2013

The Application of Evolutionary Algorithms for Energy Efficient Grooming of Scheduled Sub-Wavelength Traffic Demands in Optical Networks

Ala Shaabana
University of Windsor

The Application of Evolutionary Algorithms for Energy Efficient Grooming of Scheduled Sub-Wavelength Traffic Demands in Optical Networks

By

Ala Shaabana

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada
2013

The Application of Evolutionary Algorithms for Energy Efficient Grooming of Scheduled Sub-Wavelength Traffic Demands in Optical Networks

by

Ala Shaabana

APPROVED BY:

____
Dr. Kemal Tepe
Department of Electrical and Computer Engineering

____
Dr. Ziad Kobti
School of Computer Science

____
Dr. Arunita Jaekel, Advisor
School of Computer Science

May 13th, 2013

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyones copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

In recent years there has been a growing recognition of the need for developing energy efficient network design approaches for WDM backbone networks as well. The typical approach has been to switch off some components such as line cards and router ports during low demand periods, and has focussed on traditional static and dynamic traffic models. In this paper, we present a new approach that exploits knowledge of demand holding times to intelligently share resources among non-overlapping demands and reduce the overall power consumption of the network. We consider the fixed-window scheduled traffic model (STM), and present i) a Genetic Algorithm (GA) and ii) a Memetic Algorithm (MA) based strategy that jointly minimizes both power consumption and transceiver cost for the logical topology. Simulation results clearly demonstrate that both of the proposed algorithms outperform traditional holding time unaware (HTU) approaches; the GA leads to additional improvements even compared to the shortest path holding time aware (HTA) heuristic. However, the MA manages to achieve similar results to the GA while taking up 4 to 5 times less computational resources and time to compute.

DEDICATION

To my parents, for without your patience, effort, and support, I would not have come this far.

To K.J.I., for without your encouragement, inspiration, and love, this would have become an even more difficult task to achieve.

Last but never least, to my friends, for your all of your support and love, through the good and the bad.

ACKNOWLEDGEMENTS

I wish to first thank God, for always helping me understand how to tread through this difficult path, and through the path that lies ahead. Mom and dad, I thank you for never giving up; for your love, courage, support, and determination, and your wise guidance the past 24 years. Without you both I, most certainly, would have never even had the opportunity to enter any post-secondary academic institution. Dr. Jaekel, Dr. Kobti, Dr. Tepe, and Dr. Bandyopadhyay, I thank you for what you have taught me during my early years of learning to become a researcher. I also wish to thank you for your comments, questions and criticisms of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Optical communication is communication at a distance to carry information using light. It can be performed visually or by using electronic interfaces. The earliest and most basic forms of optical communication date back several millennia, while the earliest electrical device created to do so was invented in 1880, called the photophone – a device that allowed for the transmission of speech on a beam of light. An optical communication system uses a transmitter, a channel, and a receiver. The transmitter encodes a message into an optical signal, the channel carries the signal to its destination, and finally the receiver reproduces the message from the received optical signal.

An optical network is composed of the fiber-optic cables that carry channels of light, combined with the equipment deployed along the fiber to process the light. The capabilities of an optical network are necessarily tied to the physics of light and the technologies for manipulating lightstreams (or wavelengths). Hence, the evolution of optical networks has been marked with major paradigm shifts as breakthrough technologies are developed [1]. One of the earliest technological advances was the ability to carry multiple channels of light on a single fiber-optic cable. Each wavelength is carried at a different optical frequency and multiplexed onto a single fiber, giving rise to Wavelength Division Multiplexing (WDM).

Increased wavelength rate, combined with a greater number of wavelengths per fiber has expanded the capacity of optical networks by several orders of magnitude over a period of 25 years. However, transmission capacity is only *one* important factor. Historically, the contents of each wavelength have undergone electronic processing at numerous points nt he network. As networks increased greatly in size, this necessitated the use of a tremendous amount of electronic terminating and switching equipment, which presented challenges in cost, power consumption, heat dissipation, physical space, and maintenance.

1.2 Motivation

In the past decade, the immense growth in high-bandwidth applications, such as Video-on-Demand and online media sharing, has given rise to a corresponding increase in energy consumption of the network equipment [2]. Researchers have realized the importance of designing energy-minimized green networks to utilize the available power efficiently and consequently reduce the network operational cost. For a middle-sized country, for example, a 1% improvement of the total energy consumption can lead to a reduction of 5 billion US dollars per year in electricity cost [3]. It is therefore necessary to develop robust optimization strategies for the design of energy-efficient core networks. The typical approach is to switch off some network components during low traffic periods.

There have been many different approaches have been proposed for reducing power consumption in communication networks, including putting network interfaces and components to sleep [2], switching off line cards [4, 5], or even entire links or nodes [6, 7]. Our approach differs from these existing schemes in that we consider the applications that require *periodic* use of bandwidth at predetermined times. Unlike the static or dynamic traffic demands, this type of demands (also called scheduled traffic demands [8]) is periodic and predictable, so resource allocation can be optimized in both space and time.

Typically, optimization problems of this kind are solved by Integer Linear Programs (ILPs). Although ILPs achieve an optimal solution, they can only do so for smaller network sizes, once networks such as the 14 node NSFNET and the 20 node ARPANET come into the equation, the ILP tends to become computationally intractable, as it takes too long and consumes too many computational resources to find its solution. The proposed Genetic Algorithm (GA) and Memetic Algorithm (MA) are able to achieve moderate energy improvements at a much shorter time, and using up less computational resources. On the other hand, our MA consumes even less computational resources and achieves results that are as good as or better than the results reported by the GA in a significantly less amount of time.

1.3 Solution Outline

We present a GA-based approach as well as an MA-based approach to route a set of periodic, sub-wavelength traffic demands over the network and show that consideration of demand holding times can play an important role in reducing the overall energy consumption of a network. The primary goal for these two approaches is to route the traffic demands in such a way that the maximum number of lightpaths can be switched *off* at any given time, hence reducing the overall power consumption. Additionally, the other objective is to reduce the total number of lightpaths needed to realize the logical topology, such that the capacity constraints of the lightpaths are not exceeded. Specifically, we try to implement each logical edge using as few lightpaths as possible, which in turn reduces the need for optical transceivers.

Our GA is able to reduce the energy consumption more so than a simple shortest path holding-time-aware heuristic, presented in [9]. Furthermore, where ILPs become too computationally expensive to apply to large networks, the GA can be applied to achieve solutions with good energy efficiency. In contrast, the MA builds upon the GA and improves it even further by adding local search capabilities, allowing it to further optimize the GA's solution and achieve more energy efficient routing and power consumption while at the same time using up even less computational resources and time due to its ability to detect population convergence.

1.4 Thesis Organization

The remainder of this thesis is as follows. In Chapter 2, we review optical networks and their underlying technologies. Furthermore, we review the biological motivation behind evolutionary algorithms such as GAs and MAs. We then review GAs and MAs, as well as past research that has looked into using evolutionary algorithms in optical networks and other applications. In Chapter 3, we present our solution and methodology, as well as a network model upon which we based our experimentation. In Chapter 4 a summary of the results and experiments carried out to test the proposed algorithms. Finally, Chapter 5 concludes the work done in this thesis and suggests some possible future research directions.

Chapter 2

Review

2.1 Optical Networks

Optical fiber communication is a method of transmitting information from one place to another by sending pulses of light through an optical fiber, forming an electromagnetic carrier wave that is modulated to carry information. Because of its advantages over electrical transmission, copper wire communications have largely been replaced by optical fibers in core networks. The process of communicating using fiber-optics involves the following basic steps:

1. Creating the optical signal using a transmitter.
2. Relaying the signal along the fiber, ensuring the signal does not become distorted or weak via the use of amplifiers.
3. Recieving the optical signal.
4. Converting the signal into an electrical signal.

Optical communications are used by many telecommunications companies to transmit various signals, including but not limited to: telephone signals, Internet communication, and cable television signals. Optical fiber has large advantages over existing copper wire in long-distance and high-demand applications due to lower attenuation and interference. At the time of writing this paper, optical fibers are relatively cheap. However, this was not the case more than a decade ago. Since infrastructure development within cities was a difficult and time consuming process, and optical fiber systems and networks were complex and expensive to install and operate. Hence, optical fiber communication systems were primarily installed in long-distance applications, where they can be used to their full transmission capacity, offsetting their high cost.

Today, however, the price for rolling out fiber to the home has currently become *more* cost-effective than that of rolling out a copper-based network. Since 1990, the telecommunications industry has laid a vast network of intercity and transoceanic fiber optic communication lines. By 2002, an intercontinental network of 250,000 km of submarine communications cable with a capacity of 2.56 Tb/sec was completed.

2.1.1 Technologies

Modern fiber-optic communication systems generally include:

1. An optical transmitter to convert an electrical signal into an optical signal to send into the optical fiber.

2. A cable containing bundles of multiple optical-fibers that is routed

through underground conduits and buildings.

3. Various kinds of amplifiers.

4. An optical receiver to recover the signal as an electrical signal.

2.1.1.1 Transmitters

The most commonly used optical transmitters are semiconductor devices such as Light-Emitting Diodes (LEDs) and laser diodes. In optical communications, semiconductor optical transmitters must be designed to be compact, efficient, and reliable, while operating in an optimal wavelength range, and directly modulated at high frequencies.

2.1.1.2 Amplifiers

The transmission distance of an optical fiber network has typically been limited by fiber attenuation and by fiber distortion. By using opti-electronic repeaters, these problems have been eliminated. The problems of limited transmission distance due to fiber attenuation fiber distortion have been eliminated via the use of opto-electronic repeaters. The repeaters convert the signal into an electrical signal, and then use a transmitter to send the signal again at a higher density than it was before. These repeaters tend to be very expensive due to the high complexity with modern Wavelength-Division Multiplexed (WDM) signals [10].

Instead, engineers use optical amplifiers, which amplify the optical signal directly, bypassing the process of converting the signal into an electrical signal. Naturally, they have largely replaced repeaters in new installations.

2.1.1.3 Fiber Cables

An optical fiber consists of a core and a cladding (made of high-quality silica glass, although plastic can be used as well), as well as a buffer (a protective outer coating), in which the cladding guides the light along the core by using total internal reflection. Total internal reflection is a phenomenon that occurs when a propagating light wave strikes a medium boundary at an angle larger than a particular critical angle with respect to the normal to the surface. Connecting two optical fibers is done by fusion or mechanical splicing. Due to the microscopic precision required to align the fiber cores, it also requires interconnection technology and special training [10].

2.1.1.4 Receivers

Photodetectors, the main components of optical receivers, convert light into electricity. A Photodetector is typically a semiconductor-based photodiode. Several types of photodiodes include:

1. $p-n$ photodiodes.

2. $p-i-n$ photodiodes.

3. Avalanche photodiodes.

Metal-semiconductor-metal (MSM) photodetectors are also used due to their suitability for circuit integration in regenerators and WDMs [10].

2.1.2 Comparison with Electrical Transmission

Optical fiber is generally chosen for systems requiring higher bandwidth or spanning longer distances than electrical cabling can accommodate. Specifically, the main benefits of fiber optics are [10]:

1. Very low data loss rate (thereby allowing long distances between amplifiers and repeaters).
2. Due to its reliance on light rather than electricity for transmission, and the dielectric nature of fiber optics, there is an absence of ground currents and other signal and power issues that are otherwise common in long parallel electric conductor runs.
3. Its high data-carrying capacity (for perspective, thousands of electrical cables would be required to replace a single high bandwidth fiber cable).

In short distance and relatively low bandwidth applications, electrical transmission is often preferred because of its [10]:

1. Lower material cost, where large quantities are not required.
2. Lower cost of transmitters and receivers.
3. Capability to carry electrical power as well as signals.
4. Ease of operating transducers in linear mode.

At higher power outputs, optical fibers are susceptible to fiber fuse (occurs when a fiber is subjected to a shock or otherwise suddenly damaged), resulting in the destruction of the fiber core and damage to transmission components. Furthermore, optical fibers are more difficult and expensive to splice than electrical conductors [10].

Because of these benefits of electrical transmission, optical communication is not common in short box-to-box, backplane, or chip-to-chip applications. In certain situations, optical fiber may be used even for short distance or low bandwidth applications, due to other important features:

1. Lighter weight to carry in transport.

2. Not electromagnetically radiating and difficult to tap without disrupting the signal — this is an asset in high security situations.

3. Resistance to corrosion due to non-metallic transmission medium.

4. No sparks — eliminates the concern for flammability.

5. Immunity to electromagnetic interference.

6. High electrical resistance, making it safe to use near high-voltage equipment or in geographically challenging areas.

7. Smaller cable size — important where pathway is limited.

Optical fiber cables can be installed in buildings with the same equipment that is used to install copper and coaxial cables, with some modifications due to the small size and limited pull tension and bend radius of optical cables [10].

2.2 The Biology

The theory of evolution is central to this thesis. As such it makes sense to discuss the biological side of the theory before we dive into the gritty computational side. We will first highlight some definitions that will be used often in the rest of this work. A *gene* is a sequence of DNA bases that code for a trait, like eye or hair color. An *allele* is a value of a trait. For example the eye color gene could have a blue allele or a hazel allele in different people [11]. Hence, the definition accepted by most biologists is "Evolution is the variation of allele frequencies in populations over time". Interestingly enough, "evolution is the survival of the fittest" is a good description of many evolutionary computation systems. When we use evolutionary computation to solve a problem, we operate on a collection, or population, of data structures (or creatures). These "creatures" will have explicitly computed fitnesses used to decide which of them will be partially or completely copied (have offspring) [11]. This fundamental difference in the notion of fitness is a key difference between biological evolution and most evolutionary computation.

Evolution produces new forms over time. This is clear from fossil record examinations and from looking at molecular evidence or "genetic fossils". This ability to produce new forms, in essence to innovate without outside direction other than the imperative to have children that live long enough to have children themselves, is the key feature that we wish to reproduce in this research.

There are two opposing forces that drive evolution: variation and selection. Variation is the process that produces new alleles, and, more slowly, genes. Furthermore, variation can also change which genes are or are not expressed in a given individual. The simplest method of doing this is sexual reproduction with its interplay of dominant and recessive genes. In contrast, evolutionary computing operates on populations of data structures. It accomplishes variation by making random changes in these data structures and by blending parts of different structures via processes called *mutation* and *crossover*, together referred to as *variation operators*. There are good and bad mutations operating on a population of data structures. A good mutation is one that increases the fitness of a data structure, while a bad mutation is one that reduces the fitness of a data structure [11].

Selection is the process whereby some alleles survive and others do not. In short, variation builds up genetic diversity, while selection reduces it. In terms of evolutionary computing, selection is accomplished with any algorithm that favours data structures with a higher fitness score. There are many possible methods to achieve selection.

2.3 Evolutionary Algorithms

Nearly three decades of research and development have demonstrated that the mimicked search process of natural evolution can yield very robust, direct computer algorithms, even though these imitations are crude simplifications of biological reality [12]. The result of these efforts is Evolutionary Algorithms (EA). Based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a specific problem.

The population evolves towards improving regions of the search space by means of randomized processes of selection, mutation and recombination (sometimes recombination is not used in some algorithms). Moreover, the population is arbitrarily initialized. The environment delivers quality information (i.e. fitness value) of the individuals, and the selection process favours those individuals of higher fitness to reproduce more often than worse individuals [12]. Finally, the recombination mechanism allows the mixing of parental information while passing it to their descendants, while mutation introduces innovation into the population (some algorithms do not check if the "innovation" is good or bad, and proceed nonetheless).

In order to solidify our description, we will introduce some notational conventions. Let $f : \mathbf{R}^n \rightarrow \mathbf{R}$ denote the objective function to be optimized, and without loss of generality we assume a minimization task in the following: let $\Phi : I \rightarrow \mathbb{R}$ (with I being the space of individuals) be the fitness function. Generally speaking, fitness and objective function values of an individual are not required to be identical, such that Φ and f are distinguished mappings, however f is always a component of Φ. Meanwhile $\overrightarrow{a} \in I$ is used to denote an individual, $\overrightarrow{x} \in \mathbf{R}^n$ indicates an object variable vector. Moreover, $\mu \geq 1$ and $\lambda \geq 1$ denote the size of the parent population and the size of the offspring population (created by recombination and mutation at each generation) size, respectively. A population at generation t, $P(t) = \{\overrightarrow{a}_1(t), ..., \overrightarrow{a}_\mu(t)\}$, consists of individuals $\overrightarrow{a}_i(t) \in I$. $r_{\Theta_r} : I^\mu \rightarrow I^\lambda$ denotes the recombination operator which might be controlled by additional parameters summarized in the set Θ_r.

Similarly, the mutation operator $m_{\Theta_m} : I^{\mu} \to I^{\lambda}$ modifies the offspring population, also being controlled by some parameters Θ_m. Although introduced in this thesis as macro-operators transforming populations into populations, both mutation and recombination can be reduced to local operators $m'_{\Theta_m} : I^{\mu} \to I^{\lambda}$ and $r'_{\Theta_r} : I^{\mu} \to I^{\lambda}$, respectively, that create *one* individual when applied. To choose the parent population of the next generation, selection $s_{\Theta_s} : (I^{\lambda} \cup I^{\mu+\lambda}) \to I^{\lambda}$ is applied. The fitness function $\Phi : I \to \mathbb{R}$ is calculated for all individuals of a population during the evaluation step, and $\iota : I^{\mu} \to \{true, false\}$ us used to denote the termination criterion.

Thus, we can represent this with the following algorithm:

```
begin
    $t = 0$;
    $initialize P(0) = \{\overrightarrow{a}_1(0), ..., \overrightarrow{a}_\mu(0)\}$ ;
    $evaluate P(0) = \{\Phi(\overrightarrow{a}_1(0)), ..., \Phi(\overrightarrow{a}_\mu(0))\}$ ;
    while $\iota(P(t)) \neq true$ do
        $recombine : P'(t) = r_{\Theta_r}(P(t))$ ;
        $mutate : P''(t) = m_{\Theta_m}(P'(t))$ ;
        $evaluate : P''(t) : \{\Phi(\overrightarrow{a''}_1(t)), ..., \Phi(\overrightarrow{a''}_\lambda(t))\}$ ;
        $select : P(t+1) = s_{\Theta_s}(P''(t) \cup Q)$ ;
        $t = t + 1$ ;
    end
end
```

Algorithm 1: Evolutionary Algorithm Skeleton, as described in [12]

Where $Q \in \{\theta, P(t)\}$ is a set of individuals that are additionally taken into account during the selection step. The evaluation process yields a multiset of fitness values, which are not necessarily identical to objective function values [12]. However, fitness values are used here as a result of the evaluation process, since the selection criterion operates on fitness instead of objective function values. The evaluation of objective function values is *always* necessary during the calculation of fitness, this is so the information is available and can easily be stored in an appropriate data structure.

2.4 Genetic Algorithms

Genetic Algorithms (GAs) are the most common of evolutionary algorithms. Generally, a population of candidate solutions to an optimization problem is evolved towards better solutions. Each solution has a set of properties (chromosomes or genotype) which can be mutated and altered. Traditionally, solutions are represented in binary as strings of 1s and 0s, however other encodings are possible [13]. An iterative process, the evolution typically starts from a population of randomly generated individuals, with the population in each generation referred to as a generation. The fitness of every individual – the value of the objective function in the optimization problem being solved – is evaluated in every generation. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (typically recombined or randomly mutated) to form a new generation [13]. The new generation of candidate solutions is then used in the next iteration of the algorithm. Traditionally, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Typically, a GA requires:

1. A genetic representation of the solution domain.

2. A fitness function to evaluate the solution domain.

An array of bits is usually the standard representation of each candidate solution. They are typically used because their parts are easily aligned due to their fixed size, however arrays of other types and structures can be used in the same way. Variable length representations are also possible, but crossover implementation becomes more complex when one has to account for different sizes of chromosomes.

Eventually, structures in the search space were progressively modified in this model by operators selected by an adaptive plan, judging about the quality of previous trials by means of an evaluation measure [12]. In 1975 John Holland showed how to interpret the reproductive plans in terms of genetics, economics, game-playing, pattern recognition and parameter optimization [14].His GAs were applied to parameter optimization for the first time by K. De Jong [12], who laid the foundations of this application technique.

Today there exist *many* numerous modifications of the original GA – usually referred to as Canonical GAs – are all applied to many fields in Computer Science. However, many of these applications show enormous differences to the canonical GA, as we will explain shortly.

2.4.0.1 Representation and Fitness

For initialization, GAs assume a bounded subspace $\Pi_{i=1}^{n}[u_i, v_i] \subset \mathbf{R}^n$ with $u_i < v_i$, and work on bit strings of fixed length l – that is, $I = \{0, 1\}^l$ [12]. The bit string is logically divided into n segments – typically of equal length l_x (that is, $l = n \cdot l_x$) – in order to apply canonical GAs to continuous parameter optimization problems of the form $f : \Pi_{i=1}^{n}[u_i, v_i] \rightarrow \mathbf{R}(u_i < v_i)$. Each segment is interpreted as the binary code of the corresponding object variable $x_i \in [u_i, v_i]$.

A segment decoding function $\Gamma^i : \{0,1\}^{l_x} \rightarrow [u_i, v_i]$ typically looks like:

$$\Gamma^i(a_{i1}...a_{il_x}) = u_i + \frac{v_i - u_i}{2^l_x - 1}(\sum_{j=1}^{l_x} a_{ij}2^{j-1}) \quad (2.1)$$

where the i-th segment of an individual $\overrightarrow{a} = (a_1 1...a_n l_x) \in \{0,1\}^{n \cdot l_x} = I$ is denoted by $(a_i 1...a_i l_x)$. Combining the segment-wise decoding functions Γ^i to an individual-decoding function $\Gamma = \Gamma^1 \times ... \times \Gamma^n$, fitness values are obtained by setting $\Theta(\overrightarrow{a}) = \delta(f(\Gamma(\overrightarrow{a})))$, where again δ denotes a scaling function assuring positive fitness values such that the best individual receives the largest fitness [12]. A linear scaling is more commonly used to take into account the *worst* individual of the population $P(t-\omega)$ ω time steps before $(t - \omega < 0 \Rightarrow P(t-\omega) = P(0))$:

$$\delta(f(\Gamma(\overrightarrow{(}a)), P(t-\omega))) = max\{f(\Gamma(\overrightarrow{a_j}))|\overrightarrow{a_j} \in P(t-\omega)\} - f(\Gamma(\overrightarrow{a})) \quad (2.2)$$

where ω is referred to as the scaling window. This representation method is a special technique developed for the application of canonical GAs to parameter optimization problems [12]. The wide range of alternative representations based on the binary code allows canonical GAs to be applied to various different problems.

2.4.0.2 Mutation

Traditionally referred to as a “background operator” [14], in canonical GAs works on the bit string level. Particularly, it inverts single bits of individuals, with the probability p_m of this event typically being small – $p_m \approx 1 \cdot 10^{-3}$ per bit, for our MA, we have applied a mutation probability of $p_m \approx 2 \cdot 10^{-3}$, which we will discuss later. This kind of mutation depends neither on the number n of object variables, nor on the total length l of the bit string, instead it is ruled by randomness, like the Monte Carlo method. The reason we did not apply mutation to our GA was that it did not produce a significant difference in results. For a single individual, mutation $m'_{p_m} : I \rightarrow I, m'_{p_m}(s_1, ..., s_l) = (s'_1, ..., s'_l)$ works as follows:

$$s'_i = \begin{cases} s_i & \chi_i > p_m \\ 1 - s_i & \chi_i <= p_m \end{cases}$$

$\chi_i \in [0, 1]$ here is the product of a uniformly distributed function, thereby a uniform random variable, sampled anew for each bit.

2.4.0.3 Recombination

Emphasis is placed mainly on crossover in canonical GAs, the recombination operator of GAs, as the main variation operator which researchers hope recombines useful segments from different individuals. Crossover is again an operator working entirely on bit string representation, and completely ignores the genetic code and epigenetic apparatus [12]. It also does not respect the semantic boundaries of the encoded variables. An eternal parameter p_c

(or crossover rate) indicates the probability per individual to undergo recombination. Typical values for p_c are in the range of $[0.6, 1.0]$. Our rate of recombination for both our Memetic and GAs is 1.0.

When two parent individuals $\overrightarrow{s} = (s_1, ..., s_l)$, $\overrightarrow{v} = (v_1, ..., v_l)$ have been selected (at random) from the population, crossover forms two offspring individuals $\overrightarrow{s'}$ and $\overrightarrow{v'}$. Represented by:

$$\begin{aligned} \overrightarrow{s'} &= (s_1, ..., s_{\chi-1}, s_\chi, v_{\chi+1}, ..., v_l) \\ \overrightarrow{v'} &= (v_1, ..., v_{\chi-1}, v_\chi, s_{\chi+1}, ..., s_l) \end{aligned} \tag{2.3}$$

$\chi_i \in [0, 1]$ still is the product of a uniformly distributed function, thereby a uniform random variable, and one of the two offspring individuals is randomly selected to be the overall result of a crossover. This is referred to generally as *one-point crossover*, and can be extended to a more generalized m-point crossover by sampling more than one breakpoint and alternately exchanging each second resulting segment [15]. On the other hand, the *uniform crossover* operator drives the number of crossover points to an extreme by performing the random decision on whether to exchange information between parents or not for each new bit of the genotype [16]. Interestingly, neither a theoretical nor empirical evidence exists to decide upon the question of which crossover operator is most appropriate, despite several investigations on this topic [12].

2.4.0.4 Selection

Similarly to EAs, selection in canonical GAs is based on a probabilistic survival rule, combined with a fitness-dependent chance to have different partners for producing more or less offspring. For proportional selection $s : I^{\mu} \rightarrow I^{\mu}$, the reproduction probabilities of individuals $\overrightarrow{a}_j$ are given by their relative fitness, that is:

$$\forall i \in \{1, ..., \mu\} p_s(\overrightarrow{a}_i) = \frac{\Theta(\overrightarrow{a}_i)}{\Sigma^{\mu}{}_{j=1}\Theta(\overrightarrow{a}_j)} \tag{2.4}$$

Sampling μ individuals according to this probability distribution should yield the next generation of parents. This mechanism fails in the case of negative fitness or minimization tasks, this is when the scaling function described earlier comes in.

Having described the canonical GAs methodology above, it is time to put it together for a conceptual algorithm:

Interestingly, Back and Schwefel report in [12] that the position of the select operation did not have to be at the beginning; they experimented by placing it at the end and the difference was marginally small.

2.5 Memetic Algorithms

While GAs have been inspired in trying to emulate biological evolution, Memetic Algorithms (MAs) try to mimic *cultural* evolution. Essentially, MAs are a marriage between population-based global search and the local

```
begin
    $t = 0$ ;
    initialize $P(0) = \{\overrightarrow{a}_1(0), ..., \overrightarrow{a}_\mu(0) \in I^\mu\}$ ;
        where $I = \{0, 1\}^l$ ;
    evaluate $P(0) = \{\Phi(\overrightarrow{a}_1(0)), ..., \Phi(\overrightarrow{a}_\mu(0)) \in I^\mu\}$ ;
        where $\Phi(\overrightarrow{a}_k(0) = \delta(f(\Gamma(\overrightarrow{a}_\mu(0)))), P(0))$ ;
    while $(\iota(P(t) \neq true))$ do
        select: $P(t+1) = s(P''(t))$ recombine:
        $\overrightarrow{a'}_k(t) = r'_{\{p_c\}}(P(t)) \forall k \in \{1, ..., \mu\}$ ;
        mutate: $\overrightarrow{a''}_k(t) = m'_{\{p_m\}}(\overrightarrow{a'}_k(t)) \forall k \in \{1, ..., \mu\}$ ;
        evaluate: $P''(t) = \{\overrightarrow{a''}_1(t), ..., \overrightarrow{a''}_\mu(t)\}$ ;
            where $p_s(\overrightarrow{a''}_k(t)) = \frac{\Phi(\overrightarrow{a''}_k(t))}{\sigma^\mu{}_{j=1}\Phi(\overrightarrow{a''}_j(t))}$ ;
        $t = t + 1$ ;
    end
end
```

Algorithm 2: Canonical GA algorithm Skeleton as described by Holland in [14]

search heuristic made by each of the individuals. Genetic programmers normally regard MA as a special kind of GA with a local search implementation, typically hill-climbing.

Early in the history of the application of EAs to real-world problems, it became apparent that canonical GAs, namely ones using a simple binary representation, n-point crossover, bitwise mutation, and fitness proportionate selection could not possibly compete with tailor-made algorithms [17]. This empirical observation resonated well with theoretical and experimental studies on the so-called "baldwin effect" and on "lamarckian evolution" [17] that focused on how *learning* could affect the process of evolution. Therefore, the global search dynamic of EAs needed to be complimented by local search refinement provided by a suitable hybridization using problem-specific solvers including heuristics, and approximate and exact algorithms.

Hence, by means of specialized crossover and mutation operators, sophisticated problem-specific representations, smart population initialization, complex fitness functions, local search heuristics and local and exact algorithms, domain-specific knowledge was added to the EA framework. As of late, Richard Dawkins' concept of "memes" [18] has been picking up speed within the MA literature as they can be thought of as representing "evolvable" strategies for problem solving, thus breaking the mould of a fixed and static domain knowledge captured once during the design of MAs and left untouched afterwards. Therefore, Dawkins' Memes, and their extensions as evolvable search strategies provide a critical link to the possibility of open-ended combinatorial and/or continuous problem solving [17].

2.5.1 Local Search

The global search capacity of the evolutionary part of an MA takes care of exploration, trying to identify the most promising search space regions; the local search part scrutinizes the surroundings of some initial solution, thereby exploiting it in this way. For a vast majority of combinatorial optimization problems and, as it is also becoming more clear in recent research, also for many *continuous* optimization problems, this combination leads to some of the best performing heuristic optimization algorithms [19]. The local search can be integrated within the evolutionary cycle mainly in two ways. The first is the so-called "life-time learning", that is, the application of the local search to a candidate solution. In this case, the metaphor is the cultural development of the individuals which is then transmitted to the other solutions over the subsequent generations. The second way is the application of the local search during the solution generation phase, that is, the generation of a perfect child. This class of Memetic implementations aims at selecting the most convenient offspring amongst the potential offspring solutions [19]. This aim can be achieved, for example, by applying a local search to select the most convenient cutting point in a GA crossover.

2.5.1.1 Neighbourhoods and Local Optima

Essentially, a solution s' is deemed a *neighbour* of s if the former can be reached from the latter by a single step (using a so-called "move" operator) [19]. Moves can typically be regarded as modifications of some parts of a solution. Under an appropriate distance measure between solutions, these moves can thus be seen as "local", hence the name Local Search. There are two things that the reader must keep in mind:

1. Neighbourhoods are – more often than not – *symmetrical.*
2. The move operator allows the implicit definition of neighbourhoods, by referring to the potential transitions attainable upon the application of the operator.

There are some intrinsic differences in combinatorial and continuous search spaces, due to the differences in the types of underlying *search spaces.* Combinatorial spaces are finite for finite size problems, while continuous search spaces are infinite and hence not enumerable [19]. These differences cause the local optima and the way how one is searching for an improved candidate to be different as well (a local optimum is the best solution in its local neighbourhood). We will now explore these differences. First, let S denote the search space.

The number of candidate solutions in the neighbourhood of a current candidate solution s is enumerable in combinatorial problems, and a local optimum can be defined as a candidate solution s^l for which it holds that $\forall s \in \Phi(s^l)$ we have $f(s) \leq f(s^l)$, where $f : s \rightarrow \mathbf{R}$ is the objective function. Since one simply needs to enumerate all neighbouring candidate solutions and check whether they are better or not than the current candidate solution, it's easy to verify whether the current candidate solution is a local optimum or not. This check can be done in polynomial time if the number of neighbours is polynomial in the instance size, and the objective function is computable in polynomial time, which is the typical case for many neighbourhood definitions and optimization problems [19]. Our problem is finite instance is finite, and hence involves *combinatorial* search, and hence continuous search is out of the scope of this thesis. We now will very briefly summarize the idea behind continuous search, for further details [19] contains a much more comprehensive review.

In the case of continuous optimization problems, the decision space is, in principle, a dense set, and is thus composed of an *infinite* amount of points. Therefore, enumeration becomes impossible for the search of the optimum, and cannot be used. We can formally define a local optimum in a continuous space S as a point $s^o \in S$, such that

$$f(s^o) \leq f(s) \ \ \forall s \in S, \ 0 \leq ||s^o - s|| \leq \varepsilon \tag{2.5}$$

The neighbourhood of the local optimum s^o is the set of points encircled in the region limited by the magnitude of ε.

2.5.1.2 Classifications

There are many various perspectives from local search can be classified [19], and based on these classifications one must consider two important points. First, every optimization algorithm can be seen as a logical procedure composed of two sets of operations: trial solution generation and trial solution selection. Second, the classifications should not be considered in a binary way (i.e. it's not one or the other), but more as *properties* of the phases of the procedure. As a concrete example, let's take the idea behind this work into consideration. The algorithm is not fully stochastic or fully deterministic, but instead has a certain degree of stochastic logic and determinism. Furthermore, our algorithm is self-adaptive, meaning it has two behaviours: exploitation and exploration. Hence when the population is diverse it acts like a local search procedure (following the greedy approach), and when the population converges its goal becomes to diversify the search (following an approach that is closer to the Steepest Descent).

1. According to the nature of the search logic:
    - **Deterministic**: Deterministic generation of the trial solution.
    - **Stochastic**: Randomized generation of the trial solution.
2. According to the *amount* of solutions involved:

- **Single-solution**: The algorithm processes and influences only one solution.
- **Multiple-solution**: The algorithm processes and influences more than one solution, usually employed for interacting with and joining generic trial solutions.

3. According to the pivot rule:

- **Steepest Descent**: The algorithm generates a set of solutions and selects the most promising one only after having explored all other solutions.
- **Greedy**: The algorithm performs the replacement as soon as it detects a solution that outperforms the current best and starts over the exploration.

2.5.1.3 Algorithm

The following algorithm provides a general outline of a single-solution meta-heuristic. It receives an initial solution and iteratively picks a neighbour and decides whether or not to accept this neighbour as the new current solution or not. The algorithm may use a memory structure that modulates this process to select which neighbourhood should be used to select the neighbour, whether to accept the latter as the new current solution or not, and even to support some high-level strategy for intensifying or diversifying the search [19].

The possibility of performing some sort of incremental evaluation of neighbours is one of the most distinctive features of local search strategies in

```
begin
    InitializeMemory($M$) ;
    while ($TerminationCriterion(M) \neq true$) do
        $\Phi \leftarrow$ PickNeighbourhoodStructure($M$) ;
        $s' \leftarrow$ PickNeighbour($\Phi, s$) ;
        SELECT($s, s', M$) ;
        UpdateMemory($s, M$)
    end
    return $s$ ;
end
```
Algorithm 3: Typical Single-solution Local Search Algorithm

combinatorial domains. That is, computing $f(s')$ as $f(s') = f(s) + \delta f(s, s')$ where $\delta f(s, s')$ is a term that depends on the influence exerted on s to get s' and can be typically computed in a simplistic and efficient way. More often than not, this means the cost of *exploring* the neighbourhood of a solution is not much higher than a few full evaluations. This allows the practical use of some intensive local procedures [19].

2.6 Approaches to Optical Network Optimization

Memetic and GAs have been applied extensively to solve problems in optical networks. In the following section we review some of the prominent solutions to WDM traffic grooming issues that have been attempted using GAs. Moreover, we also review some MA solutions to more generic problems. At the time of writing this paper, MAs are not very well explored in terms of traffic grooming in WDM optical networks. As such, we instead review some closely related applications of MAs to problems that range from particle swarm optimization to the Travelling Salesman Problem (TSP). Related

Literature is listed and discussed chronologically.

In 1994, Radcliffe et al. introduced a formal, representation-independent form of an MA(i.e. a GA incorporating a local search mechanism) . They claim that, as expected, given the decomposable nature of the evaluation function and the large number ofpossible alleles to the TSP, the MA significantly outperformed the GA. The GA failed by a large margin to match the performance achieved by repeatedly generating 2-opt solutions [21].

Later, in 1999 Gazen et al. proposed a method based on GAs for optimizing the logically re-arrangeable multihop lightwave networks. The algorithm takes topologies as individuals of its population, and tries to find optimal ones by mating, mutating and eliminating them [22]. Although the GA produced high quality solutions to this problem, it required long running times. The authors claim that improvements to the GA are still possible, and that a more compact representation, faster evaluation algorithms and very finely tuned set of parameters will increase the time performance of the GA and the quality of the results. Krasnogor et al. proposed a new hybridization scheme in 2000 for an MA which is composed of two hybridization processes: a GA and a Monte Carlo (MC) Method. They claim the proposed research reached optimal and near optimal molecular confirmations in the Protein Folding problem. They further report that when applied to the Travelling Salesman Problem (TSP), the algorithms did not reach an optimal solution but followed the intended behaviour [23].

To minimize the total network facility cost for the traffic demand at each evolutionary stage of the network, Datta et al. proposed a simulated annealing (SA) approach in 2003 for near-optimal routing of static connections in a mesh-restorable network, and provide a generalized framework for network evolution in large networks with complex demand sets [24]. They show that the SA finds the solution close to the ILP optimal solution. The authors conclude the scheme can be used as a heuristic to arrive to near-optimal solutions in cases of complex demand sets and moderately large networks, where the run-time of the ILP becomes practically infeasible. The proposed framework can be implemented in networks that collect information through link-state protocols and employ source-based routing. The authors note the scheme is highly inexpensive, fast, and can be ideally employed for all backbone networks. The methodology can be extended to heterogeneous networks, wherein one can study the impact of switching architectures on route selections [24].

Meanwhile, Kuri et al proposed a branch and bound (B&B) algorithm for exact resolution and an alternative Tabu search (TS) algorithm for approximate resolution. Furthermore, a greedy graph vertex coloring approach is used to solve the wavelength assignment problem. The authors claim to be able to obtain approximate solutions very close to the optimal ones by modifying the TS parameters. Kuri et al. further report that the time correlation among scheduled lightpath demands (SLDs) in a set Δ has a significant effect on the average gain in WDM channels [8]. Particularly, they have found the number of required wavelengths to be significantly smaller than the number of demands because of the time and space wavelength reuse.

In 2005, Prathombutr et al. considered the grooming problem of static demands as an optimization problem. Specifically, they proposed e a Multiple Objective Evolutionary Algorithm (MOEA) that deals with encoding, routing and wavelength assignment schemes. They claim the algorithm is able to:

1. Maximize traffic throughput.

2. Minimize the number of transcievers.

3. Minimize average propagation delay or average hop counts.

The authors claim that the results showed that the MOEA performed better in any cases than that of the Maximizing Single-hop Traffic (MST) heuristic and the Maximizing Resource Utilization (MRU) heuristic, with the acceptable processing time. Krasnogor began to look into creating MAs with a less complicated foundation, and defined a syntactic model which enables a better understanding of the interplay between the different component parts of a MA. By using the defined syntactic model and taxonomy, the writers claim the process of identifying which of the many components and interactions of these complex algorithms relate to which of those design issues should be facilitated. While this model is *not* applicable to every implementation of a MA, it would certainly be beneficial to keep this model in mind to inform design decisions.

Also in 2005, Tsenov et al. proposed a way for combined use of two non-traditional algorithms by solving topological problems on telecommunications concentrated networks. Specifically, Tsenov suggests simulated annealing (SA) and GA as viable solutions. Results show that “such an approach may lead to good results” [25]. However, more research is needed to investigate the impact of the parameterization of the algorithms. Chen et al. attempted find a routing-tree with minimal multicast cost which satisfies a delay constraint and a destination constraint defined in their previous work in 2002 [26]. The authors report the ILP was able to find an optimal solution for routing the request with fewer than 8 destinations, however an optimal solution for routing the request with more than 8 destinations could not be found in an affordable time [27]. In contrast, the GA can also find the equivalent to the light forest found by the ILP. Specifically, they report the results show that the GA can always find a better solution than 3-Phase Model, but the computation time is still high, and the reduction of computational time will remain a challenge to the GA method.

In 2007, Roy et al. proposed a simple GA that minimizes the number of required Add-Drop Multiplexers (ADM)s based on the shortest path and a possible alternate shortest path. They claim that the distinguishing feature of this algorithm is in introducing a catalyst to direct the solution. Furthermore, the authors state there have been some different approaches to this problem, however there exists no solution that can be applied in general, making all the published solutions too specific to apply to a broader area [28]. Hence they introduced a simple Routing and Wavelength Assignment (RWA) mechanism that aims to minimize ADM employing the shortest path and possible alternate shortest path. The standard deviation of the individuals is taken as a performance index of the generation, and the algorithm converges when the performance index becomes 0. In the case of bi-directional ring networks, the authors were able to achieve 30% reduction of ADMs, and observed that as generations proceed, some better chromosomes appear due to the evolutionary mechanism. However, they were not able to achieve significant results in other cases (such as all-to-all uniform and non-uniform traffic in a uni-directional ring network).

Meanwhile, an indirect encoding EA using a construction heuristic for the Shared-Path-Protection (SPP) problems in WDM optical networks under Shared-Risk Link Group (SRLG) constraints is proposed by Zhang et al.. Experimental results show that the EA/G outperforms the conventional GA in tuning the control parameters, which indicates that the combination of local information and global statistical information can improve the performance of the EA [29]. This work also shows that there is further room for researching the meta-use of EAs to fine tune the parameters of existing programs, however this only applies to problems where the quality of the solution depends on the parameter settings.

In 2008, Chabarek et al. advocated a broad approach to addressing this problem that includes making power-awareness a primary objective in the design and configuration of networks, and in the design and implementation of network protocols. In 2009, Shen et al. note that although at the time of writing this paper the backbone network is only consuming a small fraction of the total network energy, the percentage is perceived to significantly increase with the popularity of bandwidth intensive user applications. Furthermore, because energy consumption of the backbone network is confined to a few buildings, the energy density within these key locations is also an important issue [3]. Thus they developed a Mixed Integer Linear Program (MILP) optimization model and 2 heuristics based on the lightpath bypass model. Experimental results show the strategy of lightpath bypass can significantly cut power consumption over non-bypass designs, ranging from 25% to 45%.

Meanwhile, Huang et al. report that power aware networking is not well explored, and they intend to address this problem by *grooming* traffic [30]. Hence, they provide formulations for green optical network design and show a simple algorithm working at one of the layers identified by the model. Moreover, Yetinger et al. investigated the grooming problem from a power consumption perspective and develop a formulation which combines the objectives of minimizing the number of lightpaths and electronically routed traffic. The authors report that the results obtained suggest that minimizing the number of lightpaths or amount of traffic switched alone may be inefficient in terms of overall power consumption even for a small network, and a power-aware grooming strategy may help reduce the power consumption of optical networks significantly for low to moderate traffic loads, which is actually the operating regime for most of today's real world networks [31].

Bathula et al. propose a multi path selection approach to minimize the energy consumption of the optical core network. These wavelength routed paths may have to forgo minimum distance paths and choose a path which is at a larger distance [7]. At the same time, Idzikowski estimated and compared the potential energy savings of three different approaches to make line cards idle by reconfiguring the routing at the Internet Protocol (IP) and/or WDM layer [5]. Idzikowski et al. argued that their work indicates that energy aspects should be included in daily IP routing reconfigurations done by network operators. Furthermore, they argue that it should also motivate equipment vendors to provide line cards with a convenient and fast functionality to be switched on and off.

Finally, in 2011 Coiro et al. considered a circuit-switched WDM optical network, and based on topological and power consumption considerations as well as on-link considerations proceeded to propose several link-ordering criteria, and applied them to an optical link switch off algorithm. The authors aimed to minimize the energy consumed by WDM optical links by reducing the number of fibers powered on into the whole network.

Chapter 3

Energy Efficient Grooming in Optical Networks

3.1 Introduction

In this chapter, we first define our problem, and then describe our two approaches to solving the problem of energy efficient grooming of sub-wavelength traffic demands in optical networks. Specifically, we developed a canonical GA and further transposed the GA into an MA in order to quickly and efficiently route a set of periodic, sub-wavelength traffic demands over the network and show that consideration of demand holding times *can* play an important role in reducing the overall energy consumption of the network.

Hence, we present two techniques:

1. A new Canonical GA.

2. An MA built on top of the GA that improves the performance and

results presented by the GA.

We have shown, through simulations, that both approaches can be used to handle larger networks with many demands – situations in which ILPs would become computationally intractable and not practical – and lead to significant improvements in resource utilization, compared to the holding-time-unaware(HTU) techniques (algorithms in which it is not known how long a network resource (typically a lightpath) is used, or kept "on" for). While the GA approach is able to further reduce the energy consumption over the simple shortest path holding time aware (HTA) heuristic [9] presented in our previous work in 2012. Furthermore, the proposed MA slightly outperforms the GA in terms of energy reduction, however, it significantly outperforms the GA in terms of computational resources, reducing the computational time needed by at least 2 orders of magnitude.

3.2 Problem Definition

Before we discuss our proposed canonical GA and the further proposed MA, we must first formally define our problem. Suppose we have a logical topology of a small network with four end nodes and four logical edges (i.e. lightpaths), represented by circles and solid lines, respectively, as shown in Figure 3.1a.

Demand $q_1(q_2)$ is active and routed over lightpaths l_1 and l_2 (or l_3 and l_4). As shown in 3.1b, θ_i, and ω_i represent the start and end time of demand q_i, and are used to partition the entire time period into a number of consecutive time intervals $i_1, i_2, ..., i_max$ (with $i_max = 4$ in our case). The bandwidth

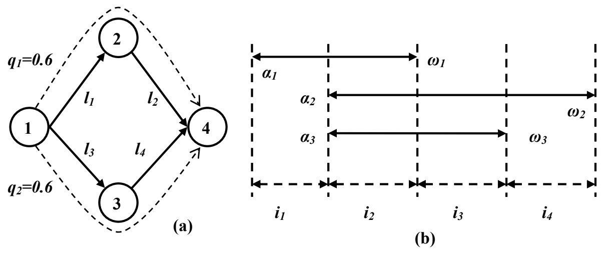


Figure 3.1: a) Logical topology and Traffic routing. b) Overlapping demands.

requirement for each demand is expressed as a fraction of the lightpath capacity. Now, suppose a new demand – q_3 – with a bandwidth requirement of 0.3 needs to be routed from node 1 to node 4, starting at interval i_2. There are only two ways to do this:

- Combine q_3 with q_1 on to lightpaths l_1 and l_2, or
- Combine q_3 with q_2 on to lightpaths l_3 and l_4.

When considering the demand holding times, we note that option 1 will require *more* energy, as both l_1 and l_2 need to remain *active* for one extra interval (up until the end of i_3). In contrast, however, if q_3 is routed over l_3 and l_4, then l_1 and l_2 can be switched *off* at the end of interval i_2. Thus, by selecting an appropriate route for each demand, power consumption can be greatly reduced by turning *off* the transponders corresponding to a lightpath when it is not carrying any traffic. For a medium-large network, there may be hundreds or even thousands of individual demands, and *many* possible paths between each pair of nodes. Hence, it is necessary to develop efficient techniques to determine a route for each demand such that the overall energy consumption – as well as the number of lightpaths needed to implement each logical edge – are reduced as much as possible.

3.3 Network Model

Our experimentation and tests were done on 10-node, 14-node and 20-node network models. The 14-node network is to emulate the 14-node National Science Foundation Network (NSFNET) as in Figure 3.2. Initially created to link researchers to the nation's NSF-funded supercomputing centres, through further public funding and private industry partnerships it developed into a major part of the internet backbone.

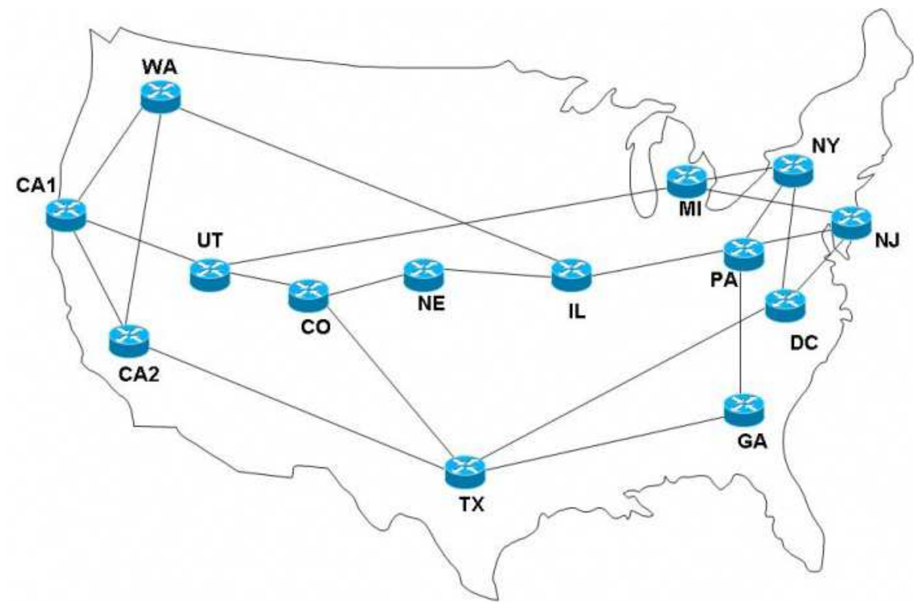


Figure 3.2: 14-node 21-link NSFNET

The network operates in connectionless mode using the Internet Protocol(IP)[32] as the basic networking mechanism. End-to-end reliability is maintained using the Transmission Control Protocol(TCP)[33], which assembles and reorders datagrams received over possibly diverse and unreliable paths using retransmissions as necessary. The User Datagram Protocol (UDP)[34] provides direct IP datagram access for transaction services, including routing and network control in some cases.

The 20-node network is to emulate the 20-node Advanced Research Projects Agency Network (ARPANET) as in Figure 3.3. The world's first operational packet switching network, it was the first ti implement TCP/IP, and the progenitor of what was to become the global internet.

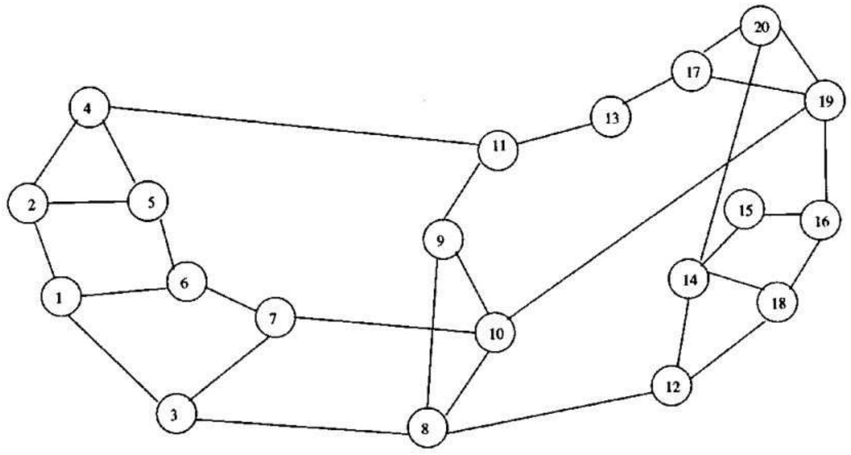


Figure 3.3: 20-node 32-link NSFNET

Our algorithms both take the same two inputs: the network topology and the demand matrix. The first line contains text in the form (n e), where n is the number of nodes and e is the number of edges. Let N be the network topology, we can represent each connection in the network topology in the form of (n_i, n_j), where n_i is the source node and n_j is the destination node.

On the other hand, the demand matrix is inputted similarly, however the first line contains (d n), where d is the number of demands, and n is the number of nodes. The rest of the file contains the demands in the form of the tuple. A demand $q \in \mathcal{Q}$ is represented by a tuple $(s_q, d_q, n_q, \alpha_q, \omega_q, \tau_q)$, where s_q and d_q are the source and destination respectively, n_q represents the bandwidth requirement for the demand, α_q, ω_q are the start and end times of the demand, and τ_q is the demand holding time.

3.4 Genetic Algorithm

As discussed in section 2.2 and 2.4, a GA (GA) is a technique that is based on the evolution theory for difficult solving optimization and search problems [35, 36, 22]. The general idea behind GAs is that we can build a better solution if we somehow combine the "good" parts of other solutions, just like nature does by combining the DNA of living beings, producing new offspring or "generations". It has been applied in a wide range of studies in solving optimization problems, especially problems that are not well-structured and interact with large numbers of possible solutions. To ensure that the new population is at least as fit as the previous generation, the best performing chromosomes from the previous generation can replace the poorest performing chromosomes of the current generation, a process called Elitism [37]. The algorithm terminates once a termination criterion is met. The steps of a standard GA [27] are outlined in Algorithm 4.

```
Input: Population
Output: New, elitist population
$Population_\alpha$ = GeneratePopulation()
compute_fitness_of_individuals($Population_\alpha$)
while (criterion == True) do
    $Parent_\beta$ = selectParent($Population_\alpha$)
    doCrossover($Parent_\beta$)
    mutate($Parent_\beta$)
    $Offspring_\gamma$ = compute_fitness_of_individuals($Parent_\beta$)
    repopulate_with_offspring($Offspring_\gamma$, $Population_\alpha$)
end
```

Algorithm 4: Genetic algorithm

3.4.1 GA Based Energy Minimization for Scheduled Traffic

The primary goal of the proposed GA is to route the traffic demands in such a way that the maximum number of lightpaths can be switched *off* at any given time, thus reducing the overall power consumption. Another objective is to reduce the total number of lightpaths needed to realize the logical topology, such that the capacity constraints of the lightpaths are not exceeded. In other words, we try to implement each logical edge using as few lightpaths as possible, which in turn reduces the need for costly optical transceivers. In the following sections we define our chromosome representation, specify the initial population, describe the fitness function, and discuss the strategies and validity for crossover and mutation for our proposed GA approach.

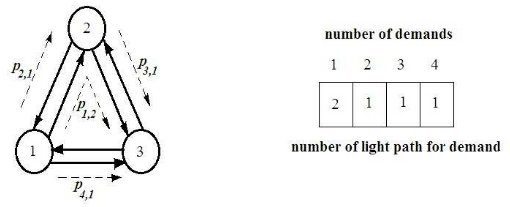
3.4.1.1 Chromosome representation

For each demand q to be routed, we pre-compute a set of up to k paths over the logical topology, where $p_{q,k}$ is the k^{th} potential path for demand q. We

Table 3.1: Potential paths for scheduled demands

demand (q)	$p_{q,1}$	$p_{q,2}$
$1(1 \to 3)$	$1 \to 3$	$1 \to 2 \to 3$
$2\ (1 \to 2)$	$1 \to 2$	$1 \to 3 \to 2$
$3\ (2 \to 3)$	$2 \to 3$	$2 \to 1 \to 3$
$4\ (1 \to 3)$	$1 \to 3$	$1 \to 2 \to 3$

represent the chromosome as an array of integers, specifying the selected path for routing each demand. So, the length of each chromosome is equal to the number of demands, and the integer in position q indicates the path along which demand q will be routed. For example, let us consider the simple topology with 3 nodes and 6 logical edges (shown as solid arrows), in Fig. 3.4(a). There are 4 demands to be routed over the network, and we pre-compute $k = 2$ potential paths for each demand. Table 3.1 shows the two potential paths for each demand. The chromosome in Fig. 3.4(b) indicates that demand 1 is routed using the second pre-computed path for that demand (i.e. along the path $p_{1,2} = 1 \to 2 \to 3$). Similarly, demand 2 is routed using the first pre-computed path ($p_{2,1}$) for that demand and so on. Based on the information in Table 3.1, the routing corresponding to the chromosome in Fig. 3.4(b) is shown (using dashed lines) Fig. 2 (a).



(a) logical topology and s-d pair routing *(b) a chromosome representation for 4 demands*

Figure 3.4: Chromosome representation for a given traffic routing.

3.4.1.2 Initial population

Each chromosome in the initial population specifies a single valid path (from a set of k potential paths) for each demand. The potential paths are calculated beforehand, using a modified version of Dijkstra's algorithm [38]. However, our GA does not depend on the particular algorithm used to generate the potential paths for the initial population, and any suitable routing algorithm can be used. Chromosomes for the initial population are generated by randomly selecting a path for each demand. The initialization ensures the validity of the paths as genes in the chromosome, but does not consider the energy consumption of the optical network.

3.4.1.3 Fitness function

After generating each new individual, it is necessary to evaluate its fitness value. The fitness function for a chromosome consists of two components, as shown in eqn 3.1.

$$fitness = \sum_{i} T_i \sum_{l \in L} n_{l,i} + a \sum_{l} n_l \tag{3.1}$$

The first term in eqn 3.1 represents the total energy consumption for the logical topology. Here $n_{l,i}$ represents the number of active lightpaths needed for logical edge l during interval i and T_i is the duration of interval i. So, if the capacity of a single lightpath is $OC-192$, and the total traffic traversing logical edge l in interval i is $OC-220$, then 2 lightpaths need to be active during interval i to accommodate the traffic and $n_{l,i} = 2$. Subsequently, in another interval j, if the total traffic on l is reduced to $OC-180$, then only one lightpath needs to be activated during that interval and $n_{l,j} = 1$. In order to obtain more energy efficient grooming, we try to minimize the number of *active* lightpaths at each given time interval, which in turn reduces the overall static power consumption. The second term in eqn 3.1 attempts to reduce the overall (transceiver) cost for the logical topology, by minimizing the number of lightpaths n_l needed to implement each logical edge l. We have seen that the number of active lightpaths $n_{l,i}$ for l can vary, depending on the traffic in interval i. So, the number of transceivers used for implementing l is determined by the *maximum* number of active lightpaths needed for l in any given interval, i.e. $n_l = max\{n_{l,i} | i = i_1, i_2, \ldots, i_{max}\}$. Finally a is a constant (weight) representing the relative cost of adding a new lightpath compared to increasing the energy consumption. Since our goal is to reduce energy consumption and transceiver cost, lower fitness values indicate 'better' chromosomes.

3.4.1.4 Selection, crossover and mutation

Selection of individuals from the initial population as parents is carried out using the Roulette-Wheel selection method [35, 36], where chromosomes with better (lower) fitness values are more likely to be selected for crossover. The routine to generate a member of the initial population can be summarized as follows:

1. Remove a chromosome e from the list of network demands.

2. Determine if it is legal to place e into the search subspace Π, as defined by $\Pi_{i=1}^{n}[u_i, v_i] \subset \mathbf{R}^n$ with $u_i < v_i$. If it is good, then calculate how

"good" it would be using the following:

$$\frac{numCommon + Size + 1}{penalty + 1} \tag{3.2}$$

where $numCommon$ is the number of neighbours that e has in common with the solutions already placed in the specified subspace. Meanwhile $Size$ is the number of solutions already in the subspace, and $penalty$ is the cost (as used in the evaluation function) of placing e in that subspace. The implications of this heuristic are two-fold. First, to place the current solution within the neighbourhood of similar solutions to it. Second, it tries to minimize the penalty caused by placing e by biasing the roulette wheel to those subspaces which can accommodate e with lower penalties.

3. Construct and execute a roulette wheel

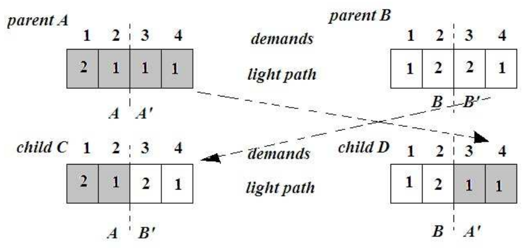


Figure 3.5: Example of single-point crossover.

To produce new offspring (children) from the selected parents, we have used $k - point$ crossover($k = 1, 2, or 3$,selected randomly) for each crossover operation [39]. Fig. 3.5 shows an example of single-point crossover, with two parent chromosomes, *parentA* and *parentB*, for the network and demand set shown in Fig. 3.1(a). Two routing schemes corresponding to the two new child chromosomes, *childC* and *childD*, are shown in Fig. 3.5. As discussed earlier, a value of r in the q^{th} position indicates that the r^{th} pre-computed route $p^{q,r}$ (given in Table 3.1) is used for demand q.

The crossover operation does not create any new gene that is not present in the initial population. Mutation can be applied after the process of crossover to attempt to improve the fitness value of an individual. Mutation is performed in each round after crossover has been completed. First a single candidate chromosome is selected randomly for mutation (all chromosomes have an equal probability of being selected). Then a particular gene in that chromosome is selected (again randomly), and the specified route is changed to a different value.

We note that an important feature of our GA is that both the crossover and mutation operations are guaranteed to generate valid chromosomes. So, there is no need need to "repair" the resulting chromosomes. This is because each pre-computed route for a demand is a valid path from the source to the destination over the given topology. Since multiple lightpaths may be used, as needed, to implement a logical edge, capacity constraints will never invalidate the set of selected routes corresponding to a chromosome (although it does affect the 'fitness' of the chromosome).

3.4.1.5 Termination condition

In each iteration a new generation of chromosomes is created, until a termination condition is met. Common termination conditions include:

1. A solution is found that satisfies the minimum criteria.
2. A fixed number of generations is reached.
3. An allocated budget (computational time/money) is reached.
4. The highest ranking solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results.
5. Manual Inspection.

We terminate the GA after a fixed number of generations (500 generations in our simulations). This value was determined by trial and error, after observing that the highest ranking solution's fitness reaches a plateau by 500 generations.

3.5 MAs

Particularly, the main difference between MAs and GAs is that MAs are essentially GAs which were modified to use some kind of interaction with local searchers. From an optimization point of view, MAs have repeatedly shown that they are orders of magnitude more accurate than canonical GAs for certain problem domains. It is generally argued the trade-off between the exploration abilities of the underlying GA and the exploitation abilities of the local searchers used is the main reason behind the success of MAs.

The catch for this is a greater number of fitness evaluations, and often a loss of diversity within the population, however local searchers and MAs are now being designed to maintain diversity within the population while trying to obtain an energy efficient solution. With regard to the mutations, as we can see there is a 15% chance of mutation at each iteration.

The overall pseudo-code of the MA is hence as follows:

```
begin
    Initialize Population $Parents$ ;
    while $(TerminationCriterion \neq true)$ do
        LocalSearch $(Parents, P_{ls})$ ;
        $MatingPool$ = SelectMating($Parents$) ;
        if $random(0,1) \leq 0.08$ then
            $randomChrom$ = random($0, Parent$) ;
            Mutate($randomChrom$) ;
        end
        $Offsprings$ = doCrossover($MatingPool$) ;
        $Parents$ = Select($Parents, Offsprings$) ;
    end
end
```

Algorithm 5: MA Pseudo-code

In this strategy, the $Select()$ procedure is a (μ, λ) or a $(\mu + \lambda)$ selection strategy, with the +-strategy having the highest pressure, and the ,-strategy having the lowest pressure. $Select_mating()$ is a roulette wheel selection method. A given individual can be modified several times during its life span either by local search or by mutation in the case of +-strategy, because the strategy allows an individual to persist. The best individual is never modified by the local search method.

As we can see from the algorithm, it is a very similar structure to our GA discussed above, with the main difference being the use of a local search method before every iteration. In order to more technically define MAs, let's consider a search space S (of phenotypes) and a representation space C (of genotypes), and let $p : S \to C$ be the representation function which, given any solution in search space S, returns the chromosome in C that represents it. Let f be the fitness function, which would be convenient to regard as the mapping $f : C \to \mathbf{R}^+$. Our aim is to maximize fitness, and the set of global optima will be denoted by $C^* \subset C$.

Let Q be a stochastic unary move operator over C. It would be convenient for the moment to accommodate the stochastic element of such an operator through a control set K_Q, from which a *control parameter* will be drawn to determine which of the *possible* moves actually occurs. For an example, in the case of binary strings a binary mask might be used as the control parameter with the presence of a 1 at position i indicating that the i-th bit should be mutated. This makes the functional form for Q:

$$Q : S \times K_Q \to S \tag{3.3}$$

A chromosome $x \in C$ will be said to be *locally optimal with respect to Q or Q-opt* if no chromosome of higher fitness than x can be generated from it by a single application of Q, that is, if and only if

$$\forall k \in K_Q \ : \ f(Q(x,k)) \leq f(x) \tag{3.4}$$

then let $C_Q \subset C$ be the set of Q-opt chromosomes in C, that is

$$C_Q \equiv \{x \in C | x \equiv \text{Q-opt}\} \tag{3.5}$$

A GA applied to the task of optimizing f over C has some goal such as finding some or all optima in C^* or making rapid improvements towards more fit chromosomes. It is trivial that for any move operator Q, all chromosomes in C^* are Q-opt, and therefore $C^* \in C_Q$. This makes it sufficient to formulate the search instead over C_Q.

Given a representation space C, a move operator Q and the subspace C_Q of local optima as above, define a *hill-climber* (local search paradigm) to be an stochastic, parametrised operator that, given a chromosome $x \in C$, returns a local optimum in C_Q. Therefore a hill-climber H with control set K_H is any function

$$H \; : \; C \times K_H \rightarrow C_Q \tag{3.6}$$

Note that there is *no* requirement that the solution returned be in any sense "near" the starting solution, although this will often be the case in practice [20]. As we have already mentioned, GAs produce new chromosomes by recombination of two parents followed by some small level of mutation, so that if

$$X : C \times C \times K_X \rightarrow C \tag{3.7}$$

is the recombination operator, with a control set K_C, and

$$M : C \times K_M \rightarrow C \tag{3.8}$$

is the mutation operator, with a control set K_M, the combined generating reproductive function $\mathbb{R}_g$ would typically be given by the composition of mutation and recombination $\mathbb{R}_g = M \circ X$, yielding:

$$R_g \;:\; C \times C \times K_M \times K_X \to C \tag{3.9}$$

defined by:

$$R_g(x, y, k_M, k_X) \equiv M(X(x, y, k_X), k_M) \tag{3.10}$$

However, if R_g is further composed with a hill-climber H (with respect to some unary move operator Q), and restricted to C_Q, a memetic reproduction function $R_m \equiv H \circ M \circ X$ results in:

$$R_m \;:\; C_Q \times C_Q \times K_H \times K_M \times K_X \to C_Q \tag{3.11}$$

defined by:

$$R_m(x, y, k_H, k_M, k_X) \equiv H(M(X(x, y, k_X), k_M), k_H) \tag{3.12}$$

3.5.1 Local Search

Let us first take a simple example in order to simplify the way Local Search works. Suppose we have a population P of chromosomes – or candidate solutions – and we divide this population into N neighbourhoods of chromosomes, as shown in Figure 3.5.1. We can define a "neighbour" of a chromosome via any criteria that suits us, for the purposes of this thesis, we simply considered adjacent chromosomes to be the neighbours. Note that we can use an optimization algorithm to find out the best way to choose the

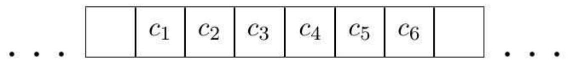


Figure 3.6: A neighbourhood of Chromosomes or Candidate Solutions.

neighbours, however this is beyond the scope of this work. The chromosomes are referred to as $(c_1, c_2, ..., c_n)$ where n is the number of chromosomes in that neighbourhood. When local search is implemented, it searches through the neighbourhoods within population P and chooses the most *locally* optimal chromosome from each neighbourhood. As we have mentioned in section 2.5.1, the local search can be integrated within the evolutionary cycle mainly in two ways. The first is the application of the local search to a candidate solution, called lifetime learning. In this case, the metaphor is the cultural development of the individuals which is then transmitted to the other solutions over the subsequent generations. We have implemented our method using the second way, which is the application of the local search during the solution generation phase, that is, the generation of a perfect child. This class of Memetic implementations aims at selecting the most convenient offspring amongst the potential offspring solutions [19]. Hence this way we make sure that our most locally optimal or near-locally-optimal solutions always make it to the next generation, sharply reducing our time needed to find the most near-optimal solution, as well as giving us better results than the canonical GA.

The local search and diversification process is described in Algorithm 6. Parents is a set of solutions to which local search will be applied, with probability P_ls. The self adaptation of the local search to either exploitation or exploration behavior is governed by the *adapt* variable. This variable determines the degree by which uphill moves will be allowed. *adapt* is inversely proportional to the spread of fitnesses within the population, when the latter converges, the former rises. One consequence of this is that each individual in the population will become more “nervous”, and try to move away from its initial position, thereby forcefully exploring the search space. Eventually, the fitnesses will spread, lowering the population adaptation rate. The best fitness is always maintained as we do not allow the modification of the best individual via local search. This process of adaptation, exploration, exploitation and acception of solutions is achieved via our $ApplyMove$ subroutine described in Algorithm 7.

```
begin
    $adapt = \frac{1}{|maxFitness - minFitness|}$ ;
    $size = sizeOf(Parents)$ ;
    for $i = 0; i < size$ do
        $chrom = Parents[i]$ ;
        if $(p_{ls} \geq random(0,1)) \wedge (chrom\ \neg bestSolution)$ then
            ApplyMove($chrom$) ;
        end
        $i{+}{+}$;
    end
end
```

Algorithm 6: Local Search Procedure

3.5.2 Mutation

The mutation model that we have implemented is the same one implemented in canonical GAs. Specifically, we flip a single bit of an individual, with a P_ls probability of this event happening. Via rigorous trial and error, we have

```
begin
    $previousFitness$ = fitness($chrom$) ;
    Modify($chrom$) ;
    $nextFitness$ = fitness($chrom$) ;
    if $previousFitness > nextFitness$ then
        Accept Configuration, solution is good ;
    end
    else
        $delta_e = nextFitness - previousFitness$ ;
        $threshold = e^{-k} \cdot \frac{delta_e}{adapt}$ ;
        if $random(0,1) < threshold$ then
            Accept configuration, even if worse than previous one ;
        end
        else
            Reject any changes ;
        end
    end
end
```

Algorithm 7: Apply Move Procedure

```
Move(initialPosition, size)
begin
    $newPosition = random(0, initialPosition)$ ;
    for $i = 0; i < numberOfDemands$ do
        $temp[0][i] = chrom[initialPosition][i]$;
        $i++$;
    end
    for $i = 0; i < numberOfDemands$ do
        $chrom[initialPosition][i] = parent[newPosition][i]$ ;
        $i++$;
    end
    for $i = 0; i < numberOfDemands$ do
        $chrom[newPosition][i] = temp[0][i]$ ;
        $i++$;
    end
end
```

Algorithm 8: Move procedure

arrived at a probability of 8% being the best mutation rate, as it does not converge the population too quickly, nor does it ruin the population diversity. This kind of mutation is ruled by pure randomness, and hence it does not depend on the number n of object variables, nor on the length l of the bit string. On a single individual, mutation $m'_{p_m} : I \to I, m'_{p_m}(s_1, ..., s_l) = (s'_1, ..., s'_l)$ works as follows:

$$\forall i \in \{1, ..., l\} s'_i = \begin{cases} s_i & \chi_i > P_m \\ 1 - s_i & \chi_i \leq P_m \end{cases}$$

Where P_m is the probability of mutating a bit in the individual. For our purposes we chose $P_m \approx 1 \cdot 10^{-3}$. Let us take a simple example to further explain our Mutation mechanism. Suppose we have a population P. We choose a random chromosome, P_r, to be mutated. The basic structure of chosen chromosome is an array of integers, specifying the selected path for routing each demand.

Hence, Figure 3.7 shows a typical chromosome, containing:

- Demand 1, path 1.
- Demand 2, path 2.
- Demand 3, path 3.
- Demand 4, path 2.
- Demand 5, path 1.

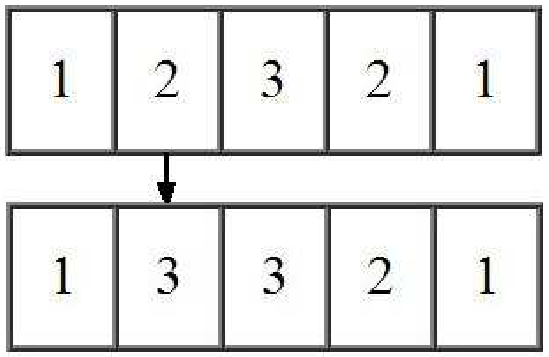


Figure 3.7: Chromosome Mutation within the population.

We then randomly select a gene, and set a new, “mutated” value from the potential lightpaths that can be routed correspondingly to demands. For example, if our random choice lands on the demand 2, we can change the path that it is routed on, as in Figure 3.7. In this case it was rerouted to path 3.

Below, we show our algorithm used to achieve mutation, where $Position$ is a random position chosen before calling the function where $Position \in [0, Parent]$, and hence can specify any chromosome in the parent population:

```
*Mutate(Position)*
begin
    $randomBit = \text{Random}(0, numberOfDemands)$ ;
    $randomPath = \text{Random}(1, k)$ ;
    $Offspring[random_bit][random_path] =$
    $(demands[random_bit][index] \cdot k) + n$ ;
    recalculateFitness($Position$) ;
end
```

Note that k specifies the number of paths chosen when invoking the algorithm, between 3 and 5. Hence the algorithm chooses a random chromosome and a random bit (or allele) in that chromosome, and changes its value according to the path and the random path chosen, thereby "flipping" its bit, in a sense. The reason for this is that the data format for our input is not specifically 1's and 0's, and hence we were required to come up with a simplistic method of changing the bit value randomly, for better or for worse. However, the randomization algorithm uses the uniformly distributed Mersenne twist method, and hence we can – to some degree – guarantee the same probability for all chromosomes we specify from a subspace of chromosomes to be chosen. Hence we can specify a subspace of "good" chromosomes to be mutated, and reinserted into the population.

Chapter 4

Experimental Results

In order to assess our GA based approach, we have run simulations with different demand sets on a number of well known networks such as the 14-node NSFNET and 20-node ARPANET [40]. For each network topology and size of demand set, the results reported in this section represent the average values of at least five runs. We also experimented with different values of the constant a ($a = 0$, 3, 10, and 20) in the fitness function of eqn. 3.1. We have found that the changing the value of a did not produce a significant change in the results, so we have reported the results corresponding to $a = 10$ in this section; results for other values of a follow a very similar pattern. The experiments for the GA were run on a Amazon EC2 Virtual “Elastic Cloud” server, with 8GB of RAM memory and 4 Amazon EC2 Compute Units (with each Compute Unit being equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor). However, the MA experiments, although using the same data sets, were run on a small 2GB RAM memory server, utilizing 1 Amazon EC2 Compute Unit.

Simulation results clearly demonstrate that knowledge of demand holding times result in significant savings over HTU approaches. Furthermore, the proposed GA based algorithm leads to additional improvements, compared to a HTA shortest path heuristic. Furthermore, they demonstrate that by using an MA we can reduce the amount of computational resources and time used while further improving upon the solutions presented by the GA.

4.1 Energy consumption within Genetic Algorithm

Fig. 4.1 shows the overall energy consumption for different networks, normalized to the energy consumption for the HTU case. It is clear that knowledge of demand holding times significantly reduces energy consumption (26% - 40%), even using a simple shortest path routing approach. This reduction is achieved by simply switching off lightpaths when they are not carrying any traffic. Additional improvements of 8% - 13% are then achieved using our proposed GA, even compared to the holding time aware shortest path approach.

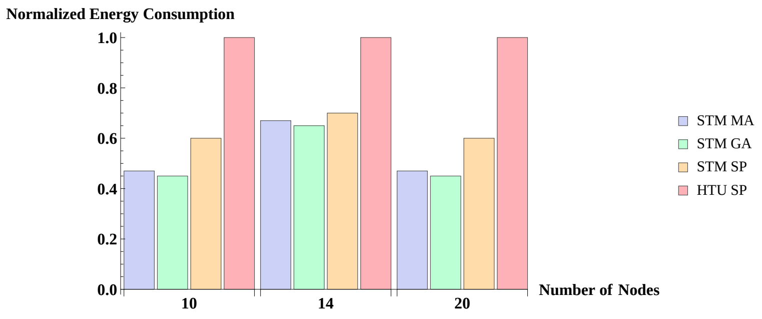


Figure 4.1: Comparison of the energy consumption for different approaches.

However, no significant additional improvement in terms of energy consumption was achieved by the MA. The significance of the MA does not come from the improvement of its results compared to the GA. Rather, the improvement comes from the time taken to achieve these results by the MA, which we will discuss at the end of this chapter.

4.2 Number of lightpaths

The objective of our GA algorithm was not only to minimize energy consumption, but also to reduce the total transceiver cost for the network by minimizing the number of lightpaths needed to implement each logical edge. The second term of the fitness function addresses this criterion. Fig. 4.2 shows the total number of lightpaths needed to construct the logical topology capable of handling all traffic demands. Knowledge of demand holding times reduces the number of lightpaths required, by allowing reuse of WDM channels by non-overlapping demands. The proposed GA outperforms holding-time-aware (HTA) shortest path routing, and HTU case by an average of 15% and 17% respectively. The MA achieved the same number of lightpaths needed as the GA.

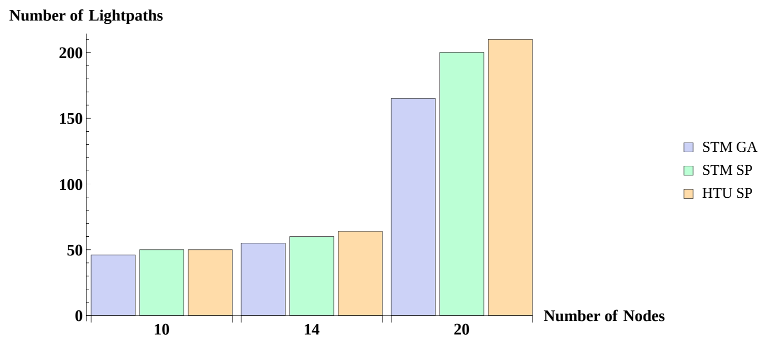


Figure 4.2: Comparison of the number of lightpaths needed for different approaches.

4.3 Numerical Results

In this section, we present our numerical results obtained for the canonical MA as well as the MA. a is the weight of a new lightpath, we have tested for $a = 0, 3, 10$, and 20, we have also set the bandwidth capacity of the optical fiber to be $G = 160$Gbit/s. For each value of A in our figures, we took the average of five test cases, and plotted the improvement percentage against the value of A. Figure 4.3 shows the average of 5 data sets of 10 node networks. From the average between the initial population and the final population, we see an initial improvement of about 14% in energy efficiency when $a = 0$, where a is the weight cost of a new lightpath. As a increases, we see an increase to 15% when $a = 10$, and finally we see a decrease to 15% when $a = 20$.

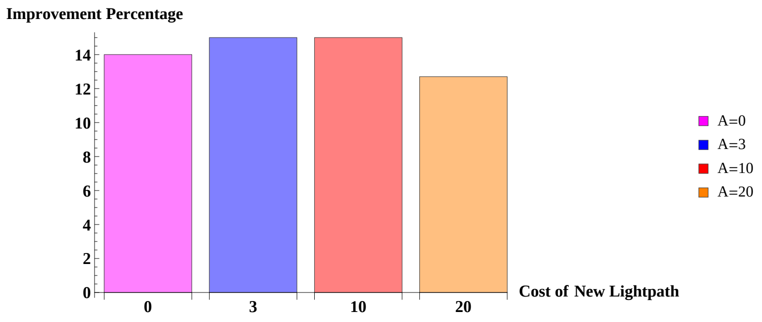


Figure 4.3: Average Results of GA on 10 Node Network.

In the case of the 14-node architecture we used the same testing conditions. Figure 4.4 indicates the algorithm performed a little less efficiently, in which it showed a 15% initial improvement, and went lower as the cost of a increased. However, the improvement did not go below 10%. Therefore we can safely assume that the efficiency will not go lower than 10% when dealing with a network like the NSFNET.

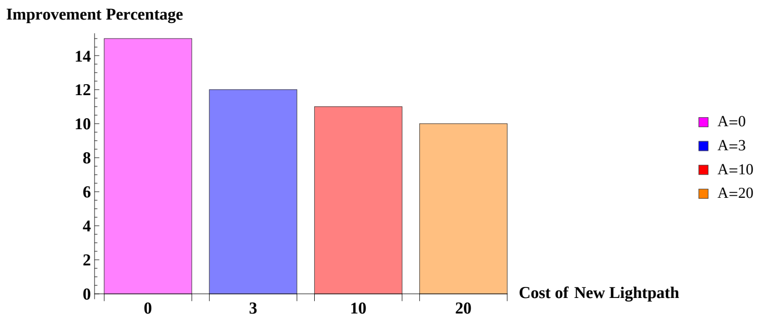


Figure 4.4: Average Results of GA on 14 Node NSFNET.

Finally, in the case of the 20-node ARPANET architecture, Figure 4.5 show an initial improvement of approximately 12%. When $a = 3$, we see an improvement of 10%. However, the improvement rises to 12% when $a = 10$ and subsequently to 13% when $a = 20$. Hence, our canonical GA is a good improvement and replacement to ILPs as it takes less time and at the same time achieves as high as a 13% - 15% efficiency improvement on large networks such as the 20-node ARPANET.

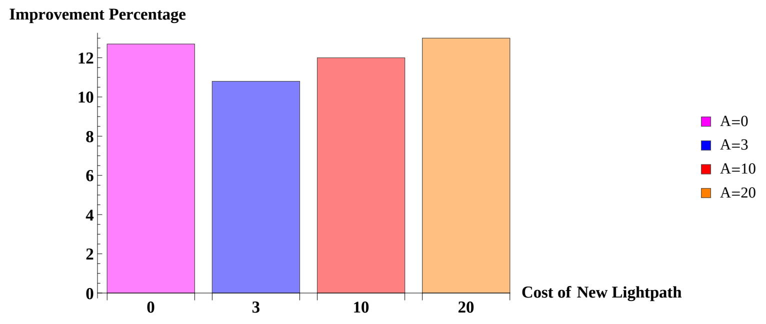


Figure 4.5: Average Results of GA on 20 Node ARPANET.

On the other hand, Figure 4.6 shows the results from our proposed MA applied to the same dataset. As we can see, the results are the same or better than the GA, with a 12% to a 15% improvement rate. The running time for all five test cases was 5 hours.

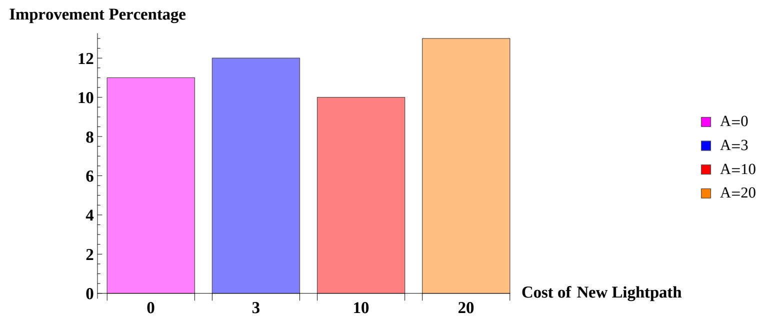


Figure 4.6: Average Results of MA on 10 Node Network.

Figure 4.7 shows the results for the 14 node NSFNET architecture. Similarly to the GA, there is a downward trend in terms of efficiency as the value of A goes up. The average efficiency is between 8% and 13%. However, the

GA took approximately 45 hours to run on a 14 node topology, while the MA took approximately 11.5 hours to produce these results, which are generally as good as or better than the GA.

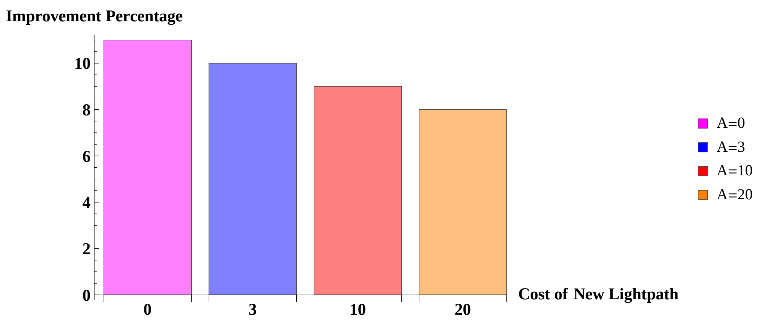


Figure 4.7: Average Results of MA on 14 Node NSFNET.

Finally, Figure 4.8 shows the results of the 20 node ARPANET architecture. Curiously however, it goes opposite to our 14 node case in Figure 4.7. Starting out at approximately 5% improvement, and going up to as high as 12% efficiency as the weight of A, the cost of adding a new lightpath, increases. Hence, the more expensive placing lightpaths becomes, the more energy efficient the results become in the case of the ARPANET architecture.

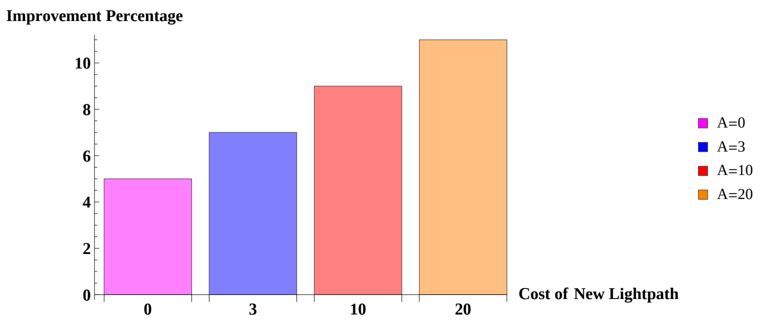


Figure 4.8: Average Results of MA on 20 Node ARPANET.

As the figures and the results show, there is approximately a 10% to 15% fluctuation of improvement in terms of the GA and the MA. These fluctuations can be attributed to the randomizations within the algorithm, such as mutation rates, local search probabilities, and chromosome positioning – which we chose to be random for the purposes of this thesis. There is much room for research in terms of optimizing these parameters to achieve the best results.

It must be noted, however, that at the time of writing this paper, there are no known techniques that can generate exact solutions for practical networks with large demands. In future work, it is possible to create an ILP in order to have an optimal solution benchmark to compare against, or even to brute force small instances of this problem and compare our proposed solution.

4.4 Chronological Analysis

Although our GA achieved an improvement between 8% - 13% of energy reduction compared to the holding time aware and unaware shortest path approaches, it took a significant amount of time to achieve so. Indeed, as the number of nodes in the network increased, the time taken by the algorithm to compute the solutions increased as well. However, the proposed MA shows its significance over the GA especially over the time taken to achieve results that are also more energy efficient than those achieved by the GA. The graph in figure 4.9 highlights the difference in computational time between the two algorithms.

Table 4.1: Computational times for MAs and GAs

Nodes (n)	GA	MA
10	20 hours	5 hours
14	45 hours	11.4 hours
20	80 hours	20 hours

In Figure 4.9 we plotted the nodes (x-axis) vs the amount of time taken in hours (y-axis), as the number of nodes increases, the time needed for the GA to finish computing the solutions sharply increases. Although GAs still a significant amount of time less than an ILP would, they can still take a considerable amount of time to compute nonetheless. This sometimes happens during the selection process, as it becomes time-consuming to compute the fitness for each individual every cycle [41], however there have been proposed works to circumvent this issue [42, 43, 44]. A comparison is made in terms of computational time required to run each algorithm is demonstrated in Table 4.1.

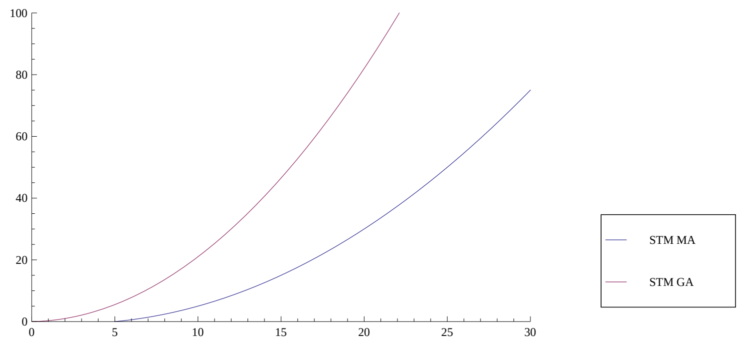


Figure 4.9: Comparison of computational time between MA and GA, plotted on a time vs node graph.

We speculate that this is not due to the MA doing less computations than a GA (in fact, the MA does more computations per cycle than the

GA does due to the local search capability). However, it is likely due to the MA's capability of detecting when a population converges. Once the local search detects that there are no more better solutions to be obtained, it stops. Thereby saving multiple useless computations that would lead to the same solution.

Chapter 5

Conclusion

Since its inception, the internet has experienced an exponential growth in both users and content availability. In order to sustain this growth, new technologies must be developed in order to provide a reliable form of high performance communication. Optical communication is communication at a distance to carry information using light. It can be performed visually or by using electronic interfaces. An optical communication system uses a transmitter, a channel, and a receiver.

In the past decade, the immense growth in high-bandwidth applications such as multimedia streaming and sharing has given rise to a corresponding increase in energy consumption of the network equipment [2]. Researchers have realized the importance of designing energy-minimized green networks to utilize the available power efficiently and consequently reduce the network operational cost. It is therefore necessary to develop robust optimization strategies for the design of energy-efficient core networks. The typical approach is to switch off some network components during low traffic periods.

We presented a GA-based approach as well as an MA-based approach to route a set of periodic, sub-wavelength traffic demands over the network. Moreover, we have shown that consideration of demand holding times can play an important role in reducing the overall energy consumption of optical networks. Our primary goal for these approaches was to route the traffic demands in such a way that the maximum number of lightpaths can be switched off at any given time, hence reducing the overall power consumption. Furthermore, our other objective was to reduce the total number of lightpaths needed to realize the logical topology, such that the capacity constraints of the lightpaths are not exceeded. Specifically, we have implemented each logical edge using as few lightpaths as possible, which in turn reduces the need for optical transceivers.

Results show that our GA is capable of achieving improvements of approximately 10% to 15%, while our MA is capable of achieving results of approximately 10% to 14%. However, the MA achieves similar results at 4 to 5 times less time than the GA. At the time of writing this paper, we have found no research that applies GAs to energy optimization in optical networks with static sub-wavelength traffic demands. There is even less research applying MAs to solve optical network energy optimization problems. Hence we believe there is room to be explored in this area, and the parameters of our research can be optimized greatly to achieve greater results.

5.1 Future Work

While the proposed GA and MA perform better than ILPs in terms of computational time and resources, there is still potential for future improvement. One of the fundamental strengths of GAs and MAs is the diversity of their parameters. It is very possible to achieve better results after optimizing the parameters to make them more optimized for a specific architecture.

In particular, MAs have great potential in energy optimization problems in optical networks. In terms of parameters, the way the Local Search mechanism defines neighbourhoods can be changed to something more complex than "chromosomes that happen to be next to each other during solution population", while this is a common and viable implementation, it may not be the most optimized for our purposes. Furthermore, we have only used one of *many* "move" mechanisms in Local Search, there are countless others that can be explored and exploited in order to achieve more optimized results.

It is also possible to change the way the MA switches from "exploitation" to "exploration" behaviour within its local search sub-routine. By changing the way the adaptation variable is defined, it is possible to achieve a more customized exploration and exploitation behaviour, where exploration and exploitation modes are invoked at specified times within the generations. A natural variation of the scheme presented in Algorithm 6 is one where every individual in the population has its own *adapt* variable and the local search/diversification process is applied according to it.

With respect to our current implementation, more experimentation and data analysis should be applied, not only with different size and complexity of instances of networks, but also with other network optimization problems, as this has proven to be a promising direction in optical network optimization problems.

Chapter 6

Bibliography

[1] J. M. Simmons, *Optical Network Design and Planning.* Springer, 2008.

[2] M. Gupta and S. Singh, “Greening of the internet,” in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 19–26, ACM, 2003.

[3] G. Shen and R. S. Tucker, “Energy-minimized design for ip over wdm networks,” *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 1, no. 1, pp. 176–186, 2009.

[4] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, “Power awareness in network design and routing,” in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pp. 457–465, IEEE, 2008.

[5] F. Idzikowski, S. Orlowski, C. Raack, H. Woesner, and A. Wolisz, “Saving energy in ip-over-wdm networks by switching off line cards in low-

demand scenarios," in *Optical Network Design and Modeling (ONDM), 2010 14th Conference on*, pp. 1–6, IEEE, 2010.

[6] A. Coiro, M. Listanti, A. Valenti, and F. Matera, "Reducing power consumption in wavelength routed networks by selective switch off of optical links," *Selected Topics in Quantum Electronics, IEEE Journal of*, vol. 17, no. 2, pp. 428–436, 2011.

[7] B. G. Bathula and J. M. Elmirghani, "Green networks: Energy efficient design for optical networks," in *Wireless and Optical Communications Networks, 2009. WOCN'09. IFIP International Conference on*, pp. 1–5, IEEE, 2009.

[8] J. Kuri, N. Puech, M. Gagnaire, E. Dotaro, and R. Douville, "Routing and wavelength assignment of scheduled lightpath demands," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 8, pp. 1231–1240, 2003.

[9] A. Shaabana, F. Luo, Y. Chen, and A. Jaekel, "A genetic algorithm-based approach for energy efficient grooming of scheduled sub-wavelength traffic demands in optical networks," *Submitted to IEEE Globecom 2013*.

[10] V. Alwayn, *Optical network design and implementation*. Cisco Systems, 2004.

[11] D. Ashlock, *Evolutionary computation for modeling and optimization*. Springer Science+ Business Media, 2006.

[12] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[13] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[14] J. H. Holland, *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

[15] K. A. De Jong, "Analysis of the behavior of a class of genetic adaptive systems," 1975.

[16] G. Syswerda, "Uniform crossover in genetic algorithms," 1989.

[17] N. Krasnogor, "Memetic algorithms," in *Handbook of Natural Computing* (G. Rozenberg, T. Bck, and J. Kok, eds.), pp. 905–935, Springer Berlin Heidelberg, 2012.

[18] R. Dawkins, "The selfish gene," *Oxford University Press*, p. 192, 1976.

[19] M. Oca, C. Cotta, and F. Neri, "Local search," in *Handbook of Memetic Algorithms* (F. Neri, C. Cotta, and P. Moscato, eds.), vol. 379 of *Studies in Computational Intelligence*, pp. 29–41, Springer Berlin Heidelberg, 2012.

[20] N. J. Radcliffe and P. D. Surry, "Formal memetic algorithms," in *Evolutionary Computing* (T. Fogarty, ed.), vol. 865 of *Lecture Notes in Computer Science*, pp. 1–16, Springer Berlin Heidelberg, 1994.

[21] N. J. Radcliffe, “Equivalence class analysis of genetic algorithms,” *Complex Systems*, vol. 5, no. 2, pp. 183–205, 1991.

[22] C. Gazen and C. Ersoy, “Genetic algorithms for designing multihop lightwave network topologies,” *Artificial Intelligence in Engineering*, vol. 13, no. 3, pp. 211–221, 1999.

[23] N. Krasnogor, J. Smith, *et al.*, “A memetic algorithm with self-adaptive local search: Tsp as a case study,” in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000), volume*, pp. 987–994, 2000.

[24] P. Datta, M. Sridharan, and A. K. Somani, “A simulated annealing approach for topology planning and evolution of mesh-restorable optical networks,” in *8th IFIP Working conference on optical networks design and modeling (ONDM)*, vol. 16, Citeseer, 2003.

[25] A. Tsenov, “Simulated annealing and genetic algorithm in telecommunications network planning,” *International Journal of Computational Intelligence*, vol. 2, no. 1, pp. 240–245, 2005.

[26] M.-T. Chen and S.-S. Tseng, “Multicast routing under delay constraint in wdm network with different light splitting,” in *Proceedings of International Computer Symposium (ICS 2002)*, vol. 1, 2002.

[27] M.-T. Chen and S.-S. Tseng, “A genetic algorithm for multicast routing under delay constraint in wdm network with different light splitting,” *Journal of Information Science and Engineering*, vol. 21, no. 1, pp. 85–108, 2005.

[28] K. Roy and M. K. Naskar, "Genetic evolutionary algorithm for static traffic grooming to sonet over wdm optical networks," *Computer Communications*, vol. 30, no. 17, pp. 3392 – 3402, 2007. Special Issue Concurrent Multipath Transport.

[29] Q. Zhang, J. Sun, G. Xiao, and E. Tsang, "Evolutionary algorithms refining a heuristic: A hybrid method for shared-path protections in wdm networks under srlg constraints," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 51–61, 2007.

[30] S. Huang, D. Seshadri, and R. Dutta, "Traffic grooming: a changing role in green optical networks," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pp. 1–6, IEEE, 2009.

[31] E. Yetginer and G. N. Rouskas, "Power efficient traffic grooming in optical wdm networks," in *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*, pp. 1–6, IEEE, 2009.

[32] J. Postel, "User datagram protocol," *DARPA Network Working Group Report RFC-791*, September 1981.

[33] J. Postel, "User datagram protocol," *DARPA Networkw Working Group Report RFC-793*, September 1981.

[34] J. Postel, "User datagram protocol," *DARPA Network Working Group Report RFC-791*, August 1980.

[35] D. E. Goldberg, "Genetic algorithms in search, optimization, and machine learning," 1989.

[36] J. H. Holland, *Adaptation in natural and artificial systems*. Cambridge, MA, USA: MIT Press, 1992.

[37] L. Davis, "Handbook of genetic algorithms," 1991.

[38] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2001.

[39] A. Bari, S. Wazed, A. Jaekel, and S. Bandyopadhyay, "A genetic algorithm based approach for energy efficient routing in two-tiered sensor networks," *Ad Hoc Networks*, vol. 7, no. 4, pp. 665–676, 2009.

[40] M. Sridharan, M. V. Salapaka, and A. K. Somani, "A practical approach to operating survivable wdm networks," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 1, pp. 34–46, 2002.

[41] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.

[42] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature extraction, construction and selection*, pp. 117–136, Springer, 1998.

[43] D. Thierens and D. Goldberg, "Convergence models of genetic algorithm selection schemes," in *Parallel problem solving from naturePPSN III*, pp. 119–129, Springer, 1994.

[44] H. Muhlenbein, “Evolution in time and space-the parallel genetic algorithm,” in *Foundations of genetic algorithms*, Citeseer, 1991.

VITA AUCTORIS

NAME:	Ala Shaabana
PLACE OF BIRTH:	Baghdad, Iraq
YEAR OF BIRTH:	1989
EDUCATION:	Holy Names High School, Windsor, ON, 2006
	University of Windsor, B.Sc., Windsor, ON, 2011
	University of Windsor, M.Sc., Windsor, ON, 2013