optical WDM Networks. *EvoStar - EvoWorkshops*, April 15-17, 2009, Tübingen, Germany.