İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

A GA/HEURISTIC BASED HYBRID TECHNIQUE FOR ROUTING AND WAVELENGTH ASSIGNMENT IN WDM NETWORKS

M.Sc. Thesis by

A.Çağatay TALAY, B.Sc.

Date of Submission : 22 November 2003

Date of Defense Examination : 14 January 2004

Supervisor (Chairman) : **Assoc. Prof. Dr. Sema OKTUĞ**

Members of the Examining Committee : **Assoc. Prof. Dr. Ayşegül GENÇATA (İTÜ)**

Assoc. Prof. Dr. Levent AKIN (BÜ)

DECEMBER 2003

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

WDM AĞLARDA YOL VE DALGABOYU ATAMA İÇİN BİR GENETİK ALGORİTMA/SEZGİSEL YÖNTEM MELEZ TEKNİĞİ

YÜKSEK LİSANS TEZİ

Müh. A.Çağatay TALAY

Tezin Enstitüye Verildiği Tarih : 22 Aralık 2003

Tezin Savunulduğu Tarih : 14 Ocak 2004

Tez Danışmanı : Doç. Dr. Sema OKTUĞ

Diğer Jüri Üyeleri : Yrd. Doç. Dr. Ayşegül GENÇATA (İTÜ)

Doç. Dr. Levent AKIN (BÜ)

ARALIK 2003

Acknowledgements

This thesis would not have happened without the help the people mentioned below. I am most grateful to my thesis supervisor Assoc. Prof. Dr. Sema OKTUĞ for her invaluable guidance, support, suggestions, and motivation during the preparation of this dissertation.

I also would like to express my sincere gratitude and give my warmest thanks to Sanem SARIEL for her great encouragement and for helping me out with the problems I encountered in all the stages of my research.

I am grateful to my family their help and support. I would like to thank them for putting up with all the inconveniences I caused.

December 2003 A. Çağatay TALAY, B.Sc.

CONTENTS

ABBREVIATIONS

ACO : Ant Colony Optimization
ACTS : Advanced Communications Technologies And Services
ADM : Add Drop Multiplexer
BON : Business Object Notation
CS : Classifier System
DCR : Different Common Ring
DNA : Deoxyribose Nucleic Acid
DQDB : Dual Queue Dual Bus Man (IEEE 802.6)
EA : Evolutionary Algorithm
EC : Evolutionary Computation
EH : Evolutionary Hardware
EON : European Optical Network
EP : Evolutionary Programming
ES : Evolution Strategy
FSM : Finite State Machine
FTEL : Fujitsu Telecommunications Europe Ltd.
GA : Genetic Algorithm
GP : Genetic Programming
id : Identification Number
LAN : Local Area Network
LCS : Learning Classifier System
MAN : Metropolitan Area Network
MCA : Microcanonical Annealing
NGOS : Network Grade-Of-Service
NP : Non-Deterministic Polynomial
NWR : Network Wavelength Requirement
OOA&D : Object-Oriented Analysis And Design
OPEN : Optical Pan-European Network
OXC : Optical Cross Connect
PON : Passive Optical Network
RCA : Random Cost Algorithm
SA : Simulated Annealing
SGA : Goldberg's Simple Genetic Algorithm
SWP : Sparse Wavelength Path
TS : Tabu Search
TSP : Traveling Salesman Problem
VWP : Virtual Wavelength Path
WOTAN : Wavelength-agile Optical Transport and Access Network
WP : Wavelength Path

LIST OF TABLES

LIST OF FIGURES

A GA/HEURISTIC BASED HYBRID TECHNIQUE FOR ROUTING AND WAVELENGTH ASSIGNMENT IN WDM NETWORKS

SUMMARY

Recently, there has been considerable progress in the area of all-optical networks which are based on wavelength division multiplexing (WDM). WDM technology provides the capacity required by backbone networks. WDM based all optical networks offering multi-gigabit rate per wavelength may soon become economical as the underlying backbone in wide area networks.

When individual static traffic requirements are to be routed independently on a single wavelength end-to-end based on wavelength continuity constraint, the problem is to determine the route, fibers, and wavelength each connection will use. This problem and its variants have attracted considerable interest, with a variety of solution approaches including heuristics, genetic algorithms (GAs), and integer linear programming (ILP) techniques. Those heuristic algorithms can produce good solutions in a reasonable amount of time. However, they have restricted applicability in a practical environment because they have a number of fundamental problems including high time complexity, lack of robustness and no performance guarantee as input changes.

Our approach is to incorporate advanced heuristics into an overall genetic algorithm. Genetic Algorithms are a class of stochastic, global optimization algorithms that model the biological principles of Darwin's theory of evolution and Mendel's principles of classical genetics. The theory of evolution centers on the principle of natural selection that mainly states that those individuals that have a certain characteristics. The characteristics that give the individuals some advantage above others are more likely to survive and reproduce. If this characteristic is inheritable, then some of these individuals' offspring will be born with it and thus have the advantage over the others. After a few generations, the number of individuals with the favorable trait will increase in the population.

The GA/heuristic hybrid approach adopted in this thesis, inspired by Davis' work, employs an object-oriented network model, an adaptive overall GA framework, and heuristic operators. Such an object-oriented network model would appear to be a natural representation for network problems, rather than a linear bit-string encoding. Using heuristic operators allows problem-specific knowledge to be applied, while operator-probability adaptation should allow the blend of operators to be adjusted according to their relative productivities during execution. Two metrics for network effectiveness assessment are used: one is based on NWR (network wavelength requirement), and the other is based on a simplified model of network cost. The results obtained with the hybrid technique are compared with those obtained from the recent wavelength-allocation heuristics. It is observed that the proposed hybrid technique has very promising results when compared with the results of the other techniques under various parameters.

WDM AĞLARDA YOL VE DALGABOYU ATAMA İÇİN BİR GENETİK ALGORİTMA/SEZGİSEL YÖNTEM MELEZ TEKNİĞİ

ÖZET

Son yıllarda, optik ağlarda, özellikle de Dalgaboyu Atamalı WDM ağlarda (Wavelength Routed WDM Networks) önemli ilerlemeler sağlanmıştır. Optik iletim teknolojisi, çok büyük bant genişliği ve oldukça yüksek hızlarda iletim yeteneğine sahip bir teknolojidir. Bu nedenle WDM teknolojisinin gelecekte yaygın olarak ağların omurgasını teşkil edeceği öngörülmektedir.

Statik trafik isteklerinin yol seçim işleminin tek bir dalgaboyunda uçtan uca yapıldığı düşünüldüğünde, optik ağların ulaşım katman dizaynında, her bağlantının kullanacağı yolun ve hangi dalgaboyunun kullanılacağına ne şekilde karar verileceğinin seçimi önemli bir faktördür. Bu problem ve varyantları önemli ölçüde dikkat çekmiş ve içlerinde, sezgisel yöntemlerin, genetik algoritmaların, ve lineer programlamanın da olduğu, çeşitli çözüm önerileri ortaya atılmıştır. Bu yöntemler, makul zaman kısıtları içerisinde iyi sonuçlar üretmelerine rağmen, pratik hayatta uygulanabilirlikleri sınırlı ölçülerde kalmaktadır.

Bu çalışmada kullanılan yaklaşım, başarılı sezgisel yöntemlerle, genetik algoritmaları birleştirerek çözüme ulaşmaya çalışmaktır.

Genetik Algoritmalar, Darwin'in evrim teorisini ve Mendel'in kalıtım prensiplerini modelleme yaklaşımına dayanan stokastik, global en iyileme yöntemleridir. Evrim teorisinin temelinde doğal seçim kavramı yatmaktadır. Bu teoriye göre, doğal seçim sonucunda diğerlerine göre bazı avantajları olan bireylerin yaşama ve sonraki kuşaklara yavru bırakma olasılıkları daha yüksektir. Bu avantajı sağlayan özellikler kalıtımsal ise bu özelliğe sahip bireylerden doğan yavrularında bir kısmı bu özelliği taşıyacak ve dolayısıyla aynı avantajlara sahip olacaklardır. Birkaç kuşak sonucunda bu iyi özelliği sağlayan bireylerin sayısı toplumda artacaktır.

Bu çalışmada benimsenen Genetik Algoritma/Sezgisel yaklaşım yöntemi, Davis'in çalışmalarından esinlenilerek, nesneye dayalı bir ağ modeli, adaptif bir genetik algoritma, ve sezgisel yaklaşımlara dayanan operatörlerden oluşmaktadır. Bu şekilde bir nesneye dayalı ağ modeli, ikili sayı katarlarıyla yapılan kodlamadan daha doğal bir temsil olarak düşünülmektedir. Sezgisel yöntemlere dayanan operatörlerin kullanımı, bu yöntemlerin probleme dayalı bilgilerinin kullanımına olanak vermiştir. Ayrıca, operatör olasılık adaptasyonunun kullanılmasıyla operatörlerin koşturma sırasındaki verimliliklerine göre uygulanmaları sağlanmaya çalışılmıştır. Çalışmada iki farklı uygunluk fonksiyonu (verimlilik yargısı) kullanılmıştır. Bunlardan birincisi ağın kullanması gereken minimum dalgaboyu sayısına, diğeri ise basitleştirilmiş bir ağ maliyet modeline dayanmaktadır. Önerilen teknikle elde edilen sonuçlar, daha önce önerilmiş olan bazı sezgisel yöntemlerin sonuçlarıyla karşılaştırılarak başarımının daha iyi olduğu görülmüştür.

1. INTRODUCTION

Telecommunications is a vital and growing research field. It is important not only in its own right, but also for the services it provides to other disciplines. Moreover, there seems to be a demand for an expanding set of telecommunication services of increasing bandwidth. One particular technology, namely multi-wavelength all-optical transport networks, has the potential to provide the required bandwidth if such broadband services are widely adopted [1].

Recently, there has been considerable progress in the area of all-optical networks, in particular the networks based on wavelength division multiplexing (WDM). WDM offering multi-gigabit rate per wavelength may soon become economical as the underlying backbone in wide area communication networks. However, the development of such networks presents a challenging range of difficult design and optimization problems.

There are many different potential approaches for the design and optimization of all-optical networks. These can be classified as: (integer) linear programming, problem specific heuristics, and modern heuristic search methods. These heuristic search methods can be listed as evolutionary algorithms (EAs) [2-4], simulated annealing (SA) [5] and tabu search (TS) [6]. In particular, EAs have attracted growing interest in recent years to solve complicated problems in many fields including telecommunications. Applications in the telecommunications area have included network design, call routing, frequency assignment, wavelength allocation, capacity planning, admission control, network management, and many others [7].

When individual static traffic requirements are supposed to be routed independently on a single wavelength end-to-end, on an optical WDM network, issues to be considered are determining the route, fibers, and wavelength of each connection. This problem and its variants have attracted considerable interest, with a variety of solution approaches mentioned above. The heuristic algorithms mentioned can produce good solutions in some amount of time. However, they have restricted applicability in a practical environment because they have a number of fundamental

problems including high time complexity, lack of robustness and no performance guarantee with respect to the optimal solutions.

Nagatsu *et al.* [31] presented a heuristic where the initial routing is accomplished with Dijkstra's algorithm using the number of paths carried by each link as the link weights. Next, iterative rerouting, constrained by hop count, is used to equalize link weights. Finally, wavelengths are assigned to the paths, aiming for minimum network wavelength requirement (NWR). Nagatsu *et al.* then extended their heuristic to include preplanned path restoration in the single-link failure condition. Whereas in [97] Nagatsu and Sato modified the original heuristic to include multiple fiber links, with only a restricted number of wavelengths per fiber. Then, in [96], Nagatsu *et al.* combined the two earlier extensions into a single heuristic. In both [96] and [97], the objective was to minimize the average number of ports on the network's optical cross-connects (OXCs).

Wuttisittikulkij and O'Mahony [28] developed a simple, fast and non-iterative heuristic, which works with *k*-lowest hop count paths to obtain a low NWR.

Wauters and Demeester [29] presented an integer-linear-programming approach that aims for maximum throughput, and two heuristics. The first of them is based on a version of Dijkstra's algorithm modified such that, in extending an already found part of a route, only some arcs are considered. The considered arcs have a wavelength channel free that is available on all links of the already found part of the route. This algorithm produces a low cost allocation, while keeping NWR low. Their second algorithm (HRWA), is based on iterative application of a modified version of Yen's *k*-shortest path algorithm, and aims to produce the lowest cost allocation that still achieves minimum NWR.

This thesis describes the application of GA/heuristic hybrids to one of the main problems in the design of optical WDM networks. This problem can be divided into two subproblems: optical-path routing, and wavelength assignment in WDM networks. Our approach is to incorporate advanced heuristics into an overall genetic algorithm. While GAs are good as general-purpose search and optimization procedures, they are never the best approach for a specific problem [8]. A powerful alternative is the development of problem specific EAs-GA/heuristic hybrids. These adopt a problem specific encoding and employ problem specific operators that together combine the best existing heuristics for the problem within an overall GA framework.

Our hybrid GA/heuristic approach adopts a problem-specific encoding based on object-oriented manner. It also develops and employs problem-specific operators that together combine the best existing heuristics for the problem with an overall GA framework. It is almost possible to develop a problem-specific heuristic/GA hybrid that will outperform either the heuristic approach or a traditional GA alone. This technique is an example to show it.

In this thesis, optical, mesh multiple-fiber-link topology is used. This architecture is expected to have its place in future ultra-high-capacity transport networks [9]. While hierarchical networks have many notable benefits, particularly in simplifying network management, the less-constrained flat architecture still has potential. Similarly, although more modular networks (e.g. multi-ring) have several desirable qualities, the arbitrary-mesh topology may well form the core of future transport networks. Also, whereas the use of wavelength converters can result in greater flexibility under failure conditions, careful wavelength-path (WP) routing remains competitive during normal operation. Moreover, in all three areas, the choices adopted lead to more potential networks and hence larger search spaces during both topology design and network allocation, particularly in employing WP rather than virtual wavelengthpath (VWP) routing. Overall, then, these problems will provide a suitable testbed for development of problem specific EAs.

The hybrid approach is shown to provide excellent solutions, of comparable or higher quality than those obtained with either heuristic approaches or simple GAs alone, and to require less computational effort than GAs.

The thesis is organized as follows: The problem domain is described in Section 2. The detailed information about evolutionary algorithms is given in Section 3. Section 4 gives the information about the technique used here. Section 5 gives the detailed information about heuristics used for wavelength assignment in WDM networks that are used for comparison. Section 6 presents the simulation environment and results obtained. Finally, Section 7 concludes the thesis.

2. WAVELENGTH ASSIGNMENT AND ROUTING PROBLEM IN WDM NETWORKS

A basic property of single mode optical fiber is its enormous low-loss bandwidth of several tens of Terahertz. However, due to dispersive effects and limitations in optical device technology, single channel transmission is limited to only a small fraction of the fiber capacity. To take advantage of the full capacity of fiber, the use of wavelength division multiplexing (WDM) technology has become the option to be considered. With WDM, a number of distinct wavelengths are used to implement separate channels [12]. An optical fiber can carry several channels in parallel. Each channel transmits data on a particular wavelength. The number of wavelengths that each fiber can carry simultaneously is limited by the physical characteristics of the fiber and the state of the optical technology used to combine these wavelengths onto the fiber and isolate them off the fiber. With currently available commercial technology, a few tens of wavelengths can be supported within the low-loss window at 1550 nm, but this number is expected to grow rapidly in the next few years. Therefore, optical fiber links employing WDM technology have the potential of delivering an aggregate throughput in the order of terabits per second.

Unfortunately, due to the mismatch between aggregate fiber capacity and peak electronic processing speed, upgrading existing point-to-point fiber link capacity creates the well-known electro-optic bottleneck [1] rather than achieving the multiterabit per second throughput of the fiber. Overcoming the electro-optic bottleneck involves the design of properly structured architectures to interconnect the fiber links. Optical WDM network is should be designed to exploit the unique features of fibers and WDM. Such networks optical information highway capable of supporting a wide range of applications that involve the transport of massive amounts of data and/or require very fast response times. Such applications include video on demand and teleconferencing, telemedicine applications, multimedia document distribution, remote supercomputer visualization, and many more to come. Consequently, optical WDM networks have been subject of extensive research both theoretically and experimentally [13,1 4].

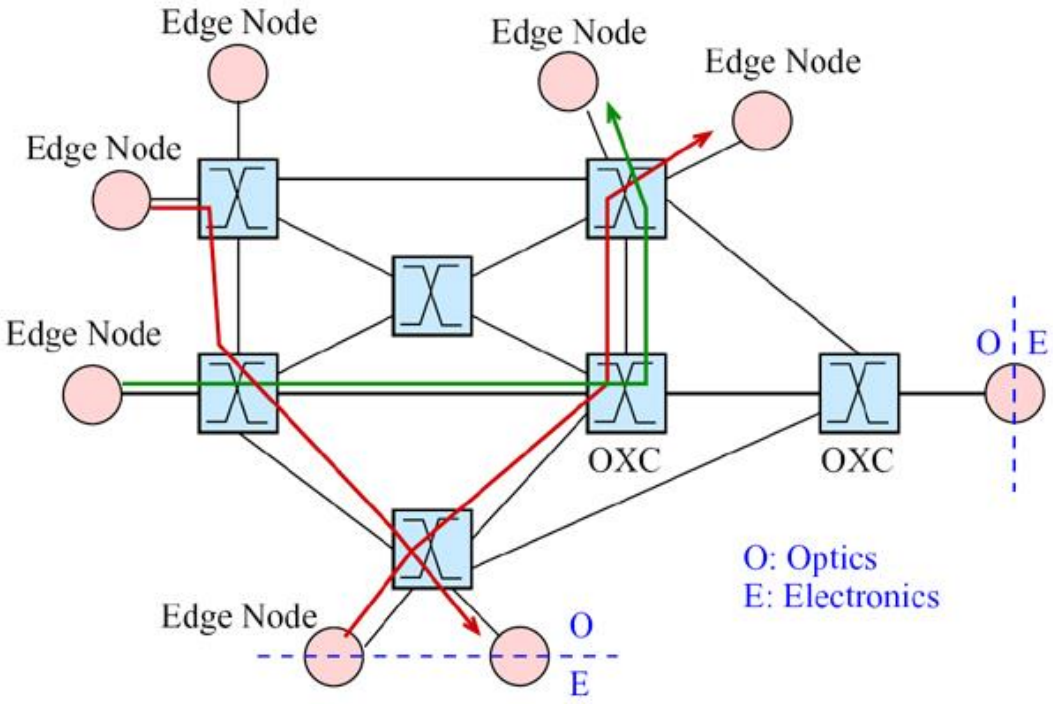


Figure 2.1 A wavelength routed WDM network

The architecture for wide-area WDM networks which is widely expected to form the basis for a future all-optical infrastructure is built on the concept of wavelength routing. A wavelength routing network, shown in Figure 2.1, consists of two types of nodes: optical cross-connects (OXCs), which connect the fibers in the network, and edge nodes which provide the interface between non-optical end systems (such as IP routers, ATM switches, or supercomputers) and the optical core. Access nodes provide the terminating points (sources and destinations) for the optical signal paths; the communication paths continue outside the optical part of the network in electrical form.

The services that a wavelength-routed network offers to end systems attached to edge nodes are in the form of logical connections implemented using lightpaths. Lightpaths, are clear optical paths between two edge nodes, and are shown in Figure 2.1 as red and green directed lines. Information transmitted on a lightpath does not undergo any conversion to and from electrical form within the optical network, and thus, the architecture of the optical network nodes can be very simple because they do not need to do any signal processing. Furthermore, since a lightpath behaves as a literally transparent “clear channel" between the source and destination edge node, there is nothing in the signal path to limit the throughput of the fibers except the capacity of the channel.

The OXCs provide the switching and routing functions for supporting the logical connections between edge nodes. An OXC takes in an optical signal at each wavelength at an input port, and can switch it to a particular output port, independent of the other wavelengths. An OXC with *N* input and *N* output ports capable of handling *W* wavelengths per port can be thought of as *W* independent *NxN* switches. These switches have to be preceded by a wavelength demultiplexer and followed by a wavelength multiplexer to implement an OXC, as shown in Figure 2.2. Thus, an OXC can cross-connect the different wavelengths from the input to the output, where the connection pattern of each wavelength is independent of the others. By appropriately configuring the OXCs along the physical path, a logical connection (lightpath) may be established between any pair of edge nodes.

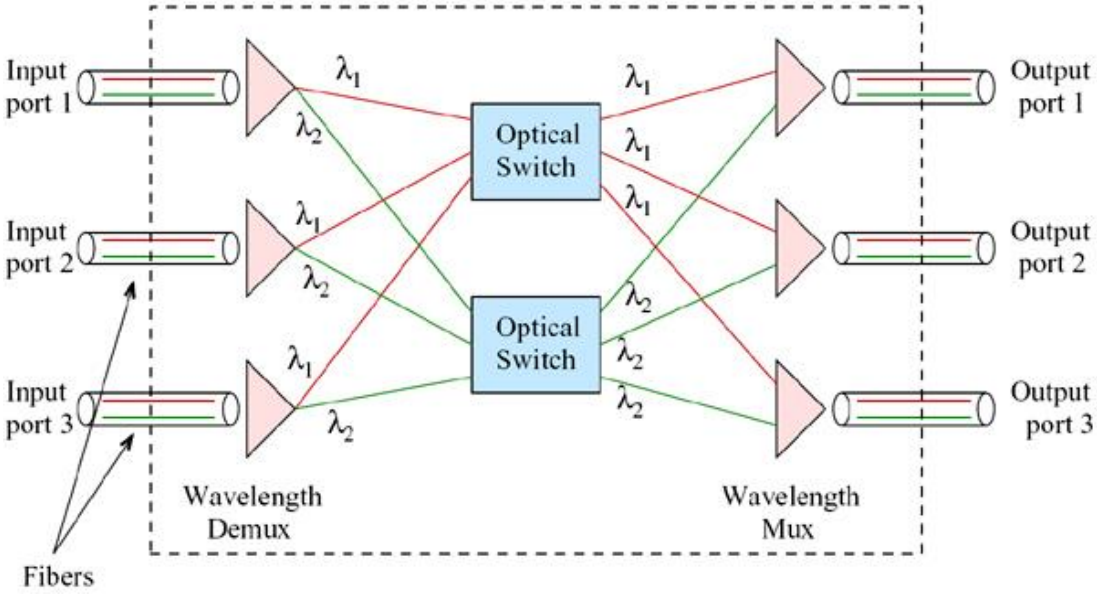


Figure 2.2 A 3x3 optical cross-connect (OXC) with two wavelengths per fiber

A unique feature of optical WDM networks is the tight coupling between routing and wavelength selection. As can be seen in Figure 2.1, a lightpath is implemented by selecting a path of physical links between the source and destination edge nodes, and reserving a particular wavelength on each of these links for the lightpath. Thus, in establishing an optical connection we must deal with both routing (selecting a suitable path) and wavelength assignment (allocating an available wavelength for the connection). The resulting problem is referred to as the routing and wavelength assignment (RWA) problem [15], and is significantly more difficult than the routing problem in electronic networks. The additional complexity arises from the fact that routing and wavelength assignment is subject to the following two constraints:

1. *Wavelength continuity constraint:* a lightpath must use the same wavelength on all the links along its path from source to destination edge node. This constraint is illustrated in Figure 2.1 by representing each lightpath with a single color (wavelength) along all the links in its path.

2. *Distinct wavelength constraint:* all lightpaths using the same link (fiber) must be allocated distinct wavelengths. In Figure 2.1 this constraint is satisfied since the two lightpaths sharing a link are shown in different colors (wavelengths).

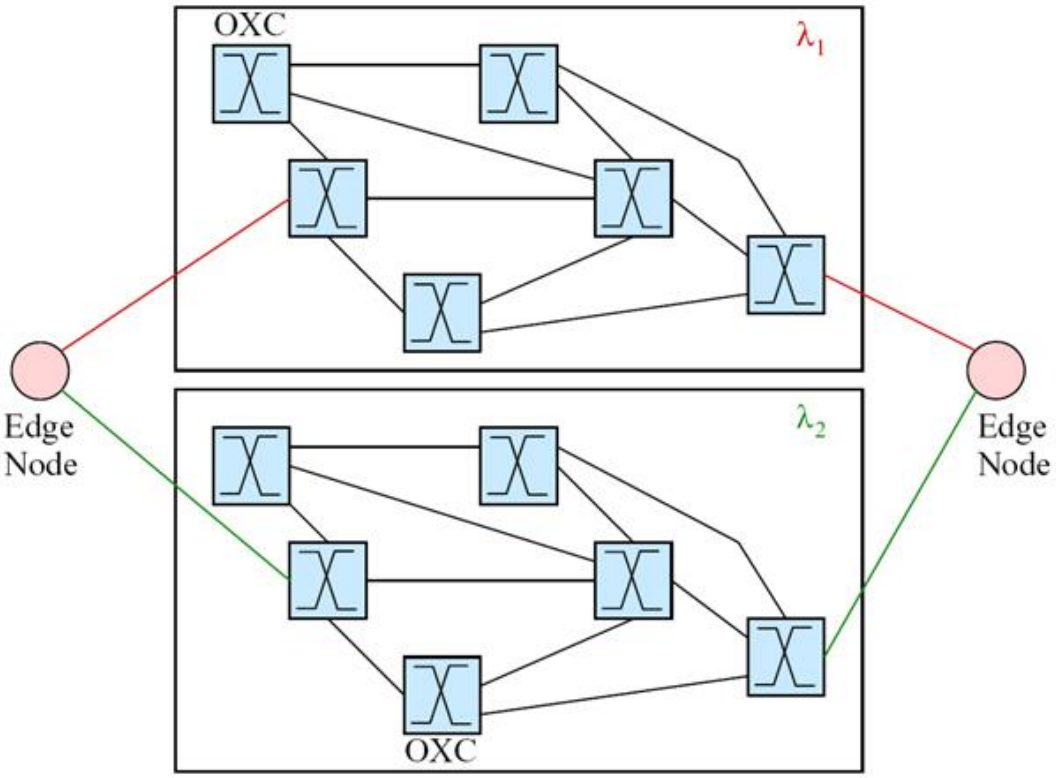


Figure 2.3 The RWA problem with two wavelengths per fiber

The RWA problem in optical networks is illustrated in Figure 2.3, where it is assumed that each fiber supports two wavelengths. The effect of the wavelength continuity constraint is represented by replicating the network into as many copies as the number of wavelengths (in this case, two). If wavelength *i* is selected for a lightpath, the source and destination edge node communicate over the *i*-th copy of the network. Thus, finding a path for a connection may potentially involve solving W routing problems for a network with W wavelengths, one for each copy of the network.

The wavelength continuity constraint may be relaxed if the OXCs are equipped with wavelength converters [16]. A wavelength converter is a single input/output device that converts the wavelength of an optical signal arriving at its input port to a different wavelength as the signal departs from its output port, but otherwise leaves the optical signal unchanged. In OXCs without a wavelength conversion capability, an incoming signal at port p_i on wavelength λ can be optically switched to any port p_j, but must leave the OXC on the same wavelength λ. With wavelength converters, this signal could be optically switched to any port p_j on some other wavelength λ'. That is, wavelength conversion allows a lightpath to use different wavelengths along different physical links.

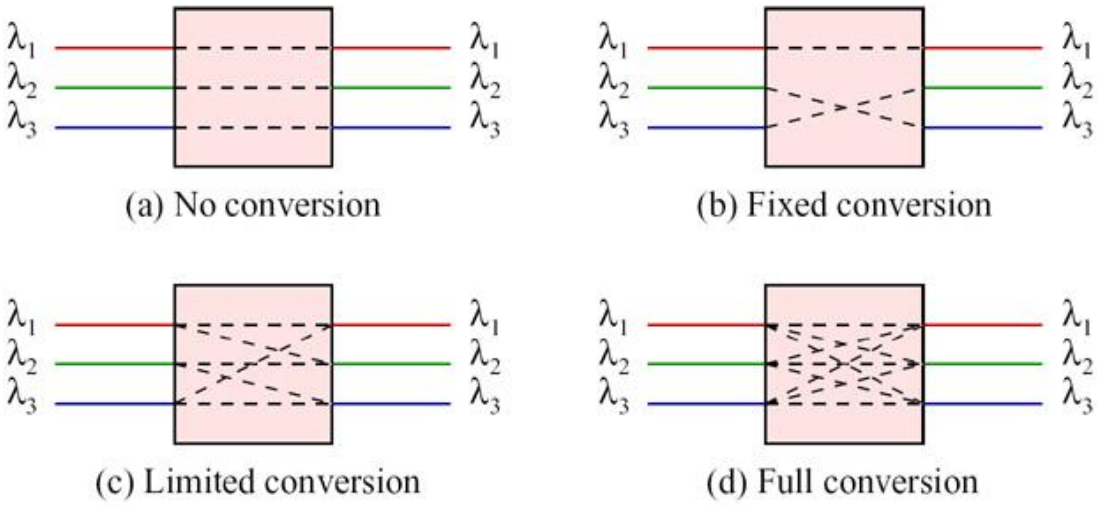


Figure 2.4 Wavelength conversion

Different levels of wavelength conversion capability are possible. Figure 2.4 illustrates the differences for a single input and single output port situation; the case for multiple ports is more complicated but similar. Full wavelength conversion capability implies that any input wavelength may be converted to any other wavelength. Limited wavelength conversion [17] denotes that each input wavelength may be converted to any of a specific set of wavelengths, which is not the set of all wavelengths for at least one input wavelength. A special case of this is fixed wavelength conversion, where each input wavelength can be converted to exactly one other wavelength. If each wavelength is converted only to itself, then we have no conversion.

The advantage of full wavelength conversion is that it removes the wavelength continuity constraint, making it possible to establish a lightpath as long as each link along the path from source to destination has a free wavelength (which could be

different for different links). As a result, the RWA problem reduces to the classical routing problem, that is, finding a suitable path for each connection in the network. Referring to Figure 2.3, full wavelength conversion collapses the *W* copies of the network into a single copy on which the routing problem is solved. On the other hand, with limited conversion, the RWA problem becomes more complex than with no conversion. To see why, note that employing limited conversion at the OXCs introduces links between some of the network copies of Figure 2.3. For example, if wavelength λ_1 can be converted to wavelength λ_2 but not to wavelength λ_3, then links must be introduced from each OXC in copy 1 of the network to the corresponding OXC in copy 2, but not to the corresponding OXC in copy 3. When selecting a path for the connection, at each OXC there is the option of remaining at the same network copy or moving to another one, depending on the conversion capability of the OXC. Since the number of alternatives increases exponentially with the number of OXCs that need to be traversed, the complexity of the RWA problem increases accordingly.

Wavelength conversion (full or limited) increases the routing choices for a given lightpath (i.e., makes more efficient use of wavelengths), resulting in better performance. Since converter devices increase network cost, a possible middle ground is to use sparse conversion, that is, to employ converters in some, but not all, OXCs in the network. In this case, a lightpath must use the same wavelength along each link in a segment of its path between OXCs equipped with converters, but it may use a different wavelength along the links of another such segment. It has been shown that by implementing full conversion at a relatively small fraction of the OXCs in the network is sufficient to achieve almost all the benefits of conversion [18, 19].

Routing and wavelength assignment is the fundamental control problem in optical WDM networks. Since the performance of a network depends not only on its physical resources (e.g., OXCs, converters, fibers links, number of wavelengths per fiber, etc.) but also on how it is controlled. The objective of an RWA algorithm is to achieve the best possible performance within the limits of physical constraints. The RWA problem can be cast in numerous forms. The different variants of the problem, however, can be classified under one of two broad versions: a static RWA, whereby the traffic requirements are known in advance, and a dynamic RWA, in which a sequence of lightpath requests arrive in some random fashion.

2.1 Static Routing and Wavelength Assignment

If the traffic patterns in the network are reasonably well-known in advance and any traffic variations take place over long time scales, the most effective technique for establishing optical connections (lightpaths) between edge nodes is by formulating and solving a static RWA problem. For example, static RWA is appropriate for provisioning a set of semipermanent connections. Since these connections are assumed to remain in place for relatively long periods of time, it is worthwhile to attempt to optimize the way in which network resources (e.g., physical links and wavelengths) are assigned to each connection, even though optimization may require a considerable computational effort.

A solution to the static RWA problem consists of a set of long-lived lightpaths that create a logical (or virtual) topology among the edge nodes. This virtual topology is embedded onto the physical topology of optical fiber links and OXCs. Accordingly, the static RWA problem is often referred to as the virtual topology design problem [20]. In the virtual topology, there is a directed link from edge node s to edge node d if a lightpath originating at s and terminating at d is set up (refer also to Figure 2.1), and edge node s is said to be \one hop away" from edge node d in the virtual topology, although the two nodes may be separated by a number of physical links. The type of virtual topology that can be created is usually constrained by the underlying physical topology. In particular, it is generally not possible to implement fully connected virtual topologies: for *N* edge nodes this would require each edge node to maintain *N-1* lightpaths and the optical network to support a total of *N*(*N-1*) lightpaths. Even for modest values of *N*, this degree of connectivity is beyond the scope of current optical technology, both in terms of the number of wavelengths that can be supported and in terms of the optical hardware (transmitters and receivers) required at each edge node.

In its most general form, the RWA problem is specified by providing the physical topology of the network and the traffic requirements. The physical topology corresponds to the deployment of cables in some existing fiber infrastructure, and is given as a graph $G_p(V, E_p)$, where *V* is the set of OXCs and E_p is the set of fibers that interconnect them. The traffic requirements are specified in a traffic matrix $T=[\rho p_{sd}]$, where ρp_{sd} is a measure of the long-term traffic owing from source edge node s to destination edge node d [21]. Quantity ρ represents the (deterministic) total officered load to the network, while the p_{sd} parameters define the distribution of the offered traffic.

Routing and wavelength assignment are considered together as an optimization problem using mixed integer programming (MIP) formulations. Usually, the objective of the formulation is to minimize the maximum congestion level in the network subject to network resource constraints [20, 22]. While other objective functions are possible, such as minimizing the average weighted number of hops or minimizing the average packet delay, minimizing network congestion is preferable since it can lead to linear programming (MILP) formulations. While we do not present the RWA problem formulation here, the interested reader may refer to [20, 21, 22]. These formulations turn out to have extremely large numbers of variables, and they are intractable for large networks. This fact has motivated the development of heuristic approaches for finding good solutions efficiently.

Before describing the various heuristic approaches, we note that the static RWA problem can be logically decomposed into four subproblems. The decomposition is approximate or inexact, in the sense that solving the subproblems in sequence and combining the solutions may not result in the optimal solution for the fully integrated problem, or some later subproblem may have no solution given the solution obtained for an earlier subproblem, so no solution to the original problem may be obtained. However, the decomposition provides insight into the structure of the RWA problem and is a first step towards the design of effective heuristics. Assuming no wavelength conversion, the subproblems are as follows.

1. Topology Subproblem: Determine the logical topology to be imposed on the physical topology, that is, determine the lightpaths in terms of their source and destination edge nodes.

2. Lightpath Routing Subproblem: Determine the physical links which each lightpath consists of, that is, route the lightpaths over the physical topology.

3. Wavelength Assignment Subproblem: Determine the wavelength each lightpath uses, that is, assign a wavelength to each lightpath in the logical topology so that wavelength restrictions are obeyed for each physical link.

4. Traffic Routing Subproblem: Route packet traffic between source and destination edge nodes over the logical topology obtained.

A large number of heuristic algorithms have been developed in the literature to solve the general static RWA problem discussed here or its many variants. Overall, however, the different heuristics can be classified into three broad categories: (1)

algorithms which solve the overall MILP problem sub-optimally, (2) algorithms which tackle only a subset of the four subproblems, and (3) algorithms which address the problem of embedding regular logical topologies onto the physical topology.

Suboptimal solutions can be obtained by applying classical tools developed for complex optimization problems directly to the MILP problem. One technique is to use LP-relaxation followed by rounding [23]. In this case, the integer constraints are relaxed creating a non-integral problem which can be solved by some linear programming method, and then a rounding algorithm is applied to obtain a new solution which obeys the integer constraints. Alternatively, genetic algorithms or simulated annealing [24] can be applied to obtain locally optimal solutions. The main drawback of these approaches is that it is difficult to control the quality of the final solution for large networks: simulated annealing is computationally expensive and thus, it may not be possible to adequately explore the state space, while LP-relaxation may lead to solutions from which it is difficult to apply rounding algorithms.

Another class of algorithms tackles the RWA problem by initially solving the first three subproblems listed above; traffic routing is then performed by employing well-known routing algorithms on the logical topology. One approach for solving the three subproblems is to maximize the amount of traffic that is carried on one-hop lightpaths, i.e., traffic that is routed from source to destination edge node directly on a lightpath. A greedy approach taken in [25] is to create lightpaths between edge nodes in order of decreasing traffic demands as long as the wavelength continuity and distinct wavelength constraints are satisfied. This algorithm starts with a logical topology with no links (lightpaths) and sequentially adds lightpaths as long as doing so does not violate any of the problem constraints. The reverse approach is also possible [26]: starting with a fully connected logical topology, an algorithm sequentially removes the lightpath carrying the smallest traffic flows until no constraint is violated. At each step (i.e., after removing a lightpath), the traffic routing subproblem is solved in order to find the lightpath with smallest the flow.

The third approach to RWA starts with a given logical topology, thus the first of the four subproblems listed above are directly solved. Regular topologies are good candidates as logical topologies since they are well understood and results regarding bounds and averages (e.g., for hop lengths) are easier to derive. Algorithms for routing traffic on a regular topology are usually simple, so the traffic routing subproblem can be trivially solved. Also, regular topologies possess inherent

load balancing characteristics which are important when the objective is to minimize the maximum congestion.

Once a regular topology is selected as the logical topology to implement, it remains to decide which physical node will represent each given node in the regular topology (this is usually referred to as the node mapping subproblem), and which sequence of physical links will be used to realize each given edge (lightpath) in the regular topology (this path mapping subproblem is equivalent to the lightpath routing and wavelength assignment subproblems discussed earlier). This procedure is usually referred to embedding a regular topology in the physical topology. Both the node and path mapping subproblems are intractable, and some heuristics have been proposed in the literature [26, 27]. For instance, a heuristic for mapping the nodes of shuffle topologies based on the gradient algorithm was developed in [27].

Given that all the algorithms for the RWA problem are based on heuristics, it is important to be able to characterize the quality of the solutions obtained. To this end, one must resort to comparing the solutions to known bounds on the optimal solution. A comprehensive discussion of bounds for the RWA problem and the theoretical considerations involved in deriving them can be found in [20]. A simulation-based comparison of the relative performance of the three classes of heuristic for the RWA problem is presented in [22]. The results indicate that the second class of algorithms discussed earlier achieves the best performance.

2.2 Dynamic Routing and Wavelength Assignment

Since dynamic routing and wavelength assignment is out of scope of the thesis, only a brief concept is given. More detailed information can be found in the literature.

Under a dynamic traffic scenario, edge nodes submit to the network requests for lightpaths to be set up as needed. Thus, connection requests are initiated in some random fashion. Depending on the state of the network at the time of a request, the available resources may or may not be sufficient to establish a lightpath between the corresponding source-destination edge node pair. The network state consists of the physical path (route) and wavelength assignment for all active lightpaths. The state evolves randomly in time as new lightpaths are admitted and existing lightpaths are released. Thus, each time a request is made, an algorithm must be executed in real time to determine whether it is feasible to accommodate the request, and, if so, to

perform routing and wavelength assignment. If a request for a lightpath cannot be accepted because of lack of resources, it is blocked.

Because of the real-time nature of the problem, RWA algorithms in a dynamic traffic environment must be very simple. Since combined routing and wavelength assignment is a hard problem, a typical approach to designing efficient algorithms is to decouple the problem into two separate subproblems: the routing problem and the wavelength assignment problem. Consequently, most dynamic RWA algorithms for wavelength routed networks consist of the following general steps:

1. Compute a number of candidate physical paths for each source-destination edge node pair and arrange them in a path list.

2. Order all wavelengths in a wavelength list.

3. Starting with the path and wavelength at the top of the corresponding list, search for a feasible path and wavelength for the requested lightpath.

The specific nature of a dynamic RWA algorithm is determined by the number of candidate paths and how they are computed, the order in which paths and wavelengths are listed, and the order in which the path and wavelength lists are accessed.

3. EVOLUTIONARY ALGORITHMS

This chapter includes a brief overview and historical perspective on evolutionary algorithms (EAs), and an introduction to genetic algorithms (GAs). A tutorial introduction to GAs, Goldberg's Simple Genetic Algorithm and GA/heuristic hybrids is given. A survey of evolutionary telecommunications, i.e. the application of EAs to the telecommunications problem domain, is presented.

3.1 Overview of Evolutionary Algorithms

The history of EAs is longer than most practitioners in the field realise. However, this has been remedied in recent times by both a concise history and summary of the state of the art by Back et al. [32], as well as by David Fogel's edited volume of both early and important papers in evolutionary computation [4]. The origin of EAs dates back to pioneering work in the 1950s even before adequate computing power for them was available [32]. In particular, David Fogel [4] has highlighted early research by Friedman [33], Fraser [34], Box [35] and Friedberg [36, 37]. In his 1956 master's thesis, Friedman [33] described how control circuits employing 'basic neural elements' could be evolved for autonomous robots using random selection and mutation. His work anticipated by decades more recent efforts in robot control, evolving artificial neural networks, and evolving electronic circuits. Friedman's research was theoretical -there was no practical implementation- but by analysis of fitness landscapes, he endeavored to show the superiority of his suggested approach over purely random search.

Then, as part of a computer based simulation study of genetic systems in 1957, Fraser [34] introduced an approach which clearly anticipated the bit-string GA later studied by Holland [38], Goldberg [2] and others. Fraser's algorithm employs diploid bit-strings and a multi-point crossover operator for recombination, with different crossing probabilities at different points along the genotype so as to vary linkage between the genes. His algorithm is generational, with a selection mechanism that produces more offspring than required and then removes the excess according to

some function of their phenotype fitness. This foreshadowed the (μ, λ) selection method of evolution strategies (ESs).

Box [35], also in 1957, introduced evolutionary operation as an online approach to improving the productivity of an industrial process. During an agreed period, a systematic set of mutations is applied to the current production settings, with performance data gathered from the running plant. At the end of the period, the best settings found become the new current production settings, and systematic experimentation recommences. While Box's approach has been criticized by others (e.g. Goldberg [2]), it nevertheless represents a successful early application of both mutation and fitness based selection.

Finally, Friedberg et al. [36, 37] in 1958 developed some of the earliest research on the evolution of computer programs. These are represented as binary instructions, and a credit assignment mechanism is used to apportion credit to individual instructions. The approach suffered from a number of limitations motivated and exacerbated by the restricted computing power available to Friedberg et al. Nevertheless, the work clearly had similarities to the later development of both GAs and classifier systems (CSs). Moreover, concepts such as Holland's implicit parallelism and theory [38] were anticipated, the importance of incorporating domain knowledge into the algorithm was recognized, and a number of insights obtained that were subsequently rediscovered by the evolutionary computation (EC) community [4].

3.1.1 Genetic Algorithms (GAs)

GAs were introduced by Holland in 1962 [38, 39], originally intended as a general model of adaptive processes [32], but subsequently widely adopted as optimizers [2, 3, 8]. They are inspired by natural genetic processes, and thus have a population of chromosomes, each analogous to the DNA in a natural organism [40]. The population evolves by simulated evolution, employing genetic operators modeled on those in nature. The overall process is driven by fitness, determined by decoding the chromosome and applying a problem-specific objective function, analogous to the reproductive success of an individual organism matured from its DNA.

Their popularity can arguably be traced back to two extensively cited books by Goldberg [2] and Davis [8] in 1989 and 1991 respectively. As well as a clear description of the basics of GAs, Goldberg [2] provides a complete Pascal

implementation, a survey of applications, advanced operators and techniques, and an introduction to CSs. In his book, Davis [8] describes the step-by-step development of a GA/heuristic hybrid from a simple GA, and then includes a wide-ranging edited collection of application case studies of hybrid GAs.

While the historical distinctions between EAs have blurred [4], GAs are now, by far, the strongest research area. However, to make this assertion clearly, GA/heuristic hybrids [8] must also be regarded as GAs. Some researchers would argue [41], though, that such algorithms bear little resemblance to the canonical GA, and should simply be described as EAs.

3.1.2 Evolutionary Programming (EP)

EP originated with Lawrence Fogel in 1962 [42-44] as an attempt to create artificial intelligence [32]. The population of individuals being evolved is typically finite state machines (FSMs), and the emphasis is on the evolution of behavior from parents to offspring, rather than on modeling genetic operations as in GAs [40]. The inspiration for EP is an abstraction of evolution at the species level, consequently recombination (crossover) is not usually included, although even in the early days, this was not always so [41]. Although originally more focused, EP does not now impose any restriction on the representation of an individual in the population [4, 40]: rather a 'natural' representation for the problem is selected (cf. hybrid GAs). Mutation is stochastic, usually determined such that small variations are common and large variations less so; in addition, adaptation of mutational variation is now often applied during a run. Selection, although originally deterministic, is now typically tournament-based [4].

3.1.3 Evolution Strategies (ESs)

ESs were developed by Rechenberg and Schwefel in 1965 [45, 46], with the aim of solving mainly experimental parameter optimization problems [32]. An individual in the population is an array of real-valued parameters: there is no encoding. ESs were originally based on a 'population' of size one, with the parent competing each time for survival with a single offspring: the (1+1) ES. Now, however, ESs typically use a larger population, size μ. From this, λ offspring are obtained by mutation (and sometimes recombination). The best μ individuals survive deterministically, selected from either the combined pool of parents plus offspring, termed $(\mu + \lambda)$, or just from amongst the offspring, (μ, λ). Mutation of an individual's parameters is Gaussian

(normal) about the current position. In addition, strategy parameters are usually included in the individual, one for each problem parameter; each of these determines the mutation rate (standard deviation) for the corresponding parameter. The strategy parameters are themselves mutated, although according to a global log-normal distribution. While ESs have achieved a measure of success, particularly within civil engineering [40], its popularity has been limited by much of the early literature being in German. This was remedied in 1995 with a book by Schwefel [46].

3.1.4 Genetic Programming (GP)

As has already been noted, efforts to evolve computer programs date back to the earliest days of EC, with the work of Friedberg et al. in 1958 [36, 37]. However, these early machine-code based efforts were not particularly fruitful. Better results awaited more powerful computers, and also more expressive representations. In particular, the FSMs of early EP, the if-then rules of CSs, and the LISP s-expressions of what came to be known as GP have all had a measure of success. While several researchers in the 1980s suggested the use of LISP s-expressions for program evolution, including Hicklin, Fujiki and Cramer [4], it was the work of Koza from 1989 [47] which established the GP field.

In GP [48], an individual in the population is usually a single LISP s-expression (i.e. a parse tree), composed from a user-specified, problem-specific set of functions and terminals. GP crossover, usually the only genetic operator, is based on the exchange of sub-trees. Fitness of the individuals is established by evaluating an individual program, either directly or indirectly, as a solution to the problem. Selection was originally stochastic and fitness-proportional [48], but subsequently tournament-based selection has been preferred [49].

In the last decade, GP has been developed by Koza [49, 50] and others [51-53], and has also diversified to include a wide-range of representations and operators [54]. It has even returned to the successful evolution of machine-code programs [55].

3.1.5 Classifier Systems (CSs)

CSs were introduced by Holland and Reitman in 1978 [56] as an attempt to develop adaptive behavior by evolving a set of classifiers (i.e. condition-message pairs). In a CS, a set of detectors (i.e. the program input) posts messages to a message list [2]. Each classifier (in parallel) is activated when it detects a matching condition, and

competes to post its message. While messages are binary, the classifier conditions can include ‘wild cards' (usually represented as ‘#'), allowing an individual classifier to respond to a range of conditions. The message list in turn can trigger effectors, resulting in program output. Evolution of the set of classifiers is achieved by assigning credit to successful classifiers; they in turn pass some of this credit on to other classifiers that contributed to their success (the ‘bucket brigade'). Periodically, accumulated credit is used as fitness to replace a limited subset of the population with a modified GA, hopefully resulting in an improvement of the overall adaptive behavior of the system as a whole. Subsequent development of CSs have included evolving entire sets of classifiers (the 'Pittsburgh' approach) rather than individual classifiers (the ‘Michigan' approach) [4], and a wide variety of different credit-assignment mechanisms [40].

3.1.6 Evolutionary Hardware (EH)

A relatively recent development in EAs has been the emergence of the EH field. This includes EAs where the evolution occurs in purpose-built processors (rather than in software on general-purpose processors), EAs where the fitness evaluation occurs in hardware (rather than e.g. software simulation), or EAs that combine both of these. It could be argued that EH is not really a distinct field, as the only difference is in the implementation of the algorithms or their fitness evaluation. Indeed, much of the work on ESs could be considered EH. Nevertheless, there is a growing community focusing on this specialized aspect of EAs, and the interested reader is referred to the books by Sipper [57], Thompson [58] and Mange & Tomassini [59].

3.2 Genetic Algorithms

This section provides an introduction to GAs. Its main purpose, following some background on both GAs in general and Goldberg's Simple Genetic Algorithm in particular, is to introduce GA/heuristic hybrids. This latter approach is the central theme of the thesis.

3.2.1 Introduction to GAs

3.2.1.1 Goal of Optimization

When engineers attempt to optimize the parameters of a problem, or search for the global maximum (or minimum) of an objective function, what is their actual goal? In a large and complex problem space, where it may never be possible to find the true global optimum, surely a 'good' solution is sufficient? In thinking about optimizing a problem, it is possible to be biased into thinking that only the true, global optimum is sufficient. However, in complex systems, as in the natural world, a solution that is 'good enough' may suffice. The goal is not so much to find the global optimum solution, but rather a solution that is the best that can be achieved with the time and resources available. It is with complex problems such as these, that the robustness and generality of modern heuristic search methods, such as EAs, becomes apparent.

3.2.1.2 Evolutionary Algorithms

EAs are based on a simplification of the mechanisms of natural evolution i.e. natural selection and genetics. As has already been noted, a variety of different EAs have been proposed [4]. However, they all share a common basis of simulating the evolution of individual structures using selection, reproduction and mutation. These processes in turn depend on the apparent fitness of the individuals in the context of their environment. Thus EAs maintain a population of structures that reproduce under the influence of selection and other genetic operators, such as mutation and, in some cases, recombination. Reproduction focuses attention on high-fitness individuals, thus exploiting the available fitness information. Recombination and mutation modify those individuals, thus promoting exploration of new possibilities. Although EAs are simplistic from a biologist's point of view, nevertheless, they are still sufficiently complex to act as robust and powerful adaptive search techniques [40].

3.2.1.3 Genetic Algorithms

Against this general background of EAs, GAs can be distinguished by two key features. First of all, they operate in a search space using genetic operators on fixed-length strings that encode solutions in the problem space, rather than on the

problem space directly. In addition, they use a particular recombination operator on these fixed-length strings that is unique to GAs, called crossover.

3.2.1.4 Chromosomes

The use of fixed-length strings in GAs is inspired by the encoding of the genetic inheritance of an individual biological organism in the DNA that makes up its chromosomes. The strands of DNA in the cells of, say, a human being, encode its entire genetic heritage in a four-'letter' alphabet. Thus, in a GA, each individual in the population contains a (fixed-length) string (also sometimes called a chromosome), usually, but not always, using a binary alphabet. The use of such a binary representation follows from schema theory [2], which suggests that a two-symbol encoding promotes more effective search.

3.2.1.5 Fitness

The fitness of a biological organism is measured by its interaction with the environment. Its fitness determines (to some extent) how long it will survive, what opportunities it will have to reproduce, and consequently, how many offspring (or children) it will have. In a GA there is not usually an environment as such, but rather the fitness of an individual is determined by decoding its chromosome (i.e. the fixed-length binary string) and thereby interpreting it as a point in the problem space. The (possibly modified) value of the objective function for the problem is then fed back to the GA as the fitness of that individual. The fitness value is then used to determine the extent to which that individual is allowed to reproduce into the next generation.

3.2.1.6 Crossover

What then of the evolution of the population from one generation to the next? As well as reproduction as a function of fitness (which in a GA may simply be fitness-proportional), many biological organisms use sexual reproduction to allow the blending of genetic characteristics from two organisms to be reflected in their offspring, such that at least some of the children will be 'fitter' than either of their parents. Thus in GAs, parents that are selected to reproduce will (with a certain probability) not simply duplicate their chromosomes in their children (i.e. cloning), but instead two parents will recombine their chromosomes using the crossover

operator such that the chromosomes of their children are a combination of both their parents.

3.2.1.7 Mutation

In the natural world, another factor that operates on biological organisms' genetic heritage is mutation -the apparently random modification of some of the genetic material inherited by an organism. Most such mutations would appear to be harmful, but occasionally one would be beneficial, resulting in increased fitness for the organism, and consequently the mutated genetic material is passed on to more and more organisms as the generations pass. Inspired by this, GAs include the possibility of the mutation of a child's chromosome during reproduction by randomly changing (at low probability) each of the bits (or letters for a no binary alphabet) of its chromosome. For GAs though, this operator is regarded more much less important than fitness-proportional reproduction and crossover, and so the probability of any single bit being mutated is usually kept very low [2].

3.2.1.8 Population and Generations

In contrast with natural populations, the population of individuals in a GA is often maintained at a constant size (for simplicity). The initial starting population (which has no analogue in the natural world) is usually randomly generated, although it may be seeded with a few individuals encoding better-than-average attempts at the problem. In addition, many GAs are generational -a new population of individuals is created from the old by reproduction, crossover and mutation, and then the old generation is discarded.

3.2.1.9 Distinctive Qualities

What then are the main differences between GAs and other search algorithms? Goldberg [2] identifies four differences. GAs work with a coding of the problem's parameters, not with the parameters directly. They employ the objective function values themselves, rather than derivatives. They search using a population of points, instead of a single point. Finally, GAs incorporate probabilistic choices, rather than purely deterministic.

Because the problem parameters are coded as finite length (binary) strings, and the GA then operates on these directly using only objective function values, i.e. without

problem-specific information (at least for a 'pure' GA) or even derivatives, a very broad range of problems can be tackled successfully. Although it should be noted that considerable thought can still be required to develop a suitable binary encoding for a complex problem domain. Clearly, using a 'database' of points in the problem space (i.e. the population of strings), rather than a single point, immediately reduces the chance of getting trapped in a local optimum. Finally, the probabilistic choices in the heart of the GA, far from indicating directionless search, are actually used to guide the algorithm to explore areas of the search space most likely to lead to improvement. Further, by re-running a GA several times with the same parameters (apart from the random seeds), the probability of finding a solution of a given quality can be greatly enhanced.

3.2.2 The Simple Genetic Algorithm

3.2.2.1 Overall Algorithm

To make the description of the genetic algorithm above a little more concrete, it is perhaps beneficial to examine Goldberg's Simple Genetic Algorithm (SGA) [2]. An overall flowchart for the algorithm is given in Figure 3.1 (a). After creating the initial population (random binary strings), assessing the fitness of each individual and calculating overall population statistics, the algorithm enters its main loop. Each generation is created from the one before, using selection, reproduction, crossover and mutation. The population statistics are computed again, including calculating average fitness and identifying if a new best-of-run individual has been found. The old generation is then replaced by the new. The algorithm continues to loop until a termination condition, usually a maximum number of generations, is met.

3.2.2.2 Create Next Generation

The 'Create next generation' subroutine can be further expanded, as shown in Figure 3.1 (b). First two individuals are selected from the old population. From these, two new offspring are created which, with a certain probability, are recombined using crossover. Then the new individuals, whether simple clones of their parents or not, are subject to mutation. Finally, the fitness values of the new individuals are determined by decoding them and applying the objective function. The subroutine continues to loop until the new population has been completely filled with offspring.

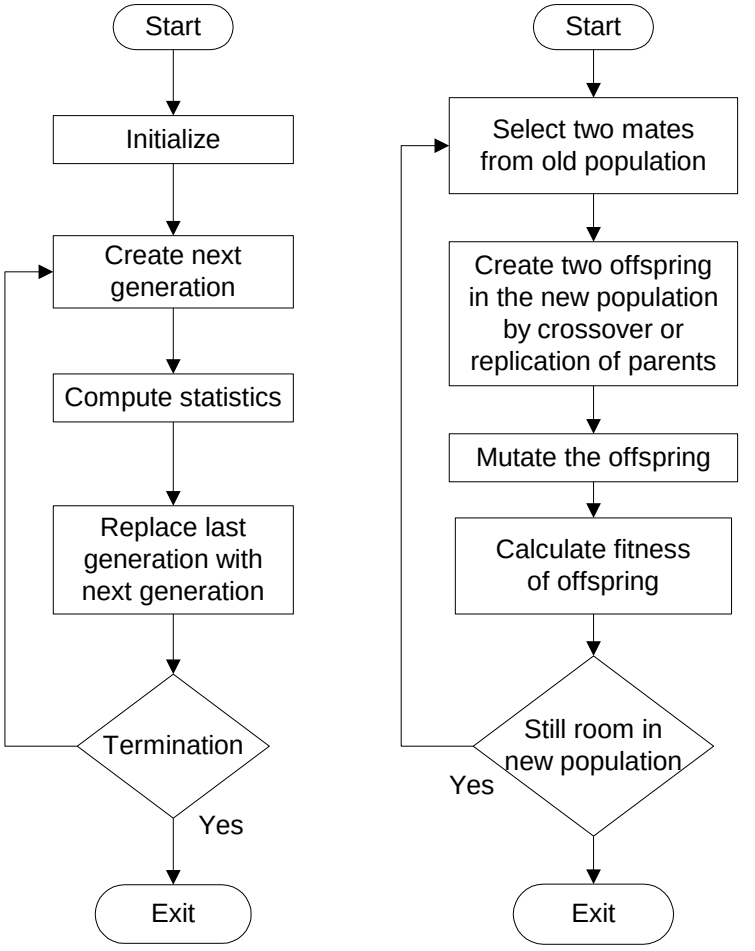


Figure 3.1 Simple Genetic Algorithm flowcharts

Another way of looking at the details of the SGA is in terms of the relationship of one generation to the next (the new population to the old). The SGA is generational, i.e. it uses non-overlapping populations. Each generation is created entirely from the one before, and then the old one is discarded. In contrast with natural populations, the population size is kept constant: each generation has the same number of individuals as the preceding one.

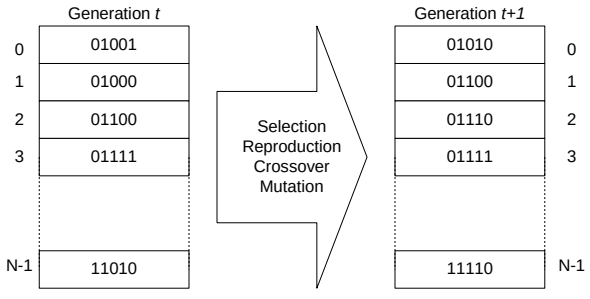


Figure 3.2 From one generation to the next

This is illustrated in Figure 3.2, which shows generation t of N individuals transformed into generation t + 1 by the genetic operators: selection, reproduction, crossover and mutation.

3.2.2.3 Selection & Reproduction

By way of illustration, consider a tiny example population of six strings, each of length 5 bits. The strings encode unsigned integers (x), and the objective function (fitness) is $f(x) = 2x^2 + x$ (Table 3.1).

Table 3.1 Sample problem strings and fitness values

No.	String	x	Fitness ($2x^2+x$)	% of Total
0	01001	9	171	5.2
1	01000	8	136	4.2
2	01100	12	300	9.2
3	01111	15	465	14.2
4	10100	20	820	25.1
5	11010	26	1378	42.1
Total			3270	100

The simplest form of selection is fitness-proportional. The probability of a string being selected is simply its fraction of the sum-of-fitness of the whole population (Figure 3.3). Fitness-proportional selection can be imagined to operate by analogy with a roulette wheel (or a 'wheel of fortune'). For every vacancy in the new population, say, the roulette wheel is spun, and where the ball lands (by randomly selecting a number between 0.0 and 1.0, representing the circumference of the wheel), that individual is reproduced in the new population.

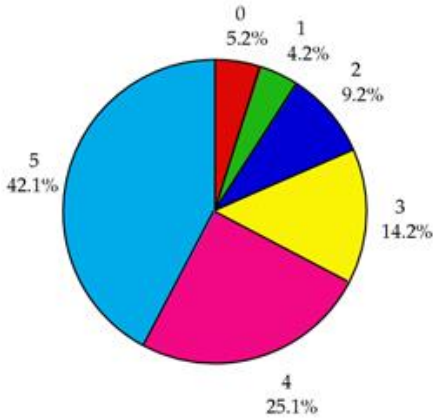


Figure 3.3 Roulette-wheel selection

The chance an individual has of being selected and reproduced is thus proportional to its share of the wheel, and hence to its fitness. Selection is done 'with replacement' i.e. just because an individual is selected once does not mean it cannot be selected again. The wheel remains the same, and is simply spun again until the new population is filled. Therefore, the number of offspring each individual in the old population can expect to have in the new, is just that fraction of the new population that corresponds to its proportion of the total fitness in the old. In practice though, because of stochastic effects, an individual may have more (or less) than the expected number of offspring -in the limit, a very fit individual might have no offspring, or a very unfit individual might have many.

3.2.2.4 Crossover

For every pair of individuals in the new population (which are just clones of their parents at this stage), the crossover operator is applied with a certain probability (typically 0.6). Simple crossover operates by selecting a crossover point at random somewhere along the strings, e.g. if the strings are length k, a crossover point j would be selected between 1 and k-1. All the bits between 0 and j-1 inclusive would remain unaffected, but the bits between j and k-1 would be swapped between strings (Figure 3.4). Between two strings from the tiny example population, say:

0 1 1 0 | 0

0 1 1 1 | 1

with k = 5 and j = 4 (indicated by the '|'s), the resulting crossover would yield:

0 1 1 0 1

0 1 1 1 0

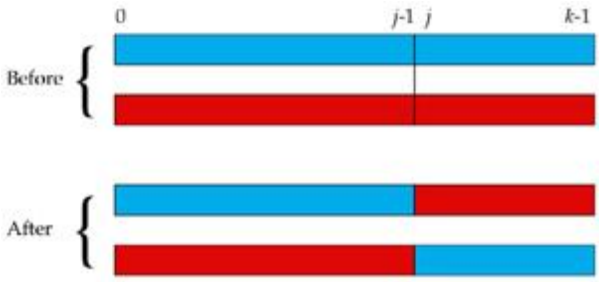


Figure 3.4 Crossover

3.2.2.5 Mutation

For GAs, fitness-proportional reproduction and crossover are the primary genetic operators, but something more is still needed. As a result of selection and crossover, the population will gradually converge on more promising areas of the search space. However, in any given generation, all members of the population may come to have the same bit value (say 1) at a particular bit position -perhaps all those individuals with a 0 at that location proved unfit and died out. But what if later in the evolution of the population, better fitness could be found by incorporating a 0 at that bit position (as other locations have changed over time)? Selection and crossover alone can never bring back a 0 at that bit position -hence the need for mutation. In a binary string alphabet, mutation simply consists of, at low probability (say 0.001), toggling the bit at each position in a new individual -from 0 to 1 or from 1 to 0. For non-binary alphabets, some variation must be used -perhaps selecting a letter at random from the alphabet, excluding the letter currently at that location. Although mutation brings needed diversity back into a population that might otherwise converge prematurely on less than ideal points in the search space, it must be used sparingly [2]. Just as mutation in the natural world is often harmful, so mutation in GAs often reduces the fitness of the individuals affected -it is after all just a form of random walk.

3.2.2.6 Improving the SGA

Although the SGA is a reasonable starting point, a wide variety of ways have been suggested to extend it [2]. These are usually inspired by either additional aspects of natural evolution, or by integrating other search algorithms or domain knowledge into the GA. Instead of fitness-proportional selection, for example, fitness ranking can be used, where an individual's rank in the fitness-sorted population determines its selection probability, rather than its absolute fitness value. Another possibility is tournament selection where, whenever an individual is to be selected, a small number of structures, chosen randomly, but not according to fitness, compete directly in terms of their fitness values for the right to mate.

Alternatively, rather than abandoning fitness-proportional selection, the fitness values themselves can be suitably modified to maintain adequate selection pressure and avoid premature convergence. Variations include windowing, i.e. rescaling fitness values using the history of the last few generations, or using linear, sigma truncation or power-law rescaling mechanisms.

The SGA is strictly generational, but this too can be altered. One, or a few, of the better individuals can be passed unchanged into the new population (called elitism). At the other extreme, the population as a whole can be left largely unchanged, and only a few (usually the worst) individuals replaced (known as steady state reproduction).

Modified or additional genetic operators can also be introduced. Instead of single-point crossover, used in the SGA, multi-point (usually two-point) crossover can be employed. In the limit, uniform crossover can be used to create an offspring by taking a bit at random, for each position along the string, from either parent.

The possibilities for extending the SGA are considerable, of which just a few have been mentioned, but some caution should be exercised. While for any particular problem, some of these changes will be beneficial, others may prove counter-productive. The next section describes a more promising alternative.

3.2.3 Genetic Algorithm/Heuristic Hybrids

3.2.3.1 Incorporating Other Search Algorithms

Most of the extensions to the SGA mentioned above were biologically inspired. What about making use of other optimization techniques or domain-specific knowledge? One straightforward approach is based on the recognition that GAs are good for generalized search, but having located optimal or near-optimal regions in the search space, can find it difficult to locate the optimal point itself. Calculus, greedy algorithms and problem-specific heuristic search can, however, be very good at local search. By combining GAs and local search algorithms, the overall hybrid algorithm can outperform its individual components –"the genetic algorithm finds the hills and the hill-climber goes and climbs them" [2]. There are two main approaches. Either run the GA for a reasonable period and then run the local search algorithm on the best (or better few) individuals (Figure 3.5 (a)), or incorporate the local search into the overall algorithm (Figure 3.5 (b)). In the latter case, the local search can itself be used as an operator (alongside crossover and mutation, say), with the results of the local search re-encoded (if necessary) as an offspring -essentially acting like an enhanced mutation operator (i.e. Lamarkian evolution). Alternatively, local search can be applied before function evaluation, but no change made to the string itself (i.e. Baldwinian evolution). The fitness of each individual is simply its fitness after the application of local search each time. Because the results of the

local searches are only fed back into the population in the form of fitness values, not actual strings, this approach (at the expense of considerable additional processing) helps to reduce the risk of a super individual being found by local search, whose genetic material would then dominate future generations, leading to premature convergence.

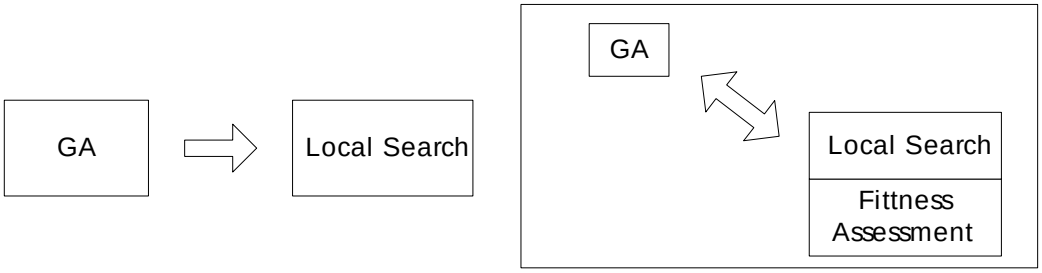


Figure 3.5 Incorporating local search

3.2.3.2 Domain-specific Knowledge

As discussed above, one way of incorporating problem-specific knowledge into a GA is by, say, adding an enhanced mutation operator, which decodes a string to a point in the problem space, applies a local search heuristic, and then re-encodes the result. This approach is a comparatively minor extension to a traditional GA, that nevertheless can bring considerable performance enhancement. However, it is possible to go a good deal further! There is a pragmatic 'school of thought' in GA research [8] that sees GAs solely as a means to the solution of engineering problems. Here, the approach is to adopt a problem-specific encoding, and then develop and employ problem-specific operators that together combine the best existing heuristics for the problem with an overall GA framework -a hybrid algorithm. The motivation for this approach is simple, according to Davis [8]: "Although genetic algorithms using binary representation and single-point crossover and binary mutation are robust algorithms, they are almost never the best algorithms to use for any [specific] problem". It is almost always possible to develop a problem-specific GA/heuristic hybrid that will outperform either the heuristic(s) or a traditional GA alone.

3.2.3.3 Domain-specific Encoding

Rather than using a binary string, the encoding chosen is the same as the current algorithm (i.e. the best known algorithm used on the problem before the development of the hybrid) [8]. However, if the problem were a new one, then a 'natural' encoding would be used. For example, if the problem is to optimize the ten real-valued parameters that control the overall design of an airplane, then an individual in the GA would simply consist of a list of ten real numbers, perhaps constrained to the physical limits of the parameters. By contrast, in a traditional GA, each parameter would be quantized and encoded as a binary string, with an individual consisting of the concatenation of the ten encoded individual parameters. In other problems the contrast might be even more marked, especially where the natural encoding was, say, two-dimensional or some complex pointer-based structure in computer memory.

3.2.3.4 Domain-specific Operators

The method then proceeds to developing domain-specific versions of the traditional crossover and mutation operators [8]. For the airplane example, one offspring might be constructed by selecting parameters at random (with equal probability) from either parent; the other offspring would get the complementary set (uniform crossover). Another possibility might be average crossover, with one offspring generated by averaging the values of the corresponding parameters in the two parents. Possible mutation operators include simply choosing a single parameter at random and giving it a random value (within established limits); or each of the parameters on an individual (with a certain probability) could be modified by a small random amount - real-number creep. Finally, operators based on existing domain- or problem-specific heuristics can be developed and incorporated as enhanced mutation operators, as was described previously for the traditional GA. In terms of the airplane example, these might include well-established relationships between, say, wing and tail area, or overall engine power and fuselage length-whatever could be obtained from domain experts and existing design software. However, the choice of encoding and the development of problem-specific operators require considerable ingenuity. It is an iterative process, involving online experimentation, consultation with domain experts, and experience drawn from the growing body of literature on hybrid algorithms. Clearly, efforts to improve a hybrid algorithm should only continue as long as the fitness of the problem solutions discovered by the hybrid remains less than required.

3.3 Applying Genetic Algorithm/Heuristic Hybrids

In this section a brief overview is given of some prior work from other areas using GA/heuristic hybrid algorithms, the solution approach adopted in the thesis, and the basics of which were explained above.

In 1989, Jog et al. [60] tackled the traveling salesman problem (TSP) using a generational GA/heuristic hybrid algorithm, and building on a number of even earlier papers in this area. Their algorithm used the usual permutation representation, but employed both an heuristic crossover operator and two local-search mutation operators applied with fixed operator probabilities. Their experiments focused on the effects of including or excluding operators from the hybrid, as well as varying the population size. They concluded that larger population sizes, using problem-specific crossover and including local-search heuristics of increasing sophistication were all beneficial to the final solution quality.

An early and influential reference in this area, as already noted, is the Handbook of Genetic Algorithms edited by Lawrence Davis in 1991 [8]. After a step-by-step development of a GA/heuristic hybrid from a simple GA, the book includes a wide-ranging collection of application case studies of (mainly) hybrid GAs. These include the parametric design of aircraft [61], dynamic anticipatory routing in circuit-switched telecommunications networks [62], robot arm trajectory generation [63], international security modeling [64], airplane control strategy design [65], neural network design [66], industrial plant control [67], multiple-fault diagnosis [68], conformational analysis of DNA [69], interpretation of passive sonar traces [70], engineering design optimization [71], scheduling [72] and the TSP [73]. Between them, the case studies employ a wide variety of representations and operators.

For example, the chapter by Bramlette & Bouchard [61] from [8] concerns the parametric design of a modern fighter aircraft. Their design required eight parameters to be optimized, including fuselage width, height and length, wing loading and engine size. They investigated not only an integer-coded generational GA, but also a wide variety of stochastic approaches including various types of SA, hill climbing and random search, used both individually and in combination. Overall, they found that iterated SA, and a hybrid GA using stochastic hill climbing as the mutation operator, obtained the best results on their test problem.

Also in [8], Davidor [63] developed trajectories for robot arms using a hybrid GA which represents the trajectories as order-dependent variable length strings of (intermediate) arm positions, with each position specified by the inter-link angles in the arm configuration. A problem-specific two-point crossover is introduced which chooses two random arm configurations as crossover points in one parent. It then locates the two most similar configurations in the other parent. Then the configuration sequences between these points (and including the second crossover point) are exchanged between the parents to form the children. Mutation is guided by problem information to focus on those points in an arm trajectory that are least fit. In addition, two further problem-specific mutations are used that can add (duplicate) and delete arm configurations from the trajectories. In experiments on one particular target trajectory, Davidor's hybrid GA performed well in comparison with both random search and hill climbing, and was found to be further improved by directly incorporating hill climbing within the GA.

As a third example from [8], Grefenstette [65] sought to develop strategies for sequential decision tasks in a dynamic system, specifically evasive maneuvers allowing a plane to avoid a pursuing missile. Within his generational GA, each strategy is represented by a set of condition-action rules that relate the sensor inputs (missile range, missile bearing, missile speed, etc.) and the system response (i.e. the turning rate of the plane). The problem-specific crossover operator exchanges complete rules between individuals, whilst trying not to separate successful rule sequences. Of the two mutation operators provided, one specializes existing rules, and the other randomly modifies individual clause within a rule. In addition, the GA can be seeded with rules generated by domain experts that are combined with randomly generated rules within the individual strategies that form the initial population. In experimental runs, the GA performed well providing a high probability of missile evasion, although no direct comparison was made by Grefenstette with other non-trivial approaches.

More recently, in their paper, Schnecke & Vornberger [74] address constrained placement problems -both the facility layout problem and VLSI macro-cell layout generation- using a hybrid GA. Their steady-state algorithm uses a binary slicing-tree representation to express the relative placement of the blocks, three problem-specific mutations, and a form of gene-pool recombination. It employs a multi-population approach, with each sub-population using parameter adaptation, including the operator probabilities. Their hybrid GA produced competitive results for

VLSI macrocell layout generation, without the use of sophisticated routing algorithms, and obtained the best results yet reported on large facility layout problems.

A hybrid GA for computer aided process planning for job shop machining, considering operation selection, machine selection, setup selection, cutting tool selection, and operations sequencing simultaneously, was presented by Zhang et al. in [75]. Their representation was an ordered list of operations, each one providing details of the corresponding machine, tool and tool approach direction choices. A problem-specific form of cyclic crossover and three mutation operators for machines, tools and tool approach direction were applied using fixed operator probabilities. For the one complex problem examined, Zhang et al. found that their hybrid GA was superior to a heuristic approach.

An adaptive evolutionary planner/navigator for mobile robots was described by Xiao et al. [76] in 1997. Their system uses the same GA/heuristic hybrid algorithm for both offline planning and online navigation. The steady-state hybrid represents paths as a linked list of knot points (i.e. intersection points) between straight line segments. It uses a problem-specific crossover and seven mutation operators, with their probabilities adapted during the course of a run. Their system has been developed over a period of time, and demonstrates robust online and offline performance over a wide range of challenging environments. Their work was subsequently adapted by Smierzchalski & Michalewicz [77] for modeling ship trajectories in collision situations. The path representation was extended to include ship speeds on the line segments, the concept of time introduced to allow for moving obstacles, due allowance made for dynamic as well as static constraints, and a new speed mutation operator added. However, apart from these, the earlier planner/navigator was largely adopted unchanged, except that for simplicity Smierzchalski & Michalewicz omitted operator probability adaptation, preferring instead to use equal fixed operator probabilities.

In 2000, Magyar et al. [78] described a GA/heuristic hybrid for the three-matching problem. This problem requires a set of nodes to be clustered in disjoint groups of three (i.e. triplets), with the cost of each triplet simply that of the minimum-spanning tree required to interconnect its nodes, and the overall cost taken to be the sum of the triplet costs. Magyar et al.'s hybrid uses a permutation representation and incorporates several problem-specific recombination and local-improvement operators. It employs operator probability adaptation, and the results obtained were

better than those of a local-search heuristic, SA and a grouping GA over twenty large problem instances.

Although this section has focused on GA/heuristic hybrids, it should perhaps be mentioned that many researchers have hybridized GAs with other modern search heuristics. As one example, a paper in 1999 by Bhandarkar & Zhang [79] addresses the problem of image segmentation using EC. As well as developing a problem-specific generational GA, Bhandarkar & Zhang hybridize this with three different stochastic annealing algorithms in turn: SA, microcanonical annealing (MCA) and the random cost algorithm (RCA). In the GA, a two-dimensional integer representation is used, combined with problem-specific crossover, mutation and a local-search operator. All three operators follow deterministic changes in their operator probabilities during a run, with local search and mutation increasing, and crossover decreasing. The hybridization is achieved by wrapping a stochastic annealing replacement mechanism around the GA framework (excluding the local search operator). Thus children that had been created using GA crossover and mutation only replace their parents if they satisfy the appropriate annealing condition. Overall, they found that the three hybrids all produced higher quality results than the GA, particularly so for the SA-GA combination.

Overall, as in evolutionary telecommunications, the GA/heuristic hybrid approach has been readily applied to a wide range of problem domains, only a few of which have been considered here. Researchers have shown considerable ingenuity in developing problem-specific algorithms for difficult design and optimization tasks, and the GA/heuristic hybrid approach clearly has considerable promise.

3.4 Operator Probability Adaptation

One aspect of GA/heuristic hybrid algorithm explored in this thesis is its potential use of operator probability adaptation mechanisms. Consequently, in this section some attempts to classify the many approaches to parameter control in EAs will be mentioned, as well as briefly examining a selection of prior work, taken from areas other than evolutionary telecommunications, employing such mechanisms.

A detailed survey and classification of parameter setting was recently completed by Eiben et al. [80]. They based their work in part on their own earlier, although more limited, classification scheme [81]. In [80], they argue that there are only two main criteria for classifying adaptation mechanisms: what is changed, and how the

change is made. For the first, they considered parameters controlling representation, evaluation function, mutation operators and their probabilities, crossover operators and their probabilities, parent selection, replacement operator (survivor selection) and the population itself as potential targets for adaptation. As to how the changes are made, they divided parameter setting into two main areas: parameter tuning and parameter control. In the former case, parameters remain fixed throughout the course of the EA run, although they can be tuned between runs by the experimenter. In the latter approach, however, the parameter values change during the run, with this mechanism further divided into three categories of parameter control. In the deterministic method, the changing parameter value follows a preset schedule -there is no feedback from the evolutionary process. With adaptive control, there is feedback, but this only changes the parameter value using some heuristic, rule-based approach. The final possibility is self-adaptation -the parameter values participate in the ongoing evolutionary process; thus the adaptation mechanism is itself evolutionary.

An earlier and more focused classification by Tuson [82, 83], considered only operator-probability adaptation algorithms. In his study, he classifies these into two groups: co-evolutionary methods (equivalent to Eiben et al.'s self-adaptive parameter control), that encode operator probabilities into each member of the population; and learning-rule methods (equivalent to adaptive parameter control), that adapt the probabilities based on measurement of the operator productivities. Turning now from surveys to individual papers employing operator-probability adaptation, an early paper by Fogarty [84] studied varying mutation probability for an industrial optimization problem, i.e. minimizing combustion stackloss in the common flue of multiple burner furnaces. He compared two fixed mutation schemes (one of which applied reduced levels of mutation to more significant bits) with two corresponding deterministic schedules that gradually reduced mutation throughout the run. He found that deterministic variation of the mutation probability improved the GA's online performance in both cases when starting from a conservative initial population.

Davis [85] presented an operator-probability adaptation mechanism incorporated into a steady-state GA/heuristic hybrid with an integer representation and five problem-specific operators. The adaptation mechanism was found to provide improved performance over both naive and tuned fixed probabilities, and to be

competitive even with a deterministic schedule derived from a substantial number of experimental runs.

Davis subsequently described his approach in more detail in [8]. The operator productivity is computed by crediting operators with any improvement over the best member of the population; a decreasing fraction of this credit is also passed back to the operators responsible for creating the individual's parents, and so on, to a given depth. After a defined interval, the productivity of an operator is taken to be its total credit over the interval divided by the number of individuals it has created. Operator probabilities are then adjusted by reassigning a fraction of their probabilities in proportion to their productivity. Davis also mentions the possibility of preventing an operator's probability falling below an assigned minimum value, as well as the potential of his approach to measure the productivity of individual problem-specific heuristic operators. Rather than using a credit-assignment approach, in 1994 Srinivas & Patnaik [86] adapted both crossover and mutation probabilities in a generational binary-string GA such that the smaller the gap between the best-in-population fitness and the average fitness, the higher the operator probabilities. However, to preserve the better individuals, the probabilities are also reduced the nearer an individual is to the best-in-population fitness. In particular, the mechanism ensures the best individual is (almost) always preserved unchanged (although a small background level of mutation prevents premature convergence). Their experimental results show that across a range of test functions their adaptive algorithm was in most cases significantly better than a GA with fixed operator probabilities, showing both more rapid convergence and a reduced tendency to get trapped in local optima.

Julstrom [87] presented ADOPP, an operator-probability adaptation mechanism for steady-state GAs, in which all operators back to a given depth in an operator tree are credited with a decreasing fraction of any improvement over population median fitness. At all times, actual operator probabilities (and not just a fractional part) are determined by their recently earned credit. However, his work was limited to two-operator GAs, and provided no comparison with fixed operator probabilities.

Tuson & Ross [83, 88] describe an investigation of Corne's COBRA (COst Based operator Rate Adaptation) algorithm, which instead of adjusting individual operator probabilities, assigns operators to a fixed set of probabilities according to their operator-productivity rank over a given interval. The operator productivity is determined by the average improvement of a child individual over its parent(s).

Although COBRA has shown itself to be beneficial for timetabling problems, Tuson & Ross found it provided only equal or worse solution quality over a wide range of other test problems, compared with carefully chosen fixed operator probabilities (for both steady-state and generational GAs). They concluded that operator productivity (at least as Corne defines it) may misguide COBRA as, in some cases, although an operator's productivity is high, its preferred probability is low.

Xiao et al. [76] describe an adaptive evolutionary planner/navigator for mobile robots. Their adaptation algorithm not only takes account of improvements in path fitness over its parent(s), but modifies this by the time taken to execute an operator and any change in the number of line segments in the path (as more line segments require both more memory and processing time). Operator probability in one interval of a given number of trials directly reflects the operator's relative figure of merit over the previous interval. Overall, they found this adaptive algorithm gave better results than either equal or pre-tuned fixed operator probabilities.

Magyar et al. [78] incorporated an operator-probability adaptation method into their GA/heuristic hybrid for the three-matching problem. Their GA is generational and the operators obtain credit by improvements of the children over their parents, but they are also penalized to an equivalent extent if there is no improvement. However, no credit is given to chains of operators, although the credit (or debit) obtained by an operator is moderated by the time taken to execute it. The actual probabilities are adjusted using an exponential moving average similar to Davis' scheme [8]. Whilst Magyar et al. did obtain slightly poorer results than with a non-adaptive, but finely-tuned, set of fixed operator probabilities, this was only by a very small margin, and at a considerable saving in computer time over the substantial effort involved in parameter tuning. Overall, while a wide range of different mechanisms have been proposed for perator-probability adaptation, in general researchers seem to consider such a mechanism advantageous for GAs. In addition, although this approach has not always proved superior to carefully pre-tuned probabilities, nevertheless the advantages in computer time are clear, with one run using an adaptive mechanism producing results (almost) equivalent to the many runs required for parameter tuning.

4. HEURISTICS USED FOR WAVELENGTH ASSIGNMENT IN WDM NETWORKS

This section provides details of four overall heuristics for wavelength-path (WP) routing, fiber choice and wavelength allocation, plus one for virtual-wavelength-path (VWP). They have all been implemented for making comparison with the proposed GA/heuristic hybrid technique, and the corresponding classes are listed in Table 5.3 and illustrated in Figure 5.3.

4.1 Wuttisittikulkij & O'Mahony's Study

Wuttisittikulkij and O'Mahony [28] have developed a simple, fast and non-iterative WP allocation heuristic, which works with k-lowest hop count paths 1 (and multiple fibers) to obtain a low network wavelength requirement (NWR).

First, for all source-destination node pairs, the k-lowest hop count paths between them are determined. Then, in the basic (unsorted) variant of the heuristic, the node pairs are considered in order of their end-node ids. For each wavelength channel required, the path currently with the lowest free wavelength end-to-end is selected from the choice of k paths. If there is more than one path with the same lowest wavelength free then, of these, the one with the lowest hop count is chosen. However, if there is still not a unique choice, then a random selection is made between all those with the lowest wavelength and hop count.

For the sorted variant, rather than simply allocating in order of their end-node ids, the node pairs are instead pre-sorted in decreasing order of the minimum number of hops between them. It was hoped by the paper's authors that this might improve the resulting NWR, as paths with a greater number of hops are often found to be more difficult to allocate.

Their heuristic has been implemented as class WaveWutti95, the parameters for which are given in Table 4.1. The first of these (ww95.theK) is simply the constant k. The third, ww95.firstFiberOnly, occasionally avoids the need to edit the network file from which the wavelength-routed network is built: if true, the router uses only the

first fiber on each cable in the network. This option can be useful if a single-fiber network design is under consideration. The remaining parameter, ww95.sortPaths, determines whether the paths are allocated in hop-count order or not.

Table 4.1 WaveWutti95 parameters

Parameter	Default	Description
ww95.theK	4	Number of alternative paths, k
ww95.sortPaths	False	Allocate the paths in hop-count order?
ww95.firstFiberOnly	True	Use only first fiber on each cable?

4.2 Wauters & Demeester's Study

In [306], Wauters and Demeester presented two WP allocation heuristics. The first of these is a non-iterative algorithm based on a version of Dijkstra's algorithm modified such that in "extend[ing] an already found part of a route, only those [cables] are considered which have a wavelength channel free which is available on all [cables] of the already found part of the route". Although not stated in [29], the allocation presumably proceeds in arbitrary order of source-destination node pairs, such as according to end-node ids, and one wavelength channel at a time. This heuristic results in a low-cost allocation, whilst also keeping NWR low.

Their second algorithm (HRWA), is based on iterative application of a modified version of Yen's k-shortest path algorithm [30], and aims to produce the lowest cost allocation that still achieves minimum NWR. The modification to Yen's algorithm required can be achieved by simply incorporating as a sub-routine the modified version of Dijkstra's algorithm already referred to above. An overall flowchart for the HRWA heuristic is given in Figure 4.1.

The HRWA heuristic starts (Step 1a) by determining the k-shortest paths for each node pair using the modified version of Yen's algorithm. For each wavelength channel, in arbitrary node-pair order, that path is selected with the lowest free wavelength end-to-end; if there is more than one, the shortest amongst these is chosen (Step 1b). Then, the first wavelength channel (i.e. wavelength-routed path; p_{max}) in the network that uses the highest wavelength number (λ_{max}), found by searching the cables in order of their end-node ids, is located (Step 2). Two different attempts are then made to move this wavelength channel to a lower wavelength number. The first (Step 2a) attempts to reroute it on a lower wavelength number

(and the lowest available end-to-end) using one of the k-shortest paths (and the shortest path is selected if there is more than one at the lowest available wavelength

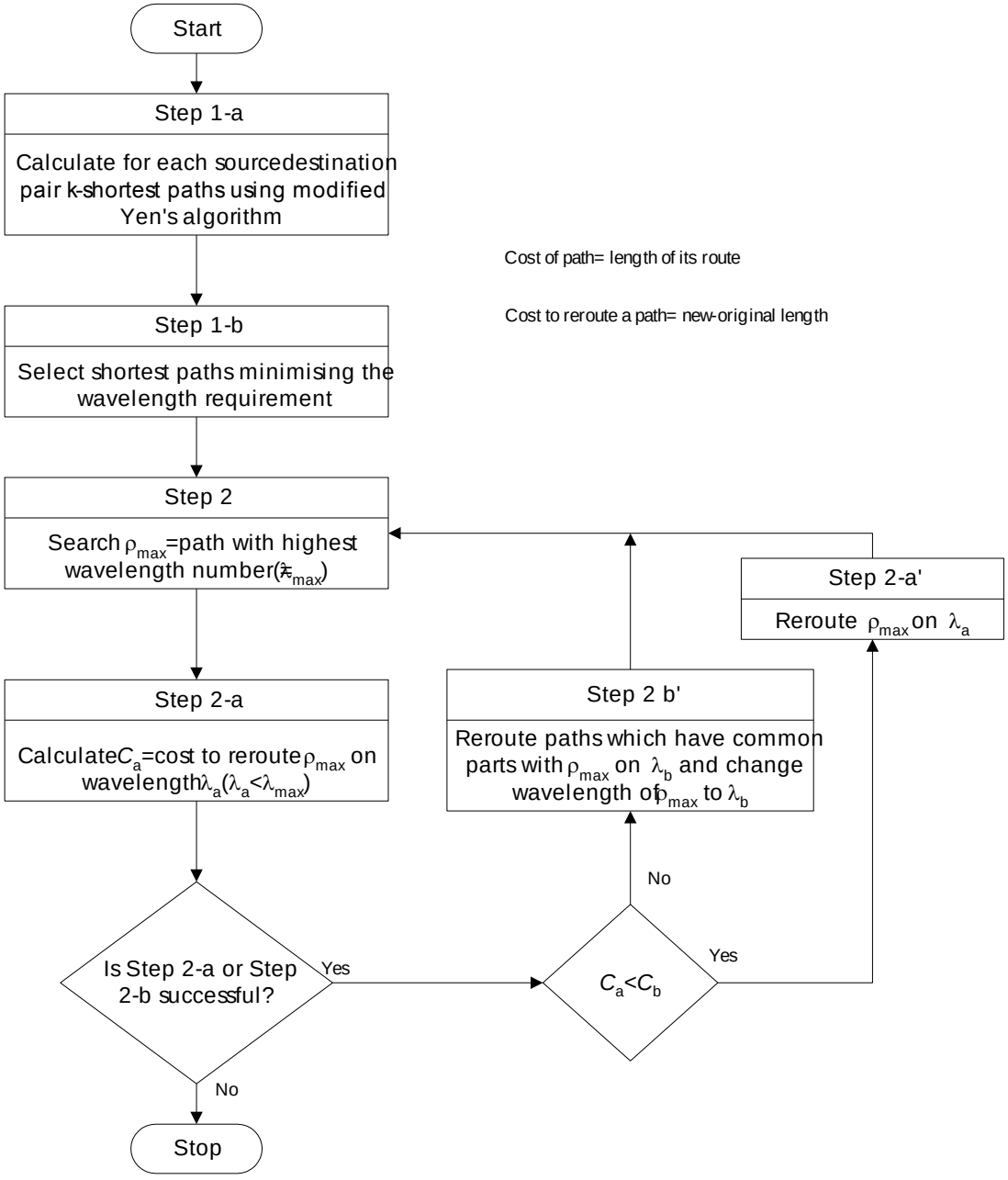


Figure 4.1 Flowchart of HRWA

number). The second (Step 2b) attempts to shift it to a lower wavelength number by shifting out (rerouting) all the paths blocking that particular wavelength number end-to-end. The wavelength chosen is the one that results in the smallest increase in overall path length, given the constraint that all rerouting must be to lower wavelength numbers than the original wavelength of the channel being shifted down. Both these steps have been selected as wavelength allocation operators, and they are described in the previous section and illustrated by examples in Figure 5.5.

If both Step 2a and 2b are feasible, then the one that causes the least growth in overall network path length is selected. Having changed the network allocation, the heuristic then proceeds to another iteration in the hope of further improvements. It should be noted however that the modified k-shortest paths have to be recalculated, when required, each iteration due to the effect of the reallocations. If at any point no improvements can be made, the algorithm terminates.

Both Wauters & Demeester's heuristics have been implemented as classes WaveWauDijk and WaveWauHRWA respectively, the parameters for which are given in Tables 4.2 and 4.3. These parameters are directly equivalent to those used by WaveWutti95, and similar considerations apply.

Table 4.2 WaveWauDijk parameter

Parameter	Default	Description
wadi.firstFiberOnly	True	Use only first fiber on each cable?

Table 4.3 WaveWauHRWA parameters

Parameter	Default	Description
wahr.theK	4	Number of alternative paths, k
wahr.firstFiberOnly	true	Use only first fiber on each cable?

4.3 Nagatsu et al.'s Study

In contrast to other approaches, an overall allocation heuristic for VWP networks is described by Nagatsu et al. [31]. This is a two-stage algorithm, which first establishes reasonable initial routes, and then improves these in a second iterative rerouting stage, aiming for minimum NWR.

The initial routing (Step 1) is achieved by first determining both the minimum number of hops and the number of wavelength channels required between each source-destination node pair. Individual channels are then routed in priority order, with the highest priority given to the node pair with the greatest product of minimum hop count and channels remaining to be routed. The routing itself is achieved by maintaining a weight associated with each cable; this is simply the number of channels (paths) using it. A channel is actually routed by assigning it to the path with

the lowest sum of link weights using Dijkstra's algorithm; any affected weights are then updated.

Once all the channels have been given an initial route, rerouting can begin (Step 2). For each iteration, two different attempts are made to improve the eventual NWR. Both involve identifying those cables (called maximum-accommodating cables) with a fiber carrying the highest number of channels, and which will therefore have to use the highest wavelength number after allocation. First, those paths which use the largest number of maximum-accommodating cables are identified as candidates, and an attempt is made to reroute them using a minimum-link-weight path. If the rerouted path would be on a lower wavelength than the original, then it is adopted, otherwise the original path is restored. If, after trying all candidates' paths, no improvement is achieved, the second attempt is made. This time, any path which uses one or more maximum-accommodating cables is a candidate, and each one is rerouted onto a new path that contains as few as possible of the maximum-accommodating cables it was originally using (and not including, of course, any maximum-accommodating cables that were not part of the original path). Throughout, Nagatsu et al. impose an upper limit on path length: no path is ever adopted which is more than, say, 2 hops longer than the minimum-hop-count path between that node pair. In addition, they impose a limit, say 50, on the number of iterations of the second stage.

Finally, wavelength assignment is simply achieved by a greedy method, which assigns channels to the lowest available wavelength on any fiber of a cable.

In the same paper [31], Nagatsu et al. also described a WP allocation heuristic derived straightforwardly from their earlier VWP one. They argue that the routes used by the VWP paths they obtain would also make excellent choices for the equivalent WP-routed network. Consequently, after their VWP algorithm has terminated, all the routes used by the network paths are adopted unchanged, but their associated fiber and wavelength allocations discarded. Then, starting from the lowest wavelength number, each path in turn is checked to see whether it can be allocated end-to-end on that wavelength. After each iteration, the wavelength number is incremented, and all the remaining paths checked again. The algorithm terminates when the last path has been successfully re-allocated.

Both of Nagatsu et al.'s heuristics have been implemented as classes WaveNagVWP and WaveNagWP respectively. The latter has no parameters of its

own, being based on the VWP heuristic, so both share the parameters given in Table 4.4. The first of these, as before, is used to limit the heuristic to just the first fiber of each cable. The other two correspond to the two parameters of the VWP heuristic discussed above.

Table 4.4 WaveNagVWP parameters

Parameter	Default	Description
wnvw.firstFiberOnly	true	Use only first fiber on each cable?
wnvw.step2Count	50	Maximum iterations of Step 2
wnvw.hopIncrMax	2	Maximum allowed increase in path hop count over minimum achievable

5. THE PROPOSED GA/HEURISTIC HYBRID TECHNIQUE

This chapter presents key details of genetic algorithm/heuristic hybrid approach for routing and wavelength assignment in WDM networks. Our approach is to incorporate advanced heuristics into an overall genetic algorithm. Our GA/heuristic hybrid approach adopts a problem-specific encoding based on object-oriented manner, and then develops and employs problem-specific operators that together combine the best existing heuristics for the problem with an overall GA framework. It is almost possible to develop a problem-specific heuristic/GA hybrid that will outperform either the heuristic approach or a traditional GA alone, and this is the approach that will be used here.

5.1 Cost Model to be Considered

Many researchers [23, 28, 31, 89-91] have considered a low network wavelength requirement (NWR) (i.e. the number of distinct wavelengths in the network) to be desirable as a design objective for wavelength routed networks. This is due to the technological limitations, and resulting cost implications, of carrying more than a small number of wavelengths over national or even international distances [92].

The actual metric described here also includes a penalty [93, 94] added for each unsatisfied traffic requirement (i.e. wavelength channel that remains unrouted), to avoid the false minimum of carrying no traffic at all, which would otherwise have a network wavelength requirement of zero. A suitable penalty would be a number of wavelengths comparable with, but smaller than, the expected network wavelength requirement (e.g. 5 wavelengths for an expected network wavelength requirement in the range 6-10). This provides a balance between allocating all traffic requirements by the end of the (evolutionary) algorithm run, while avoiding undue focus on early satisfaction of the constraints. Also an additional per fiber penalty has been introduced, to penalize a network design with more than one fiber using the highest wavelength number. For every fiber using the current highest wavelength, a small penalty (say 0.01 wavelengths) is added to the overall assessment. For example, this enables two network designs to be distinguished, both with a wavelength

requirement of say 8, but one with 5 fibers using the highest wavelength, and the other only 2. The first would have an assessment of 8.05, the second, 8.02, indicating the latter's superior quality. The selected value of per-fiber penalty should be such as to avoid increasing the assessment by a full wavelength, even if many fibers use the highest wavelength number. A value comparable to the inverse of the total number of fibers in the network is recommended.

Network wavelength requirement was presented as a suitable design metric for wavelength-routed networks above. However, an objective that consists of a simplified assessment of the cost of the whole network including, at an appropriate level, the cost of exceeding the minimum network wavelength requirement, is arguably superior, and is the metric used in this study. Our network cost model is specific to wavelength-routed all-optical transport networks [95]. It should be noted that although, in assessing similar designs, others have included failure restoration [90, 96] and restrictions on the number of wavelengths per fiber [96, 97], for simplicity, these are not considered here.

Separate models for both links and nodes are required to determine the cost of a network. The intension is to approximate the relative contribution to purchase, installation and maintenance costs of the different network elements, while ensuring that the model is not too complex for use in the inner loop of a design procedure. Given the network path, the capacity of link (*i, j*) is taken to be:

$$V_{ij} = \lambda_{ij} V_{\lambda} \qquad (\mathbf{5.1})$$

where λ_{ij} is the total number of wavelengths in use on the link (i.e. across all its fibers) and V_{λ} is the granularity of the wavelength channels (i.e. 10Gbit/s). The wavelength-requirement capacity of link (*i, j*) can be formulated as:

$$V_{\lambda,ij} = \lambda_{req,ij} F_{ij} V_{\lambda} \qquad (\mathbf{5.2})$$

where $\lambda_{req,ij}$ is the wavelength requirement of the link (*i, j*) (i.e. the highest wavelength number of any fiber on the link) and F_{ij} is the number of fibers in use on link (*i, j*). The wavelength-requirement capacity is thus the capacity that would result if all the dark wavelengths on all (partially used) fibers of the link were filled, up to the current link wavelength requirement. As capacity is one of the elements in the link cost calculation, wavelength requirement capacity thus allows the additional costs of uneven fiber loading and dark wavelengths to be assessed in equivalent capacity terms, and thus included at an appropriate level in the overall link cost.

The capacity cost of link (*i, j*), i.e. the link cost omitting any contribution from wavelength requirement, is:

$$C_{V,ij} = V_{ij}^{\alpha} L_{ij} \qquad (\mathbf{5.3})$$

where α a constant and L_{ij} is the length of link (*i, j*) in km. Whereas, the wavelength-requirement cost of link (*i, j*) is:

$$C_{\lambda,ij} = V_{\lambda,ij}^{\beta} L_{ij} \qquad (\mathbf{5.4})$$

where β is a constant. Increasing capacity necessarily implies increased cost due, for example, to wider transmission bandwidth, narrower wavelength separation and/or increasing number and speed of transmitters and receivers. With α=1, linear dependence of cost on capacity is assumed, but with α<1, the cost can be adjusted to rise more slowly with increases in the link capacity. The linear link length dependency approximates the increasing costs of, for example, duct installation, fiber blowing and/or the larger number of optical amplifiers with increasing distance. By setting $\beta \neq \alpha$, the link model can be adjusted to allow for a different dependency of cost on wavelength requirement capacity. The overall cost of link (*i, j*) is then taken to be:

$$C_{ij} = \gamma C_{V,ij} + (1-\gamma) C_{\lambda,ij} \qquad (\mathbf{5.5})$$

where γ is a constant ($0 \leq \gamma \leq 1$). Different values of γ allow the model to reflect the relative contributions to link cost of the capacity cost and the wavelength-requirement cost. With γ=1, the link only takes account of the capacity used, and ignores the link wavelength requirement; whereas, with γ=0, the actual capacity used is ignored, and the focus is entirely on the wavelength requirement. An intermediate value of γ, such as 0.5, is regarded as being more representative of the actual cost involved.

For the nodes, a node effective distance was used as a way of representing the cost of nodes in an optical network in equivalent distance terms. It can be regarded as the effective distance added to a path as a result of traversing a node. Although [98] used it to influence the selection of paths in minimum cost network topology design, thereby reflecting the relatively high costs of optical switching, here it is only used as part of the node cost model. The node effective distance of node *i* was taken to be:

$$N_i = K_0 + n_i K_n \quad (\mathbf{5.6})$$

where n_i is the degree of node *i*, i.e. the number of links attached to it. The constants K_0 and K_n were taken here as 200km and 100km, respectively. Node effective distance thus increases, as the switch becomes more complex. Node capacity is the sum of the effective capacity of all attached links, where V_i is the capacity of node *i* in Gbit/s and the effective capacity of link (*i, j*) is given by:

$$V_{e,ij} = \gamma V_{ij} + (1-\gamma) V_{\lambda,ij} \quad (\mathbf{5.7})$$

The node capacity is therefore depend to some extent on the wavelength requirements of its attached links. The node cost can then be formulated as:

$$C_i = 0.5 N_i V_i \quad (\mathbf{5.8})$$

The cost is thus derived as if the node were a star of links, each of half the node effective length, and each having the same capacity as the node itself. Further, if all the links attached to a node are of the same capacity (have equally balanced fibers), the node costs would grow approximately with the square of the node degree, corresponding, for example, to the growth in the number of crosspoints in a simple optical space switch. Overall, the relative costs of nodes and links can be adjusted by setting the values of K_0 and K_n appropriately.

The network cost is then taken to be the sum of the costs of all individual links and nodes comprising the network. However, the metric used also includes a penalty, added for each individual traffic requirement unsatisfied, to avoid the false minimum of carrying no traffic at all, which would otherwise have a network cost of zero.

5.2 Genetic Algorithm Framework

5.2.1 Overall Structure

We introduce an object oriented-oriented approach for implementing the genetic algorithm. The classes discussed in this section form the overall GA framework. As well as the central framework classes (GeneticAlgorithm, Population, Individual and OperatorPool), the abstract classes from which all genetic operators are derived (Operator, UnaryOp and BinaryOp) are described, additionally one very special operator, Reproduction is introduced. The relationships between most of these are illustrated in a class diagram, Figure 5.1.

5.2.1.1 GeneticAlgorithm Class

The GeneticAlgorithm class is central to GA/heuristic hybrid approach. It supports a generational, rather than steady-state, algorithm [8], and depends heavily on two other classes: Population, which holds the population of evolving network designs, and OperatorPool, which not only manages the pool of genetic operators used for population evolution, but also runs the operator-probability adaptation mechanism.

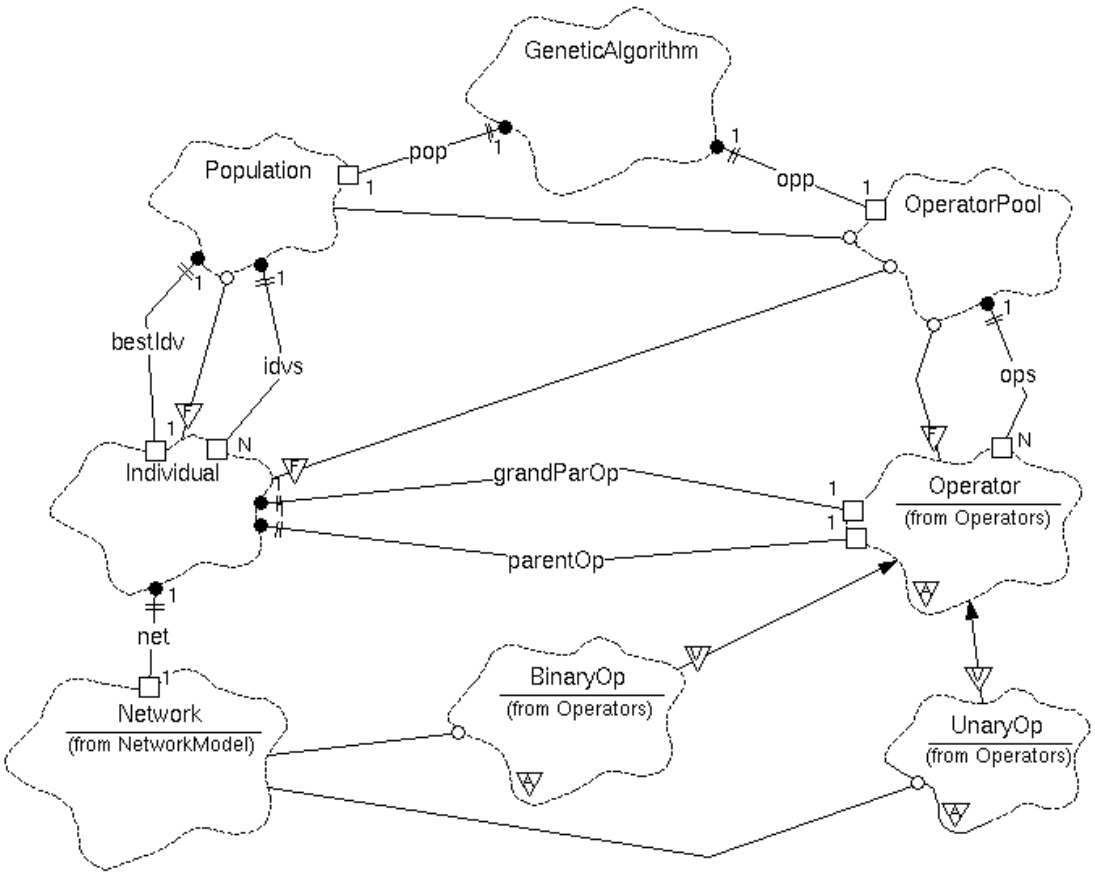


Figure 5.1 GA class diagram

The main GeneticAlgorithm class constructor takes three arguments: parameter databases built from both the network specification and the parameter files, plus a pseudo-random number generator object (of class RandomGen). After extracting the GA's one parameter, the maximum number of trials, the constructor uses the parameter databases and random number generator to build both population and operator-pool objects. Both of these, as well as the maximum number of trials, can subsequently be retrieved with the appropriate member functions. In addition, a GA object maintains one additional data member, the current number of trials, which is initialized to zero, and can also be accessed via a member function.

The principal member function of the GeneticAlgorithm class is run(). This takes the desired number of generations as an integer argument, and runs the GA for that number of generations. However, it will terminate prematurely, although always at the end of a complete generation, if either the number of trials exceeds the maximum specified or the appropriate interprocess signal has been received. On completion, the function returns a Boolean value indicating whether it would be able to continue for at least one more generation ('run on'). However, if the desired number of generations is specified as zero (the default), the function continues without returning until the maximum number of trials is exceeded i.e. the run is completed, or the signal has been received. Nevertheless, although it allows for runs of multiple generations in this way, if recording of population and operator statistics is required in every generation, the function is only invoked for one generation at a time.

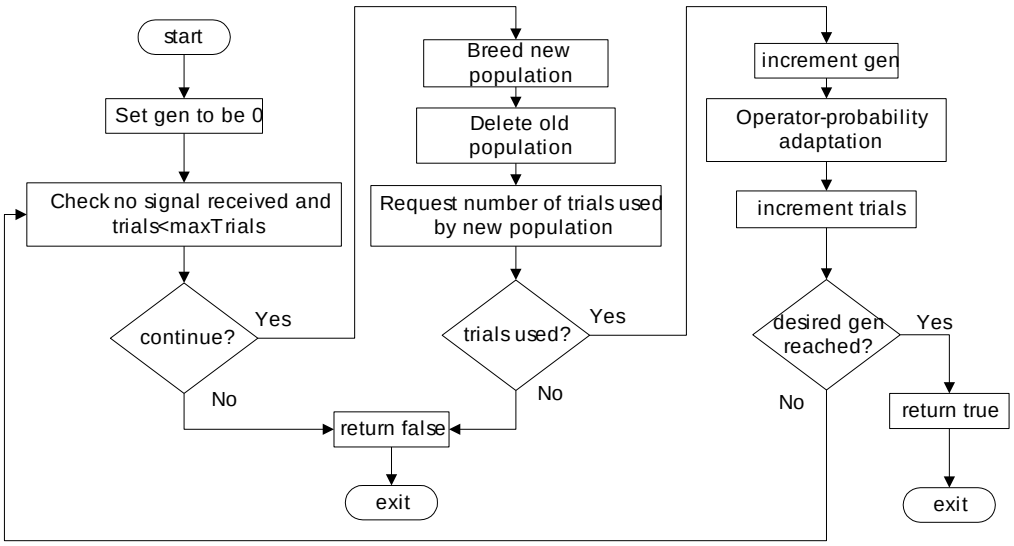


Figure 5.2 Flowchart of the GeneticAlgorithm run() member function

A flowchart of the run() member function is given in Figure 5.2 below. This figure shows the overall generational nature of the process, represented by the main loop, including breeding of a new population and its replacement of the old, as well as the use of operator-probability adaptation. However, there are a number of decision points that can end the iteration. As well as the receipt of the appropriate interprocess signal, exceeding the maximum number of trials or completing the desired number of generations, there is one further possibility. First note that network objects both cache their last assessment result, and also count the number of trials they have used in self-assessment since their construction. This count is

made available via a member function, and so the number of trials (i.e. assessments) used in constructing a population can be obtained by summing over all its constituent network objects. Consequently, if a new population is bred without consuming any trials, then not a single network has been changed in the breeding process: the GA is 'spinning' (i.e. has converged to identical or almost identical population members), and the run is terminated.

5.2.1.2 Population Class

Population class is not only responsible for containing a population of individuals (each a network object plus additional accounting information), but also maintaining several important population statistics.

Table 5.1 Population parameters

Parameter	Default	Description
pop.size	---	Population size
pop.winSize	1	Window size for best-in-window credit assignment, w
pop.select	---	Selection operator
pop.tournament.size	4	Tournament size for tournament selection
pop.initOp	RLS	Population initialization operator; default is WaveRandKP

There are two main Population class constructors. One of these, taking both network specification and parameter databases plus a random number generator, creates a full population. The other, omitting the network specification parameter database, creates an empty population, to which individuals can subsequently be added. The former constructor is used for initial population construction, whereas the latter is intended as part of breeding a new population from an old one. Both constructors extract and store several parameters (Table 5.1): the population size, the window size (w) for best-in-window fitness, the selection function, and, if the latter is tournament selection, the tournament size 4. Member functions are provided to allow the population size, window size, and selection function to be subsequently retrieved. In addition, for the former constructor, a choice of initialization operator may also be given. This is applied to every network object in the population as it is constructed, and can be used to build a random initial population.

Expanding a non-full population is achieved using a member function which enables an individual to be added to the population. At any point the current size of the population can be accessed via another member function, and then, when the population is full, a further member function is automatically invoked to refresh the

population statistics. (The latter is also invoked at the end of initial population construction.)

The stored statistics include, at the end of the last generation, the highest fitness (i.e. assessment) of any individual in the population, the lowest fitness, the average fitness and the median fitness. Further, a record is maintained of the lowest fitness in each generation over a user-defined number of generations. In addition, a copy of the best individual (i.e. of lowest fitness value) seen up to the end of the last generation is made. To access the statistics, member functions are provided for highest, lowest, average, median and best-in-window fitness, as well as one to access a const pointer to the best individual (and hence the best-so-far fitness).

Two further member functions are also available: an overloaded '[]' operator allows indexed access to the individuals in the population, and the total number of trials used by the population can be obtained, as was mentioned in the description of the GeneticAlgorithm class above.

5.2.1.3 Individual Class

The Individual class is simply a Network with some additional accounting information to support operator credit assignment.

The main constructor takes two arguments: the parameter databases for network specification and some parameters. From the latter database, a single parameter is taken which determines whether the individual constructs a Network or a PhyNetwork using the network database. However, construction of an individual from parameter databases in this way is computationally relatively expensive, and as a result, after the first, the remainder of the individuals in a population are actually created using the copy constructor (both in the initial population and when breeding).

An individual stores a pointer to its network object, as well as pointers to both its parent and grandparent operators. These latter objects are used by the credit-assignment mechanism to ensure that two- and three-operator sequences are properly rewarded. However, they are not actually maintained by the individual itself, but rather by its friend class, OperatorPool. For member functions, as well as allowing access to its network object as a const pointer, an individual also allows indirect retrieval of the underlying network fitness (i.e. self-assessment) using its own member function.

5.2.1.4 OperatorPool Class

In addition to containing and owning the set of operators utilized in a particular GA/heuristic hybrid algorithm, the operator pool is responsible for both breeding the population and managing the credit-assignment and operator-probability adaptation mechanisms.

The main OperatorPool class constructor takes two arguments: a parameter database and a pseudo-random number generator. The random number generator is stored, and from the parameter database several parameters are retrieved (Table 5.2). The number of operators (excluding simple reproduction) is stored, and then the names and initial probabilities of the expected number of operators are retrieved from the database. Each of these is constructed and stored, as is the reproduction operator object, whose initial probability is simply whatever remains to ensure the sum of the initial probabilities is 1.0. The remaining parameters, each of which is stored, influence the credit-assignment and operator-probability mechanisms.

Table 5.2 OperatorPool parameters

Parameter	Default	Description
opp.numOps	---	Number of operators (excluding Reproduction), s; must be ≥ 1
opp.ops[0].name	---	Name of operator 0
opp.ops[0].prob	---	Initial probability of operator 0
opp.adaptProp	0.15	Adaptation proportion, ψ, $0 \leq \psi \leq 1$
opp.adaptNumGens	4	Adaptation generation gap, ϵ; must be ≥ 1
opp.creditK	0.5	Credit constant, κ; must be ≥ 0
opp.fitCredType	best	Credit type: best, winBest, median or parent
opp.fitCredDiv	children	Credit divisor: children or trials

As well as a member function to obtain the number of operators and an overloaded '[]' operator to retrieve const pointers to the individual operators themselves, the OperatorPool class provides only two member functions: breed() and adaptOpProb().

The first of these creates a new population from an old one. Until the new population is full, it first chooses an operator from the pool at random, according to their current probabilities. Next, for unary operators, it selects an individual from the old population using the designated selection function (e.g. tournament selection), creates a new individual from this individual, and then applies the chosen operator. Both the operator's children and count of trials are incremented, by 1 and the number of trials actually used (which may be 0, 1 or more), respectively. Then, the operator itself, as well as the individual's parent and grandparent operators, are

awarded any credit earned, according to the applicable credit type. Next, the individual's grandparent and parent operator data members are updated (the parent operator is copied to the grandparent, and the current to the parent). Finally, the new individual, possibly modified, is added to the new population. A similar sequence, except that two parent individuals are used, is executed for binary operators. The second member function periodically adjusts the individual operator probabilities according to one of the algorithms discussed below.

5.2.1.5 Operator, UnaryOp, BinaryOp and Reproduction Classes

Operators themselves are created by deriving classes from either unary operator (UnaryOp) or binary operator (BinaryOp) classes. In general, those derived from UnaryOp are analogous to mutation [2] in a traditional GA, but can be powerful heuristics that utilize arbitrary amounts of problem-specific information. Likewise, BinaryOp subclasses are analogous to traditional crossover [2], although once again, there is considerable potential for problem-specific heuristics. As well as including the abstract classes representing operators (Operator, UnaryOp and BinaryOp) and routers (Router, UnaryRouter and BinaryRouter), the Operators class category includes a large number of concrete classes derived from these. The operators range from those of modest effect, analogous to traditional mutation or crossover, intended for use within a GA framework, through more powerful problem-specific operators, up to complete design heuristics, intended for use in isolation.

The Operator class' main constructor requires a single argument, the operator name, which can be up to four characters, and must be unique. For example, WaveKSPMutate uses 'WKM'. In addition to its name, an operator has data members for: the number of children it has generated, the number of trials it has used, its probability of being applied, and the credit it has earned. All the latter four data members are maintained by the friend class OperatorPool and all five, including its name, can be accessed via member functions. In addition, the Operator class provides two virtual member functions, with the overloaded name apply(), that take either one or two network arguments, and return a Boolean to indicate success or failure. In the same way, it provides two virtual Boolean tests: isUnary() and isBinary(). All four virtual member functions are implemented in the base class to simply return false.

As the abstract base class of all unary operators, UnaryOp redefines as abstract virtual, the version of apply() with only one network argument, forcing all its derived classes to re-implement it for themselves. In addition, it also re-implements isUnary() to return true. On the other hand, the BinaryOp class makes the complementary changes, redefining the other version of apply() as abstract virtual to force its own descendants to work with two networks at a time, and re-implementing isBinary() to return true.

Reproduction, derived from UnaryOp, is a very special concrete class: it does nothing, apart from return true, when applied! The reason for this is that implementation of the overall GA framework, reproduction (i.e. cloning) of parent individuals is actually performed by the operator pool. As a result, when simple reproduction is applied, as an operator, to the child individual, no further actions are required on its part. A reproduction operator object is still necessary, however, as it is capable of earning credit for its part in two- and three-operator sequences, just as any other operator, and is thus very much still part of the overall operator-probability adaptation mechanism.

5.2.2 Operators for Routing, Fiber & Wavelength Assignment

Six routers developed for WP routing, fiber choice and wavelength allocation, plus one for virtual-wavelength-path (VWP). These are listed in Table 5.3, and the relationships between the classes illustrated in Figure 5.3.

Table 5.3 Routers for wavelength allocation

Class	Name	Description
WaveRandKSP	WRKS	Wavelength-allocation random k-shortest paths
WaveNop	WNOP	Wavelength-allocation no-op
WaveWutti95	WW95	Wuttisuttikulkij & O'Mahony (W&O) [28]
WaveWauDijk	WADI	Wauters & Demeester (W&D1) [29]
WaveWauHRWA	WAHR	Wauters & Demeester HRWA (W&D2) [29]
WaveNagVWP	WNVW	Nagatsu et al. VWP [31]
WaveNagWP	WNWP	Nagatsu et al. WP (NHS) [31]

Wavelength-allocation random k-shortest paths is intended for the random, but reasonable initialization of WP routing, fiber choice and wavelength allocation for those networks in the initial population of a GA/heuristic hybrid. For each of the wavelength channels required between a node pair, a random path is selected from the k-shortest, and the wavelength channel routed along it on the lowest wavelength available end-to-end, irrespective of fiber choice (i.e. on each of the cables along the route, the lowest fiber id is used which has the required free wavelength). This

router has been implemented as class WaveRandKSP, the parameters for which are given in Table 5.4.

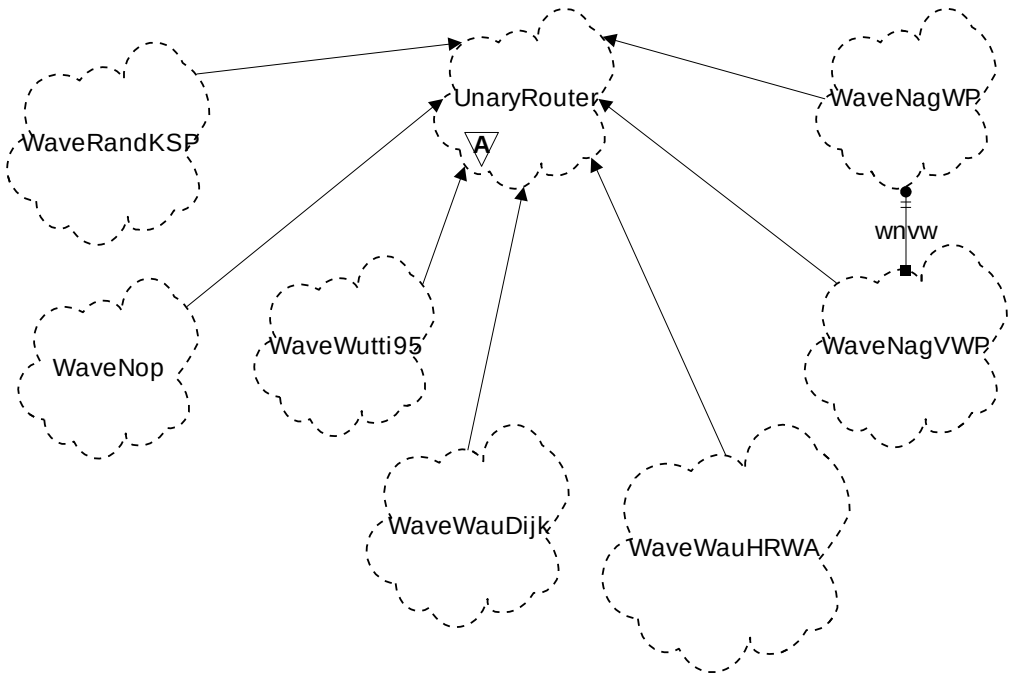


Figure 5.3 Wavelength assignment routers class diagram

Wavelength-allocation no-op is the simplest of routers: it does nothing at all! It is implemented as class WaveNop, and is intended for use as the router data member

Table 5.4 WaveRandKSP parameters

Parameter	Default	Description
wrks.theK	4	Number of alternative paths, k
wrks.firstFiberOnly	true	Use only first fiber on each cable?
wrks.cacheKSP	true	Cache k-shortest paths?

of a wavelength-routed network object whose routing, fiber and wavelength allocation is to be determined by the external application of a router or GA/heuristic hybrid. If a wavelength-allocation no-op object is not installed in those circumstances, the network would reroute itself (possibly incorrectly) whenever it carried out a self-assessment.

The remaining routers in Table 5.3 are all overall wavelength-allocation heuristics published in the literature. Details of these, and their implementation, can be found in the next section.

As well as the routers, described above, four operators developed for WP routing, fiber choice and wavelength allocation, three unary and one binary. These are listed in Table 5.5, and the relationships between the classes illustrated in Figure 5.4.

Table 5.5 Operators for wavelength allocation

Class	Name	Description
WaveKSPMutate	WKM	Wavelength-allocation k-shortest path mutation
WaveWauReroute	WWRR	Wavelength-allocation path reroute operator
WaveWauShiftOut	WWSO	Wavelength-allocation path shift-out operator
WaveKSPCO	WKCO	Wavelength-allocation k-shortest path single-point crossover

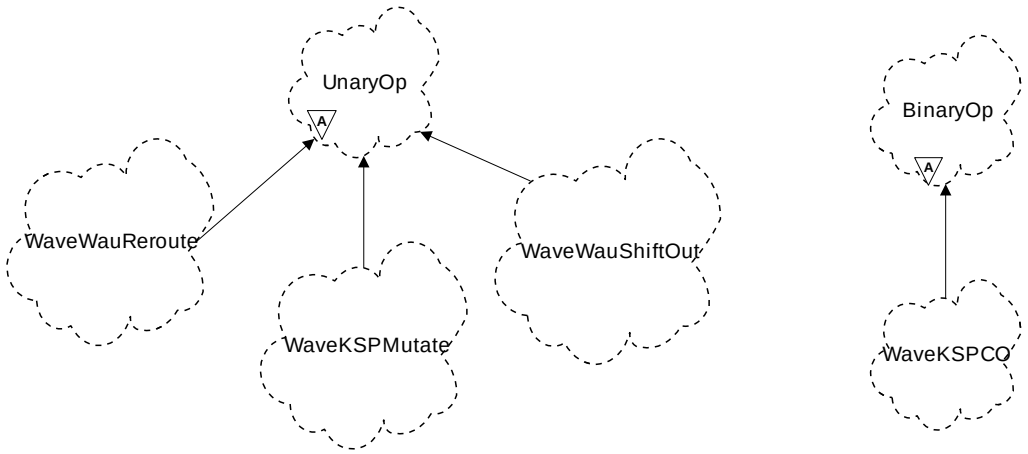


Figure 5.4 Wavelength assignment operators class diagrams

For a random requirement (i.e. an individual wavelength channel of those required between one of the node pairs in the network), wavelength-allocation k-shortest path mutation reroutes the wavelength channel on a randomly chosen k-shortest path (where k is an operator parameter), using the lowest wavelength available end-to-end on that path (irrespective of fiber choice). The operator has been implemented as class WaveKSPMutate, the parameters of which are given in Table 5.6.

Table 5.6 WaveKSPMutate parameters

Parameter	Default	Description
wkm.theK	4	Number of alternative paths, k
wkm.firstFiberOnly	true	Use only first fiber on each cable?
wkm.cacheKSP	true	Cache k-shortest paths?

The wavelength-allocation path reroute operator locates the first wavelength channel in the network which has the highest wavelength number (found by

searching the cables in order of their end-node ids), and attempts to reroute it on a lower wavelength number (and the lowest available end-to-end) using one of the k-shortest paths (where k is an operator parameter, and the shortest path is selected if there is more than one at the lowest available wavelength number). It is based on Step 2a in Wauters and Demeester's HRWA algorithm [29]. As an example, Figure 5.5 (a) shows a 6-node, 7-link network, with single-fiber cables, over which four WP wavelength channels have been routed. The wavelength channel using the highest wavelength number (λ_2) is labeled λ_{max}. In Figure 5.5 (b), after applying the reroute operator, that channel has been successfully rerouted on a lower wavelength, λ_1. As a result, the network wavelength requirement has been reduced from three to two wavelengths. The operator has been implemented as class WaveWauReroute, the parameters of which are given in Table 5.7.

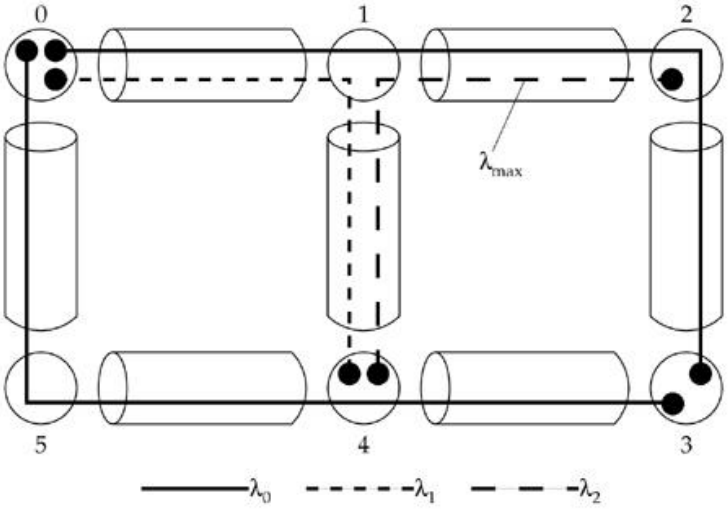
Table 5.7 WaveWauReroute parameters

Parameter	Default	Description
wwrr.theK	4	Number of alternative paths, k
wwrr.firstFiberOnly	true	Use only first fiber on each cable?

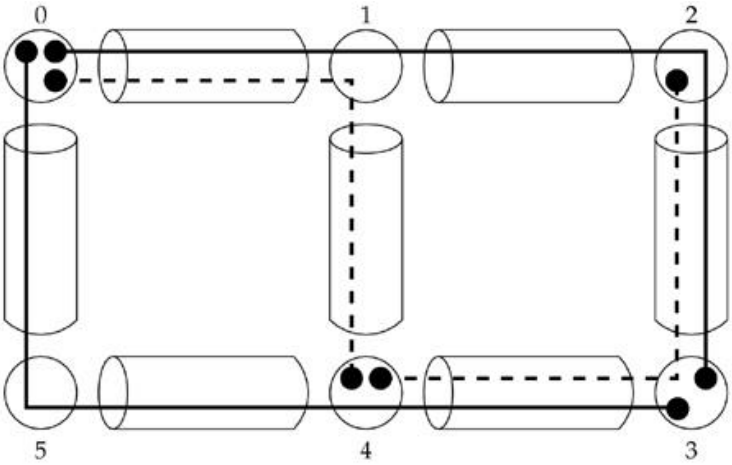
Table 5.8 WaveWauShiftOut parameters

Parameter	Default	Description
Wwso.theK	4	Number of alternative paths, k
Wwso.firstFiberOnly	true	Use only first fiber on each cable?

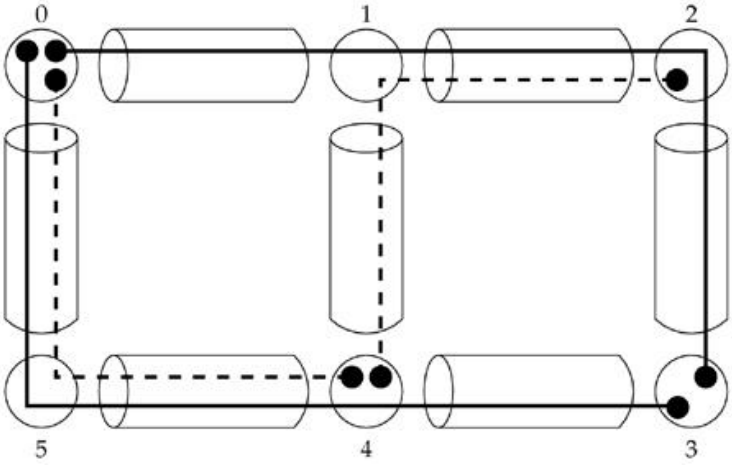
The wavelength-allocation path shift-out operator locates the first wavelength channel in the network which has the highest wavelength number, and attempts to shift it to a lower wavelength number by shifting out (rerouting) all the paths blocking that particular wavelength number end-to-end. The wavelength chosen is the one that results in the smallest increase in overall path length, given the constraint that all rerouting must be to lower wavelength numbers than the original wavelength of the channel being shifted down. It is based on Step 2b in HRWA [29]. Illustrating this, Figure 5.5 (c) shows the original network of Figure 5.5 (a) after shift-out operator has been applied. By rerouting the blocking wavelength channel on λ_1, it has proved possible to reduce the wavelength number of channel λ_{max} itself to λ_1. This change has again lowered the network wavelength requirement from three to two. The operator has been implemented as class WaveWauShiftOut, the parameters of which are given in Table 5.8.



a) Original Network



b) Reroute



c) Shift-out

Figure 5.5 Example of reroute and shift-out operators

Wavelength-allocation k-shortest path single-point crossover operates on two networks. The first is modified such that up to a randomly selected individual requirement (considered in order of their end-node ids), all the wavelength channels remain unchanged, but from then on the paths taken from the second network, except that the lowest wavelength actually available end-to-end in the first network is used in each case. (However, if any requirement was unsatisfied in the second network, then a random k-shortest path is used instead.) The second network undergoes the complementary recombination. The operator has been implemented as class WaveKSPCO, the parameters of which are given in Table 5.9.

Table 5.9 WaveKSPCO parameters

Parameter	Default	Description
wkco.theK	4	Number of alternative paths, k r
wkco.firstFiberOnly	true	Use only first fiber on each cable?
wkco.cacheKSP	true	Cache k-shortest paths?

5.2.3 Operator-probability Adaptation

As a variety of operators are incorporated in GA/heuristic hybrid algorithm, it might seem necessary to determine appropriate operator probabilities for each of the distinct combinations of operators used. However, this implies a large computational effort. Instead, the probabilities of individual operators being applied to an individual adapt during the course of a run using a credit-assignment algorithm, rather than being fixed throughout, as was mentioned above.

As well as avoiding the need to determine appropriate fixed operator probabilities (except for the initial probabilities, which could be selected from a few trial runs), it was also hoped that there might be performance gains in allowing the probabilities to vary during the course of a run, such that the most appropriate blend of operators was used at each stage, as determined by the observed operator productivity.

The operator-probability adaptation algorithm adopted is based on that of Davis [8]. First, the basic description of the algorithm is given, and then a number of further variants presented.

5.2.3.1 Basic Adaptation Algorithm

Whenever an individual is created whose fitness exceeds that of the best individual found up to the end of the previous generation, the improvement in fitness is credited to the responsible operator:

$$c'_u = c_u + (f_b - f_i), \qquad f_i < f_b \tag{5.9}$$

where c'_u and c_u are the current and previous values of the accumulated credit of operator u, f_b is the best fitness found at the end of the previous generation, and f_i is the fitness of individual i. In addition, as has already been mentioned, a decreasing fraction of the credit (say, κ= 0.5 and κ^2= 0.25) is awarded to the parent operator (i.e. the operator that created this individual's parent) and grandparent operator, to ensure that two- and three-operator sequences are rewarded appropriately. Thus:

$$c'_p = c_p + (f_b - f_i)\kappa, \qquad f_i < f_b \tag{5.10}$$

$$c'_g = c_g + (f_b - f_i)\kappa^2, \qquad f_i < f_b \tag{5.11}$$

where c_p and c_g are the accumulated credit of the parent (p) and grandparent (g) operators, respectively, and κ is the credit constant ($0 \leq \kappa \leq 1$).

It should be noted that, while Davis [8] maintained a complete tree (although of finite depth) of the operators responsible for the creation of an individual, here only a single (arbitrary) parent is recorded for binary operators. This greatly simplifies accounting, as instead of up to two parents, four grandparents, eight great-grandparents, etc., only a single parent and grandparent operator have to be recorded. Indeed, the overhead of maintaining and crediting a complete operator tree has already been noted elsewhere [82]. Further, although the selection of a single arbitrary parent operator means that all the operators responsible for a superior individual are not rewarded, it would be expected that these unrewarded operators would have subsequent opportunities to acquire credit, and it is their relative accumulation of credit, rather than the absolute value, that is important.

After a few generations (say, ϵ = 4), a small proportion (say, Ψ = 0.15) of the operator probabilities is reassigned according to the average credit earned per child each operator has generated. In addition, operators are given a minimum probability (say, p_{min} = 0.05 for less than 20 operators) to ensure they do not lose the opportunity to gain some credit if they, having decayed to the minimum level, are

later found to be useful to further evolve the population. Thus, every ϵ generations, for all operators, first:

$$a_u = \begin{cases} c_u / o_u, & o_u > 0 \\ 0, & o_u = 0 \end{cases} \quad \textbf{(5.12)}$$

where, for operator u, a_u is the adaptation weight and o_u the number of children (i.e. offspring) generated. The total adaptation weight is then given by:

$$a = \sum_u a_u \quad \textbf{(5.13)}$$

Now, provided some credit has been earned, and thus a is non-zero, probability reassignment can proceed. However, before this an adaptation scaling factor (K_a) must be calculated. This is the amount of the total probability that is to be reassigned, while still protecting the minimum probability level of all operators, rescaled using the total adaptation weight, thus:

$$K_a = \psi.(1-(s+1)p_{\min})/a, \quad a > 0, \ \ (s+1)p_{\min} < 1 \quad \textbf{(5.14)}$$

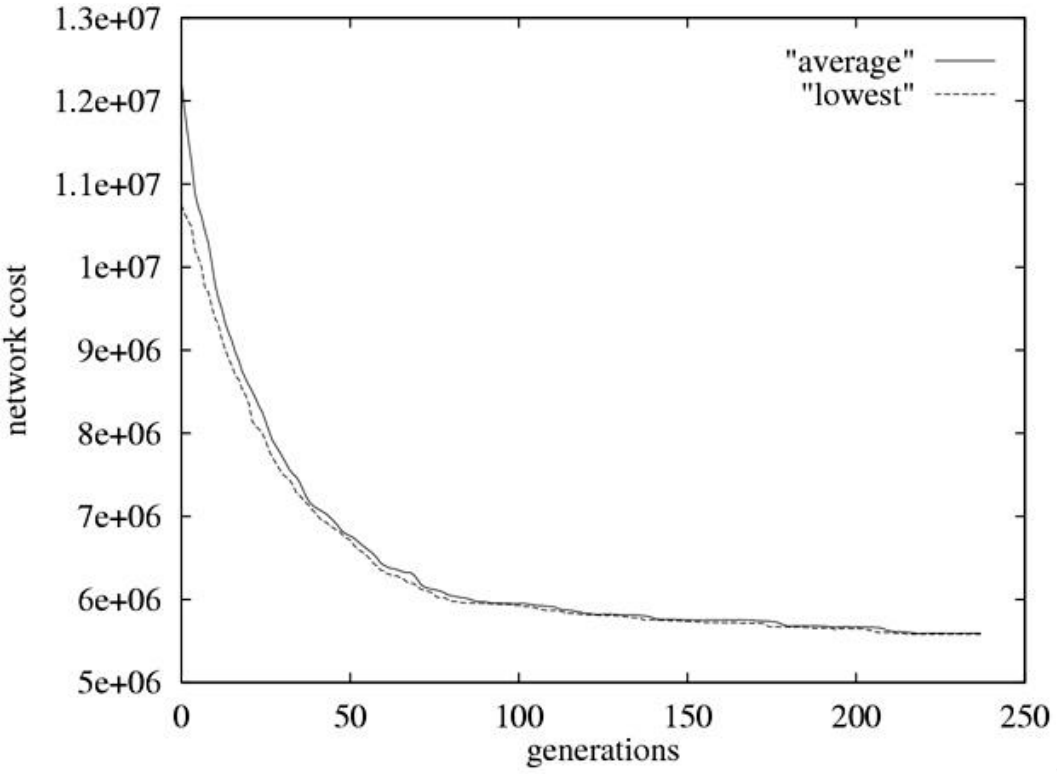
where Ψ is the adaptation proportion ($0 \leq \Psi \leq 1$), s is the number of operators (excluding reproduction, hence the addition of 1) and p_{min} is the minimum operator probability level. All the operator probabilities can then be then reassigned as follows:

$$p'_u = (p_u - p_{\min}).(1-\psi) + a_u K_a + p_{\min} \quad \textbf{(5.15)}$$

where p_u is the probability of operator u. The first term represents the unchanged part of the operator's probability, the second, that element reassigned according to adaptation weight, and the final one, the guaranteed minimum probability. It is straightforward to verify that, after the application of 5.15, the $\sum p_u$ remains 1. Finally, before continuing, the accumulated credit (c_u, number of children (o_u) and number of trials (t_u) of all operators are reset to zero.

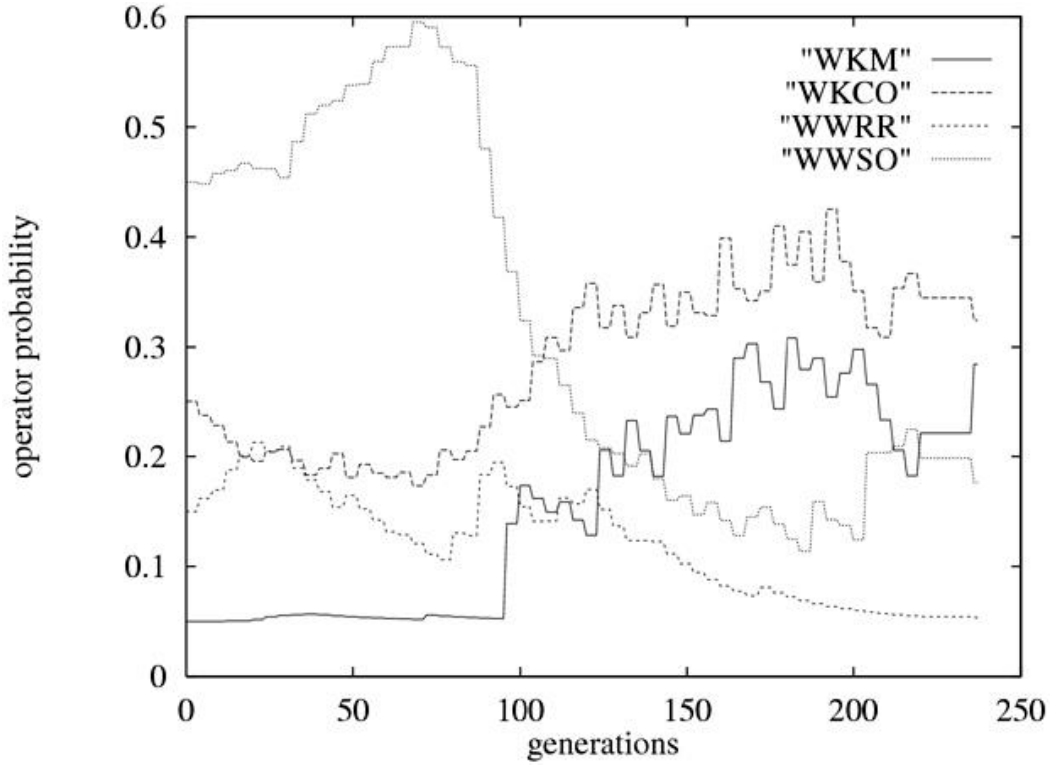
As an example of the operator-probability adaptation mechanism, Figure 5.6 (a) illustrates the progress of a typical wavelength-allocation hybrid run, showing the lowest and average population cost against generation. Figure 5.6 (b) gives the operator probabilities for the same run, where the effect of the adaptation algorithm can be clearly seen. For example, wavelength-allocation k-shortest path mutation (WKM), which started with a low probability, became much more productive after generation 100; whereas the wavelength-allocation path shift-out operator (WWSO),

which had the highest probability in early generations, was found to be much less useful towards the end of the run.



a)

Progress over a single run



b) Operator probability adaptation

Figure 5.6 Example GA/heuristic hybrid run

5.2.3.2 Variants on the Basic Algorithm

Variations to both the basic credit-assignment and operator-probability adaptation mechanisms have also been developed. First, in the basic credit-assignment algorithm, operators are rewarded as a result of improvement over the fitness of the best individual found up to the end of the previous generation. Instead, credit can be awarded for improvement over:

- best-in-window fitness (i.e. the best fitness achieved in the previous *w* generations), f_w [82, 88]
- the median fitness of the population at the end of the previous generation, f_m [87]
- or the fitness of the parent individual (or the fitter of the two parents, for binary operators), f_p [82, 88]

Thus equation 5.9 (and in a similar way, equation 5.10 and 5.11) is altered to one of:

$$c'_u = c_u + (f_w - f_i), \qquad f_i < f_w \tag{5.16}$$

$$c'_u = c_u + (f_m - f_i), \qquad f_i < f_m \tag{5.17}$$

$$c'_u = c_u + (f_p - f_i), \qquad f_i < f_p \tag{5.18}$$

Where f_w is the best-in-window fitness at the end of the previous generation, likewise, f_m is the median fitness found at the end of the previous generation, and f_p is the (better) fitness of the parent(s) of individual *i*.

For the operator-probability algorithm, in the basic approach, operator credit was divided by the number of children each operator had generated before being used to reassign operator probabilities. However, if an operator uses more than one trial per child generated, this may unfairly favor operators that are expensive in their use of trials. An alternative is to use the number of trials itself as the divisor before probability reassignment. Thus, equation 5.12 is altered to:

$$a_u = \begin{cases} c_u / t_u, & t_u > 0 \\ 0, & t_u = 0 \end{cases} \tag{5.19}$$

where t_u is the number of trials used by operator *u*. However, this change also has an interesting secondary consequence, as not all operator applications require

trial(s), due to assessment being cached by the network: if the network is unchanged, the operator has been applied ‘for free'.

6. EXPERIMENTAL RESULTS

This section reports experimental results exploring the suitability of the genetic algorithm (GA)/heuristic hybrid technique for routing and wavelength assignment in WDM networks.

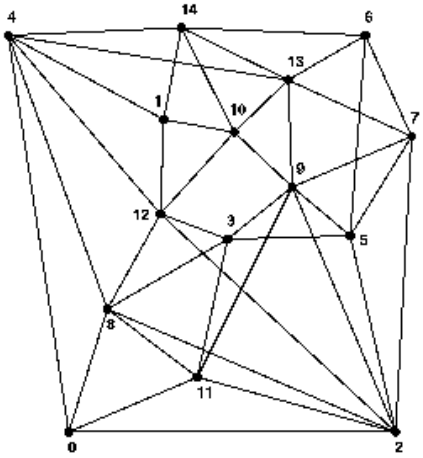
6.1 Topologies Employed

Five networks were used. Each has fifteen nodes, placed within a 1000kmx1000km area using a minimum clear radius constant of 0.5, and is intended to carry an overall static traffic requirement of 1500Gbits/s. The networks are illustrated in Figure 6.1 and traffic matrices have been recorded in Tables 6.1 and 6.2.

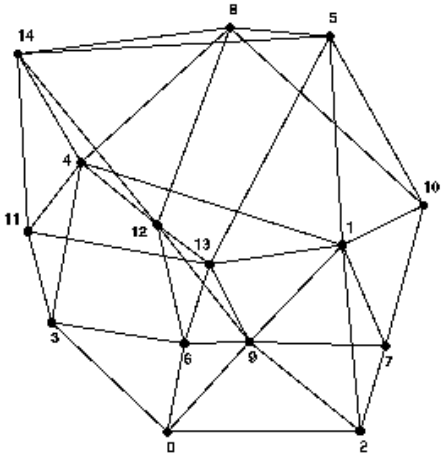
6.2 Routing and Wavelength Assignment

Once a topology has been determined, the next problem is the network routing, fiber choice and wavelength assignment. The networks considered in the thesis are flat, all-optical, with arbitrary-mesh multiple-fiber-link topologies, and employ WP routing. Consequently, as was mentioned, first the static traffic requirement between each node pair in the network has to be partitioned into an appropriate number of separate wavelength channels. Then, for each of these, a route through the network must be found, including which fiber and wavelength will be used on each link (cable) along the path. As WP routing is employed, each channel must use the same wavelength end-to-end, since no wavelength conversion is available, as well as obeying the usual constraint that on any one fiber, no wavelength is used by more than one channel.

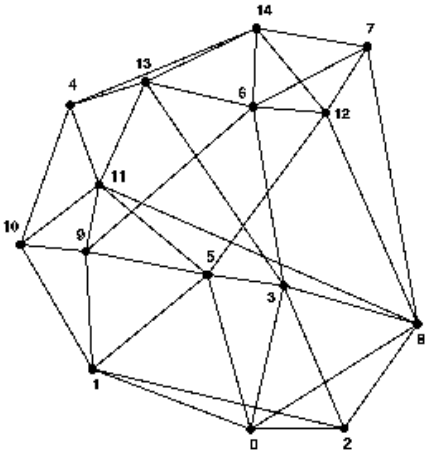
For the experiments in this section, the test networks described above were used. Each link was taken to be a cable of two fibers, and the capacity of the individual wavelength channels set to 10 Gbit/s. The total wavelength channel requirements for the networks, calculated from the traffic matrices, are given in Table 6.3.



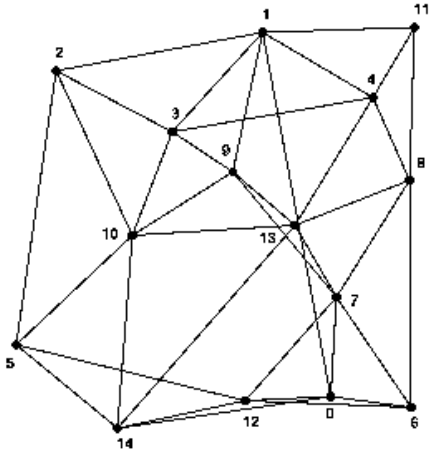
a) Test network 1



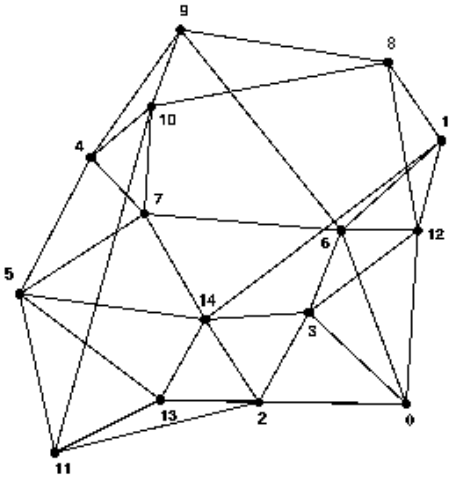
b) Test network 2



c) Test network 3



d) Test network 4



e) Test network 5

Figure 6.1 Test Networks

Table 6.1 Traffic matrix 1 (Gbit/s)

From	To							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	3.48041	28.2109	4.67568	16.6	3.78823	3.57941	6.18347
1	3.48041	0	7.10871	2.59262	11.4365	1.78551	2.46002	3.41731
2	28.2109	7.10871	0	11.0846	29.288	15.5817	10.7728	22.8053
3	4.67568	2.59262	11.0846	0	6.90465	3.28028	2.22152	4.1077
4	16.6	11.4365	29.288	6.90465	0	5.80517	8.85029	11.9654
5	3.78823	1.78551	15.5817	3.28028	5.80517	0	3.00873	8.18153
6	3.57941	2.46002	10.7728	2.22152	8.85029	3.00873	0	11.7723
7	6.18347	3.41731	22.8053	4.1077	11.9654	8.18153	11.7723	0
8	20.7896	4.09683	20.9339	6.01279	16.3052	3.6599	3.361	5.71205
9	3.91257	2.71065	11.8	4.83045	7.1551	5.8588	3.59739	7.41663
10	3.3198	4.83249	8.1056	3.2559	8.24158	2.5145	3.29398	4.78808
11	20.531	3.31738	34.0387	6.26122	12.9557	4.73639	3.5256	6.58389
12	4.91326	3.74075	8.82422	5.00864	8.82735	2.10229	2.0187	3.29897
13	3.04435	2.9381	8.30644	2.27951	7.95378	2.57128	6.69178	6.92201
14	4.76117	6.41144	10.616	2.84633	20.607	2.56515	5.15292	5.85751

From	To						
	8	**9**	**10**	**11**	**12**	**13**	**14**
0	20.7896	3.91257	3.3198	20.531	4.91326	3.04435	4.76117
1	4.09683	2.71065	20.531	3.31738	34.0387	6.26122	12.9557
2	20.9339	11.8	8.1056	34.0387	8.82422	8.30644	10.616
3	6.01279	4.83045	3.2559	6.26122	5.00864	2.27951	2.84633
4	16.3052	7.1551	8.24158	12.9557	8.82735	7.95378	20.607
5	3.6599	5.8588	2.5145	4.73639	2.10229	2.57128	2.56515
6	3.361	3.59739	3.29398	3.5256	2.0187	6.69178	5.15292
7	5.71205	7.41663	4.78808	6.58389	3.29897	6.92201	5.85751
8	0	4.18596	3.72385	18.1009	7.60589	3.08525	4.89017
9	4.18596	0	4.90572	4.62457	2.99162	4.01461	3.54067
10	3.72385	4.90572	0	3.47403	3.19385	5.09149	5.15626
11	18.1009	4.62457	3.47403	0	5.25084	3.07613	4.32944
12	7.60589	2.99162	3.19385	5.25084	0	2.1113	3.29661
13	3.08525	4.01461	5.09149	3.07613	2.1113	0	5.68844
14	4.89017	3.54067	5.15626	4.32944	3.29661	5.68844	0

Table 6.2 Traffic matrix 2 (Gbit/s)

From	To							
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	6.3571	6.9486	9.72611	5.15408	4.9919	10.0818	5.77075
1	6.3571	0	8.96659	5.72956	5.71384	11.8995	5.27966	15.0771
2	6.9486	8.96659	0	4.54633	3.58516	5.3979	4.277	16.0814
3	9.72611	5.72956	4.54633	0	9.9127	5.71345	6.84457	4.40464
4	5.15408	5.71384	3.58516	9.9127	0	7.21891	4.25471	3.83807
5	4.9919	11.8995	5.3979	5.71345	7.21891	0	3.85926	6.77956
6	10.0818	5.27966	4.277	6.84457	4.25471	3.85926	0	4.10727
7	5.77075	15.0771	16.0814	4.40464	3.83807	6.77956	4.10727	0
8	5.27834	9.97364	5.00121	6.74815	10.2409	29.0283	4.17233	5.93165
9	7.15029	7.4859	5.99148	4.62628	3.54007	4.20143	7.91675	6.06757
10	4.723	20.1798	7.10677	4.43091	4.50892	12.6031	3.55144	11.458
11	5.83994	4.93185	3.51491	17.0713	16.237	5.63684	4.43747	3.57801
12	4.23073	4.98555	2.87077	6.38872	8.4303	4.83321	4.42171	3.13825
13	5.0459	6.93556	3.71423	5.30038	5.04311	4.8917	6.44525	4.17262
14	7.36656	8.68726	5.6559	12.2995	23.3413	13.091	5.46776	6.05271

From	To						
	8	**9**	**10**	**11**	**12**	**13**	**14**
0	5.27834	7.15029	4.723	5.83994	4.23073	5.0459	7.36656
1	9.97364	7.4859	5.83994	4.93185	3.51491	17.0713	16.237
2	5.00121	5.99148	7.10677	3.51491	2.87077	3.71423	5.6559
3	6.74815	4.62628	4.43091	17.0713	6.38872	5.30038	12.2995
4	10.2409	3.54007	4.50892	16.237	8.4303	5.04311	23.3413
5	29.0283	4.20143	12.6031	5.63684	4.83321	4.8917	13.091
6	4.17233	7.91675	3.55144	4.43747	4.42171	6.44525	5.46776
7	5.93165	6.06757	11.458	3.57801	3.13825	4.17262	6.05271
8	0	4.24076	9.02353	7.19227	6.0558	5.40718	19.1824
9	4.24076	0	4.48607	3.41472	3.51687	6.12958	4.887
10	9.02353	4.48607	0	3.91681	3.47505	4.20605	7.47683
11	7.19227	3.41472	3.91681	0	6.18287	4.34215	16.9482
12	6.0558	3.51687	3.47505	6.18287	0	7.48968	7.86787
13	5.40718	6.12958	4.20605	4.34215	7.48968	0	6.06738
14	19.1824	4.887	7.47683	16.9482	7.86787	6.06738	0

Table 6.3 Total wavelength channel requirements of the test networks

Network	1	2	3	4	5
Channel requirement	268	248	258	262	268

6.2.1 Minimum Network Wavelength Requirement

The first set of experiments employ the network wavelength requirement (NWR) metric. This modifies the actual NWR by adding a penalty for each unsatisfied traffic requirement (i.e. wavelength channel that remains unrouted) to avoid the false minimum of carrying no traffic at all, which would otherwise have a NWR of zero.

The four operators developed for routing, fiber and wavelength assignment were described in previous sections. In devising experiments based on these, it was decided to compare, for two different GAs, the best of five runs with results obtained from three recent wavelength-allocation heuristics published in the literature. The first GA is a more straightforward algorithm, which uses just wavelength-allocation k-shortest path mutation (WKM) and wavelength-allocation k-shortest path single-point crossover (WKCO), while the second GA also includes the two problem-specific heuristics, wavelength-allocation path reroute (WWRR) and wavelength-allocation path shift-out (WWSO).

Both GAs had a population size of 500 and used tournament selection (of size 4); their initial operator probabilities are given in Table 6.4. Other non-default parameter values for GA1 and GA2 are listed in Tables 6.5 and 6.6. It should be emphasized that they both used the operator-probability adaptation mechanism with default parameters throughout.

Table 6.4 Initial operator probabilities for minimum NWR

GA	WKM	WKCO	WWRR	WWSO
1	0.25	0.5		
2	0.05	0.15	0.3	0.35

The two GAs were both applied five times with different random seeds to Networks 1-5. The progress of single typical runs of GA1 and GA2 on Network 1 are given in Figures 6.2 (a) and (c), showing both the average NWR of the population and the lowest (i.e. best in population) NWR, generation by generation. The operator-probability adaptation curves for the same runs are shown in Figures 6.2 (b) and (d).

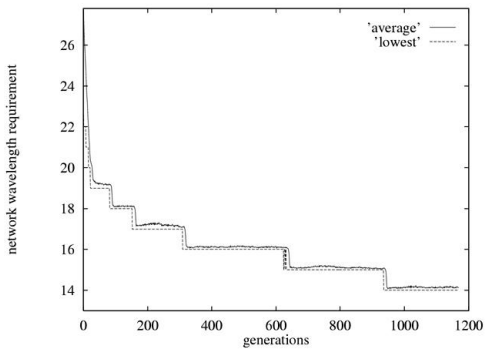
Table 6.5 Non-default parameters for GA1

Parameter	Value	Description
net.metric	NWR	Metric used for network self-assessment
net.router	WNOP	Router used to route the network prior to self-assessment
maxTrials	10.000	Maximum number of trials; must be ≥1
pop.size	500	Population size
pop.select	tournament	Selection operator
pop.initOp	WRKS	Population initialization operator
pop.phyNetwork	true	Construct the individual using a PhyNetwork (true) or a Network (false)
Wrks.theK	8	Number of alternative paths, k
Wrks.firstFiberOnly	false	Use only first fiber on each cable?
Wkm.theK	8	Number of alternative paths, k
Wkm.firstFiberOnly	false	Use only first fiber on each cable?
wkco.theK	8	Number of alternative paths, k
wkco.firstFiberOnly	false	Use only first fiber on each cable?

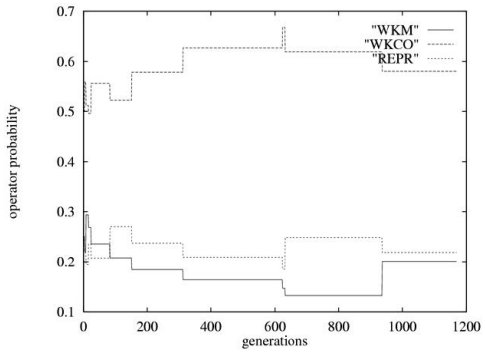
Table 6.6 Non-default parameters for GA2

Parameter	Value	Description
net.metric	NWR	Metric used for network self-assessment
net.router	WNOP	Router used to route the network prior to self-assessment
maxTrials	5.000	Maximum number of trials; must be ≥ 1
pop.size	500	Population size
pop.select	tournament	Selection operator
pop.initOp	WRKS	Population initialization operator
pop.phyNetwork	true	Construct the individual using a PhyNetwork (true) or a Network (false)
Wrks.theK	8	Number of alternative paths, k
Wrks.firstFiberOnly	false	Use only first fiber on each cable?
Wkm.theK	8	Number of alternative paths, k
Wkm.firstFiberOnly	false	Use only first fiber on each cable?
Wwrr.theK	8	Number of alternative paths, k
Wwrr.firstFiberOnly	false	Use only first fiber on each cable?
Wwso.theK	8	Number of alternative paths, k
Wwso.firstFiberOnly	false	Use only first fiber on each cable?
Wkco.theK	8	Number of alternative paths, k
Wkco.firstFiberOnly	false	Use only first fiber on each cable?

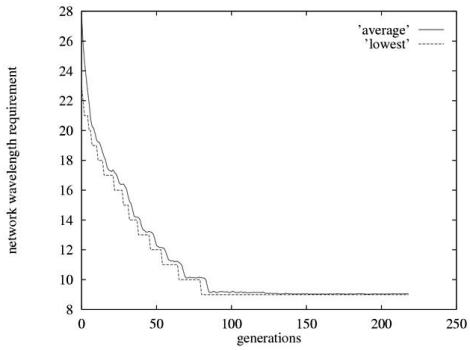
In contrast to most of the topology curves, all four figures show abrupt changes in value. This results from the integer-valued nature of NWR: whenever there is a new best individual in the population, its NWR is always at least a full unit lower than the previous best. In both Figures 6.2 (a) and (c), this improvement appears to rapidly propagate through the population, as the average NWR swiftly drops to close to the best. In addition, as the operator-probability adaptation mechanism is driven by credit assigned when a new best individual is found, the integer-valued nature of



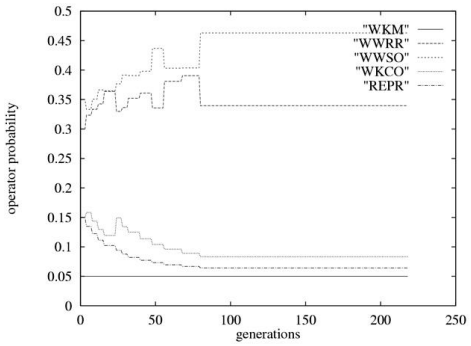
a) Progress over a single GA1 run



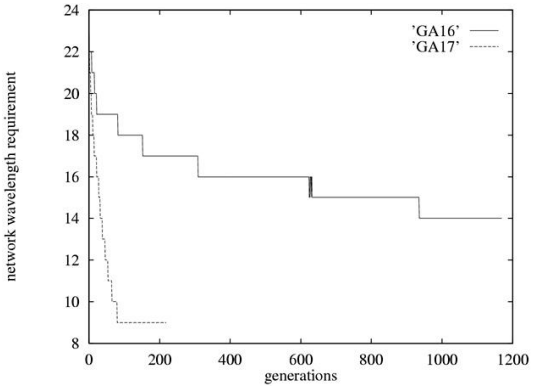
b) Operator probability adaptation for GA1



c) Progress over a single GA2 run



d) Operator probability adaptation for GA2



e) Comparing lowest NWR

Figure 6.2 Typical Minimum NWR results

NWR also results in fairly abrupt changes in operator probabilities, as can be observed in both Figures 6.2 (b) and (d).

Turning to the NWR results, there is clearly a considerable difference in the progress of the two GAs. Whilst second GA takes about 80 generations to obtain an NWR of 9, first GA is only able to achieve an NWR of 14 using nearly 1200 generations. This difference is even more apparent in Figure 6.2 (e) which directly compares the lowest NWR curves for the two GAs. Both these runs employed the same random seed, and thus started from the same initial population. From the operator-probability adaptation curves it appears that, at least for these runs, first GA chiefly depends on the crossover operator (WKCO), with a modest contribution from mutation (WKM), whereas second GA makes full use of both its problem-specific heuristic operators, shift-out (WWSO) and reroute (WWRR), with only minimal contributions from crossover and mutation.

In addition to the two GAs, a wavelength-allocation heuristic by Wuttisittikulkij & O'Mahony and two by Wauters & Demeester were also applied to Networks 1-5. Wuttisittikulkij & O'Mahony's heuristic [28] (W&O) is simple, fast and non-iterative, working with k-lowest hop count paths (and multiple fibers) to obtain a low NWR. The first heuristic by Wauters & Demeester [29] (W&D1) is based on a version of Dijkstra's algorithm modified such that in "extend[ing] an already found part of a route, only those [cables] are considered which have a wavelength channel free which is available on all [cables] of the already found part of the route"; this algorithm produces a low-cost allocation, whilst also keeping NWR low. Their second algorithm (W&D2), is based on iterative application of a modified version of Yen's k-shortest path algorithm, and aims to produce the lowest cost allocation that still achieves minimum NWR. Description of all three heuristics is given in previous sections, and their non-default parameters listed in Table 6.7. Of the three algorithms, Wuttisittikulkij & O'Mahony's is non-deterministic and so, like the GAs, it was allowed the best of five runs.

Table 6.7 Non-default parameters for W&0, W&D1 and W&D2 heuristics (minimum NWR)

Parameter	Value	Description
ww95.theK	8	Number of alternative paths, k
ww95.sortPaths	true	Allocate the paths in hop count order?
ww95.firstFiberOnly	false	Use only first fiber on each cable?
wadi.firstFiberOnly	false	Use only first fiber on each cable?
wahr.theK	8	Number of alternative paths, k
wahr.firstFiberOnly	false	Use only first fiber on each cable?

A summary of the results of applying the two GAs and three heuristics is recorded in Table 6.8, which lists the NWR, or the best NWR for five runs as appropriate, as well as the number of trials used to find the recorded NWR in the best GA run in each case. As before, the best overall result for each network is displayed in bold. In addition, the full set of results from applying the two GAs and W&0 to all five of the networks is recorded in Tables 6.9, 6.10 and 6.11, respectively.

Table 6.8 Results for minimum NWR

Network	W&O	W&D1	W&D2	GA1		GA2	
	NWR	NWR	NWR	NWR	Trials	NWR	Trials
1	8	15	10	13	6,264	9	1,874
2	7	11	7	10	7,919	7	2,291
3	8	19	9	10	8,235	7	3,467
4	8	15	9	11	9,385	8	3,997
5	9	18	9	12	8,962	8	4,448

Table 6.9 Minimum NWR results for GA1

Run	Network									
	1		2		3		4		5	
	NWR	Trials	NWR	Trials	NWR	Trials	NWR	Trials	NWR	Trials
1	14	7,181	11	6,331	**10**	**8,235**	12	6,042	13	9,673
2	13	7,968	10	9,338	11	8,945	12	5,261	12	9,217
3	**13**	**6,264**	**10**	**7,919**	10	8,909	**11**	**9,385**	**12**	**8,962**
4	14	3,861	10	8,945	11	5,812	12	8,841	13	6,215
5	14	6,665	12	5,997	11	7,983	12	7,958	13	6,122
Best	13	6,264	10	7,919	10	8,235	11	9,385	12	8,962

Table 6.10 Minimum NWR results for GA2

Run	Network									
	1		2		3		4		5	
	NWR	Trials	NWR	Trials	NWR	Trials	NWR	Trials	NWR	Trials
1	**9**	**1,874**	7	2,936	8	2,420	8	4,304	9	3,523
2	9	3,515	8	1,987	**7**	**3,467**	9	2,991	9	4,118
3	9	4,011	**7**	**2,291**	8	3,178	**8**	**3,997**	9	3,571
4	9	2,376	7	2,883	7	3,688	8	4,084	**8**	**4,448**
5	9	3,351	7	3,823	7	4,726	8	4,139	9	3,393
Best	9	1,874	7	2,291	7	3,467	8	3,997	8	4,448

Of the three heuristic algorithms, the poorest in this context is W&D1, which in aiming for low network cost, has obtained relatively high NWRs. Although the more recent W&D2 has clearly improved on this in NWR terms, the best heuristic overall is W&O, which despite its simplicity, achieves good NWR results. Although over the five test networks, first GA consistently obtained better NWRs than W&D1, it was no

match for the other two heuristics. However, second GA equaled or exceeded the results of all three heuristics, with one exception, and using considerably fewer trials than first GA. Clearly, therefore, the addition of the two heuristic operators (reroute and shift-out), has led to a marked improvement of second GA over first GA. In addition, despite these two additional operators being based on steps of the W&D2 heuristic, second GA has in many cases exceeded the results of W&D2, managing to match the best heuristic, W&O.

Table 6.11 Minimum NWR results for W&O

Run	Network				
	1	**2**	**3**	**4**	**5**
1	8	7	8	8	9
2	8	7	8	8	9
3	8	8	8	8	9
4	8	8	8	8	9
5	8	7	8	8	9
Best	8	7	8	8	9

Overall, these experimental results show that, at least for the five test networks, the GA/heuristic hybrid approach is as promising as has been hoped, as the blend of a GA with steps from Wauters & Demeester's HRWA heuristic [29] resulted in both improved performance (over the GA alone) and improved results (over the heuristic alone). In addition, the GA/heuristic hybrid results matched those of the best wavelength-allocation heuristic examined.

6.2.2 Minimum Cost Allocation

The second set of experiments on routing, fiber choice and wavelength assignment used cost model for minimum cost allocation. This changes the assessment from just looking at the NWR to an objective that consists of a simplified assessment of the cost of the whole network including, at an appropriate level, the cost of exceeding the minimum NWR. The cost model is parameterized, amongst others, by three constants, (α, β, γ). The first two, α and β, adjust the model of link cost in the network to allow for different dependencies on capacity. With $\alpha=\beta=1$, the dependency is linear -doubling the capacity doubles the cost- whereas with $\alpha = \beta<1$, the cost increases more slowly with capacity. In these experiments, α and β were always given the same value, however, γ is included in the model to fine-tune the dependency of cost on link wavelength requirement rather than just capacity alone. The remaining parameter, γ, allows the cost model to reflect the relative contributions to link and node costs of the capacity and link wavelength

requirements. With γ = 1, the cost model only takes account of the actual capacity used; whereas, with γ = 0, the capacity is ignored, and the focus is entirely on the costs resulting from wavelength requirements. An intermediate value of γ, such as 0.5, is regarded as being more representative of the actual costs involved.

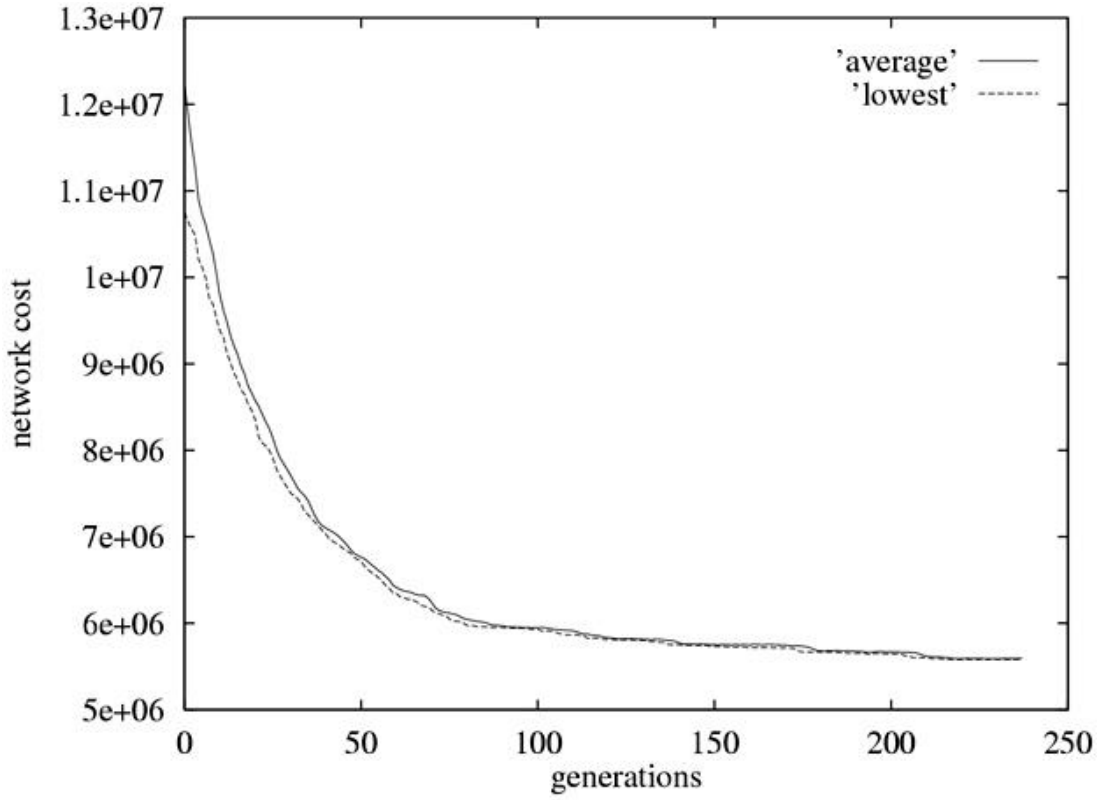
For these experiments, the same four routing, fiber and wavelength assignment operators used as in second GA. Given the outcome of the experiments on minimum NWR design, it was decided to immediately employ all four operators within a GA, rather than further explore ineffective algorithms that did not include the problem-specific heuristic operators. Instead, the emphasis was on competing, on a network cost basis, with the same three overall wavelength-allocation heuristics used for the minimum NWR study.

The three GAs used were identical except that they had different initial operator probabilities (given in Table 6.12). These were selected for problems with differing values of fl by using the usual approach of a few test runs, aiming for settings that would remain approximately constant for the first few generations across all the networks examined. All three GAs had a population size of 500, a maximum of 100,000 trials, and used tournament selection (of size 4). Other non-default parameter values for all GAs are listed in Table 6.13. It should be emphasized that they all used the operator-probability adaptation mechanism with default parameters throughout.

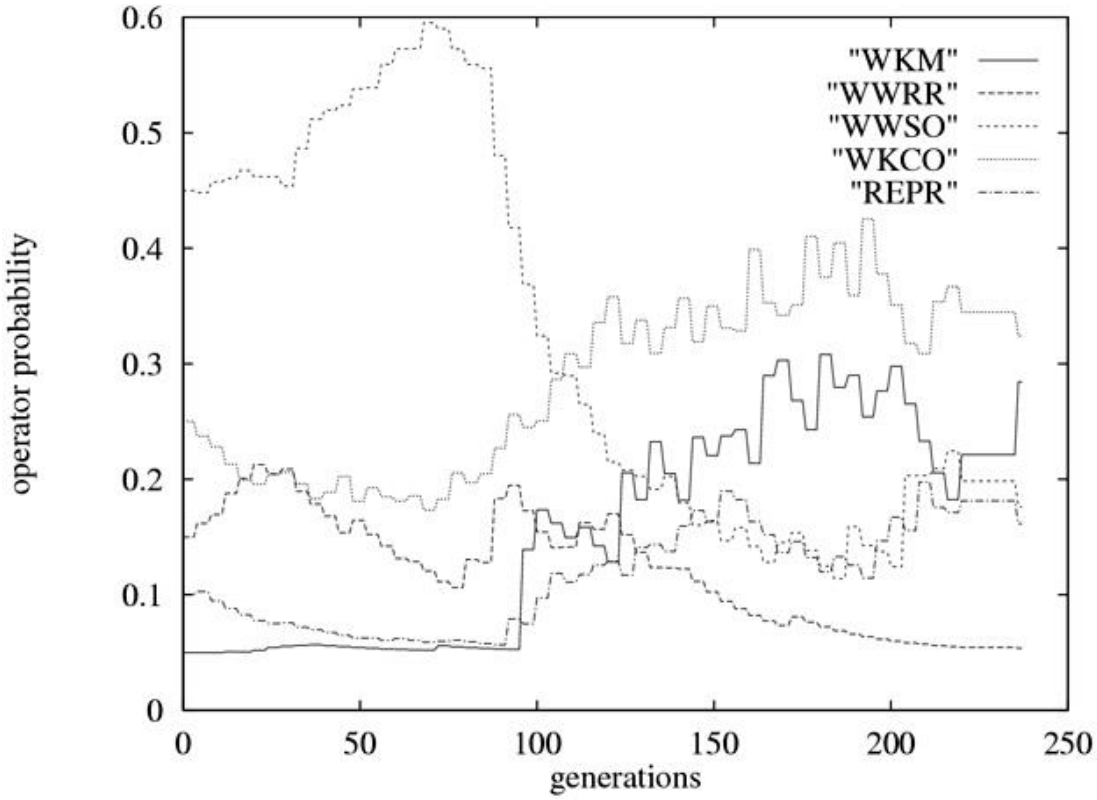
Table 6.12 Initial operator probabilities for minimum cost allocation

GA	WKM	WKCO	WWRR	WWSO
3	0.05	0.25	0.15	0.45
4	0.05	0.3	0.1	0.4
5	0.05	0.2	0.25	0.4

All three GAs were applied five times with different random seeds (i.e. best of five runs) to Networks 1-5, with γ set to 0.5, 1, and 0 for GA3, GA4 and GA5, respectively; first for α=β=1, then for α=β=0:8. For all three GAs (and the heuristics), k was set to 8 for the Networks 1-5. The progress of single typical runs of GA3 on Network 1 with (α, β, γ) = (1, 1, 0.5) and (0.8, 0.8, 0.5) are given in Figures 6.3 (a) and 5.4 (a), respectively, showing both the average network cost of the population and the lowest (i.e. best in population) cost, generation by generation. The operator-probability adaptation curves for the same runs are shown in Figures 6.3 (b) and 6.4 (b).

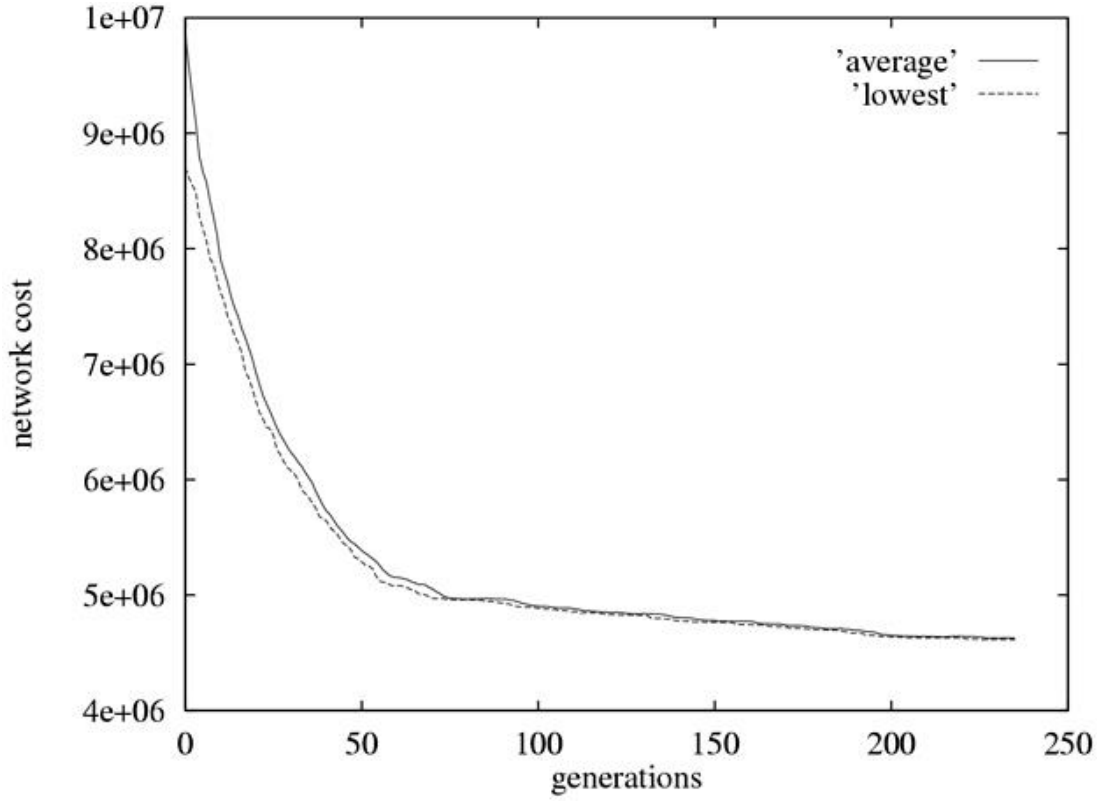


a) Progress over a single run

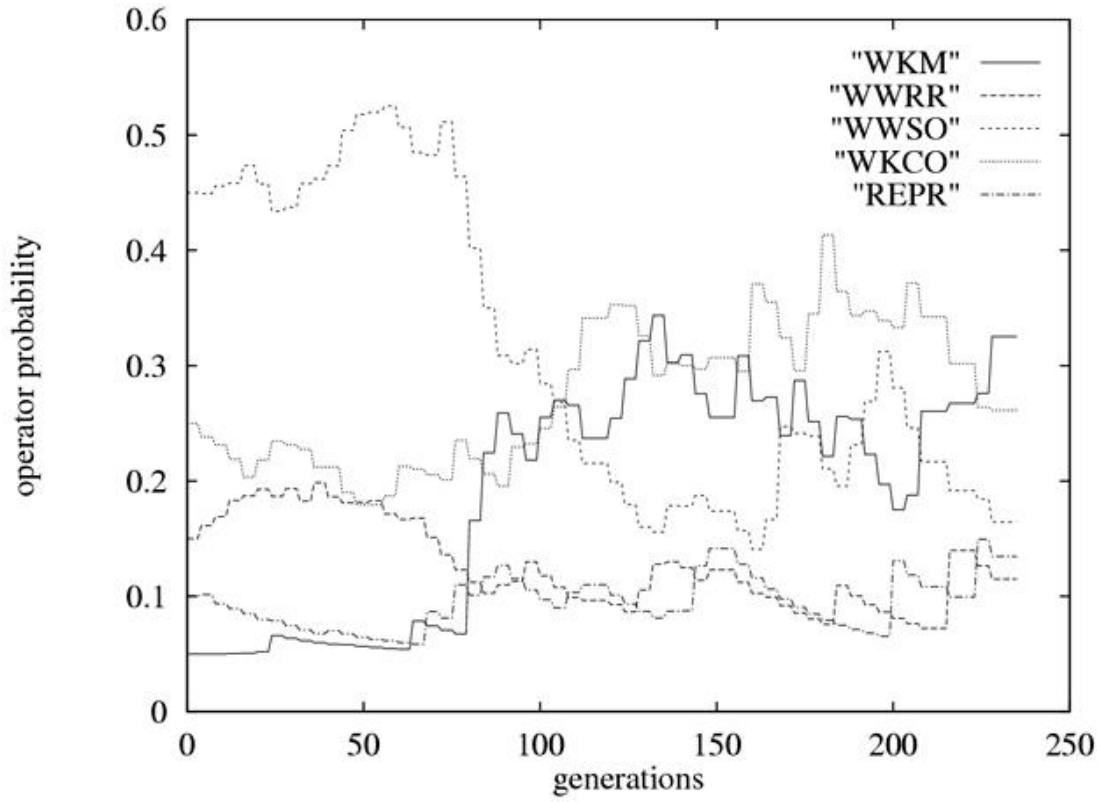


b) Operator probability adaptation

Figure 6.3 Typical results for GA3, α=β=1, and γ=0.5



a) Progress over a single run



b) Operator probability adaptation

Figure 6.4 Typical results for GA3, α=β=0.8, and γ=0.5

Table 6.13 Non-default parameters for GA3-GA5

Parameter	Value	Description
net.metric	PNCT	Metric used for network self-assessment
net.router	WNOP	Router used to route the network prior to self-assessment
MaxTrials	100,000	Maximum number of trials; must be ≥ 1
pop.size	500	Population size
pop.select	tournament	Selection operator
pop.initOp	WRKS	Population initialization operator
pop.phyNetwork	true	Construct the individual using a PhyNetwork (true) or a Network (false)
wrks.theK	8 or 4	Number of alternative paths, k
wrks.firstFiberOnly	false	Use only first fiber on each cable?
wkm.theK	8 or 4	Number of alternative paths, k
wkm.firstFiberOnly	false	Use only first fiber on each cable?
wwrr.theK	8 or 4	Number of alternative paths, k
wwrr.firstFiberOnly	false	Use only first fiber on each cable?
wwso.theK	8 or 4	Number of alternative paths, k
wwso.firstFiberOnly	false	Use only first fiber on each cable?
wkco.theK	8 or 4	Number of alternative paths, k
wkco.firstFiberOnly	false	Use only first fiber on each cable?

Both progress curves show a similar pattern of rapid initial reduction in network cost up to about generation 100, followed by more gradual improvement for the balance of the run. Examination of the operator probability adaptation curves indicates that, during this first phase of the run, the dominant operator is shift-out (WWSO), although this proves much less useful towards the end of the run. In contrast, however, after about generation 100, mutation (WKM), which started with a very low initial probability, becomes much more productive, as does crossover (WKCO). These changes probably reflect the far greater opportunity for local improvements by shift-out (and to a lesser extent, reroute) in the initial random individuals, whereas later in the run, crossover and mutation are needed to introduce diversity into the highly-evolved population, allowing further cost reductions.

Supplementing the results provided in Figures 6.3 and 6.4, further minimum cost allocation results, showing typical progress and operator-probability adaptation over single runs of both GA4 and GA5 on Network 1 for both (α, β, γ) = (1, 1, 0.5) and (0.8, 0.8, 0.5) are illustrated in Figures 6.5 - 6.8.

In addition to the GAs, the three wavelength allocation heuristics by Wuttisittikulkij & O'Mahony (W&O) and Wauters & Demeester (W&D1 and W&D2) were also applied

to Networks 1-5 for the same six combinations of (*α, β, γ*). As before, Wuttisittikulkij & O'Mahony's non-deterministic algorithm was allowed the best of five runs.

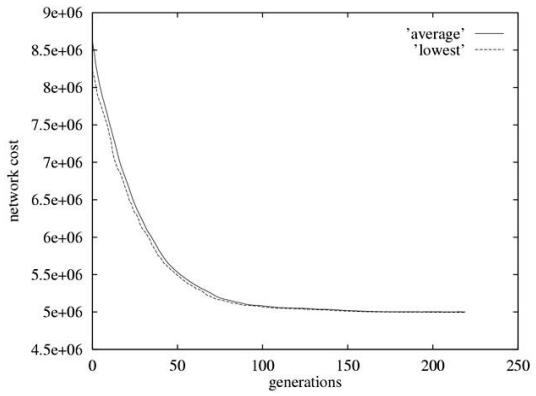
A summary of the results of applying all six GAs and heuristics is recorded in Tables 6.14-6.19, which lists the network cost, or the best cost for five runs, as appropriate. As before, the best overall result for each network, for each combination of *(α, β, γ*), is displayed in bold. In addition, the full set of results from applying the three GAs and W&0 to all five of the networks is recorded in Tables 6.20-6.31.

For *γ* = 0.5, a value regarded as being reasonably representative of actual costs, the best heuristic for both values of *α, β* was found to be W&O, which is thus providing a good trade-off between capacity cost and link wavelength requirements. Nevertheless, for all five networks the lowest network costs were actually obtained by the hybrid algorithm, GA3 (Tables 6.14 and 6.17).

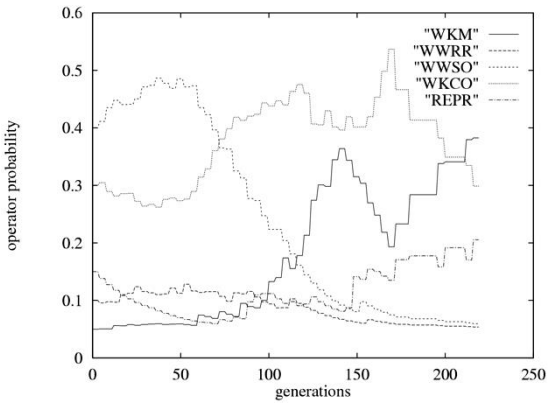
With *γ*=1, focusing the network cost on capacity rather than wavelength requirement, the best heuristic for both values of *α, β* was W&D1, and by a wide margin; this is perhaps to be expected, as W&D1 uses shortest paths (rather than k-lowest hop count or k-shortest paths, as do W&O and W&D2), thereby placing the emphasis on cost instead of wavelength requirement. However, once again, the hybrid algorithm, GA4, provided the best results for all seven networks (Tables 6.15 and 6.18).

Finally, setting *γ* = 0, so that the network cost depends entirely on wavelength requirement rather than capacity, the best heuristic for both values of *α, β* was W&O; this is no doubt due to the superiority of W&O, with respect to NWR, compared to the other two heuristics. Although for minimum NWR, GA2 matched the results of W&O, here, disappointingly, the GA5 hybrid algorithm was only able to obtain the best overall cost for four networks with *α*=*β*=1 and two networks with *α*=*β*=0:8, although its results were close to those of W&O on the other eight (Tables 6.16 and 6.19).

Overall, the experimental results show that, at least for the five test networks, the GA/heuristic hybrid approach has again shown itself to be as promising as has been hoped, as in competition with three recent heuristics, it was able to obtain the lowest network costs for γ set to both 0.5 and 1; and in the case of γ = 0, was able to approach the results of the best of the allocation heuristics.

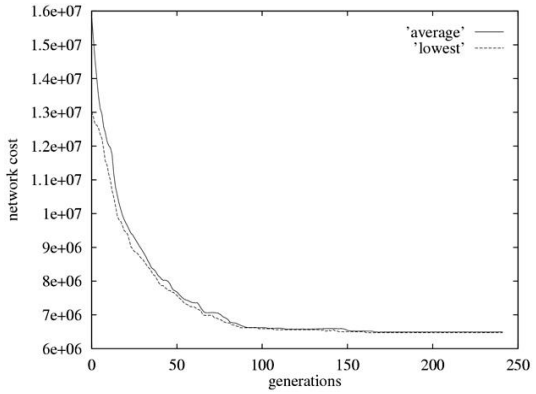


a) Progress over a single run

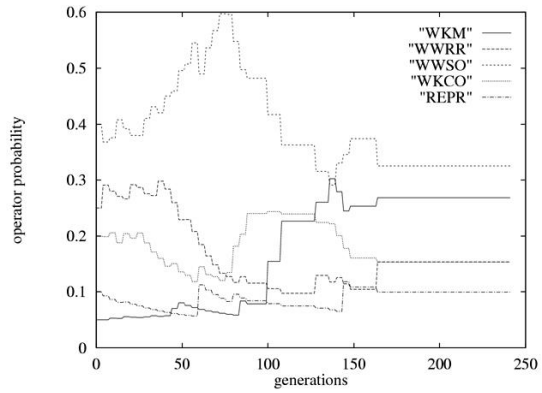


b) Operator probability adaptation

Figure 6.5 Typical results for GA4, α=β=1, and γ=1

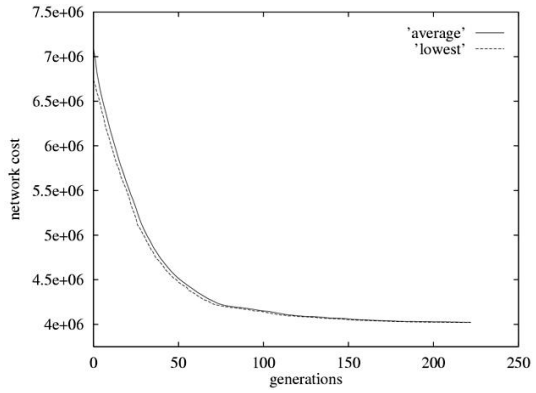


a) Progress over a single run

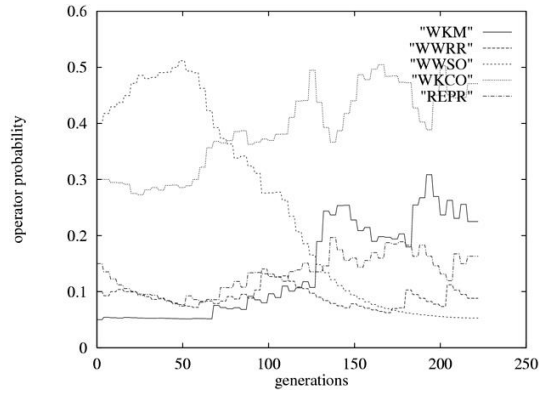


b) Operator probability adaptation

Figure 6.6 Typical results for GA5, α=β=1, and γ=0

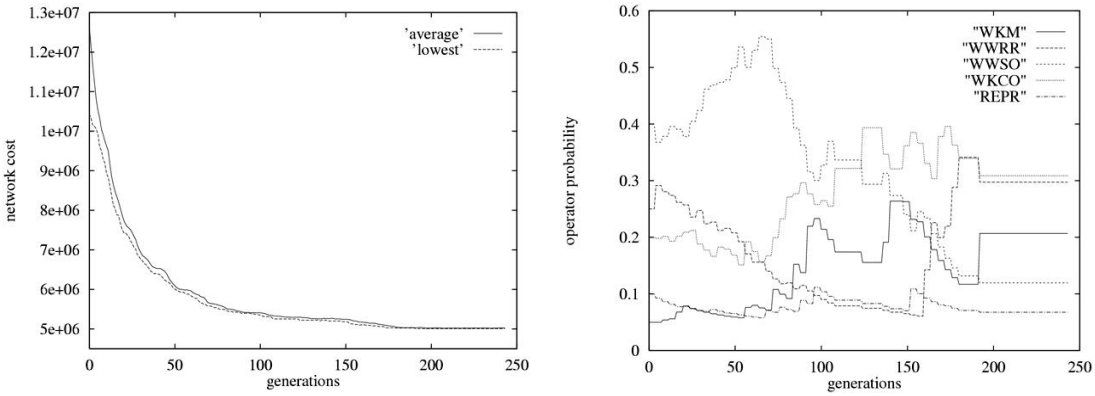


a) Progress over a single run



b) Operator probability adaptation

Figure 6.7 Typical results for GA4, α=β=0.8, and γ=1



a) Progress over a single run

b) Operator probability adaptation

Figure 6.8 Typical results for GA5, α=β=0.8, and γ=0

Table 6.14 Minimum cost allocation results for $\alpha = \beta = 1$ and $\gamma = 0{:}5$

Network	W&O	W&D1	W&D2	GA3
1	5.909	6.007	6.358	5.579
2	5.151	5.241	5.278	4.886
3	5.153	5.707	5.637	4.976
4	5.413	5.528	5.765	5.117
5	5.424	5.773	5.658	5.108

Table 6.15 Minimum cost allocation results for $\alpha = \beta = 1$ and $\gamma = 1$

Network	W&O	W&D1	W&D2	GA4
1	5.704	5.172	5.896	4.994
2	5.081	4.589	5.046	4.436
3	5.051	4.625	5.446	4.394
4	5.313	4.808	5.388	4.645
5	5.302	4.806	5.326	4.64

Table 6.16 Minimum cost allocation results for $\alpha = \beta = 1$ and $\gamma = 0$

Network	W&O	W&D1	W&D2	GA5
1	6.036	6.843	6.82	6.116
2	5.213	5.893	5.51	5.228
3	5.229	6.789	5.828	5.362
4	5.503	6.247	6.142	5.499
5	5.544	6.74	5.991	5.523

Table 6.17 Minimum cost allocation results for $\alpha = \beta = 0.8$ and $\gamma = 0.5$

Network	W&O	W&D1	W&D2	GA3
1	4.653	4.89	5.103	4.45
2	4.089	4.238	4.212	3.909
3	4.138	4.656	4.559	3.866
4	4.312	4.474	4.608	4.062
5	4.313	4.684	4.518	4.040

Table 6.18 Minimum cost allocation results for $\alpha = \beta = 0.8$ and $\gamma = 1$

Network	W&O	W&D1	W&D2	GA4
1	4.489	4.233	4.757	4.014
2	4.036	3.731	4.039	3.531
3	4.056	3.802	4.414	3.541
4	4.23	3.91	4.32	3.716
5	4.216	3.913	4.267	3.716

Table 6.19 Minimum cost allocation results for $\alpha = \beta = 0.8$ and $\gamma = 0$

Network	W&O	W&D1	W&D2	GA5
1	4.755	5.546	5.448	4.895
2	4.128	4.745	4.384	4.249
3	4.2	5.51	4.704	4.271
4	4.382	5.039	4.896	4.444
5	4.409	5.455	4.769	4.363

Table 6.20 Minimum cost allocation results for $\alpha = \beta = 1$ and $\gamma = 0.5$ using GA3

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	**5.579**	4.930	5.013	**5.117**	5.182	5.37	5.751
2	5.694	4.969	5.016	5.201	5.202	5.434	5.656
3	5.732	4.905	4.979	5.188	5.206	5.361	5.757
4	5.679	4.938	4.979	5.189	5.150	**5.344**	5.681
5	5.629	**4.886**	**4.976**	5.248	**5.108**	5.435	**5.651**
Best	5.579	4.886	4.976	5.117	5.108	5.344	5.651

Table 6.21 Minimum cost allocation results for $\alpha = \beta = 1$ and $\gamma = 1$ using GA4

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	4.996	4.436	4.395	**4.645**	4.641	**4.753**	**5.228**
2	4.996	4.437	4.396	**4.645**	**4.640**	**4.753**	**5.228**
3	4.996	**4.436**	**4.394**	**4.645**	**4.640**	**4.753**	**5.228**
4	4.995	4.438	**4.394**	**4.645**	**4.640**	**4.753**	**5.228**
5	**4.994**	4.438	**4.394**	4.646	4.641	**4.753**	**5.228**
Best	4.994	4.436	4.394	4.645	4.640	4.753	5.228

Table 6.22 Minimum cost allocation results for $\alpha = \beta = 1$ and $\gamma = 0$ using GA5

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	6.471	5.338	5.435	5.861	5.561	5.958	6.007
2	6.188	5.417	**5.362**	**5.499**	5.684	5.787	**6.001**
3	6.310	5.454	5.598	5.910	5.858	6.108	6.029
4	**6.116**	**5.428**	5.384	5.672	**5.523**	**5.781**	6.005
5	6.217	**5.228**	5.422	5.749	5.714	5.954	6.294
Best	6.116	5.228	5.362	5.499	5.523	5.781	6.001

Table 6.23 Minimum cost allocation results for $\alpha = \beta = 0.8$ and $\gamma = 0.5$ using GA3

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	4.616	4.011	4.017	4.136	4.123	4.098	4.001
2	**4.450**	3.941	**3.866**	4.093	4.117	4.065	3.982
3	4.627	**3.909**	4.016	**4.062**	**4.040**	4.169	**3.965**
4	4.532	3.909	3.928	4.144	4.151	4.185	4.034
5	4.650	3.933	3.918	4.103	4.076	4.110	4.000
Best	4.450	3.909	3.866	4.062	4.040	4.065	3.965

Table 6.24 Minimum cost allocation results for $\alpha = \beta = 0.8$ and $\gamma = 1$ using GA4

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	4.020	**3.531**	**3.541**	3.718	3.717	**3.633**	**3.686**
2	**4.014**	**3.531**	**3.541**	3.717	**3.716**	**3.633**	**3.686**
3	4.015	**3.531**	3.545	3.717	3.717	**3.633**	**3.686**
4	4.016	3.533	**3.541**	3.732	3.717	**3.633**	**3.686**
5	4.018	3.532	**3.541**	**3.716**	3.717	**3.633**	**3.686**
Best	4.014	3.531	3.541	3.716	3.716	3.633	3.686

Table 6.25 Minimum cost allocation results for $\alpha = \beta = 0.8$ and $\gamma = 0$ using GA5

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	5.005	4.518	4.597	**4.444**	4.625	4.585	4.300
2	4.910	4.335	4.281	4.546	4.581	4.346	4.381
3	5.147	4.292	**4.271**	4.463	**4.363**	4.491	**4.228**
4	**4.895**	**4.249**	4.374	4.473	4.474	**4.344**	4.237
5	4.946	4.265	4.463	4.556	4.524	4.624	4.230
Best	4.895	4.249	4.271	4.444	4.363	4.344	4.228

Table 6.26 Minimum cost allocation results for f $\alpha = \beta = 1$ and $\gamma = 0.5$ using W&O

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	5.919	**5.151**	5.185	5.445	5.524	5.732	6.093
2	5.980	5.231	5.172	5.434	5.696	5.692	**6.077**
3	6.032	5.161	5.213	**5.413**	5.441	5.811	6.164
4	5.917	5.171	**5.153**	5.416	**5.424**	**5.662**	6.101
5	**5.909**	5.169	5.154	5.477	5.526	5.761	6.232
Best	5.909	5.151	5.153	5.413	5.424	5.662	6.077

Table 6.27 Minimum cost allocation results for f $\alpha = \beta = 1$ and $\gamma = 1$ using W&O

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	5.790	**5.081**	5.053	5.367	5.386	5.540	5.966
2	5.845	5.159	5.067	5.320	5.543	**5.516**	**5.907**
3	5.897	5.109	5.089	**5.313**	5.338	5.569	5.960
4	**5.704**	5.104	**5.051**	5.330	**5.302**	5.533	5.968
5	5.782	5.088	5.078	5.365	5.411	5.615	6.025
Best	5.704	5.081	5.051	5.313	5.302	5.516	5.907

Table 6.28 Minimum cost allocation results for f $\alpha = \beta = 1$ and $\gamma = 0$ using W&O

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	6.048	5.222	5.317	5.523	5.662	5.925	**6.220**
2	6.115	5.303	5.278	5.548	5.850	5.868	6.247
3	6.167	**5.213**	5.337	5.513	**5.544**	6.053	6.368
4	6.131	5.237	5.254	**5.503**	5.547	**5.790**	6.234
5	**6.036**	5.251	**5.229**	5.589	5.641	5.907	6.439
Best	6.036	5.213	5.229	5.503	5.544	5.790	6.220

Table 6.29 Minimum cost allocation results for f $\alpha = \beta = 0.8$ and $\gamma = 0.5$ using W&O

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	4.667	4.090	4.162	4.328	4.392	4.295	4.215
2	4.715	4.146	4.159	4.317	4.527	4.260	**4.203**
3	4.755	**4.089**	4.186	**4.312**	4.329	4.354	4.263
4	4.662	4.107	4.138	**4.312**	**4.313**	**4.246**	4.217
5	**4.653**	4.105	4.140	4.350	4.394	4.315	4.306
Best	4.653	4.089	4.138	4.312	4.313	4.246	4.203

Table 6.30 Minimum cost allocation results for f $\alpha = \beta = 0.8$ and $\gamma = 1$ using W&O

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	4.565	**4.036**	4.058	4.272	4.283	4.148	4.124
2	4.605	4.088	4.072	**4.230**	4.402	**4.123**	**4.082**
3	4.651	4.050	4.090	**4.230**	4.248	4.170	4.115
4	**4.489**	4.056	**4.056**	4.242	**4.216**	4.149	4.123
5	4.552	4.041	4.081	4.260	4.302	4.203	4.157
Best	4.489	4.036	4.056	4.230	4.216	4.123	4.082

Table 6.31 Minimum cost allocation results for f $\alpha = \beta = 0.8$ and $\gamma = 0$ using W&O

Run	Network						
	1	**2**	**3**	**4**	**5**	**6**	**7**
1	4.768	4.144	4.266	4.384	4.501	4.441	**4.306**
2	4.824	4.204	4.245	4.404	4.651	4.398	4.323
3	4.859	**4.128**	4.283	4.395	4.410	4.538	4.412
4	4.836	4.158	4.220	**4.382**	**4.409**	**4.343**	4.311
5	**4.755**	4.169	**4.200**	4.439	4.487	4.427	4.455
Best	4.755	4.128	4.200	4.382	4.409	4.343	4.306

6.3 Execution Time Limitation

The experiments described in this chapter were carried out on an Intel Pentium IV 2.6 Ghz processor with 512 Mbytes of memory. A complementary limitation of the GA/heuristic hybrid approach is the amount of computer time required. For example, while a single run of GA1 takes about 41 seconds, running GA3 takes some 2 hours 35 minutes. When we run the other algorithms on the same environment we observed that, W&O takes 3 hours 10 minutes, W&D1 takes 1 hour 22 minutes and W&D2 takes nearly 2 hours 52 minutes for a single run. Nevertheless, considerable efforts were taken during implementation to keep the processing time down. As well as the beneficial effects of some of the design decisions taken, such as using unsigned integers as bit vectors, employing class-specific memory management for key classes also provided a considerable improvement in execution time. One promising direction for future work would be to exploit the obvious parallelism of the GA framework, and develop a distributed version. Possibilities would include either distributed design assessment, controlled by a master GA on a single processor, or a distributed multi-population approach, with distinct small populations occasionally exchanging promising individuals. In this way, processing limitations of an individual machine might be overcome, and networks of larger size designed.

7. CONCLUSIONS AND FUTURE WORK

This thesis addresses the problem of wavelength assignment and wavelength routing in WDM optical networks. One of the major design issues in these networks is the effective assignment of the limited number of wavelengths to connections so that a higher aggregate capacity can be achieved.

The problem of wavelength assignment and routing is proved to be NP-hard. The present literature on this topic has a large repertoire of heuristics that produce good solutions in some amount of time. These approaches, however, have limited applicability to a practical environment because they have a number of fundamental problems including high time complexity, and lack of scalability with respect to optimal solutions.

In this study, a GA/heuristic based hybrid technique is proposed in order to solve routing and wavelength assignment problem in WDM Optical Networks. Mesh multiple-fiber-link topologies with wavelength continuity constraint are selected as the problem domain.

The routing and wavelength assignment have largely been considered, with the network topology being established before the fiber and wavelength assignment is performed. Object oriented network representation is employed. Operator probability adaptation is used. The experimental results indicate that the main benefit of operator-probability adaptation is in relieving the burden of setting initial probabilities and in performance enhancement at the early phases.

The results obtained by the proposed technique are compared with those obtained by the two other approaches. Wuttisittikulkij and O'Mahony [28] developed a simple, fast and non-iterative heuristic, which works with *k*-lowest hop count paths to obtain a low network wavelength requirement. Wauters and Demeester [29] presented an integer-linear-programming approach that aims maximum throughput, and two heuristics.

Two metrics for network performance assessment are used. One is based on NWR, and the other is a simplified model of network cost. In both cases, the GAs are

compared with recent wavelength-allocation heuristics. For minimum NWR, the GA/heuristic hybrid approach fulfilled its promise, as the blend of a GA with steps from one of the heuristics resulted in both improved performance (over the GA alone) and improved results (over the heuristic alone). In addition, the GA results matched those of the best wavelength-assignment heuristic examined. Similar success is achieved with the minimum cost allocation, as in competition with three recent heuristics, the GA/heuristic hybrid approach is able to obtain the lowest network costs for two settings of the cost parameters (including the one judged as the most realistic). In another case, the technique results are comparable with those of the best of the assignment heuristic approaches.

Overall, the application of the GA/heuristic hybrid technique to routing and wavelength assignment has produced promising results. In particular, problem-specific representation and the incorporation of problem-specific heuristics have been successful to be used in routing and wavelength assignment in optical WDM networks. In many cases, a hybrid GA provided not only better performance than a GA alone, but also better results than conventional heuristics.

In future, as well as exploring a wider range of network sizes, comparisons could be made with more of the allocation heuristics in the literature. The extension to allow the representation of restoration paths for fiber and wavelength assignment is also suggested. Also, another future work could enhance initialization, selection and population dynamics, implement a distributed multi-population GA, or employ multi-objective optimization. The credit-assignment mechanism could be extended in a variety of ways, including explicit reward of operator chains. A single design process implementation integrating both topology and wavelength assignment could be another valuable future work.

REFERENCES

[1] **Mukherjee, B., 1997.** Optical Communication Networks, McGraw-Hill.

[2] **Goldberg, D.E., 1989.** Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley.

[3] **Mitchell, M., 1996.** An Introduction to Genetic Algorithms, MIT Press.

[4] **Fogel, D.B. (Ed.), 1998.** Evolutionary Computation: The Fossil Record, IEEE Press.

[5] **Kirkpatrick, S., Gelatt, C.D., Jr., & Vecchi, M.P., May 1983.** Optimization by Simulated Annealing, *Science,* **220 (4598)**, pp. 671-680.

[6] **Glover, F., 1989.** Tabu search-part I, *ORSA Journal on Computing*, **1 (3)**, pp. 190-206.

[7] **Sinclair, M.C., July 1999.** Evolutionary telecommunications: A summary, *Proc. GECCO'99 Workshop on Evolutionary Telecommunications: Past, Present and Future*, Orlando, Florida, USA, pp. 209-212.

[8] **Davis, L. (Ed.), 1991.** Handbook of Genetic Algorithms, Van Nostrand Reinhold.

[9] **Chbat, M.W., Grard, E., et al., September 1998.** Towards wide-scale all-optical transparent networking: The ACTS Optical Pan-European Network (OPEN) project, *IEEE Journal on Selected Areas in Communications*, **16 (7)**, pp. 1226-1244.

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

[11] **Schwefel, H.-P., May 1998.** On natural life's tricks to survive and evolve, Plenary Session, *IEEE Intl. Conf. on Evolutionary Computation (ICEC'98)*, Anchorage, Alaska, USA.

[12] **T. E. Stern and K. Bala, 2000.** Multiwavelength Optical Networks, Prentice Hall, Upper Saddle River, New Jersey.

[13] **O. Gerstel, B. Li, A. McGuire, G. N. Rouskas, K. Sivalingam, and Z. Zhang (Eds.), October 2000.** Special issue on protocols and architectures for next generation optical WDM networks, *IEEE Journal Selected Areas in Communications*, **18 (10)**.

[14] **G.-K. Chung, K.-I. Sato, and D. K. Hunter (Eds.), December 2000.** Special issue on optical networks, *Journal of Lightwave Technology*, **18 (12)**.

[15] **H. Zang, J. P. Jue, and B. Mukherjee, January 2000.** A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks, *Optical Networks*, **1 (1)**, pp. 47-60.

[16] **B. Ramamurty and B. Mukherjee, September 1998.** Wavelength conversion in WDM networking, *IEEE Journal Selected Areas in Communications*, **16 (7)**, pp. 1061-1073.

[17] **V. Sharma and E. A. Varvarigos, March 1999.** Limited wavelength translation in all-optical WDM mesh networks, *In Proceedings of INFOCOM '98 IEEE*, pages 893-901.

[18] **Y. Zhu, G. N. Rouskas, and H. G. Perros, December 2000.** A path decomposition algorithm for computing blocking probabilities in wavelength routing networks, *IEEE/ACM Transactions on Networking*, **8 (6)**, pp. 747-762,.

[19] **S. Subramaniam, M. Azizoglu, and A. Somani, August 1996.** All-optical networks with sparse wavelength conversion, *IEEE/ACM Transactions on Networking*, **4 (4)**, pp. 544-557.

[20] **R. Dutta and G. N. Rouskas, January 2000.** A survey of virtual topology design algorithms for wavelength routed optical networks, *Optical Networks Magazine*, **1 (1)**, pp. 73-89.

[21] **R. Ramaswami and K. N. Sivarajan, 1996.** Design of logical topologies for wavelength-routed optical networks, *IEEE Journal Selected Areas in Communications*, **14 (5)**, pp. 840-851.

[22] **E. Leonardi, M. Mellia, and M. A. Marsan, 2000.** Algorithms for the logical topology design in WDM all-optical networks, Optical Networks, **1 (1)**, pp.35-46.

[23] **Banerjee, D. & Mukherjee, B., 1996.** A practical approach for routing and wavelength assignment in large wavelength-routed optical networks, *IEEE Journal on Selected Areas in Communications,* **14 (5),** pp. 903-908.

[24] **B. Mukherjee et al., 1996.** Some principles for designing a wide-area WDM optical network, *IEEE/ACM Transactions on Networking*, 4(5), pp. 684-696.

[25] **Z. Zhang and A. Acampora, 1995.** A heuristic wavelength assignment algorithm for multihop WDM networks with wavelength routing and wavelength reuse, *IEEE/ACM Transactions on Networking*, 3(3), pp. 281288.

[26] **I. Chlamtac, A. Ganz, and G. Karmi, 1993.** Lightnets: Topologies for high-speed optical networks, *Journal of Lightwave Technology*, **11**, pp. 951-961.

[27] **S. Banerjee and B. Mukherjee, 1993.** Algorithms for optimized node placement in shuffienet-based multihop lightwave networks, *Proceedings of INFOCOM '93. IEEE*.

[28] **Wuttisittikulkij, L. & O'Mahony, M.J., 1995.** A simple algorithm for wavelength assignment in optical networks, *Proc. 21st Eur. Conf. on Opt. Comm. (ECOC'95)*, Brussels, pp. 859-862 .

[29] **Wauters, N. & Demeester, P., 1996.** Design of the optical-path layer in multiwavelength cross-connected networks, *IEEE Journal on Selected Areas in Communications*, **14 (5)**, pp. 881-892.

[30] **Yen, J.Y., 1971.** "Finding the k shortest loopless paths in a network", *Management Science*, **17 (11)**, pp. 712-716.

[31] **Nagatsu, N., Hamazumi, Y. & Sato, K.I., 1995.** Number of wavelengths required for constructing large-scale optical path networks, *Electronics and Communications in Japan, Part I – Communications*, **78 (9)**, pp. 1-11.

[32] **Back, T., Hammel, U. & Schwefel, H.-P., 1997.** Evolutionary computation: Comments on the history and current state, *IEEE Transactions on Evolutionary Computation*, **1 (1)**, pp. 3-17 (reprinted in [4]).

[33] **Friedman, G.J., 1956.** Selective Feedback Computers for Engineering Synthesis and Nervous System Analogy, M*aster's thesis*, University of California, Los Angeles, USA, (reprinted in [4]).

[34] **Fraser, A.S., 1957.** Simulation of genetic systems by automatic digital computers: I. Introduction, *Australian J. Biological Sciences*, **10**, pp.484-491 (reprinted in [4]).

[35] **Box, G.E.P., 1957.** Evolutionary operation: A method for increasing industrial productivity, *Applied Statistics*, **6 (2)**, pp. 81-101.

[36] **Friedberg, R.M., 1958.** A learning machine: Part I, *IBM J. Research and Development*, **2 (1)**, pp. 2-13.

[37] **Friedberg, R.M., Dunham, B. & North, J.H., 1959.** A learning machine: Part II, *IBM J. Research and Development*, **3**, pp. 282-287

[38] **Holland, J.H., 1975.** Adaptation in Natural and Artificial Systems, University of Michigan Press.

[39] **Holland, J.H., 1982.** Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence (2nd Ed.), MIT Press.

[40] **Heitkotter, J. & Beasley, D., The Hitch-Hiker's, 2000.** *Guide to Evolutionary Computation*, Issue **8.1**.

[41] **Esa Hyytia, Jorma Virtamo, 1998.** *Laboratory of Telecommunications Technology Report*.

[42] **Fogel, L.J., Owens, A.J. & Walsh, M.J., 1965.** Artificial intelligence through simulated Evolution, *Proc. 2nd Cybernetic Sciences Symposium*, pp. 131-155 (reprinted in [4]).

[43] **Fogel, L.J., Owens, A.J. & Walsh, M.J., 1966.** Artificial Intelligence through Simulated Evolution, Wiley.

[44] **Fogel, L.J., 1999.** Intelligence through Simulated Evolution, Wiley.

[45] **Rechenberg, I., 1965.** Cybernetic solution path of an experimental problem, Royal Aircraft Establishment, Library Translation 1122, (reprinted in [4]).

[46] **Schwefel, H.-P., 1995.** Evolution and Optimum Seeking, Wiley.

[47] **Koza, J.R., 1989.** Hierarchical genetic algorithms operating on populations of computer programs, *Proc. 11th Intl. Joint Conf. on Artificial Intelligence*, pp. 768-774 (reprinted in [4]).

[48] **Koza, J.R., 1992.** Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press.

[49] **Koza, J.R., 1994**. Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press.

[50] **Koza, J.R., Bennett, F.H., III, Andre, D. & Keane, M.A., 1999.** Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufmann Publishers.

[51] **Kinnear, K.E., Jr. (Ed), 1994.** Advances in Genetic Programming, MIT Press.

[52] **Angeline, P.J. & Kinnear, K.E., Jr. (Eds), 1996.** Advances in Genetic Programming, Volume II, MIT Press.

[53] **Spector, L., Langdon, W.B., O'Reilly, U.-M. & Angeline, P.J. (Eds), 1999.** Advances in Genetic Programming, Volume III, MIT Press.

[54] **Banzhaf, W., Nordin, P., Keller, R.E. & Francone, F.D., 1998.** Genetic Programming --- An Introduction, Morgan Kaufmann Publishers.

[55] **Nordin, P., 1997.** Evolutionary Program Induction of Binary Machine Code and its Application, Krehl Verlag.

[56] **Holland, J.H. & Reitman, J.S., 1978.** Cognitive systems based on adaptive algorithms, in Waterman, D.A. & Hayes-Roth, F. (Eds), Pattern-Directed Inference Systems, Academic Press, pp. 313-329, (reprinted in [4]).

[57] **Sipper, M., 1997.** Evolution of Parallel Cellular Machines, Lecture Notes in Computer Science, **1194**, Springer-Verlag.

[58] **Thompson, A., 1998.** Hardware Evolution: Automatic Design of Electronic Circuits in Reconfigurable Hardware by Artificial Evolution, Springer-Verlag.

[59] **Mange, D. & Tomassini, M. (Eds), 1998.** Bio-Inspired Computing Machines, Presses Polytechniques et Universitaries Romandes.

[60] **Jog, P., Suh, J.Y. & Van Gucht, D., 1989.** The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem, *Proc. 3rd Intl. Conf. on Genetic Algorithms (ICGA'89)*, George Mason University, Arlington, Virginia, USA, pp. 110-115 .

[61] **Bramlette, M.F. & Bouchard, E.E.,** Genetic algorithms in parametric design of aircraft, in [9], Ch. 10, pp. 109-123.

[62] **Cox, L.A., Jr., Davis, L. & Qiu, Y.,** Dynamic anticipatory routing in circuit-switched telecommunications networks, in [9], Ch. 11, pp. 124-143.

[63] **Davidor, Y.,** A genetic algorithm applied to robot trajectory generation, in [9], Ch. 12, pp. 144-165.

[64] **Forrest, S. & Mayer-Kress, G.,** Genetic algorithms, nonlinear dynamic systems, and models of international security, in [9], Ch. 13, pp. 166-185.

[65] **Grefenstette, J.J.,** Strategy acquisition with genetic algorithms, in [9], Ch. 14, pp. 186-201.

[66] **Harp, S.A.,** Genetic sysnthesis of neural network architecture, in [9], Ch. 15, pp. 202-221.

[67] **Karr, C.L.,** Air-injected hydrocyclone optimization via genetic algorithm, in [9], Ch. 16, pp. 222-236.

[68] **Liepins, G.E. & Potter, W.D.,** A genetic algorithm approach to multiple-fault diagnosis, in [9], Ch. 17, pp. 237-250.

[69] **Lucasius, C.B., Blommers, M.J.J., Buydens, L.M.C & Kateman, G.,** A genetic algorithm for conformational analysis of DNA, in [9], Ch. 18, pp. 251-281.

[70] **Montana, D.J.,** Automated parameter tuning for interpretation of synthetic images, in [9], Ch. 19, pp. 282-311.

[71] **Powell, D.J., Skolnick, M.M. & Tong, S.S.,** Interdigitation: a hybrid technique for engineering design optimization employing genetic algorithms, expert systems, and numerical optimization, in [9], Ch. 20, pp. 312-331.

[72] **Syswerda, G.,** Schedule optimization using genetic algorithms, in [9], Ch. 21, pp. 332-349.

[73] **Whitley, D., Starkweather, T. & Shaner, D.,** The travelling salesman and sequence scheduling: quality solutions using genetic edge recombination, in [9], Ch. 22, pp. 350-372.

[74] **Schnecke, V. & Vornberger, O., November 1997.** Hybrid genetic algorithms for constrained placement problems, *IEEE Transactions on Evolutionary Computation*, **1 (4)**, pp. 266-277.

[75] **Zhang, F., Zhang, Y.F. & Nee, A.Y.C., November 1997.** Using genetic algorithms in process planning for job shop machining, *IEEE Transactions on Evolutionary Computation*, **1 (4)**, pp. 278-289.

[76] **Xiao, J., Michalewicz, Z., Zhang, L. & Trojanowski, K., 1997.** Adaptive evolutionary planner/navigator for mobile robots, *IEEE Transactions on Evolutionary Computation*, **1 (1)**, pp. 18-28.

[77] **Smierzchalski, R. & Michalewicz, Z., September 2000.** Modeling of ship trajectory in collision situations by an evolutionary algorithm, *IEEE Transactions on Evolutionary Computation*, **4 (3)**, , pp. 227-241.

[78] **Magyar, G., Johnsson, M. & Nevalainen, O., 2000.** An adaptive hybrid genetic algorithm for the three-matching problem, *IEEE Transactions on Evolutionary Computation,* **4 (2)**, pp. 135-146.

[79] **Bhandarkar, S.M. & Zhang, H., 1999.** Image segmentation using evolutionary computation, *IEEE Transactions on Evolutionary Computation*, **3 (1)**, pp. 1-21.

[80] **Eiben, A.E., Hinterding, R. & Michalewicz, Z., 1999.** Parameter control in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation*, **3 (2)**, pp. 124-141.

[81] **Hinterding, R., Michalewicz, Z. & Eiben, A.E., 1997.** Adaptation in evolutionary computation: A survey, *Proc. 1997 IEEE Intl. Conf. on Evolutionary Computation (ICEC'97)*, Indianapolis, Indiana, USA, pp. 65-69.

[82] **Tuson, A.L., 1995.** Adapting Operator Probabilities in Genetic Algorithms, *M.Sc. thesis*, Dept. of Artificial Intelligence, Univ. of Edinburgh, UK

[83] **Tuson, A. & Ross, P., 1998.** Adapting operator settings in genetic algorithms, *Evolutionary Computation*, **6 (2)**, pp. 161-184.

[84] **Fogarty, T.C., 1989.** Varying the probability of mutation in the genetic algorithm, *Proc. 3rd Intl. Conf. on Genetic Algorithms (ICGA'89)*, George Mason University, Arlington, Virginia, USA, pp. 104-109.

[85] **Davis, L., 1989.** Adapting operator probabilities in genetic algorithms, *Proc. 3rd Intl. Conf. on Genetic Algorithms (ICGA'89)*, George Mason University, Arlington, Virginia, USA, pp. 61-69.

[86] **Srinivas, M. & Patnaik, L.M., 1994.** Adaptive probabilities of crossover and mutation in genetic algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, **24 (4)**, pp. 656-667.

[87] **Julstrom, B.A., 1995.** What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm, Proc. 6th Int. Conf. on GAs (ICGA'95), Univ. of Pittsburgh, USA, pp. 81-87.

[88] **Tuson, A. & Ross, P., 1996.** Cost based operator rate adaptation: An investigation, *Proc. 4th Intl. Conf. on Parallel Problem Solving From Nature (PPSN IV)*, Berlin, Germany.

[89] **Tan, L.G. & Sinclair, M.C., 1995.** Wavelength assignment between the central nodes of the COST 239 European optical network, *Proc. 11th UK Perf. Engineering Workshop*, Liverpool, UK, September, pp. 235-247.

[90] **Nagatsu, N., Hamazumi, Y. & Sato, K.I., 1995.** Optical path accommodation designs applicable to large scale networks, *IEICE Transactions on Communications*, **E78-B (4)**, pp. 597-607.

[91] **Baroni, S. & Bayvel P., 1997.** Wavelength requirements in arbitrary connected wavelength-routed optical networks, *Journal of Lightwave Technology*, **15 (2)**, pp. 242-251.

[92] **O'Mahony, M.J., Simeonidou, D., Yu, A. & Zhou, J., 1995.** The design of a European optical network, *Journal of Lightwave Technology*, **13 (5)**, pp. 817-828.

[93] **Richardson, J.T., Palmer, M.R., Liepins, G. & Hilliard, M., 1989.** Some guidelines for genetic algorithms with penalty functions, *Proc. 3rd Intl. Conf. on Genetic Algorithms (ICGA'89)*, George Mason University, Arlington, Virginia, USA, pp. 191-197.

[94] **Smith, A.E. & Tate, D.M., 1993.** Genetic optimization using a penalty function, *Proc. 5th Intl. Conf. on Genetic Algorithms (ICGA'93)*, University of Illinois at Urbana-Champaign, USA, pp. 499-505.

[95] **Sato, K.I., Okamoto, S. and Hadama, H., 1994.** Network performance and integrity enhancement with optical path layer technologies, *IEEE Journal on Selected Areas in Communications*, **12 (1)**, pp. 159-170.

[96] **Nagatsu, N., Okamoto, S. & Sato, K.I., 1996.** Optical path cross-connect system scale evaluation using path accommodation design for restricted wavelength multiplexing, *IEEE Journal on Selected Areas in Communications*, **14 (5)**, pp. 893-902.

[97] **Nagatsu, N. & Sato, K.I., 1995.** Optical path accommodation design enabling cross-connect system scale evaluation, *IEICE Transactions on Communications*, **E78-B (9)**, pp. 1339-1343.

[98] **Kershenbaum, A., 1993.** Telecommunications Network Design Algorithms, MacGraw-Hill.

BIOGRAPGHY

A. Çağatay Talay was born in 1976 in Ceyhan/Adana. He completed his high school education in Haydarpaşa Anatolian Technical High School. He received his B.Sc. in 2001 from the Computer Engineering Department of Yıldız Technical University, after which, he started Computer Engineering graduate program in the Institute of Science and Technology in Istanbul Technical University. He still works as a research and teaching assistant in the Computer Engineering Department of the same university since 2001.