WAVELENGTH ASSIGNMENT IN OPTICAL BURST SWITCHING NETWORKS USING NEURO-DYNAMIC PROGRAMMING

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Feyza KEÇELİ

September 2003

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

Assist. Prof. Dr. Ezhan Karaşan (Supervisor)

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

Assist. Prof. Dr. Nail Akar

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

Prof. Dr. Ömer Morgül

Approved for the Institute of Engineering and Science:

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

ABSTRACT

WAVELENGTH ASSIGNMENT IN OPTICAL BURST SWITCHING NETWORKS USING NEURO-DYNAMIC PROGRAMMING

Feyza KEÇELİ
M.S. in Electrical and Electronics Engineering
Supervisor: Assist. Prof. Dr. Ezhan Karaşan
September 2003

All-optical networks are the most promising architecture for building large-size, huge-bandwidth transport networks that are required for carrying the exponentially increasing Internet traffic. Among the existing switching paradigms in the literature, the optical burst switching is intended to leverage the attractive properties of optical communications, and at the same time, take into account its limitations. One of the major problems in optical burst switching is high blocking probability that results from one-way reservation protocol used. In this thesis, this problem is solved in wavelength domain by using smart wavelength assignment algorithms. Two heuristic wavelength assignment algorithms prioritizing available wavelengths according to reservation tables at the network nodes are proposed. The major contribution of the thesis is the formulation of the wavelength assignment problem as a continuous-time, average cost dynamic programming problem and its solution based on neuro-dynamic programming. Experiments are done over various traffic loads, burst lengths, and number of wavelength converters with a pool structure. The simulation results show that the wavelength assignment algorithms proposed for optical burst switching networks in the thesis perform better than the wavelength assignment algorithms in the literature that are developed for circuit-switched optical networks.

Keywords: Optical Burst Switching (OBS), Just-Enough-Time (JET) Protocol, Wavelength Assignment Algorithms, Reinforcement Learning, Neuro-dynamic Programming

ÖZET

OPTİK ÇOĞUŞMA ANAHTARLAMA AĞLARINDA SİNİRSEL DİNAMİK PROGRAMLAMA KULLANARAK DALGABOYU ATAMA

Feyza KEÇELİ
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. Ezhan Karaşan
Eylül 2003

Tam optik ağlar üstel artan internet trafiği taşıyan büyük ölçekli ve bant genişlikli taşıma ağları kurmak için en umut vadeden mimaridir. Yazında varolan anahtarlama örnekleri içinde optik çoğuşma anahtarlama optik iletişimin çekici özelliklerini arttırmaya en eğilimli olandır ve aynı zamanda sınırlarını da göz önüne alır. Optik çoğuşma anahtarlamanın belli başlı sorunlarından biri, kullanılan tek yönlü rezervasyon protokollerinden ileri gelen yüksek reddedilme olasılığıdır. Bu tezde dalgaboyu dağarcığında akıllı dalga boyu atama algoritmaları ile bu sorun çözülmüştür. Ağ düğümlerindeki rezervasyon tablolarına göre uygun dalgaboylarını önceliklendiren iki buluşsal dalgaboyu algoritması önerilmiştir. Bu tezin en büyük katkısı dalgaboyu atama sorununu ve sinirsel dinamik programlamaya dayanan çözümünü sürekli zaman ortalama ceza dinamik programlamaya dayanarak formüle etmesidir. Değişken trafik yükleri, çoğuşma uzunlukları ve farklı sayıda havuz yapılı dalgaboyu çevirgeçleri üzerinden deneyler yapılmıştır. Benzetim sonuçları gösteriyor ki bu tezde optik çoğuşma anahtarlama ağları için önerilen dalgaboyu atama algoritmaları yazında devre anahtarlama optik ağları için geliştirilmiş dalgaboyu atama algoritmalarından daha iyi sonuç vermektedir.

Anahtar Kelimeler: Optik Çoğuşma Anahtarlama, Tam Yeter Zaman Protokolü, Dalgaboyu Atama Algoritmaları, Takviyelendirerek Öğretme, Sinirsel Dinamik Programlama

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Ezhan Karaşan for his supervision, guidance, suggestions, instruction and encouragement through the development of this thesis.

I would like to express my special thanks and gratitude Dr. Nail Akar and Dr. Ömer Morgül for reading the manuscript and commenting on the thesis.

I would like to express my appreciation to my parents, my brother Murat and my friends, especially İnanç İnan and Tuba Bayık, for their continuous support and love through the development of this thesis, without their help, this thesis would not have been possible.

Finally, I would like to thank to Miss. Mürüvet Parlakay, for her always smiling face and sincere attitude.

Contents

List of Figures

List of Tables

Glossary

AHC	Adaptive Heuristic Critic	23
ATM	Asynchronous Transfer Mode	6
BBM	Buffered Burst Multiplexers	10
DR	Delayed Reservation	10
DWDM	Dense Wavelength-Division Multiplexing	1
E/O	Electrical-to-Optical	1
fc	Full conversion	53
FDL	Fiber Delay Line	2
IP	Internet Protocol	6
JET	Just-Enough-Time	2
JET-FA	Just-Enough-Time for Fairness	16
JIT	Just-In-Time	2
MDP	Markov Decision Process	3
MFM	Most-fit-Min	50
MFR	Most-fit-Rand	50
MPLS	Multiprotocol Label Switching	9
nc	No conversion	53
NDP	Neuro-Dynamic Programming	4
NSFNET	National Science Foundation Network	30
OBS	Optical Burst Switching	2
O/E	Optical-to-Electrical	1
O/E/O	Optical-to-Electrical-to-Optical	2
OSPF	Open Shortest Past First	28
RAM	Read Available Memory	14

Chapter 1

Introduction

Studies show that bandwidth usage in the Internet is doubling every six to twelve months [1]. Fiber is the most promising physical medium to meet such emerging transport requirements because fiber-optic cable carries information farther, faster, and more reliably than other types of cable. The enormous deliverable bandwidth of fibers can be used more effectively with the advances in DWDM (dense wavelength-division multiplexing) technology. This optical multiplexing technique allows better exploration of fiber capacity by simultaneously transmitting multiple high-speed channels on different frequencies (wavelengths) [2, 3]. However, still the bandwidth is wasted due to the requirement of optical-to-electrical (O/E) and electrical-to-optical (E/O) conversions at every node, and hence fails to take advantage of the wavelength routing capability provided by DWDM technology. What is needed to exploit the high speed of the fiber cables is then to have all-optical networks, where data is kept in the optical domain at all intermediate nodes.

Today's optical switching paradigms are circuit switching, packet switching and novel optical burst switching. Circuit switching ends up with low utilization of bandwidth due to its two-way reservation paradigm and long propagation delays between nodes since fiber cables are generally deployed over routes longer than 500km. On the other hand, packet switching is faster than circuit switching, and can efficiently use the bandwidth. However, due to the tight coupling in time between the payload and header as well as the store-and-forward nature of packet switching, each packet needs to be buffered at every intermediate node. At present, using fiber delay lines (FDLs) is the most practical way to implement optical buffers. Nevertheless, FDLs are scarce and expensive resources, moreover they can generate only a limited delay to data. In the long term, optical packet switching seems to be a promising technology, but due to its complexity it is expected to remain a research topic for some years.

In recent years, a novel paradigm, named optical burst switching (OBS), has been proposed to form an all-optical layer [4]. The incentive of this new idea is to retain advantages of the two approaches indicated above, while eliminating their shortcomings as much as possible. The first step is to change the basic block from a fixed-length packet to a burst that is a super packet with variable size. Unlike a packet, a burst is a pure payload. Each burst is associated with a control packet recording related control information of the burst, e.g., burst length and routing information. In this way, the control overhead is alleviated. A control packet goes through O/E/O conversion at each intermediate node for electronically processing, while a burst remains completely in the optical domain along the path without buffering. The bandwidth reservation is a one-way process [4]. Compared with wavelength routing, the burst starts transmission without waiting for an acknowledgement from destination and the problem of significant signaling delay can be eliminated. In addition, the separation between a control packet and its burst in both time and wavelength domain can avoid buffering as well as synchronization problem in optical packet switching [5].

According to signaling schemes, there can be various OBS protocols, e.g., Tell-n-Go (TAG) by reservation without an offset time, Just-In-Time (JIT) by open-ended reservation and Just-Enough-Time (JET) by close-ended reservation [4-6] etc. In all protocols, a burst is transmitted after its control packet without waiting for an acknowledgement. In TAG, control packet and burst is tightly coupled in time. At each intermediate node control packet is processed, and if available, bandwidth is instantaneously reserved for burst following the control packet without an offset time. In JIT, there are two types of control packets corresponding to a burst: a setup packet and a release packet. At each intermediate node, the desired bandwidth is reserved from the time at which the setup packet has been processed and freed after receiving the related release packet. On the other hand, bandwidth is reserved from the time at which the burst will arrive at the intermediate node in JET and just allocated for the burst duration indicated in the control packet. Since close-ended reservation gains best resource utilization of all, we focus on JET-based OBS paradigm in this thesis work.

1.1 Scope and Contributions of the Thesis

Among various optical switching paradigms, OBS shows advantages in terms of switching efficiency for bursty IP traffic and optical hardware feasibility. However, the high blocking probability is one of the major problems in optical burst switching due to its inherent one-way reservation protocol. Since data bursts are sent out without waiting for the acknowledgement from the receiver, the control packet thus the burst could be blocked in an intermediate node due to the resource contention. In this case, the burst has to be dropped. Since each burst must be assigned a specific path and a wavelength on every link of the assigned path, the resource contention occurs when two or more bursts on the same wavelength are desired to be routed to the same link at the same time.

In case of a reservation conflict, i.e., the wavelength on the output link is already reserved; there are three alternatives for contention resolution. First solution appears in the wavelength domain, where by means of wavelength conversion, a burst can be sent on a different available wavelength channel of the corresponding link. Second in time domain, by applying an FDL buffer, a burst can be delayed until the contention situation is resolved. Lastly in space domain, by deflection routing, a burst is sent to a different output link of the node and consequently on a different route towards its destination node.

In this thesis, we address the reservation conflict problem in wavelength domain, by using smarter wavelength assignment algorithms than previously proposed. Well-known heuristic solutions for wavelength assignment problem in circuit-switched optical networks, e.g. first-fit, random, most-used, least-loaded, min-sum etc., do not result with reasonable performance, when applied to OBS networks. Moreover, some of them cannot be implemented directly according to their definitions, unless a few adjustments are done. Interestingly, since wavelength assignment has to be done in a distributed manner in OBS networks, random wavelength assignment, usually the worst performing of all, results with lower average blocking probabilities than other conventional heuristics. Therefore, we propose two simple heuristic wavelength assignment algorithms that improve blocking probability beyond the random heuristic. The main contribution is to develop a dynamic algorithm for wavelength assignment in OBS networks based on neuro-dynamic programming (reinforcement learning).

Under the assumptions like memoryless interarrival and fixed holding times, wavelength assignment problem can be considered as a Semi-Markov Decision Process (MDP). Therefore, minimizing the average number of bursts blocked per unit time is formulated as a stochastic dynamic programming problem. However, for large problems, the exponential computational explosion with the problem dimension does not allow for an exact solution.

Neuro-dynamic programming (NDP) is a simulation-based approximate dynamic programming methodology to produce near optimal solutions for large scale dynamic optimization problems. The main idea is to construct an approximate cost-to-go function by using some features extracted from the current state of the network, and optimize it by tuning the parameters associated with these features [7]. In this thesis, two kinds of features are extracted from the network and an appropriate feature vector is generated from the combination of these features. Namely, these features are availability of wavelength converters and local availability of wavelengths. Then, simulation-based methods are employed to tune these parameters.

The main contribution of this thesis is that by using heuristic algorithms proposed and by using the neuro-dynamic programming method, together with the features defined in the spirit of proposed heuristics for the wavelength assignment problem in OBS networks, it is possible to obtain smaller average blocking probabilities than that of random wavelength assignment algorithm. Moreover, the effect of wavelength conversion and burst length to the blocking probability in OBS networks at varying traffic rates are also examined throughout the simulations.

1.2 Outline of the Thesis

This thesis is organized as follows: Chapter 2 reviews some background information about basic switching paradigms, especially optical burst switching. The used OBS protocol, JET (Just enough time), is described, together with reservation conflict solution methods. Chapter 3 includes basics of reinforcement learning which constitutes basis of neuro-dynamic programming formulation. In Chapter 4, proposed wavelength assignment algorithms are given. Two heuristic wavelength assignment algorithms are first stated, and neuro-dynamic programming formulation of wavelength assignment for OBS networks is done. Chapter 5 displays simulation environment and results for all proposed wavelength assignment methods over varying network parameters. Finally, Chapter 6 contains conclusions and directions for future research.

Chapter 2

Optical Burst Switching

Wavelength-division multiplexing (WDM) has emerged as a core transmission technology for the next-generation Internet protocol (IP) backbone network with its ability to support a number of high-speed (gigabit) channels in a single fiber, which provides enormous bandwidth at the physical layer. Therefore, there is a need to develop framework and protocols at higher layers to efficiently use the raw bandwidth available at the optical (WDM) layer. Presently, WDM is mainly deployed in the backbone of major long distance carriers as point-to-point links with a synchronous optical network (SONET) as a standard interface to higher layers in the protocol stack. This necessitates optical-to-electrical (O/E) and electrical-to-optical (E/O) conversions at every node, and hence fails to take advantage of the wavelength routing capability provided by WDM technology. Also, electronic multiplexing layers —IP, asynchronous transfer mode (ATM), and frame relay — introduce further bandwidth inefficiencies. Although there has been a dramatic increase in the speed of electronic devices in the recent past, it is not likely to catch up with the transmission speed available at the optical layer. This calls for a novel effort to minimize or eliminate electronic processing to fully benefit from the bandwidth potential provided by WDM technology. One possibility is to have an all-optical backbone using optical packet switching technology. However, this new technology needs to overcome a number of technological challenges. Besides, optical burst switching (OBS) is a viable transmission technology for the next-generation optical backbone and may provide a framework to deploy IP over WDM.

2.1 Switching Technologies for WDM

2.1.1 Circuit Switching

Circuit switching has three distinct phases, circuit set-up, data transmission and circuit teardown. One of the main features of circuit switching is its two way reservation process in phase 1, where a source sends a request for setting up a circuit and then receives an acknowledgment back from the corresponding destination. In WDM networks, circuit switching takes the form of wavelength routing, where an all-optical wavelength path, consisting of a dedicated wavelength channel on every link, is established between two remote communicating nodes. The bandwidth, therefore, would not be efficiently utilized if the subsequent data transmission does not have a long duration relative to the set-up time of the lightpath. In addition, given that number of wavelengths available is limited, not every node can have a dedicated lightpath to every other node, and accordingly, some data may take a longer route and/or go through O/E and E/O conversions. Furthermore, the extremely high degree of transparency of the lightpaths limits the network management capabilities (e.g. monitoring and fast fault recovery)

2.1.2 Packet Switching

An alternative to optical circuit switching is optical or photonic packet/cell switching in which a packet is sent along with its header. While an intermediate node is processing the header, either all optically or electronically (after an O/E conversion), the packet is buffered at the node in the optical domain. However, high-speed optical logic, optical memory technology, and synchronization requirements are major problems with this approach. In particular, popular routing methods used in systems with electronic buffers, like worm-hole routing and virtual cut-through routing, cannot be deployed effectively in optical networks due to the limited buffering time of optical signals .

2.1.3 Optical Burst Switching (OBS)

Optical burst switching is a new switching paradigm for optical networks proposed in [4]. The main motivation for considering OBS is that some traffic in broadband multimedia services is inherently bursty. More importantly, some studies have concluded that contrary to common assumption based on Poisson traffic, multiplexing a large number of self-similar traffic streams results in bursty traffic [8, 9].

In order to provide high-bandwidth transport services at the optical layer for bursty traffic in a flexible, efficient as well as feasible way, what is needed then is a new switching paradigm that can leverage the attractive properties of optical communications, and at the same time, take into account its limitations. OBS is intended to accomplish exactly that.

In OBS, a control packet is sent first to set up a connection, followed by a burst of data without waiting for an acknowledgement for the connection establishment. By definition, a burst of data means fixed or variable sized collection of packets. The control packet sets up the connection by reserving an appropriate amount of bandwidth and configuring the switches along a path. In other words, OBS uses one-way reservation protocols, and this distinguishes it from circuit switching which uses two-way reservation protocols.

OBS also differs from optical or photonic packet/cell switching mainly in that the former can switch a burst whose length can range from one to several packets to a (short) session using one control packet, thus resulting in a lower control overhead per data unit. In addition, OBS uses out-of-band signaling, but more importantly, the control packet and the data burst are more loosely coupled in time than in packet/cell switching. In fact, they may be separated at the source as well as subsequent intermediate nodes by an offset time as in the Just-Enough-Time (JET) protocol to be described later. By choosing the offset time at the source to be larger than the total processing time of the control packet along the path [10, 11], one can eliminate the need for a data burst to be buffered at any subsequent intermediate node just to wait for the control packet to get processed. Alternatively, an OBS protocol may choose not to use any offset time at the source, but instead, require that the data burst go through, at each intermediate node, a fixed delay that is no shorter than the maximal time needed to process a control packet at the intermediate node. Such OBS protocols will be collectively referred to as tell-n-go (TAG) based since their basic concepts are the same as that of TAG itself.

In the WDM layer, a dedicated control wavelength is used to provide the “static/physical” links between IP entities. Specifically, it is used to support packet switching between physically adjacent IP entities, which maintain topology, and routing tables. To send data, a control packet is routed from a source to its destination based on the IP addresses it carries (or just a label if MPLS is supported) to set up a connection by configuring all optical switches along the path. Then, a burst (e.g. one or more data IP packets, or an entire message) is delivered without going through intermediate IP entities, thus reducing its latency as well as the processing load at the IP layer. Note that, due to the limited “opaqueness” of the control packet, OBS can achieve a high degree of adaptivity to congestions or faults (e.g., by using deflection routing), and support priority-based routing as in optical cell/packet switching. In OBS, the wavelength on a link used by the burst will be released as soon as the burst passes through the link, either automatically according to the reservation made (as in JET) or by an explicit release packet. In this way, bursts from different sources to different destinations can effectively utilize the bandwidth of the same wavelength on a link in a time-shared, statistical multiplexed fashion. Note that, in case the control packet fails to reserve the bandwidth at an intermediate node, the burst which is considered blocked at this time may have to be dropped. OBS can support either reliable or unreliable burst transmissions at the optical layer. In the former, a negative acknowledgment is sent back to the source node, which retransmits the control packet and the burst later. Such a retransmission may be necessary when OBS is to support some application protocols directly, but not when using an upper layer protocol such as Transmission Control Protocol (TCP), which eventually retransmits lost data. For the unreliable case, control packet thus its burst is simply dropped, and no retransmission occurs.

In either case, a dropped burst wastes the bandwidth on the partially established path. However, since such bandwidth has been reserved exclusively for the burst, it would be wasted even if one does not send out the burst as in two-way reservation. Similar arguments apply to optical or photonic packet switching as well. In order to eliminate the possibility of such bandwidth waste, a blocked burst or an optical packet will have to be stored in an electronic buffer after going through O/E conversions, and later (after going through E/O conversions), relayed to its destination. Fiber delay lines (FDLs) providing limited delays at intermediate nodes, which are not mandatory in OBS when using the JET protocol, would help reduce the bandwidth waste and improve performance in OBS. Note that, when using TAG-based OBS protocols or optical/photonic packet switching, FDLs or optical buffers are required to delay each optical burst when the control packet or the packet header is processed, but do not help improve performance.

Summarizing the above discussions as illustrated in Table 2.1, switching optical bursts achieves, to certain extent, a balance between switching coarse-grained optical circuits and switching fine-grained optical packets/cells, and combines the best of both paradigms.

Table 2.1: A comparison between three optical switching paradigms

Switching Paradigm	*Bandwidth Utilization*	*Latency (set-up)*	*Optical Buffer*	*Proc./Sync. Overhead*	*Adaptivity (traffic& fault)*
Circuit	low	high	not required	low	low
Packet/Cell	high	low	required	high	high
Burst	high	low	not required	low	high

2.2 The Just-Enough-Time (JET) Protocol and Its Variations

The Just-Enough-Time (JET) protocol [4] for OBS has two unique features, namely, the use of delayed reservation (DR) and the capability of integrating DR with the use of FDL-based buffered burst multiplexers (BBMs), which are to be described in this section. These features make JET and JET-based variations especially suitable for OBS when compared to TAG-based OBS protocols and other one-way reservation based OBS protocols that lack either or both features.

Figure 2.1 illustrates the basic concept of JET. As shown, a source node having a burst to transmit first sends a control packet on a signaling channel, which is a dedicated wavelength towards the destination node. The control packet is processed at each subsequent node in order to establish an all optical data path for the following burst. More specifically, based on the information carried in the control packet, each node chooses an appropriate wavelength on the outgoing link, reserves the bandwidth on it, and sets up the optical switch. Meanwhile, the burst waits at the source in the electronic domain. After an offset time, T, whose value is to be determined next, the burst is sent in optical signals on the chosen wavelength.

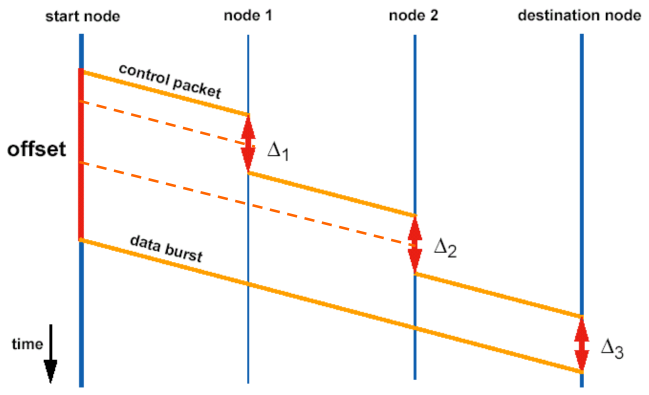


Figure 2.1: OBS using the JET protocol

2.2.1 The Use of Offset Time

For simplicity, it may be assumed that the time to process the control packet, reserve appropriate bandwidth and set up the switch is Δ time units at each node, and the receiving and transmission time of the control packet is ignorable. In a TAG-based OBS protocol or optical/photonic packet switching, a burst is sent by the source along with the control packet without any offset time (i.e., T = 0 in Figure 2.1). In addition, at each subsequent intermediate node, the burst waits for the control packet to be processed, and the two are sent to the next node without any offset time either. In this way, both the control packet and the burst will be delayed for Δ units, which will be referred to as the per node control latency. Accordingly, the minimum latency of the burst including the total propagation time, denoted by P, but excluding its transmission time, is P + Δ * H, where H is the number of hops along the path (e.g., in Figure 2.1, H = 3).

In JET, the offset time T can be chosen to be Δ* H, as shown in Figure 2.1, to ensure that there is enough headroom for each node to complete the processing of the control packet before the burst arrives. In this way, the burst will not encounter a longer latency than using TAG-based OBS protocols.

It is important to note that the burst can be sent without having to wait for an acknowledgement from its destination. At 10 Gb/s, a burst of 500 Kbytes (or 4000 average sized IP packets) can be transmitted in about 0.4 ms. However, an acknowledgement would take 1.7 ms just to propagate over a distance of merely 500km. This explains why one-way reservation protocols are generally better than their two-way counterparts for bursty traffic over a relatively long distance. Once a burst is sent, it passes through the intermediate nodes without going through any buffer, so the minimal latency it encounters would be the same as if the burst is sent along with the control packet as in optical packet switching. Of course, if a burst is extremely small, one may just as well send the data along with the control information using packet switching.

2.2.2 Delayed Reservation (DR) for Efficient Bandwidth Utilization

Figure 2.2 illustrates why delayed reservation (DR) of bandwidth is useful in achieving efficient bandwidth utilization. Using a TAG-based OBS protocol, the bandwidth on the outgoing link is reserved from t_1', the time node X finishes the processing of the first control packet. In JET, one may also reserve the bandwidth in the same way. However, it is natural to delay the bandwidth reservation till t_1, the time the first burst arrives. Here, $t_1 > t_1$' and their difference is the value of the offset time between the burst and its corresponding control packet at node X.

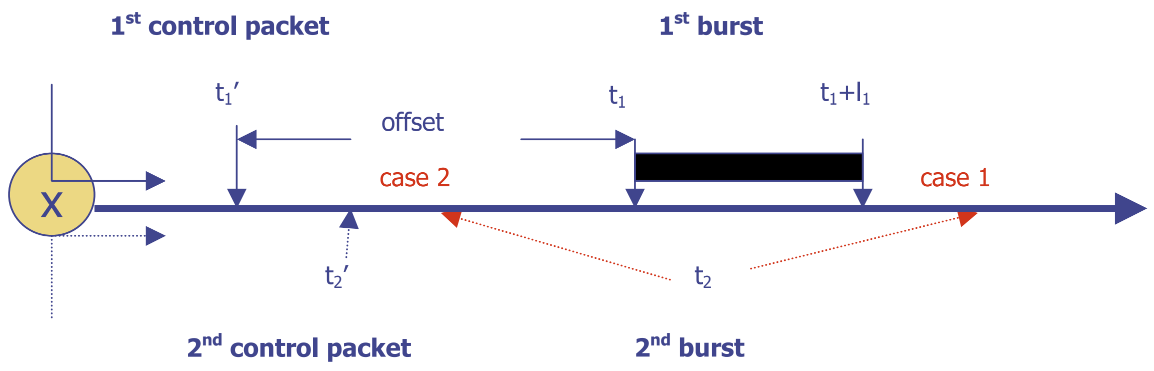


Figure 2.2: Delayed reservation (DR) and its usefulness without buffer

Note that, a way to determine the arrival time of a burst, e.g. t_1, when the processing time of a control packet may vary from one node to another, is to let the control packet carry the value of the offset time to be used at the next node. This value can be updated based on the processing time counted by the control packet at the current node. In the above example, immediately after the control packet succeeds in reserving the bandwidth, its transmission is scheduled, say, at t_1”. The value of the offset time to be used at the next node is then obtained by subtracting t_1” - t_1’ from the current value.

In addition to taking into account the arriving time of the burst, t_1, what is more important is that in JET, the bandwidth may be reserved until $t_1+ l_1$, where l_1 is the burst duration, instead of until infinity. This will increase the bandwidth utilization and reduce the probability of having to drop a burst. For example, in the case shown in Figure 2.2, namely $t_2 > t_1+ l_1$ and $t_2 < t_1$, respectively, the second burst will be dropped at node X if X has no buffer for the burst when using TAG. However, when using JET, the second burst will not be dropped in case 1, nor in case 2, provided that its length is shorter than $t_1 – t_2$.

Note that, DR goes hand in hand with the use of offset time. In addition, although burst length may vary, we may assume that the length of a burst is known before the corresponding control packet is sent. This assumption is natural in some applications such as file transfer or WWW (world wide web) downloading. However, if the burst length is unknown, one may delay the control packet until either the entire burst arrives (from an upper layer), or a certain length is reached. To take advantage of the use of an offset time in JET, thereby reducing the pre-transmission latency, an alternative is to send out the control packet as soon as possible by using an estimated value of the burst length. If it is an over-estimation, another control (release) packet may be sent to release the extra bandwidth reserved. If it is an under-estimation, then the remaining data will be sent as one or more additional bursts. JET may also support an entire session by reserving the bandwidth to infinity, and use an explicit release packet when the circuit is no longer needed (i.e. the session ends).

2.2.3 FDL’s and Pool architecture

As mentioned earlier, JET does not necessitate the use of buffer. Nevertheless, the dropping probability can be further reduced, and both bandwidth utilization and performance can be further improved, if a burst can be buffered (or delayed) at an intermediate node [4].

FDLs are equivalent to RAMs in optical networks. Since in optical domain, there is no known way of storing data, a few kilometers of extra fiber can be used to provide a maximum of few tens of microseconds delay. Figure 2.3 shows the structure for two basic types of FDLs. In the figure, if each circle denotes a time unit of delay, either structure can provide discrete delays from minimum 0 to maximum $B = 2^{n+1}-1$ time units. The difference between shared and dedicated BBM is that the latter is more complex and costly but more powerful.

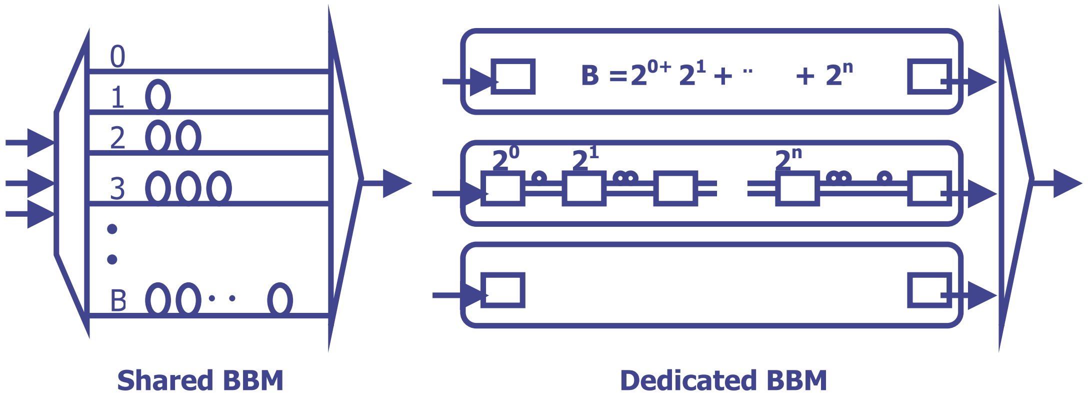


Figure 2.3: An example of (a). a shared BBM and (b). a dedicated BBM

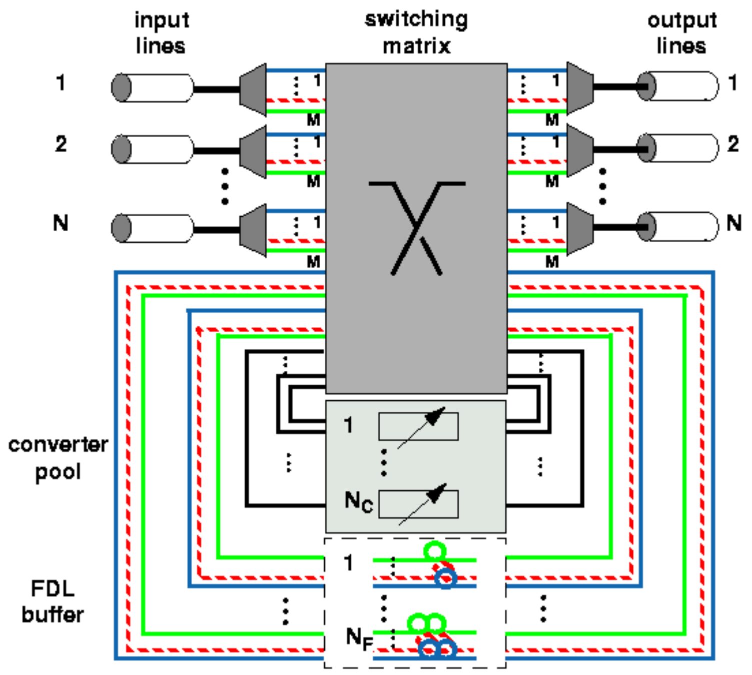


Figure 2.4: Model of an OBS node with a shared converter and a feedback FDL buffer

Both FDL buffers and wavelength converters can be shared among the switching nodes. Figure 2.4 illustrates model of an OBS node with a shared converter pool and a feedback FDL buffer. There are N links in the optical switch and M wavelengths on each fiber. Moreover, there exists a converter pool consisting of N_C converters, and an FDL pool consisting of N_F buffers.

Both FDL-based buffers and wavelength converters are costly and scarce devices. Many studies show that the performance of a network with full wavelength conversion, i.e. $N_C=M*N$ for all switches, can be achieved with a network with a fewer number of wavelength converters. A similar discussion holds also for FDL-based buffers [12]. The value of N_C and N_F can be selected appropriately such that blocking probability vs. cost trade-off is considered.

2.2.4 Adaptive Routing and Priority Schemes

The dropping probability of a burst may also be improved by implementing adaptive routing and/or assigning it with a higher priority. As mentioned earlier, a TAG-based OBS protocol does not use any offset time. Instead, a data burst goes through a fixed delay (FDL) at each intermediate node to account for the processing delay counted by the corresponding control packet. This facilitates the use of a different path to a given destination each time a source sends a new burst or retransmits a dropped burst, as well as deflection routing at intermediate nodes when a burst is blocked.

A JET-based OBS protocol can also support multi-path routing from a given source to a given destination as long as the number of hops along each path is known. To support deflection routing at an intermediate node when there is no bandwidth to reserve on the primary outgoing link, the control packet chooses an alternate outgoing link, and sets the switch accordingly so that the data burst will also follow the alternate path. If a minimal offset time based on the primary path was used, and the alternate path is longer in terms of number of hops, then the data burst needs to be delayed further in order to make up for the increase in the total processing delay counted by the control packet along the alternate path. This can be accomplished by letting the data burst go through some FDLs at one or more nodes before the offset time goes to zero, even if no blocking occurs at these nodes. We note that a JET-based protocol can support limited adaptivity even without using FDLs. Specifically, one can use an extra offset time at the source to account for a possible increase in the total processing delay of the control packet due to deflection routing. In addition to being useful for deflection routing, having an additional offset time can increase the priority of a burst. This is because the corresponding control packet will likely to succeed in reserving the bandwidth into the future, given that very few other control packets arriving earlier (or around the same time) might have reserved (or want to reserve) the bandwidth that much in advance. This property of an additional offset time can be utilized to improve fairness by assigning a higher priority to bursts which must travel for a longer distance from their sources to destinations. This variation of JET, which implements such a priority scheme, is called as JET-FA (for fairness) [4].

In summary, among various optical switching paradigms, OBS shows advantages in terms of switching efficiency for bursty IP traffic and optical hardware feasibility. However, the high blocking probability is one of the major problems in optical burst switching due to its inherent one-way reservation paradigm. Data bursts are sent out without waiting for the acknowledgements from receivers to setup the path (no end-to-end resource reservation), therefore, the burst could be blocked in an intermediate node due to the resource contention, in which case, the burst has to be dropped. Since each burst must be assigned a specific path and a wavelength on every link of the assigned path, the resource contention occurs when two or more bursts on the same wavelength are routed to the same link at the same time.

Accordingly, in case of a reservation conflict, i.e., the wavelength on this output line is already reserved, one or a combination of the following three major options for contention resolution can be applied.

Wavelength domain: By means of wavelength conversion, a burst can be sent on a different wavelength channel of the designated output line.

Time domain: By applying an FDL buffer, a burst can be delayed until the contention situation is resolved. In contrast to buffers in the electronic domain, FDL’s only provide a fixed delay and data leave the FDL in the same order in which they entered.

Space domain: In deflection routing, a burst is sent to different output line of the node and consequently on a different route towards its destination node.

However, all optical converter technologies are still not applicable for full-range wavelength conversion, which is indispensable for the burst contention resolution intent in wavelength domain. Moreover, available converter technologies are expensive and scarce. Meanwhile, the lack of optical memory makes the optical buffering to be an impractical approach. Although fiber delay lines can be used to temporarily store data in optical domain for a few tens of μs, these are not sufficient for storing longer optical bursts. Deflection routing is another solution for reducing the blocking probability. Even though the effective utilization of idle links is an advantage, the increase of the number of links used per burst as a result of deflections is a disadvantage. Burst reordering at the destination and the fairness problem are also the potential disadvantage of deflection schemes.

On the other hand, reservation conflict problem can also be attacked in the wavelength domain, by using smart wavelength assignment algorithms. When applied to OBS networks, well-known heuristic solutions for wavelength assignment problem do not perform as well as they do in networks using other switching paradigms. Interestingly, because of distributed wavelength assignment behavior of OBS, random wavelength assignment results with lower average blocking probabilities than other conventional heuristics. However, a little bit intelligence introduced to the wavelength assignment algorithm should improve blocking probability beyond the random heuristic. In this thesis work, wavelength assignment algorithm in OBS networks is considered under a dynamic traffic model. The aim is to minimize average blocking probability using neuro-dynamic programming (reinforcement learning).

Next, reinforcement learning and its related implementations will be presented to give the necessary background and motivation behind its implementation to wavelength assignment problem in OBS networks.

Chapter 3

Reinforcement Learning

Reinforcement learning is a general framework for describing learning problems in which an agent learns strategies for interacting with its environment. As seen in Figure 3.1, the agent perceives something about the state of its environment and chooses what it thinks is an appropriate action. The world's state changes (not necessarily deterministically) and the agent receives a scalar “reward” or “cost” indicating the utility of the new state for the agent. The agent's goal is to find, based on its experience with the environment, a strategy or an optimal policy for choosing actions that will yield as much reward/min cost as possible.

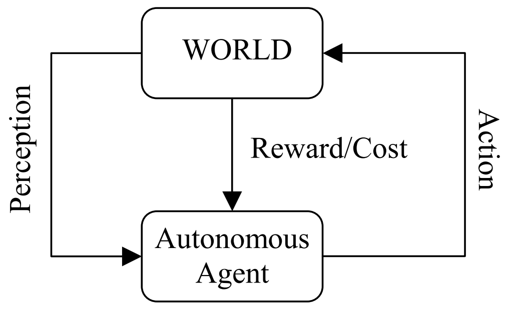


Figure 3.1: Reinforcement learning studies sequential decision problems faced by autonomous agents. Here, the agent seeks to learn an optimal policy that maximizes/minimizes the rewards/costs received over time

There are two major designs for a reinforcement learning agent. In the model-based approach, the agent learns a model of the dynamics of the world and of its rewards. Given the model, it tries to solve for the optimal control policy. In the model free approach, the agent tries to learn the optimal control policy directly, without first constructing a world model. In either approach, the agent seeks to learn a policy that maximizes/minimizes some cumulative measure of reinforcement received from the environment.

3.1 Models of Optimal Behavior

There are three models to specify how the agent should take the future into account in the decisions it makes about how to behave now.

The *finite-horizon model* is the easiest to think about: at a given moment in time, the agent should optimize its expected reward for the next *h* steps, which is given by

$$E(\sum_{t=0}^{h} r_t)$$

the agent should not worry about what will happen after that. In this and subsequent expressions, r_t represents the scalar reward received *t* steps into the future. This model can be used in two ways. In the first, the agent will have a non-stationary policy; that is, one that changes over time. On its first step, it will take what is termed an *h-step optimal action*. This is defined to be the best action available given that it has *h* steps remaining in which to act and gain reinforcement. On the next step, it will take an (*h*–1)-step optimal action, and so on, until it finally takes a 1-step optimal action and terminates. In the second, the agent does *receding-horizon control*, in which it always takes the *h*-step optimal action. The agent always acts according to the same policy, but the value of *h* limits how far ahead it looks in choosing its actions. The finite-horizon model is not always appropriate. In many cases, the precise length of the agent's life in advance may not be known.

The *infinite-horizon discounted model* takes the long-run reward of the agent into account, but rewards that are received in the future are geometrically discounted according to discount factor γ, (where $0 \leq \gamma < 1$):

$$E(\sum_{t=0}^{\infty}\gamma^t r_t) \quad ;$$

γ can be seen as an interest rate, a probability of living another step, or as a mathematical trick to bound the infinite sum. The model is conceptually similar to receding-horizon control, but the discounted model is more mathematically tractable than the finite-horizon model. This is a dominant reason for the wide attention this model has received.

Another optimality criterion is the *average-reward model*, in which the agent is supposed to take actions that optimize its long-run average reward:

$$\lim_{h\to\infty} E(\frac{1}{h}\sum_{t=0}^{h} r_t) \quad .$$

Such a policy is referred to as a gain optimal policy; it can be seen as the limiting case of the infinite-horizon discounted model as the discount factor approaches one [13]. One problem with this criterion is that there is no way to distinguish between two policies, one of which gains a large amount of reward in the initial phases and the other of which does not. Reward gained on any initial prefix of the agent's life is overshadowed by the long-run average performance. It is possible to generalize this model so that it takes into account both the long run average and the amount of initial reward than can be gained. In the generalized, *bias optimal* model, a policy is preferred if it maximizes the long-run average and ties are broken by the initial extra reward.

3.2 Markov Decision Processes

Problems with delayed reinforcement are well modeled as Markov decision processes (MDPs). An MDP consists of

- a set of states *S*,
- a set of actions *A*,
- a reward function $R : S \text{ x } A \rightarrow \Re$, and
- a state transition function $T : S \text{ x } A \rightarrow \Pi(S)$, where a member of $\Pi(S)$ is a probability distribution over the set *S* (i.e. it maps states to probabilities). We write $T(s, a, s')$ for the probability of making a transition from state *s* to state *s'* using action *a*.

The state transition function probabilistically specifies the next state of the environment as a function of its current state and the agent's action. The reward function specifies expected instantaneous reward as a function of the current state and action. The model is Markov if the state transitions are independent of any previous environment states or agent actions.

3.4 Finding a Policy Given a Model

Before looking at the algorithms for learning to behave in MDP environments, techniques for determining the optimal policy given a correct model will be explored. These dynamic programming techniques will serve as the foundation and inspiration for the learning algorithms to follow. Finding optimal policies for the infinite-horizon discounted model will be presented here, but most of these algorithms have analogs for the finite-horizon and average-case models as well. For the infinite-horizon discounted model, there exists an optimal deterministic stationary policy [14].

The optimal value of a state is defined as the expected infinite discounted sum of reward that the agent will gain if it starts in that state and executes the optimal policy. Using μ as a complete decision policy, it is written

$$V^*(s) = \max_{\mu} E(\sum_{t=0}^{\infty} \gamma^t r_t) \quad .$$

This optimal value function is unique and can be defined as the solution to the simultaneous equations

$$V^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')\right), \forall s \in S \quad ,$$

which assert that the value of a state s is the expected instantaneous reward plus the expected discounted value of the next state, using the best available action. Given the optimal value function, we can specify the optimal policy as

$$\mu^*(s) = \arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')\right) \quad .$$

3.4.1 Value Iteration

One way to find an optimal policy is to find the optimal value function. It can be determined by a simple iterative algorithm called *value iteration* that can be shown to converge to the correct V^* values [14, 15].

```
initialize $V(s)$ arbitrarily
loop until policy good enough
  loop for $s \in S$
    loop for $a \in A$
        $Q(s,a) := R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')$
    $V(s) := \max_a Q(s,a)$
  end loop
end loop
```

3.4.2 Policy Iteration

The policy iteration algorithm manipulates the policy directly, rather than finding it indirectly via the optimal value function. It operates as follows:

```
choose an arbitrary policy $\mu'$
loop
  $\mu := \mu'$
  compute the value function of policy $\mu$:
    solve the linear equations
        $V_\mu(s) = R(s,\mu(s)) + \gamma \sum_{s' \in S} T(s,\mu(s),s') V_\mu(s')$
  improve the policy at each state:
      $\mu'(s) = \arg\max_a (R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V_\mu(s'))$
end loop
```

In practice, value iteration is much faster per iteration, but policy iteration takes fewer iterations.

3.5 Learning an Optimal Policy: Model-free Methods

In the previous subsection, methods for obtaining an optimal policy for an MDP assuming that there was a model, was presented. The model consists of knowledge of the state transition probability function *T*(*s*, *a*, *s*') and the reinforcement function *R*(*s*, *a*). Reinforcement learning is primarily concerned with how to obtain the optimal policy when such a model is not known in advance. The agent must interact with its environment directly to obtain information, which, by means of an appropriate algorithm, can be processed to produce an optimal policy.

At this point, there are two ways to proceed.

- Model-free: Learn a controller without learning a model.
- Model-based: Learn a model, and use it to derive a controller.

It is still debatable in the reinforcement-learning community whether model-free or model-based approach is better. A number of algorithms have been proposed on both sides. Since model-free learning is highly related with this thesis study, only it will be examined.

The biggest problem facing a reinforcement learning agent is temporal credit assignment. How can be known whether the action just taken is a good one, when it might have far-reaching effects? One strategy is to wait until the "end" and reward the actions taken if the result was good and punish them if the result was bad. In ongoing tasks, it is difficult to know what the "end" is, and this might require a great deal of memory. Instead, insights from value iteration are used to adjust the estimated value of a state based on the immediate reward and the estimated value of the next state. This class of algorithms is known as temporal difference methods [16]. Two different temporal-difference learning strategies for the discounted infinite-horizon model will be presented next.

3.5.1 Adaptive Heuristic Critic and *TD* (λ)

The adaptive heuristic critic algorithm is an adaptive version of policy iteration [17] in which the value-function computation is no longer implemented by solving a set of linear equations, but is instead computed by an algorithm called *TD*(0). A block diagram for this approach is given in Figure 3.2. It consists of two components: a critic (labeled AHC), and a reinforcement-learning component (labeled RL). The reinforcement-learning component can be an instance of any of the *k*-armed bandit algorithms, modified to deal with multiple states and non-stationary rewards. But instead of acting to maximize instantaneous reward, it will be acting to maximize the heuristic value, *v*, that is computed by the critic. The critic uses the real external reinforcement signal to learn to map states to their expected discounted values given that the policy being executed is the one currently instantiated in the RL component.

The policy μ implemented by RL is fixed and the critic learns the value function V_μ for that policy. Here the critic is fixed and let RL component learns a new policy μ' that maximizes the new value function, and so on. In most implementations, however, both components operate simultaneously. Only the alternating implementation can be guaranteed to converge to the optimal policy, under appropriate conditions. Williams and Baird explored the convergence properties of a class of AHC-related algorithms they call "incremental variants of policy iteration" [18].

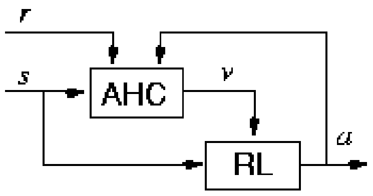


Figure 3.2: Architecture for the adaptive heuristic critic.

$\langle s, a, r, s' \rangle$ is defined to be an *experience tuple* summarizing a single transition in the environment. Here, s is the agent's state before the transition, a is its choice of action, r the instantaneous reward it receives, and s' its resulting state. The value of a policy is learned using Sutton's *TD*(0) algorithm [16] which uses the update rule

$$V(s) := V(s) + \alpha(r + \gamma V(s') - V(s)) \quad .$$

Whenever a state s is visited, its estimated value is updated to be closer to $r + \gamma V(s')$, since r is the instantaneous reward received and $V(s')$ is the estimated value of the actually occurring next state. This is analogous to the sample-backup rule [19] from value iteration - the only difference is that the sample is drawn from the real world rather than by simulating a known model. The key idea is that $r + \gamma V(s')$ is a sample of the value of $V(s)$, and it is more likely to be correct because it incorporates the real r. If the learning rate α is adjusted properly (it must be slowly decreased) and the policy is held fixed, *TD*(0) is guaranteed to converge to the optimal value function.

The *TD*(0) rule as presented above is really an instance of a more general class of algorithms called *TD*(λ), with λ = 0. *TD*(0) looks only one step ahead when adjusting value estimates; although it will eventually arrive at the correct answer, it can take quite a while to do so. The general *TD*(λ) rule is similar to the *TD*(0) rule given above,

$$V(u) := V(u) + \alpha\,(r + \gamma\, V(s') - V(s))e(u) \quad ,$$

but it is applied to *every state* according to its eligibility $e(u)$, rather than just to the immediately previous state, s. one version of the eligibility trace is defined to be

$$e(s) = \sum_{k=1}^{t} (\lambda\gamma)^{t-k}\,\delta_{s,s_k}, \quad where\ \delta_{s,s_k} = \begin{cases} 1 & if\ s = s_k \\ 0 & otherwise \end{cases} \quad .$$

The eligibility of a state s is the degree to which it has been visited in the recent past; when a reinforcement is received, it is used to update all the states that have been recently visited, according to their eligibility. When $\lambda = 0$, this is equivalent to *TD*(0). When $\lambda = 1$, it is roughly equivalent to updating all the states according to the number of times they were visited by the end of a run. Note that we can update the eligibility online as follows:

$$e(s) := \begin{cases} \gamma\lambda e(s) + 1 & if\ s = current\ state \\ \gamma\lambda e(s) & otherwise \end{cases} \quad .$$

It is computationally more expensive to execute the general *TD*(λ), though it often converges considerably faster for large λ [20, 21].

3.5.2 *Q*-learning

The work of the two components of AHC can be accomplished in a unified manner by Watkins' *Q*-learning algorithm [22, 23]. *Q*-learning is typically easier to implement. Let $Q^*(s, a)$ be the expected discounted reinforcement of taking action a in state s, then continuing by choosing actions optimally. Note that $V^*(s)$ is the value of s assuming the best action is taken initially, and so $V^*(s) = \max_a Q^*(s', a)$. $Q^*(s, a)$ can hence be written recursively as

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a') \quad .$$

Note also that, since $V^*(s) = \max_a Q^*(s',a)$, we have $\mu^*(s) = \arg\max_a Q^*(s,a)$ as an optimal policy.

Because the *Q* function makes the action explicit, the *Q* values can be estimated online using a method essentially the same as *TD*(0), but also can be used to define the policy, because an action can be chosen just by taking the one with the maximum *Q* value for the current state.

The *Q*-learning rule is

$$Q(s,a) := Q(s,a) + \alpha(r + \gamma \max_{a'} Q^*(s',a') - Q(s,a)) \quad ,$$

where $\langle s,a,r,s' \rangle$ is an experience tuple as described earlier. If each action is executed in each state an infinite number of times on an infinite run and *α* is decayed appropriately, the *Q* values will converge with probability 1 to Q^* [22, 24, 25]. *Q*-learning can also be extended to update states that occurred more than one step previously, as in *TD*(*λ*) [26].

When the *Q* values nearly converge to their optimal values, it is appropriate for the agent to act greedily, taking, in each situation, the action with the highest *Q* value. During learning, however, there is a difficult exploitation versus exploration trade-off to be made. There are no good, formally justified approaches to this problem in the general case.

AHC architectures seem to be more difficult to work with than *Q*-learning on a practical level. It can be hard to get the relative learning rates right in AHC so that the two components converge together. In addition, *Q*-learning is exploration insensitive: that is, that the *Q* values will converge to the optimal values, independent of how the agent behaves while the data is being collected (as long as all state-action pairs are tried often enough). This means that, although the exploration-exploitation issue must be addressed in *Q*-learning, the details of the exploration strategy will not affect the convergence of the learning algorithm. For these reasons, *Q*-learning is the most popular and seems to be the most effective model-free algorithm for learning from delayed reinforcement. It does not, however, address any of the issues involved in generalizing over large state and/or action spaces. In addition, it may converge quite slowly to a good policy.

3.5.3 Model-free Learning with Average Reward

As described, *Q*-learning can be applied to discounted infinite-horizon MDPs. It can also be applied to undiscounted problems as long as the optimal policy is guaranteed to reach a reward-free absorbing state and the state is periodically reset.

Schwartz [27] examined the problem of adapting *Q*-learning to an average-reward framework. Although his *R*-learning algorithm seems to exhibit convergence problems for some MDPs, several researchers have found the average-reward criterion closer to the true problem they wish to solve than a discounted criterion and therefore prefer *R*-learning to *Q*-learning [28].

With that in mind, researchers have studied the problem of learning optimal average-reward policies. Mahadevan [29] surveyed model-based average-reward algorithms from a reinforcement learning perspective and found several difficulties with existing algorithms. In particular, he showed that existing reinforcement learning algorithms for average reward (and some dynamic programming algorithms) do not always produce bias-optimal policies. Jaakkola, Jordan and Singh [30] described an average-reward learning algorithm with guaranteed convergence properties. It uses a Monte-Carlo component to estimate the expected future reward for each state as the agent moves through the environment. In addition, Bertsekas presents a *Q*-learning-like algorithm for average-case reward in his textbook [13]. Although this recent work provides a much-needed theoretical foundation to this area of reinforcement learning, many important problems remain unsolved.

In this thesis study, we deal with an average-reward model free approach with function approximation since other neuro-dynamic programming methods are computationally infeasible for such large scale / network wide problems, i.e. we are interested in the problems with a large number of states. Our approach to the wavelength assignment problem in OBS networks using reinforcement learning will be briefly explained in Chapter 4.

In [31], the problem of call admission control and routing in an integrated services network that handles several classes of calls of different value and with different resource requirements is analyzed. The problem of maximizing the average value of admitted calls per unit time (or of revenue maximization) is naturally formulated as a dynamic programming problem, but this is too complex to allow for an exact solution. Methods of neuro-dynamic programming (reinforcement learning) are based on the average reward *TD*(0) method of [32], together with a decomposition approach, to construct dynamic (state-dependent) call admission control and routing policies. This decomposition has the advantage that it allows for decentralized decision-making and decentralized training, which reduces significantly the training time. Experimental results for several example problems, of different sizes are presented. The case study involving a 16-node network shows that neuro dynamic programming can lead to sophisticated control policies involving strategic call rejections, and which are difficult to obtain through heuristics. Compared with the "Open-Shortest-Path-First" (OSPF) heuristic, the neuro dynamic programming policy reduces the lost average reward by 50% (heavily loaded 4 node network), 52% (lightly loaded 16 node network), and (except for one out of twenty experiments) by 20-70% (16 node network under different loads). This illustrates that neuro dynamic programming has the potential to significantly improve performance over a broad range of network loads.

The neuro dynamic programming method is also implemented to routing and wavelength assignment problem in WDM optical networks in [33]. In this work, a novel connection based decomposition approach for WDM optical networks is proposed. Although it does not provide as high improvements over heuristics as in global *TD*(0), it is observed that this method greatly reduces the training time when compared to global *TD*(0) method where all the parameters are updated together. The proposed decomposition approach allows obtaining distributed algorithms for each connection which only require the features associated with the specified connection. Therefore, once the parameters are obtained, in the implementation phase these distributed algorithms have the same computational complexity as used heuritics. It is presented that smaller average blocking probabilities than those of all conventional heuristic algorithms might be obtained by neuro dynamic programming method.

Chapter 4

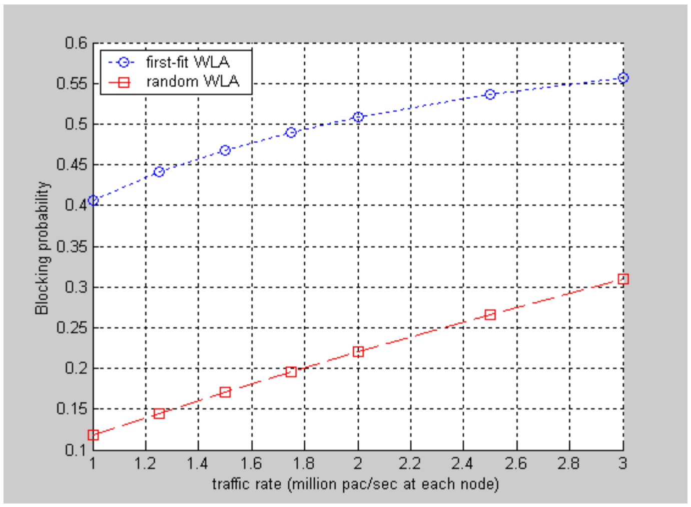
Proposed Wavelength Assignment Algorithms

One of the major problems in optical burst switching is the high blocking probability that results from the one-way reservation protocol used. Data bursts are sent out without waiting for the acknowledgements from receivers to setup the path (no end-to-end resource reservation). Therefore, the burst could be blocked in an intermediate node due to the resource contention, in which case, the burst has to be dropped. Since each burst must be assigned a specific path and a wavelength on every link of the assigned path, the resource contention occurs when two or more bursts on the same wavelength are routed to the same link at the same time.

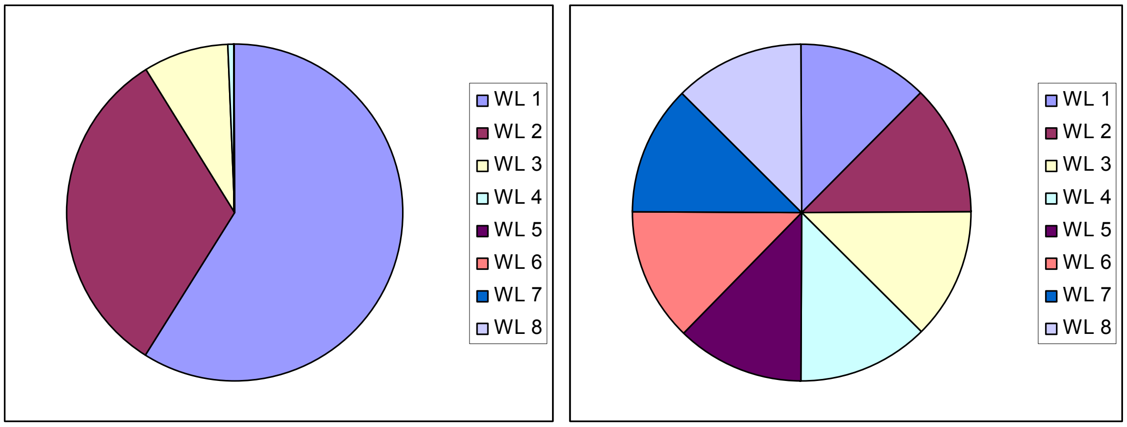
Reservation conflict problem can be solved in the wavelength domain, by using smart wavelength assignment algorithms and wavelength converters. However, all optical converter technologies are still not applicable for full-range wavelength conversion, which is indispensable for the burst contention resolution intent in wavelength domain. Moreover, available converter technologies are expensive and scarce. That is why an efficient and intelligent placement of converters with an intelligent wavelength assignment algorithm is studied throughout this thesis. Two heuristic wavelength assignment algorithms and a dynamic wavelength assignment algorithm based on neuro-dynamic programming are presented. The aim is to minimize average blocking probability and optimize the number of wavelength converters used for proposed wavelength assignment methods.

4.1 Heuristic Wavelength Assignment Algorithms

Several heuristic methods are defined for wavelength assignment in circuit-switched optical networks. Namely, random, first-fit, least-loaded, most-used etc. wavelength assignment algorithms result with acceptable blocking probability for most of the switching protocols. On the other hand, except random heuristic, most of these heuristics end up with unfair utilization of wavelengths in OBS networks. Interestingly, usually the worst performing of all, random wavelength selection algorithm, spreads the utilization of the wavelengths uniformly, and results with better performance than the others for the optical burst switching paradigm.



(a)



(b) (c)

Figure 4.1: a) Blocking probability vs. traffic rate for first-fit and random wavelength assignment algorithms over NSFNET topology with no wavelength converters.
b) Typical wavelength utilization of a link at first-fit wavelength assignment algorithm
c) Typical wavelength utilization of a link at random wavelength assignment algorithm

In Figure 4.1, performance comparison of two heuristic wavelength assignment algorithms for optical burst switching is shown. The simulations are done over single fiber NSFNET topology of Figure 4.2. Switches do not have the capability of wavelength conversion nor optical buffering. From Figure 4.1.a, it can be observed that, there is a significant difference in the blocking probability in comparison between random and first-fit wavelength assignment algorithms. Moreover, as it can be seen from Figure 4.1.b, although there exists 8 available wavelengths per fiber, first-fit assignment uses just only 4 of them resulting with an unfair utilization. Oppositely, as expected, random wavelength assignment causes approximately uniform distribution of wavelength usage. Since least-loaded and most-used heuristics are designed for multi-fiber networks, they are not included in this study.

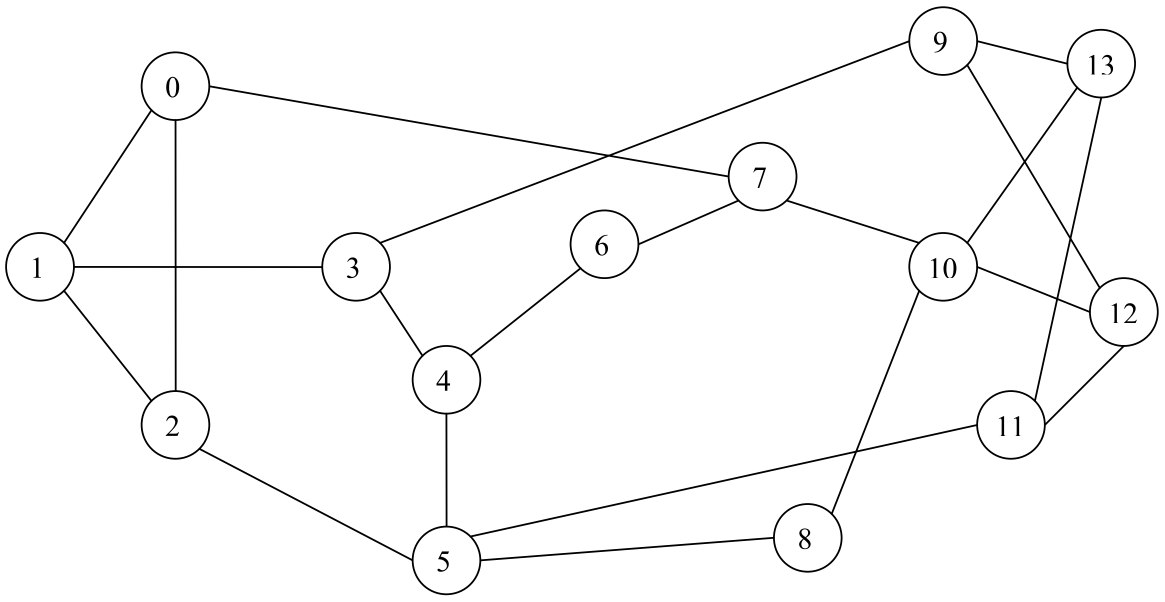


Figure 4.2: NSFNET topology

Although random wavelength assignment algorithm happens to be best performing among existing heuristic algorithms, better algorithms can be devised compared to making random decisions. The only advantage of this heuristic is that wavelength utilization among the network is distributed fairly. Since it does not use any information about the state of the network or ongoing packet arrivals, an algorithm that takes this knowledge into account during assignment procedure may improve the blocking probability.

As a direct result of one-way reservation protocols used in OBS, the data burst is sent over a wavelength without an agreement with the destination. It is probable that heading control packet may not be able to reserve the wavelength at an intermediate node along the route because of a reservation conflict. There is no guarantee that the wavelength selected at the source node will be available throughout its way from source to destination. Therefore, it may decrease the blocking probability if the assignment algorithm operates also in the way to increase the utilization along the link. Many reservation conflicts can be prevented by selecting an available wavelength such that this increases the probability of the upcoming control packet's finding an appropriate wavelength. If the assignment algorithm works in the same manner throughout the network in order to decrease the dropping probability of the succeeding control packet in the current link, burst blocking probability may be improved.

At every node, there is a reservation table which basically stores the data supplied by preceding control packets. So the information that indicates the utilization of all links adjacent to that node is available. The rationale behind the proposed heuristic wavelength assignment algorithms is to use this information about the past and future events during wavelength assignment procedure. This approach decreases the blocking probability of succeeding control packets, thus bursts. Proposed wavelength assignment algorithms designed according to this idea are named as most-fit-rand and most-fit-min wavelength assignment algorithms.

For example, assume two available wavelengths for an arriving burst of the duration (t_a,t_b). Let dark bold lines over each reservation time line show the reservations done by control packets that have arrived up to now in Figure 4.3. If wavelength *j* is selected, then another arriving burst for duration (t_c,t_d) will be dropped, but taking the duration into consideration during wavelength assignment procedure and selecting the one that leaves less free space in the reservation time line (fits most), there is a higher probability that upcoming control packets will find an available wavelength. In our example, if the former coming burst selects wavelength *i* for which the burst fits most, then the second can also be carried on wavelength *j*. Therefore, during wavelength assignment procedure using the reservation time line knowledge may help to improve blocking probability.

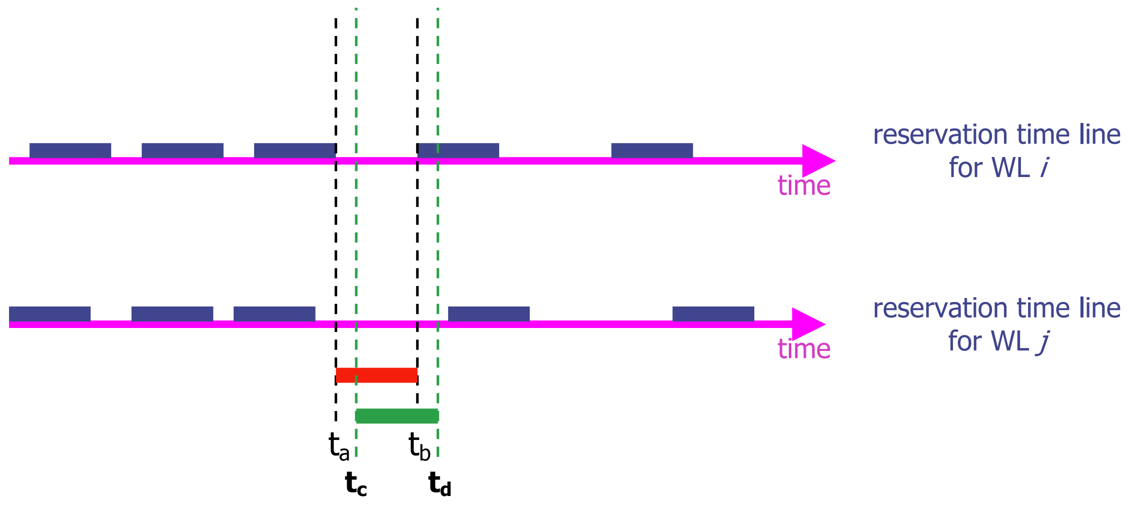

reservation time line for WL *i*
time
reservation time line for WL *j*
time
t_a
t_b
t_c
t_d

Figure 4.3: Reservation time line for WL *i* and WL *j*. Bold lines over each reservation time line show the reservations previously done Two bursts arrive at times t_a and t_c.

4.1.1 Most-Fit-Rand Heuristic Wavelength Assignment Algorithm

According to the most-fit-rand heuristic, all available wavelengths for the needed reservation period are assigned with one of low or high priorities. The reason for such an approach is to give priority to wavelengths where the reservation of the burst results with a denser timeline. Wavelength assignment is done randomly among equal priority wavelengths in order to provide fair wavelength utilization.

Let a burst of length l arrive at time t. In order to assign priorities to available wavelengths, the reservation time line is examined for periods (t-l,t) and (t+l,t+2l). Low priority is assigned to the available wavelengths with total amount of 2l free reservation time. All other available wavelengths with total amount of free reservation time less than 2l, are labeled with high priority.

The reason why the length of burst forms a criterion is that, if the duration between two reservations is smaller than the duration of a burst, it is sure that this duration is wasted because in order a control packet to make a reservation for the following burst, a continuous free duration of burst length is needed.

4.1.2 Most-Fit-Min Heuristic Wavelength Assignment Algorithm

The difference of most-fit-min from most-fit-rand wavelength assignment algorithm is that the wavelengths of high priority set are reassigned priorities according to their total amount of free reservation time for periods (*t*-*l*,*t*) and (*t*+*l*,*t*+2*l*). The larger the free time is, the lower the new priority assigned to that wavelength. As it is seen for most-fit-min the assignment among high priority set is done again for reassigned priorities, not randomly.

When the network is not densely loaded, it is observed that most of the available wavelengths are assigned to the low priority set. Therefore, during wavelength assignment, priority ties are most probably to occur also for most-fit-min heuristic. Ties of priorities are resolved randomly under all circumstances for both proposed heuristics.

4.2 Implementation of Reinforcement Learning to Wavelength Assignment Problem in OBS

4.2.1 Dynamic Programming Formulation

In this section, the problem of minimizing the average blocking rate in OBS networks is formulated as a continuous time, average cost dynamic programming problem. Let us consider a communication network where N = {1,2…N} represents the set of nodes, W={1,2…W} represents the set of wavelengths in an optical fiber, C={1,2,...C} represents the set of converters at each node, L={1,2…L} represents the set of unidirectional links.

At time t, the state of the network is represented as x_t and consists of a set of active bursts whose wavelength assignment problem has been solved. This finite set of all possible states will be referred to as the state space *S*. Although control packets arrive in time in continuous manner, it is sufficient to consider the state of the network at discrete instants when certain events take place. These events might be either a new control packet's arrival or the release of reservation of an existing connection on a link. Let us represent this finite event space as Ω.

If the system is in state x and an event e takes place, then a proper decision u should be made. Let $U(x, e)$ denote the set of all possible decisions. If e corresponds to a new control packet's arrival then $U(x, e)$ consists of reserving a possible wavelength and a wavelength converter, if necessary and available, for its following burst or simply rejecting the reservation. If e corresponds to a release of reservation of an existing connection on a link, then there is no need to make a decision since $U(x, e)$ consists of only terminating the specified reservation. Assume that the current state of the network is x, an event e occurs and a proper decision $u \in U(x, e)$ is made. Then the whole system moves to a next state which will be denoted as $\dot{x}$.

Let us denote the immediate cost that occurs when rejecting the incoming reservation request as d and the immediate cost of using a wavelength converter as c. The value of d linearly increases according to the number of nodes that the control packet passes in order to give a virtual priority to control packets that have surveyed on longer routes using the available resources (When such a burst is dropped, the resources that may be available for other bursts are wasted for the dropped one, so cost of a such drop should be higher). Then the resulting cost will be shown as $g(x, e, u)$ such that: if e corresponds to a new control packet's arrival and u corresponds to rejecting that reservation request then $g(x, e, u)=d$, otherwise if u corresponds to assigning a proper wavelength and wavelength converter pair then $g(x, e, u) =c$ and if u corresponds to assigning a proper wavelength without wavelength conversion then $g(x, e, u) =0$. If e corresponds to a reservation release, then $g(x, e, u) =0$.

Given a current state of the network x, the set of decisions like rejecting reservation request or which wavelength should be assigned to a specified data burst will be referred as policy μ. Policy μ is simply a mapping which satisfies

$$\mu(x, e) \in U(x, e)$$

and whose domain is $S \times \Omega$.

Assuming memoryless interarrival and fixed holding times for calls and a fixed policy μ, x_t evolves as a continuous time, finite state semi-Markov process. Let us represent the k^{th} event as e_k, the time of the k^{th} event as t_k, the state of the network just prior to time t_k as

x_{t_k} and the decision made at time t_k as u_{t_k} =μ(x_{t_k} , e_k). Then the average blocking cost over the infinite time horizon associated with the policy μ can be given as follows:

$$\nu(\mu) = \lim_{N\to\infty} \frac{1}{t_N} \sum_{k=0}^{N-1} g(x_{t_k}, e_k, u_{t_k}). \tag{4.1}$$

The average blocking cost might be interpreted as the average cost per unit time for the system in steady state. Under the assumption of fixed finite call holding times, the whole system is modeled as an ergodic semi-Markov process. Reaching any state *j* from any state *i* is possible. Therefore the average blocking cost of state *j* is the same as that of state *i* since the costs incurred in the process of reaching state *j* from state *i* do not contribute to average blocking cost as N → ∞. Therefore, $\nu(\mu)$ is independent of the initial state. Additionally, in ergodic processes it is known that time averages converge to ensemble averages. Therefore, average blocking cost converges to a deterministic constant with probability 1.

Let us define the differential cost-to-go of state x_{t_k} under policy μ as:

$$h^{\mu}(x_{t_k}) = \sum_{m=k}^{\infty} [g(x_{t_m}, e_m, u_{t_m}) - (t_{m+1} - t_m)\nu(\mu)], \tag{4.2}$$

where u_{t_m} = μ(x_{t_m} , e_m). It might be interpreted as the expectation of the difference of total blocking costs under policy μ over the infinite time horizon for a system initialized at state x_{t_k} compared to the system in steady state.

A policy is said to be optimal if the average blocking cost of all other policies is greater than or equal to the average blocking cost of that policy. If the optimal policy is denoted as μ*, then the optimal policy is defined as:

$$\mu^*(x,e) = \arg \min_{u \in U(x,e)} [g(x,e,u) + h^*(\dot{x})] \tag{4.3}$$

where *h**(*x*) represents the differential cost-to-go of state *x* associated with optimal policy μ*. In order to find the optimal policy, one should know the optimal differential cost-to-go of all possible states.

Optimal differential cost-to-go $h^*(x)$ values for each possible state x of the network can theoretically be computed as follows [7, 31]:

$$v^* E\{\tau \mid x\} + h^*(x) = E\{\min_{u \in U(x,e)} [g(x,e,u) + h^*(\dot{x})]\}, \quad x \in S \tag{4.4}$$

where $v^* = v(\mu^*)$ represents the average blocking cost associated with the optimal policy and τ represents the time until the next event occurs. $E\{\tau \mid x\}$ represents the expectation of the time required until the next event occurs and $\dot{x}$ represents the next state of the network. If the number of all possible states is denoted as $\mid S \mid$, then Equation 4.4 is a system of $\mid S \mid + 1$ unknowns and $\mid S \mid$ nonlinear equations for each possible state. The unknowns are $\mid S \mid$ optimal differential cost-to-go values for each possible state and the value v^*. Therefore, one more equation is needed to solve Equation 4.4.

If the vector consisting of optimal differential cost-to-go values $h^*(x)$ for all possible states x is denoted as H^*, then it is seen that if H^* solves the Equation 4.4 then $H^* + re$ also solves that equation where r is a constant and e represents a vector of ones with length $\mid S \mid$. In other words, what is important is the difference values $h^*(x) - h^*(y)$ for all possible states x and y. Therefore, if the empty system is taken as a reference state and denoted as $\hat{x}$, one can assume without loss of generalization that

$$h^*(\hat{x}) = 0. \tag{4.5}$$

Under this assumption Equation 4.4 together with the Equation 4.5 is known as Bellman equations and constitute a system with $\mid S \mid + 1$ unknowns and $\mid S \mid + 1$ nonlinear equations which is solvable.

Policy iteration and value iteration are two well known approaches to solve a dynamic programming problem as presented in Chapter 3. We comment briefly on the policy iteration, in which, one starts with an arbitrary policy μ_0 and generates a sequence of new policies μ_1, μ_{2},... . Given a policy μ_k, first the $v(\mu_k)$ value is computed according to (4.1) and $h^{\mu_k}(x)$ values for all possible states x are computed according to (4.2). This process is also known as policy evaluation step. Then, the policy improvement step is performed to find the next policy μ_{k+1} according to the following formula:

$$\mu_{k+1}(x,e) = \arg \min_{u \in U(x,e)} [g(x,e,u) + h^{\mu_k}(\dot{x})], \; x \in S \tag{4.6}$$

It is shown in [7, Proposition 2.4] that policy iteration algorithm generates an improving sequence of policies which terminates with the optimal policy μ^*.

Even for networks consisting of a few nodes and links, computation and storage of the optimal differential cost-to-go $h^*(x)$ values for every possible state x of the network by using Bellman equations might be impractical. For this reason, in the next section, the neuro-dynamic programming approach, which is an approximate policy iteration method to find near optimal policies, will be presented.

4.2.2 Neuro-Dynamic Programming Formulation

In this section, the theoretical framework of the neuro-dynamic programming solution to wavelength assignment problem in OBS networks is presented. Neuro-dynamic programming (reinforcement learning) method is an approximate dynamic programming method based on simulations, which produces near-optimal solutions to large-scale dynamic programming problems.

Neuro-dynamic programming starts with an arbitrary policy μ_k and approximates the cost-to-go function of this policy $h^{\mu_k}(\cdot)$ with an approximate cost-to-go function $\widetilde{h}(f(x),\theta_{\mu_k})$, where $f(x)$ represents a vector of features extracted from the network and θ_{μ_k} represents a vector of tunable parameters associated with each used feature. Then, policy iteration on $\widetilde{h}(f(x),\theta_{\mu_k})$ is performed to obtain better policy μ_{k+1}. This process is known as approximate policy iteration and can be summarized as follows:

- Start with a fixed policy μ_k.
- Approximate the values $h^{\mu_k}(x)$ and $v(\mu_k)$, which are hard to find, with an approximate function $\widetilde{h}(f(x),\theta_{\mu_k})$ and a scalar quantity $\widetilde{v}(\mu_k)$ respectively by using simulations.
- Use policy iteration on approximate values $\widetilde{h}(f(x),\theta_{\mu_k})$ and $\widetilde{v}(\mu_k)$ to obtain a better policy μ_{k+1}.
- Repeat this process by using μ_{k+1} instead of μ_k, until obtained blocking probabilities do not change much with each iteration.

Approximations of optimal cost-to-go functions have been commonly used in the past [31, 34-36]. Here, we are interested in the problems with a large number of states, and approximate cost-to-go functions $\tilde{h}(\cdot,\theta)$ that can be described with relatively few numbers (θ of small dimension). The main idea in these problems is to use state evaluators to rank different states and make a decision that results in the state with maximum reward or minimum cost, minimum cost in our case. The state evaluator calculates a numerical value for each state using a heuristic formula, which includes weights for the various features of the state. In other words, state evaluator calculates the approximate cost-to-go function $\tilde{h}(\cdot,\theta)$, where the weights of the features correspond to the parameter vector θ.

In Neuro-dynamic programming there are three steps to be considered

- Determining the general form of the approximate function $\tilde{h}(\cdot,\theta)$.
- Deciding on which features should be used in the approximate function $\tilde{h}(\cdot,\theta)$.
- Selecting a proper method to tune the parameter vector θ and scalar quantity $\tilde{v}$.

which will be presented in the following sub-sections.

4.2.2.1 Approximation Architecture

Selection of architecture means the choice of a parametric class of functions $\tilde{h}(\cdot,\theta)$ that suits best to the considered problem and is an important issue in function approximation.

Approximation architectures can broadly be classified into two main groups as linear and nonlinear ones. A linear architecture is of the form

$$\tilde{h}(x,\theta) = \sum_{k=0}^{K} \theta(k) f_k(x) \tag{4.7}$$

where $\theta(k)$, $k = 0, 1,\ldots, K$, are the elements of a real parameter vector θ, and $f_k : S \rightarrow \Re$ are known functions extracted from the network state. These functions are preferred to be easily computable for the simplicity of the architecture.

Assume that some training data pairs (x, $h^{\mu}(x)$) obtained by simulations under a policy μ are wished to fit using linear architecture. This issue can be formulated as a least squares problem where the aim is to minimize the squared error

$$\sum_{x} [\sum_{k} \theta(k) f_k(x) - h^{\mu}(x)]^2 \tag{4.8}$$

over all parameter vectors θ. (4.8) is a linear least squares problem even if the functions $f_k(x)$ are nonlinear and can be solved using linear algebra techniques.

In nonlinear architecture, the dependence of $\widetilde{h}(x,\theta)$ on θ is nonlinear and the least squares problem of minimizing the squared error

$$\sum_{x} [\widetilde{h}(x,\theta) - h^{\mu}(x)]^2 \tag{4.9}$$

cannot be reduced to linear algebraic problem. Therefore, (4.9) should be solved by means of nonlinear programming methods. Multilayer Perceptron [7, 37] method is most commonly used nonlinear approximation architecture in the literature and has the power of approximating arbitrary functions of feature vector $f(x)$.

In this thesis, linear architecture is used as approximation architecture due to two main reasons. First, the linear dependence of the approximation architecture to the parameter vector θ enables us to use fast and well-tested linear algebra algorithms. Second reason is to determine the features that are most relevant to the decision making process since the parameters associated with the most relevant features in decision process will dominate that of irrelevant features in magnitude.

4.2.2.2 Features

The function $h(x)$ that should be approximated is often a highly complicated nonlinear map as in the case of wavelength assignment problem in OBS networks. Therefore, it is sensible to break this complexity into smaller and less complex pieces by feature extraction. A feature is simply a mapping $f_k : S \rightarrow \Re$ where $\Re$ represents the set of real numbers. Once the set of features $f_1,\ldots,f_K$ are determined, the feature vector is formed $f(x)=(f_1(x),\ldots,f_K(x))$. These features are usually handcrafted based on available insight and prior experience on the problem. Then either using the extracted feature vector or both the feature vector and raw encoding of the state of network, *h(x)* is approximated with $\widetilde{h}(f(x),\theta)$ by the approximation map parameter θ. Throughout this thesis study, we use only feature-based approximation architecture.

For the simulations, basically two kinds of features are extracted from network state.

1) Availability of wavelength converters: The first defined feature, $f_1(x)$, is a one dimensional vector of length total node number, where each entry shows number of available wavelength converters at each node for the current state.
2) Local availability of wavelengths: Second feature, $f_2(j,l)$, composes of entries such that each entry denotes available number of wavelength j along link l for the current state.

4.2.2.3 Training Method

There are several on-line and off-line methods to train the parameter vector θ [7]. Our main goal is to train the approximation architecture so as to identify θ, $\widetilde{v}$ in such a way that $\widetilde{h}(\cdot,\theta)$ and $\widetilde{v}$ are good estimates of h^{μ} and $v(\mu)$ respectively. In this section, Sutton's [16] *TD*(0) (temporal differences) method which is a commonly used method in neuro-dynamic programming applications will be presented. This method is used in all simulations presented here.

As presented in Chapter 3, *TD*(0) method has originally been proposed for discrete time, discounted cost problems. Since wavelength assignment in OBS networks can be considered as a continuous-time, average-cost problem, average cost *TD*(0) method [32] is used which is a modified version of a discounted cost *TD*(0) method [38]. It is shown that average cost *TD*(0) method has the same convergence properties with discounted cost TD(0) method [39].

One way to approximate $v(\mu)$ in (4.1) is to sum a large number of immediate costs observed in a long trajectory of the system and then normalize by dividing with the total elapsed time. However, one should wait till the last event happens in order to find the average blocking cost $v(\mu)$ associated with policy μ. Another way is to use Robbins-Monro stochastic approximation algorithm, which can be defined as follows:

$$\tilde{v}_k = \tilde{v}_{k-1} + \eta_k [g(x_{t_{k-1}}, e_{k-1}, u_{t_{k-1}}) - (t_k - t_{k-1})\tilde{v}_{k-1}], \tag{4.10}$$

where η_k is a diminishing step size parameter with increasing k. This is an online algorithm and each $\tilde{v}_k$ value is updated after the k'th event e_k. It has been shown in [32, Theorem 1] that $\tilde{v}_k$ converges to $v(\mu)$ as $k \to \infty$. Temporal difference d_k computed after k'th event is defined as follows:

$$d_k = \underbrace{[g(x_{t_k}, e_k, u_{t_k}) - (t_{k+1} - t_k)\tilde{v}_k + \tilde{h}(x_{t_{k+1}}, \theta_k)]}_{\text{Estimated cost-to-go based on simulation}} - \underbrace{\tilde{h}(x_{t_k}, \theta_k)}_{\text{Current estimate}} \quad . \tag{4.11}$$

Temporal difference d_k represents the difference between an estimate of the cost-to-go based on the simulated outcome and the current estimate $\tilde{h}(x_{t_k}, \theta_k)$. Therefore, the temporal difference provides an indication as to whether the current estimate should be raised or lowered.

In order to minimize the squared error in (4.9), incremental gradient flow method is used which can be found in standard texts such as [7, Section 3.2.4]. By using this method the parameter vector θ is updated by

$$\theta_{k+1} = \theta_k - \gamma_k \sum_{m=0}^{\infty} \left[\nabla_\theta \left[\tilde{h}(x_{t_m}, \theta_m) - h^\mu(x_{t_m})\right]\right]\left[\tilde{h}(x_{t_m}, \theta_m) - h^\mu(x_{t_m})\right], \tag{4.12}$$

where γ_k is a diminishing step size parameter with increasing k and ∇_θ represents the gradient with respect to parameter vector θ. When the definition of $h^\mu(x)$ in (4.2) is substituted into (4.12), we get

$$\theta_{k+1} = \theta_k - \gamma_k \sum_{m=0}^{\infty} \nabla_\theta \tilde{h}(x_{t_m}, \theta_m)\, [\tilde{h}(x_{t_m}, \theta_m) - \sum_{k=m}^{\infty} [g(x_{t_k}, e_k, u_{t_k}) - (t_{k+1} - t_k) v(\mu)]] \tag{4.13}$$

Since it is known that $\tilde{v}_k$ converges to $v(\mu)$ as $k \to \infty$, (4.13) can be rewritten in terms of temporal differences as

$$\theta_{k+1} = \theta_k + \gamma_k \sum_{m=0}^{\infty} \nabla_\theta \tilde{h}(x_{t_m}, \theta_m) \sum_{k=m}^{\infty} d_k \,. \tag{4.14}$$

A more general update rule, known as *TD*(λ), where 0 ≤ λ ≤ 1, uses weighted sum of temporal differences and is defined as

$$\theta_{k+1} = \theta_k + \gamma_k \sum_{m=0}^{\infty} \nabla_\theta \widetilde{h}(x_{t_m}, \theta_m) \sum_{k=m}^{\infty} \lambda^{k-m} d_k \tag{4.15}$$

To see the intuition behind (4.15), one can refer to [7, Section 6.3]. It is seen that (4.14) is a special form of (4.15) when λ is set to 1. Therefore (4.14) is known as *TD*(1) method. (4.15) is an off-line version of *TD*(λ) since parameter vector θ should be updated after all the trajectory i_0, i_1, i_2, … is simulated. However, in the on-line version, parameter vector θ is updated as soon as temporal difference d_k becomes available. More specifically, following the state transition $(i_{t_k}, i_{t_{k+1}})$ parameter vector θ is updated as follows:

$$\theta_{k+1} = \theta_k + \gamma_k\, d_k \sum_{m=0}^{\infty} \lambda^{k-m}\, \nabla_\theta \widetilde{h}(x_{t_m}, \theta_m)\ . \tag{4.16}$$

On-line *TD*(0) method is obtained by setting λ = 0 in (4.16) and is given by the following Equation:

$$\theta_{k+1} = \theta_k + \gamma_k\, d_k\, \nabla_\theta \widetilde{h}(x_{t_k}, \theta_k)\ . \tag{4.17}$$

TD(0) does not evaluate previous gradients of $\nabla_\theta \widetilde{h}(x_{t_m}, \theta_m)$ for 0 ≤ m ≤ k in order to update θ_k as in (4.16). Therefore on-line *TD*(0) update equation is very suitable for real-time implementations. This is the main reason that *TD*(0) method is used in simulations. It should be noted that when the linear approximation architecture is used,

$$\nabla_\theta \widetilde{h}(x, \theta) = f(x) \tag{4.18}$$

Under a fixed policy μ, let us assume that θ_k values are updated according to (4.17) (on-line *TD*(0) method) and $\widetilde{v}_k$ values are updated according to (4.10). Additionally, assume that γ_k and η_k are diminishing step size parameters such that:

a) γ_k is positive, deterministic constant for k and satisfies $\sum_{k=0}^{\infty} \gamma_k = \infty$ and $\sum_{k=0}^{\infty} \gamma_k^2 < \infty$.

b) There exists a positive scalar n such that the sequence η_k satisfies $\eta_k = n\, \gamma_k$ for k.

Then, it is proven in [32, Theorem 1] that $\tilde{v}_k$ converges to $v(\mu)$ and θ_k converges to a limiting vector θ such that the squared error between $\tilde{h}(\cdot,\theta)$ and $h^{\mu}(\cdot)$ is minimized with respect to θ under the given approximation architecture.

4.2.2.4 Decomposition Approach

To obtain distributed algorithms and faster convergence especially for networks with large number of nodes and links, the immediate cost of an incoming call might be associated with its connection. This method is known as decomposition approach and was found to perform well and lead to shorter training times in [31, 33]. On the other hand, OBS network protocols use one-way reservation paradigm, and global resource information is not available at any node. At a node, processing is only done according to the information obtained only from the links of that node. For example, during wavelength assignment procedure in OBS networks, along route information is not available at the node. Therefore, there is a need for link-based wavelength selection. This directly corresponds to decomposition approach at neuro-dynamic programming methodology. That is why we use this approach throughout all NDP simulation study.

In this approach, local states x^l, associated with each link $l \in L$ are considered although they are not real states in true sense. This is because of the fact that they are affected by the global state x and they do not evolve as a Markov process. Assuming that a new control packet arrives to link l and its reservation is done, then the immediate cost of that link becomes $g^l(x_{t_k}, e_k, u_{t_k}) = g(x_{t_k}, e_k, u_{t_k})$. For all other events, the immediate cost associated with link l is set equal to 0.

Given a fixed policy μ, it can be shown that:

$$v(\mu) = \sum_{l \in L} v^l(\mu) \tag{4.19}$$

where $v^l(\mu)$ represents the average blocking cost associated with link l under policy μ. For each link, it can be introduced a scalar $\tilde{v}^l$ as an estimate of $v^l(\mu)$ and an approximation architecture $\tilde{h}^l(x^l, \theta^l)$ where θ^l represents a parameter vector that belongs to the features associated with link *l*. In this approach each link *l* has its own policy μ^l. When a reservation request arrives to link *l*, local policy μ^l is defined as:

$$\mu^l(x^l, e_k) = \arg \min_{u \in U(x,e)} [g^l(x_{t_k}, e_k, u_{t_k}) + \tilde{h}^l(\dot{x}^l, \theta^l)] \tag{4.20}$$

The main idea of decomposition approach is not to update global parameter vector θ after each event, but instead update the parameter vector θ^l which is local to each link. Under these definitions, if the incoming control packet wants to reserve the resources that belongs to link *l*, then local TD(0) algorithm for this link is given as:

$$\theta_k^l = \theta_{k-1}^l + \gamma_k^l d_k^l \nabla_\theta \tilde{h}^l(x_{t_{k-1}}^l, \theta_{k-1}^l) \tag{4.21}$$

$$\tilde{v}_k^l = \tilde{v}_{k-1}^l + \eta_k^l [g^l(x_{t_{k-1}}, e_{k-1}, u_{t_{k-1}}) - (t_k - t_{k-1})\tilde{v}_{k-1}^l] \tag{4.22}$$

$$d_k^l = g^l(x_{t_{k-1}}, e_{k-1}, u_{t_{k-1}}) + \tilde{h}^l(x_{t_k}^l, \theta_{k-1}^l) - (t_k - t_{k-1})\tilde{v}_{k-1}^l - \tilde{h}^l(x_{t_{k-1}}^l, \theta_{k-1}^l) \quad . \tag{4.23}$$

where γ_k^l and η_k^l are small diminishing step size parameters explained in the previous section. Since decomposition approach ignores some dependencies, it should not be expected to obtain better average blocking probabilities when compared to global *TD*(0) algorithm. However, at the expense of introducing an additional modelling error and obtaining higher average blocking probabilities, one can obtain distributed algorithms and faster convergence rates.

The method used in all simulations can be briefly explained as follows. First, we start with an arbitrary policy μ and apply *TD*(0) until the parameter vector and average cost value converge. Then the resulting limiting value of parameter vector θ is used to define a new policy by means of policy iteration. This process is repeated until it is observed that obtained blocking probabilities do not change much with each iteration. This approach is known as approximate policy iteration and has some weak theoretical guarantees that the policy iteration algorithm generates an improving sequence of policies [7, Proposition 2.4].

This policy iteration process might be interpreted as an actor-critic system. In this interpretation, the critic is responsible for the policy evaluation step and evaluates the performance of the current policy, in other words it calculates the estimate of h^{μ_k} by tuning the parameters. On the other hand, actor is responsible for the policy improvement step who takes into account the latest evaluation of the critic, h^{μ_k}, to obtain the next policy μ_{k+1}.

There is an alternative to the standard version of policy iteration in which policy update is performed after each update by the policy evaluation algorithm without waiting for the critic's computations to converge. Methods of this type are known as optimistic policy iteration in the literature and have been widely used in practice. Although this method has no theoretical convergence guarantees, it has been shown to perform well in some situations [35-37].

Chapter 5

Simulations and Results

Many protocols and algorithms proposed for circuit-switched optical networks results with high burst blocking rates in OBS due to its one-way reservation property. Therefore, there is a need for smart routing and wavelength assignment algorithms in order to obtain lower blocking probabilities. We compute the performance of different proposed wavelength assignment algorithms for varying available resources and traffic rates throughout the network.

Many studies in the literature assume full wavelength conversion and decreasing blocking probability is achieved by using deflection routing, FDLs etc. When resources such as wavelength converters and FDLs are fully available for each connection, it is expected to have a reasonably low blocking rate. However, these resources are scarce and expensive devices. The main motivation behind this thesis study is to achieve same blocking rate with lower cost network structure by using smarter wavelength assignment algorithms.

Well-known wavelength assignment algorithms are proposed for networks using two-way reservation protocols, and these do not seem to be appropriate for optical burst switching networks. Interestingly, usually the worst performing of all, random wavelength assignment algorithm, results with better performance over optical burst switching networks using JET protocol. This is basically because of random wavelength assignment's uniform link utilization property. This result shows the need for intelligent wavelength assignment algorithms designed especially for optical burst switching paradigm. Moreover, intuitively it can be seen that with fewer number of wavelength converters shared from a converter pool, the same blocking rate range with full wavelength conversion can be achieved.

Besides performance comparison for networks with varying number of converters at each node, we also try to match and examine the performance of proposed algorithms for varying traffic rates and burst lengths.

5.1 OBS Simulator

Optical burst switching is a new switching paradigm in optical networks. In OBS networks, a control packet is sent before transmitting the burst; and the burst is sent after an offset time depending on the OBS protocol. Since OBS uses one-way reservation, there is not a handshake between source and destination on the wavelength assignment before the transmission of the burst. This new structure is different than other switching paradigms, so well-known simulator programs are not capable of implementing optical burst switching networks. Therefore, an event-driven packet-based OBS simulator is written in Microsoft Visual C++ 6.0 environment. Due to its resilient structure, the written simulator has the following features:

- Different OBS protocols can be implemented.
- Any type of routing and wavelength assignment algorithms can be employed.
- It has the capability of implementing neuro-dynamic programming.
- Fiber Delay Lines (FDLs) and Wavelength converters can be used for reservation conflicts, and a shared pool structure for these resources is available.
- It is capable of handling fixed or variable length of data bursts.
- Offset values may be fixed or varying from source to source.
- The traffic, thus the distributions for interarrival times of packets, is adjustable.
- User defined source-destination pair matrix can be used.
- It is also capable of processing approximately $48 * 10^6$ packets/min with a 2.4 GHz processor.

5.2 Simulation Environment

Throughout all simulations, as a network environment, NSFNET topology of Figure 4.2, is used with the following properties:

- It is composed of 14 nodes, 42 unidirectional links
- Length of the links is fixed to 500 km corresponding to 1700 microsecond as the propagation delay.
- Each unidirectional link is composed of a single fiber with 8 wavelengths.
- There is traffic between each pair of nodes (fully connected) and total traffic in the network is uniformly distributed among all node pairs.
- Packet arrivals are Poisson distributed, and total packet arrival rate is varied from $1*10^6$ packets/sec to $2*10^6$ packets/sec for each node.
- JET is used as the OBS reservation protocol.
- Routing of bursts is done according to Dijktra's shortest path algorithm.
- 4 different wavelength assignment algorithms' performances in terms of blocking probability are analyzed.
- All nodes have the converter pool structure and number of converters is varied throughout simulations.
- Control packet's processing time lasts 10 µsec at each node.
- Each packet consists of 1500 bytes of data, and burst is formed from fixed number of packets. Throughout simulations, this fixed number is chosen as either 10 or 20 packets.
- Switching rate at a node is 10 Gbits/sec, so burst switching time at each node is either 12 or 24 µs according to the burst length.
- Blocking probability is averaged over $1*10^7$ packet arrivals to the system. For all simulations, the same random number generator is used to determine the packet interarrival times, so all results correspond to data traffic with exactly same characteristics.

5.2.1 Simulated Wavelength Assignment Algorithms

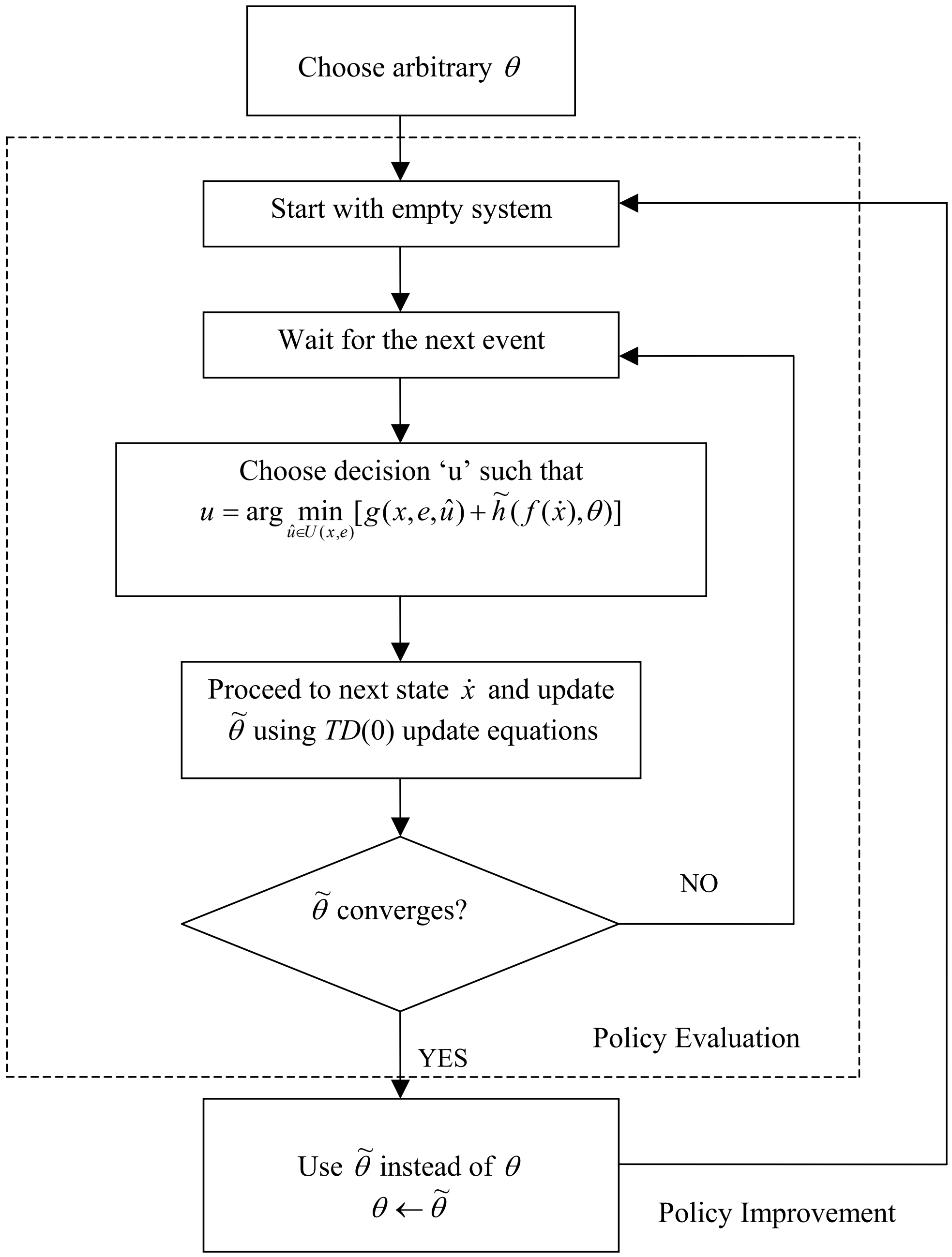
Basically, performances of proposed heuristic and dynamic wavelength assignment (DWA) algorithms of Chapter 4 are compared with the random wavelength assignment (RWA) algorithm in terms of blocking probability for varying number of wavelength converters, traffic rates and burst lengths.

As heuristics, proposed most-fit-min (MFM) and most-fit-rand (MFR) wavelength assignment algorithms are used. On the other hand, as many as feature vectors one extracts from the network, that many different dynamic wavelength assignment algorithms can be defined. As described in Section 4.2.2.2, among many, we define only two features; availability of wavelength converters ($f_1(x)$), local availability of wavelengths ($f_2(j,l)$). Combining these features, we form the feature vector such as $\mathrm{F} = \{f_1, f_2\}$.

For neuro-dynamic programming, parameter vector θ is trained by decomposition approach, since OBS network protocols use distributed algorithms. NDP method used in simulations is summarized in Figure 5.1.

In all simulations, $2.5*10^6$ event steps are used for policy evaluation. After $2.5*10^6$ event steps, the policy is updated by adapting a new parameter vector θ and this process is repeated until a steady-state behavior in average blocking probabilities is obtained. Since connection based decomposition approach is not appropriate for OBS networks (along route information is not available at a node), we use link based decomposition approach. Moreover, connection based decomposition approach has weak theoretical convergence guarantee, and it is not expected for link based approach to show better convergence behavior. Furthermore, considering large sized network, thus long feature vector because of high number of states, a convergence to a fixed point can not be observed for NDP with link based decomposition approach. However, typical improving behavior of neuro-dynamic programming is better than expected and blocking rate remains in a small range after a number of policy iterations. Throughout simulations, it is observed that blocking probability typically reaches steady-state after an average of 300 iterations. A typical improving behavior of neuro dynamic programming is shown in Figure 5.2. While blocking probability can be observed up to approximately $9*10^{-4}$ throughout heading iterations, it remains lower than $2*10^{-4}$ after sufficient number of iterations.

Since not a rigid convergence to a fixed point throughout policy iterations is observed, we pick the best policy generated in the course of algorithm, not the last one. This is also the approach of Tsitsiklis [31] for a similar situation.



Choose arbitrary θ
Start with empty system
Wait for the next event
Choose decision 'u' such that
$u = \arg\min_{\hat{u} \in U(x,e)} [g(x,e,\hat{u}) + \tilde{h}(f(\dot{x}),\theta)]$
Proceed to next state $\dot{x}$ and update $\tilde{\theta}$ using $TD(0)$ update equations
$\tilde{\theta}$ converges?
NO
YES
Policy Evaluation
Use $\tilde{\theta}$ instead of θ
$\theta \leftarrow \tilde{\theta}$
Policy Improvement

Figure 5.1: Flowchart of the NDP method

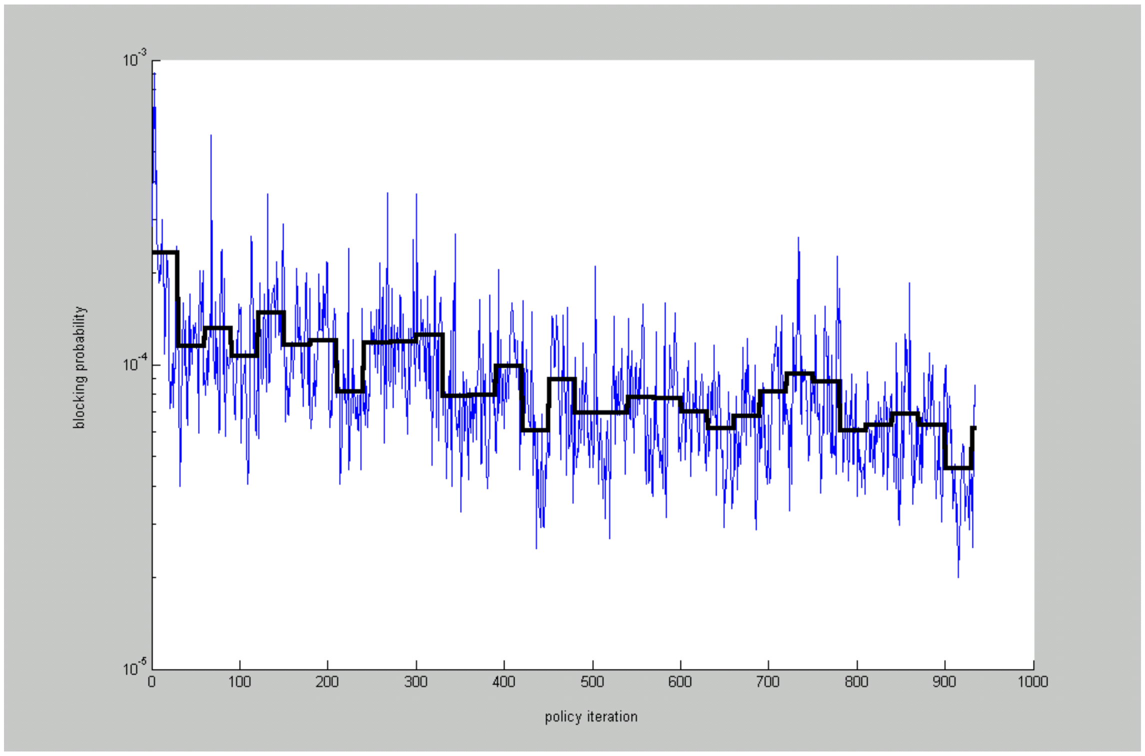


Figure 5.2: Blocking probability evaluated over $2.5*10^6$ event steps versus policy iteration

5.3 Numerical Results

In this section, simulation results of heuristic and neuro-dynamic programming algorithms on NSFNET topology are presented. The heuristic algorithms' results are obtained for five network loads using 2 different burst lengths and varying the number of converters per node. A similar set of experiments is carried out also for neuro-dynamic programming. Additionally, all types of algorithms are compared with each other for performance evaluation.

Figures 5.3-5.14 give the steady-state average blocking probability vs. traffic load for random, most-fit-min, and most-fit-rand heuristic wavelength assignment algorithms as well as by dynamic wavelength assignment algorithms using the specified features. Figures 5.15-5.18 display same results for some specific traffic loads from a different perspective, average blocking probability vs. number of wavelength converters.

The simulations are done basically for two fixed burst lengths: 10 packet bursts (12 μsec) and 20 packet bursts (24 μsec). Although the burst dropping rate of 10 packet bursts is higher, a packet base approach to dropping rate shows that more packets are dropped for 20 packet burst case. This is because of one 20 packet burst drop corresponds to two 10 packet burst drops, and such a ratio is not observed at dropping rates of bursts.

Throughout simulations, also converter numbers at network nodes are varied. It is important to mention that converter pool structure is implemented under all circumstances. What is meant by full conversion (fc) is that every wavelength on every link has its own converter at the nodes they are adjacent. Equivalently, at a node that is adjacent to *l* links with *w* wavelengths at each link, there exists *l*w* converters under full conversion assumption. Experiments are done for cases with fewer number of converters at each node such that for the same example given (fc/n) denotes *l*w/n* wavelength converters at each converter pool of each node. Moreover, the case of no wavelength conversion (nc), i.e., no wavelength converters throughout network, and sparse wavelength conversion (sc), i.e., full wavelength conversion only at some selected nodes, are also considered. For the simulations of sparse conversion, 7 network nodes of NSFNET topology, for which traffic is expected to be dense, are selected. Namely, these are nodes numbered with 0, 3, 4, 5, 7, 9 and 10 in Figure 4.2.

One of the reasons to prefer linear approximation architecture in neuro-dynamic programming is to determine the most relevant features in decision making process. When all the extracted features are combined, the entries of parameter vector θ associated with the irrelevant feature happen to be very smaller in magnitude compared to ones associated with other feature sets. Moreover, neuro-dynamic programming has the freedom to set all the parameters to zero except of those associated with the feature set that gives the best result of all. For our case, no such entry of θ close to zero is observed. This shows that availability of the features based on wavelength converters (f_1) and wavelengths (f_2) are both dominant in wavelength assignment problem in optical burst switching networks, and it is sure that NDP with only feature vector f_1 or f_2 cannot perform beyond NDP with feature vector $F = \{f_1, f_2\}$.

All the proposed assignment algorithms result with better blocking probabilities than conventional random wavelength assignment algorithm in most of the cases. When two proposed heuristics, most-fit-min and most-fit-rand, are compared, performance results are very close to each other, but most-fit-min always narrowly beats most-fit-rand. Also, when

NDP assignment algorithms are compared with proposed heuristics in terms of performance, it is observed that better results are obtained with feature vector F, composed of the features based on wavelength converters and wavelengths. Numerically, NDP with the feature F can improve the blocking probability with respect to best performing proposed heuristic, the most-fit-min wavelength assignment algorithm, up to %10, and random wavelength assignment algorithm up to %63 among different network loads under full conversion for 10 packet bursts. This result approximately holds for 20 packet bursts. On average, NDP with the feature F improves the blocking probability with respect to most-fit-min wavelength assignment algorithm by % 7.5 and with respect to random wavelength assignment algorithm by % 60.

What can be implied from simulation results of heuristic algorithms is that an equal performance of full conversion can be obtained under a situation with less number of converters throughout the network. It is directly seen that fc, fc/2 and fc/4 cases have approximately same blocking probabilities for all traffic rates and burst lengths over random, most-fit-min and most-fit-rand heuristics. For these cases, it is observed that the blocking of bursts at a node occurs because control packets cannot find an appropriate duration of bandwidth in any of the wavelengths, although there are available wavelength converters at the node. We see that although most-fit-min wavelength assignment algorithm is expected to give the best result under full conversion since it is created to utilize the wavelengths, it is not the case. This is due to the randomness in most-fit-min wavelength assignment algorithm when available wavelengths happened to have equal priorities. The system is usually not dense, and the priorities assigned to available wavelengths are usually the lowest available priority. Under such circumstances, the wavelength is selected randomly and this brings no intelligence to the assignment procedure.

The situation differs when neuro-dynamic programming is considered. For this case, the lower the number of converters at each node is, the higher the blocking probability of bursts is. NDP tries to find out the best policy that reduces the blocking probability as much as possible by efficient use of available resources. That is why the blocking probability for bursts is lowest for full conversion assumption. Different than heuristics cases, as the number of wavelength converters decreases down to fc/2, fc/4 etc., i.e. the number of available resources decreases, the performance worsens.

An intuitive explanation may state that under full wavelength conversion assumption, effect of wavelength assignment algorithm to average blocking probability is negligible, because one can always solve resource contention problem with wavelength conversion. However, as seen in Figure 5.7 and 5.13, the result obtained throughout simulations show that this is not the case. All proposed wavelength assignment algorithms improve over random wavelength assignment algorithm for fc case over all traffic loads. Although, random wavelength assignment results with a fair utilization of wavelengths, reservation timeline for the wavelengths is not efficiently used because of this randomness. Therefore, an assignment algorithm considering temporal parameters during assignment has higher number of available wavelengths for a reservation request. If there are no available wavelengths, it is not important to have many available converters, and such situations can occur more frequently for random heuristic than proposed algorithms.

When wavelength converter number at all nodes is decreased down to fc/8, performance of the proposed heuristic algorithms decreases significantly. However, this is not the case for NDP algorithms. This shows that while fc/4 converters can handle the assignment procedure efficiently and very closely to optimal full conversion case for heuristic wavelength assignment algorithms, a further decrease to fc/8 can not be endured. On the other hand, this is an expected result, since when converter number decreases the importance of wavelength assignment algorithm shows off. Under full wavelength conversion assumption, high number of available converters hinders the effect of dumb wavelength assignments. Therefore, when converter number is decreased below a threshold value, this effect cannot be efficiently removed, and this alters performance of the system. However, at the training phase, observing many network states, NDP dynamically learns about how to use the converters and wavelengths available at hand, therefore performance is better compared to heuristics especially for fc/8 and nc cases.

Another interesting observation from simulation results is that for the case nc, random wavelength assignment algorithm shows a little bit improved performance than proposed most-fit-min and most-fit-rand heuristics. This is an expected result, when link-based poor performance of random wavelength assignment algorithm appears to be behaving similar to call admission control. Similar results are also observed at highly loaded networks.

Furthermore, sparse conversion performs further beyond the expected in terms of blocking probability. As seen in Figure 5.19 and 5.20, average blocking probability even for the fc/8 case with uniform conversion where a total number of 42 converters are placed at all nodes remains significantly lower than sparse conversion case where a total number of 184 converters placed at densely loaded nodes are used. Moreover, sparse conversion results are very close to the nc case. This is due to the one way reservation paradigm used for OBS, and a wavelength conflict can occur at any node whether it is densely loaded or not. Therefore, there is a need for wavelength converters at every node for obtaining low blocking probabilities. It is more efficient to provide distributed less availability at all nodes rather than local full availability at some nodes throughout the route.

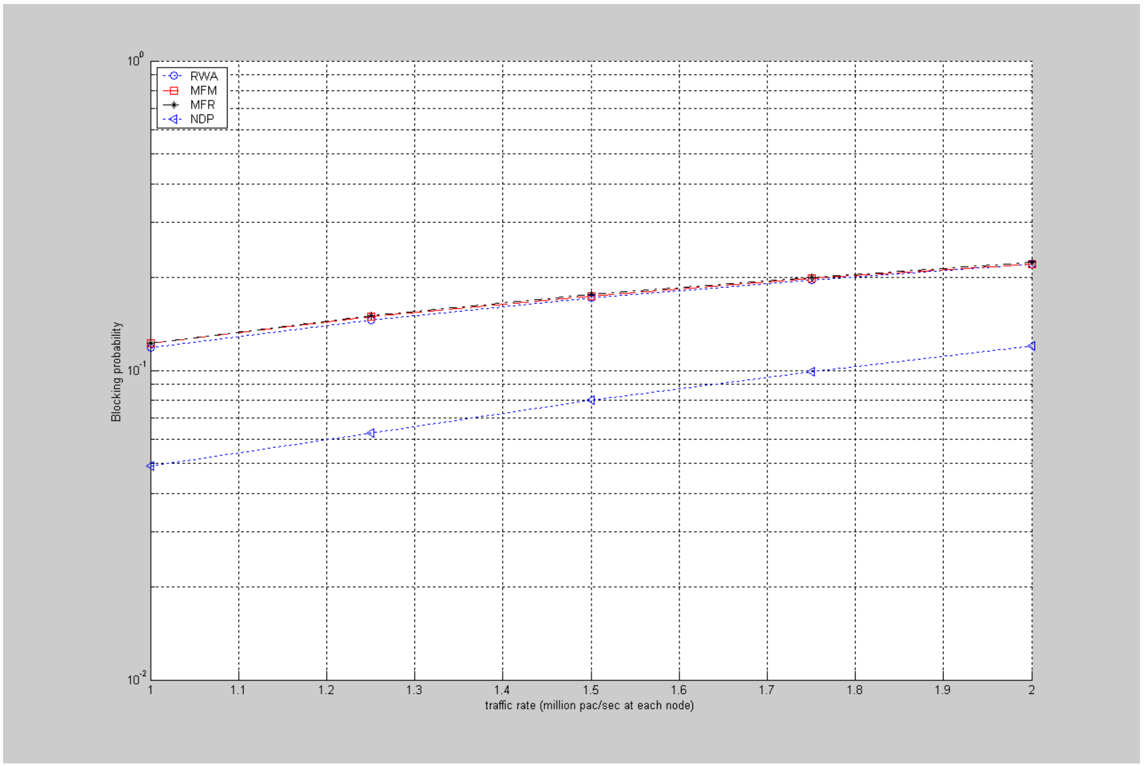


Figure 5.3: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and no wavelength conversion (nc) assumption

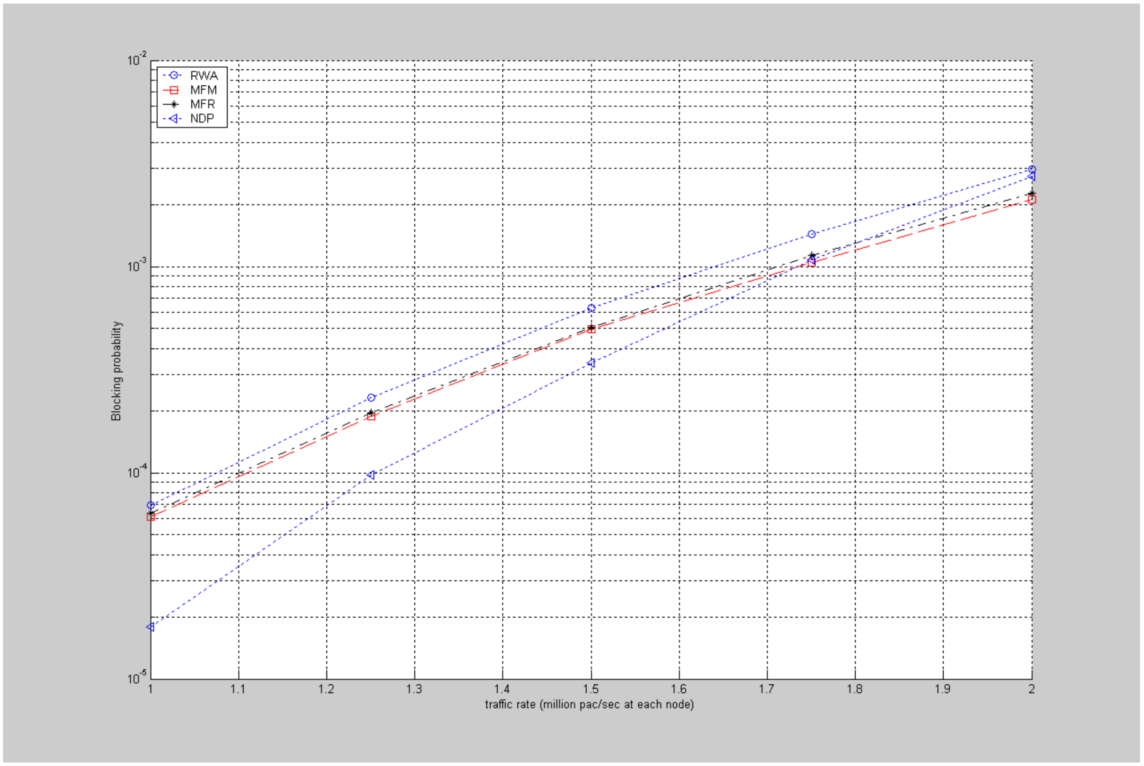


Figure 5.4: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and (fc/8) assumption

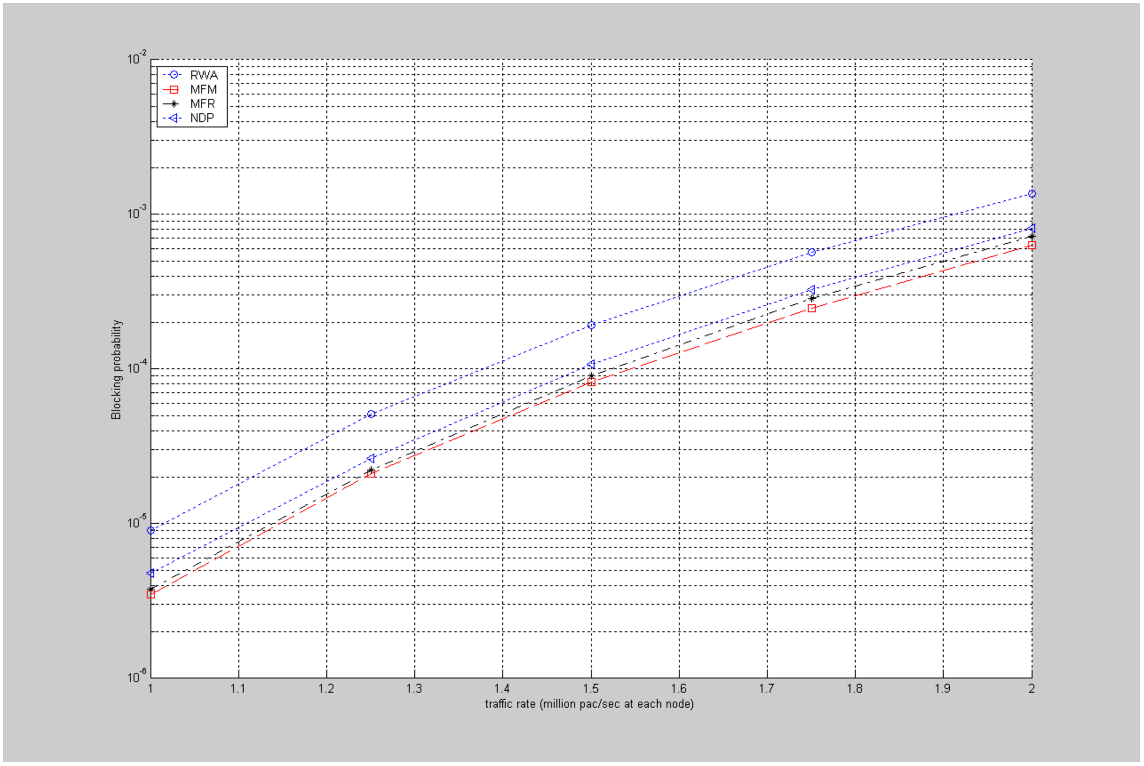


Figure 5.5: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and fc/4 assumption

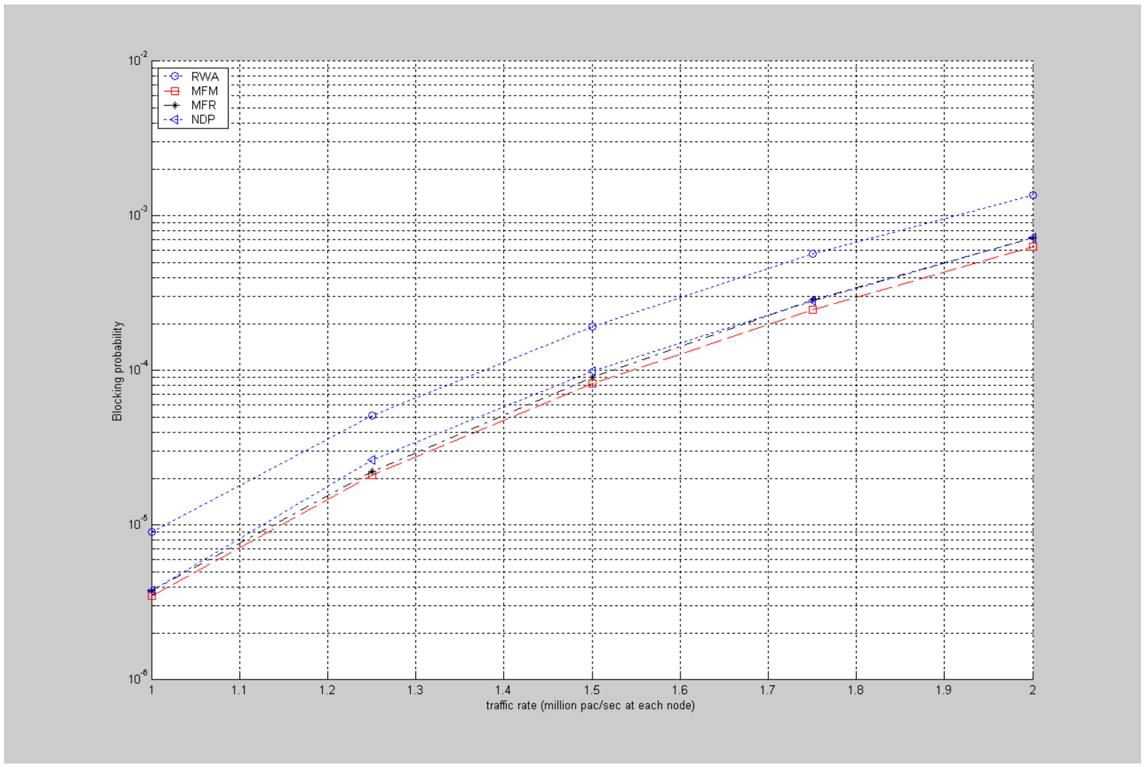


Figure 5.6: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and fc/2 assumption

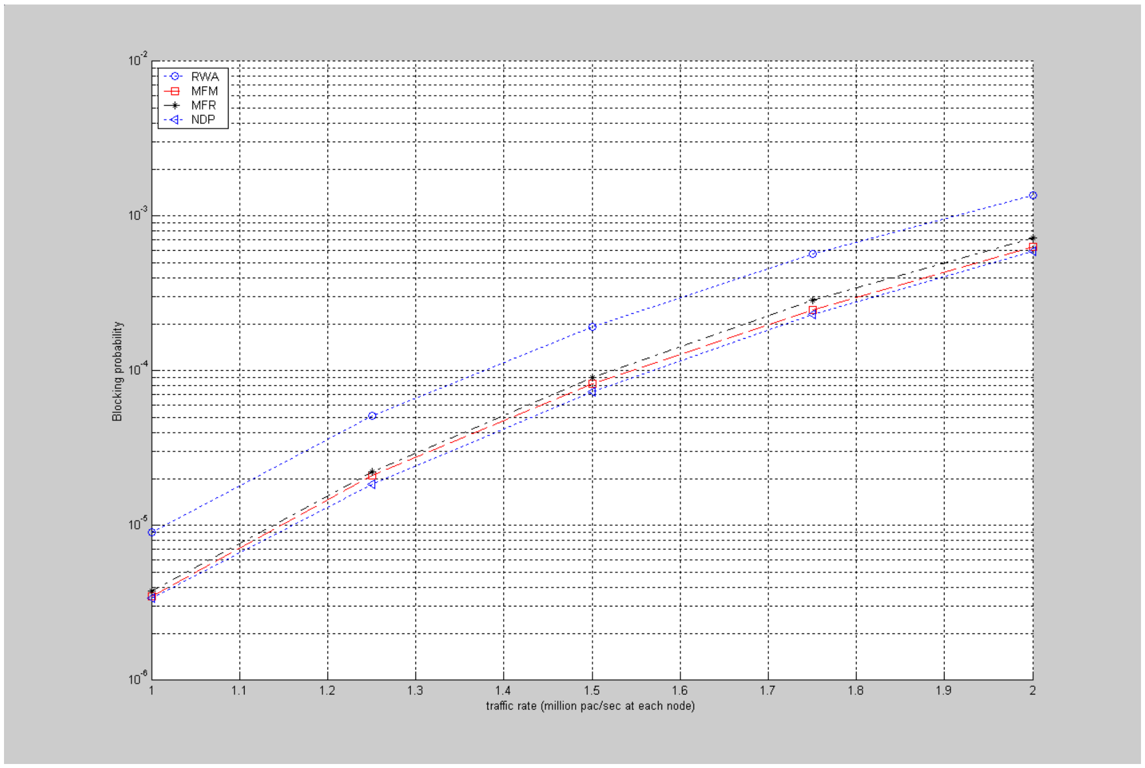


Figure 5.7: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and full wavelength conversion (fc) assumption

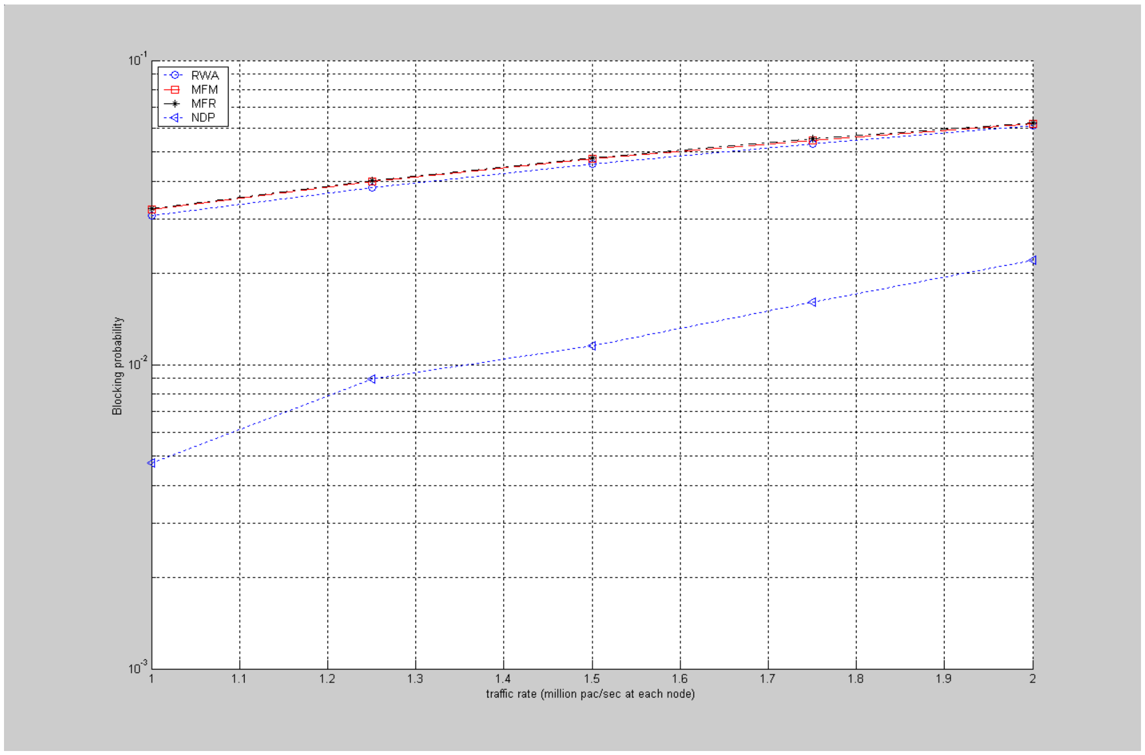


Figure 5.8: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and sparse conversion (sc) assumption

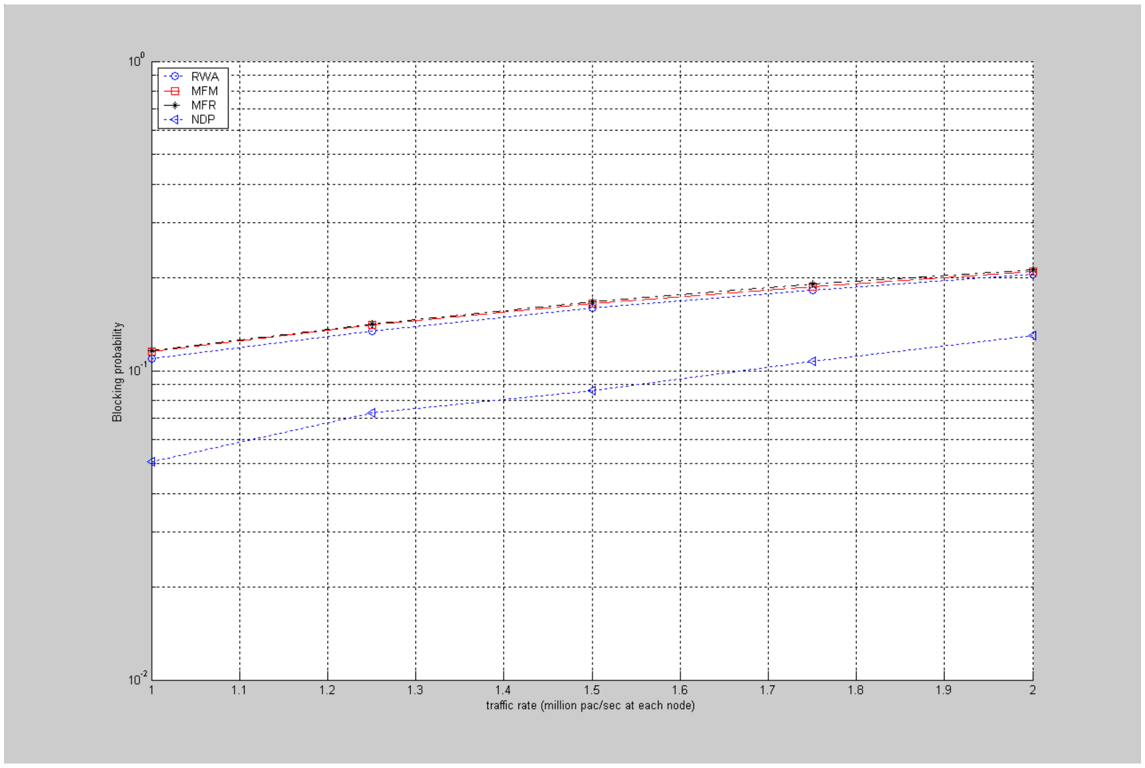


Figure 5.9: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and no wavelength conversion (nc) assumption

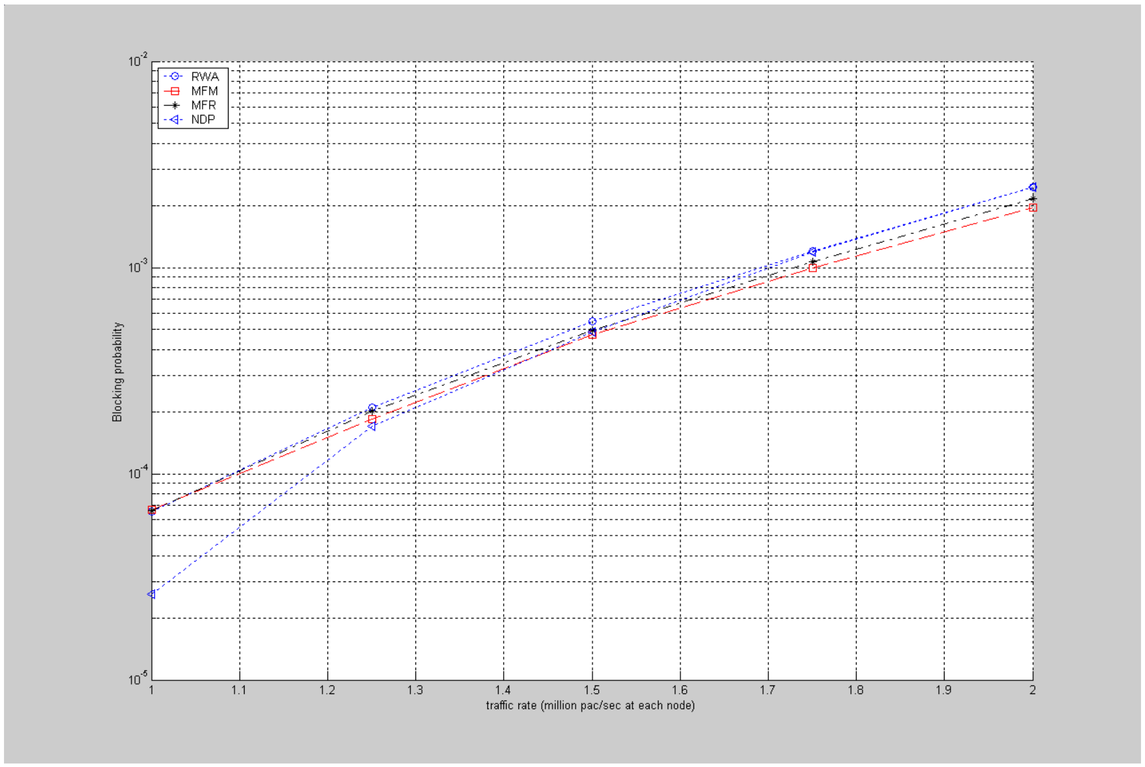


Figure 5.10: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and (fc/8) assumption

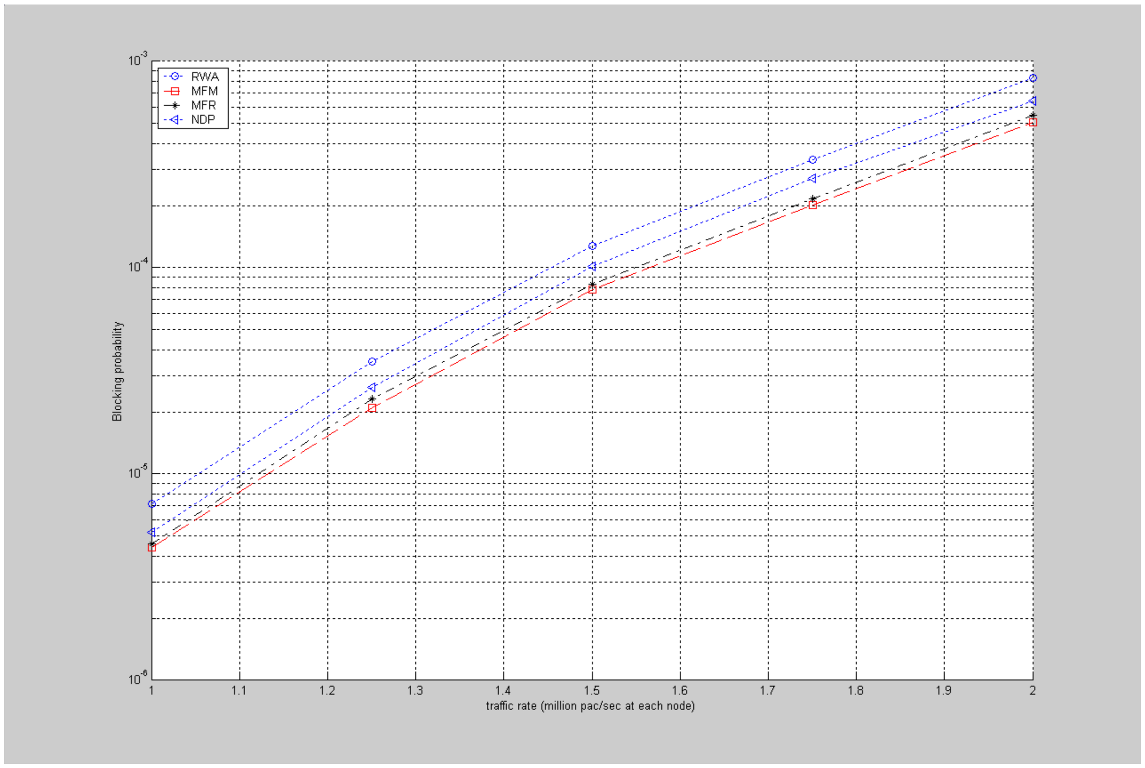


Figure 5.11: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and (fc/4) assumption

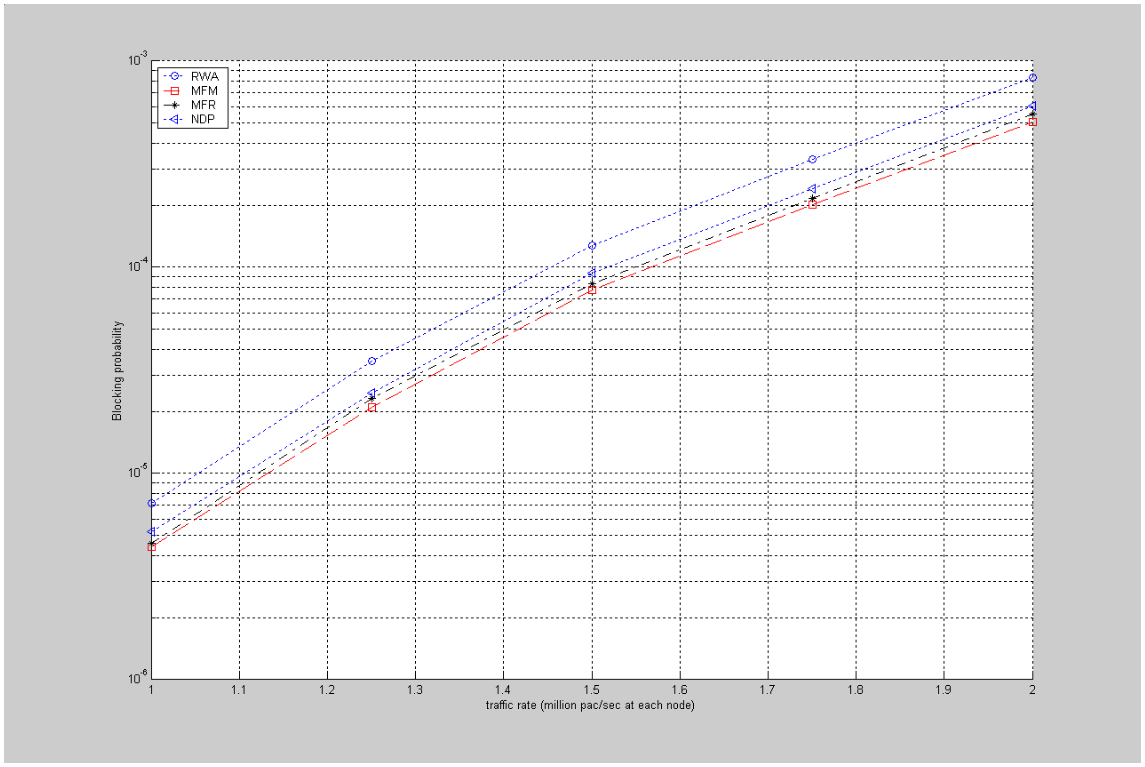


Figure 5.12: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and (fc/2) assumption

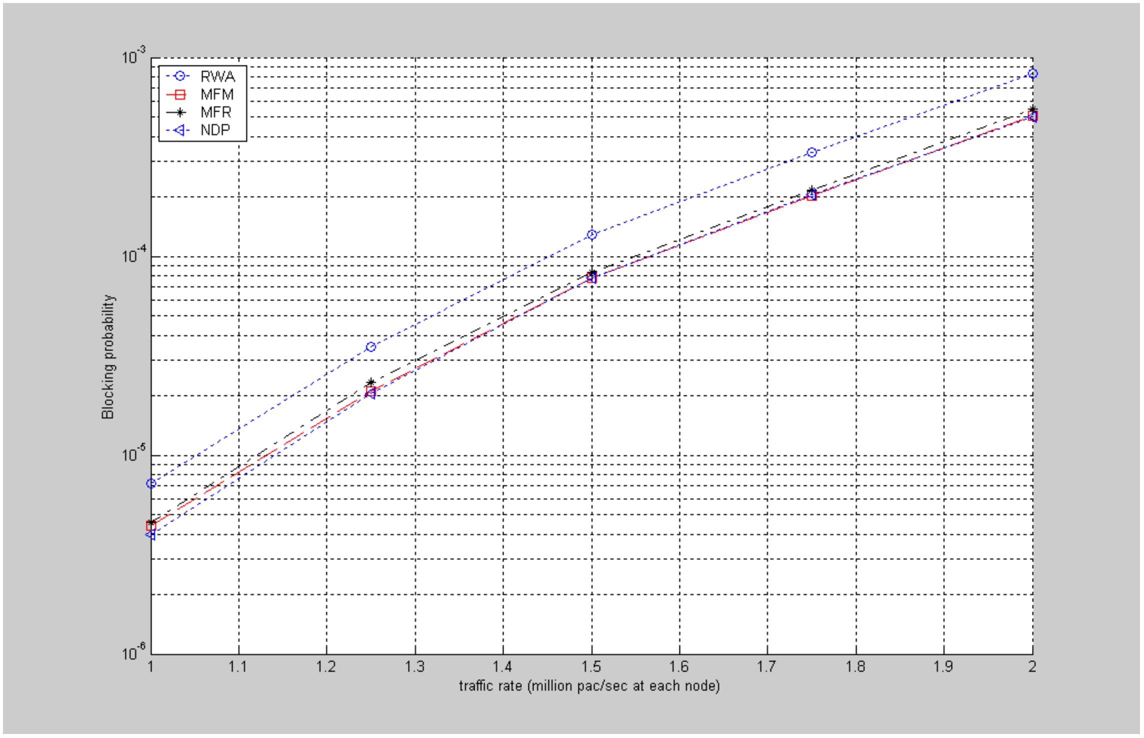


Figure 5.13: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and full wavelength conversion (fc) assumption

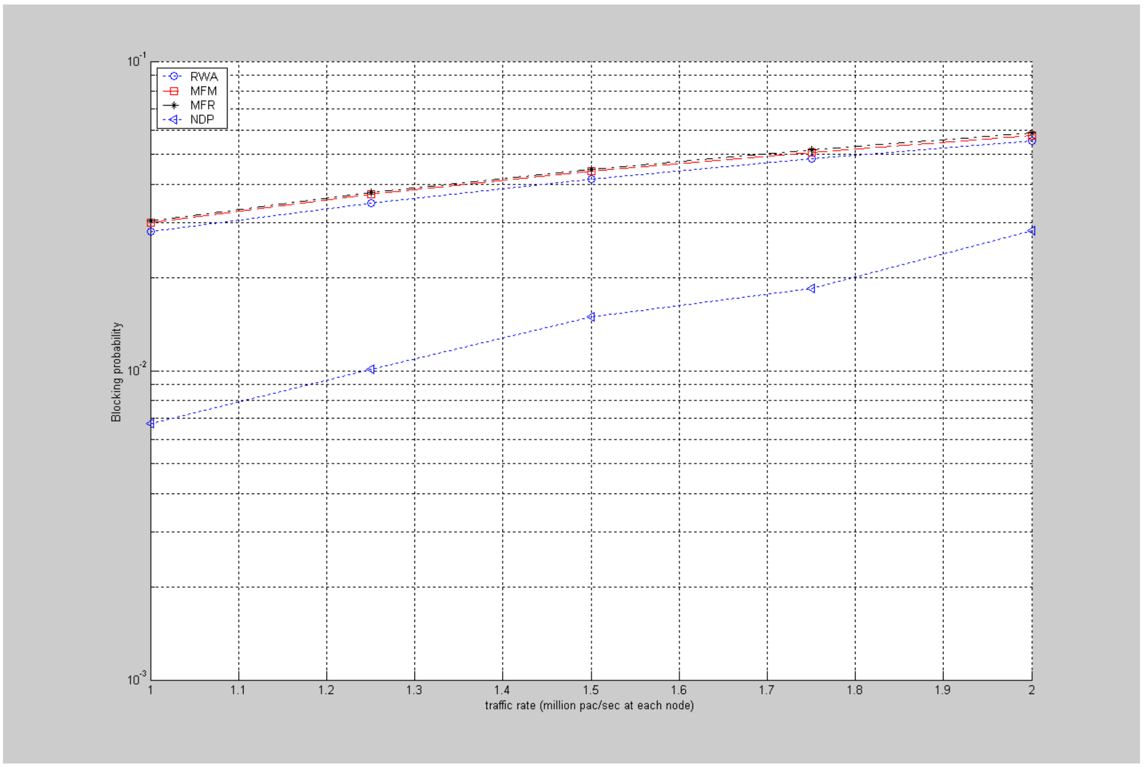


Figure 5.14: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and sparse conversion (sc) assumption

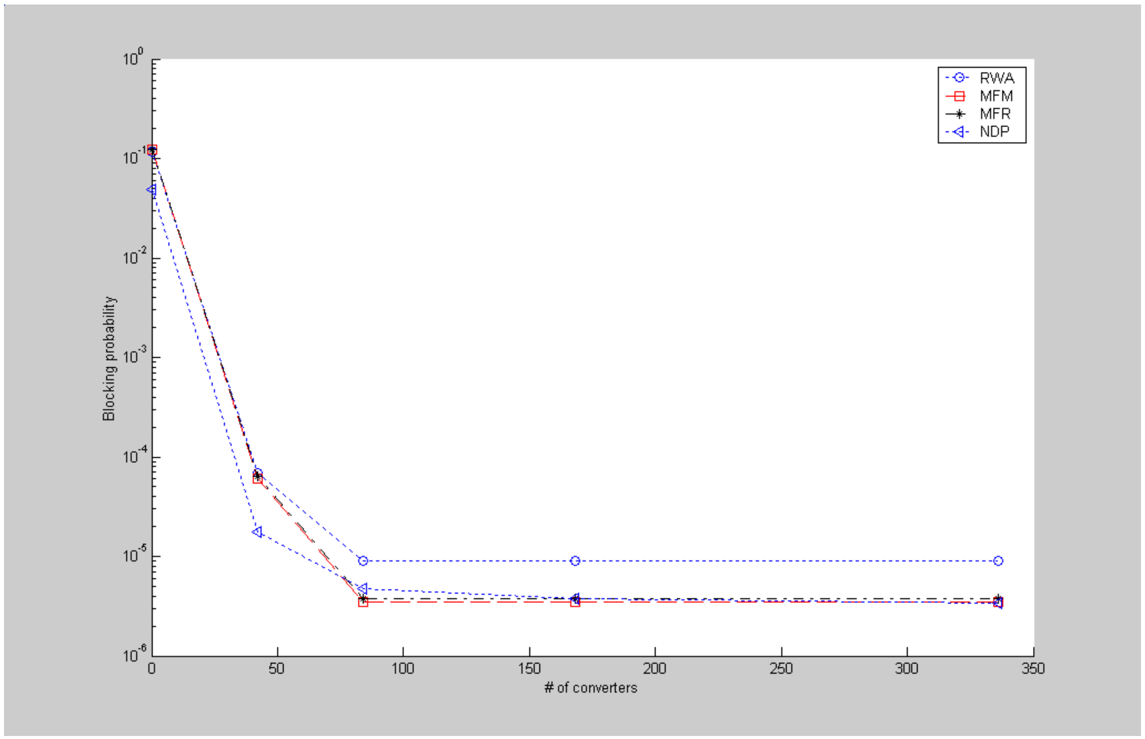


Figure 5.15: Blocking probability of random and proposed wavelength assignment algorithms vs. number of converters uniformly distributed at all nodes. Traffic rate per node is $1*10^6$ packets/sec and a burst is composed of 10 packets

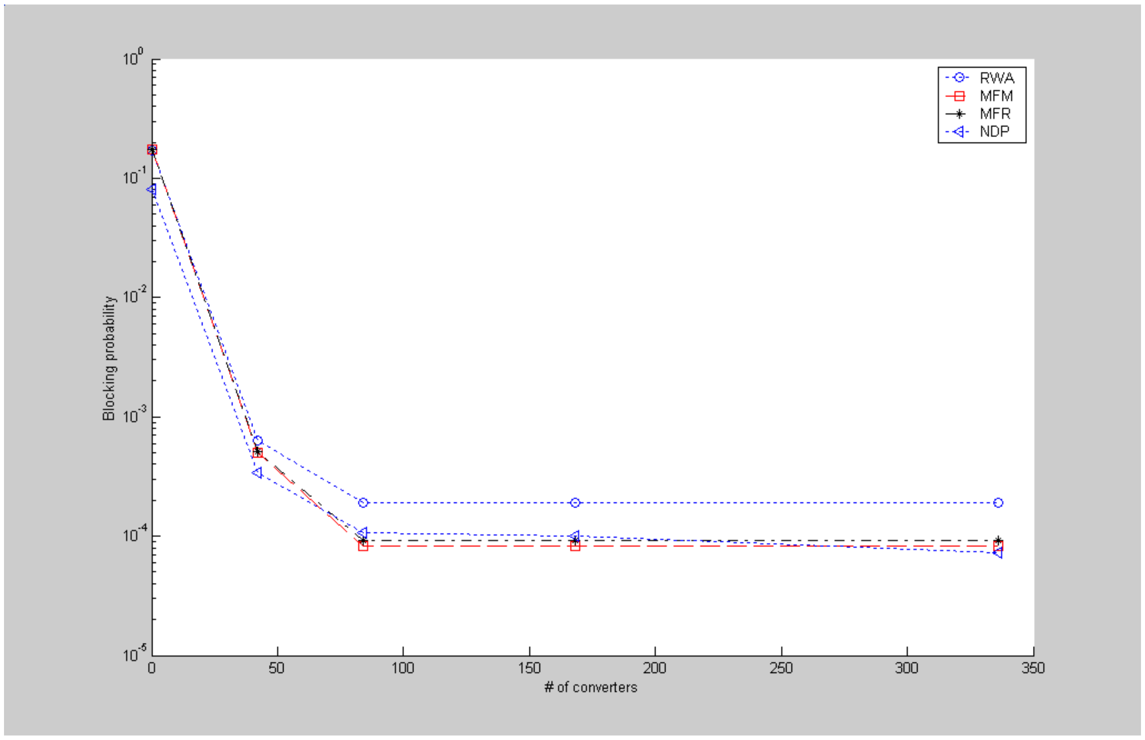


Figure 5.16: Blocking probability of random and proposed wavelength assignment algorithms vs. number of converters uniformly distributed at all nodes. Traffic rate per node is $1.5*10^6$ packets/sec and a burst is composed of 10 packets

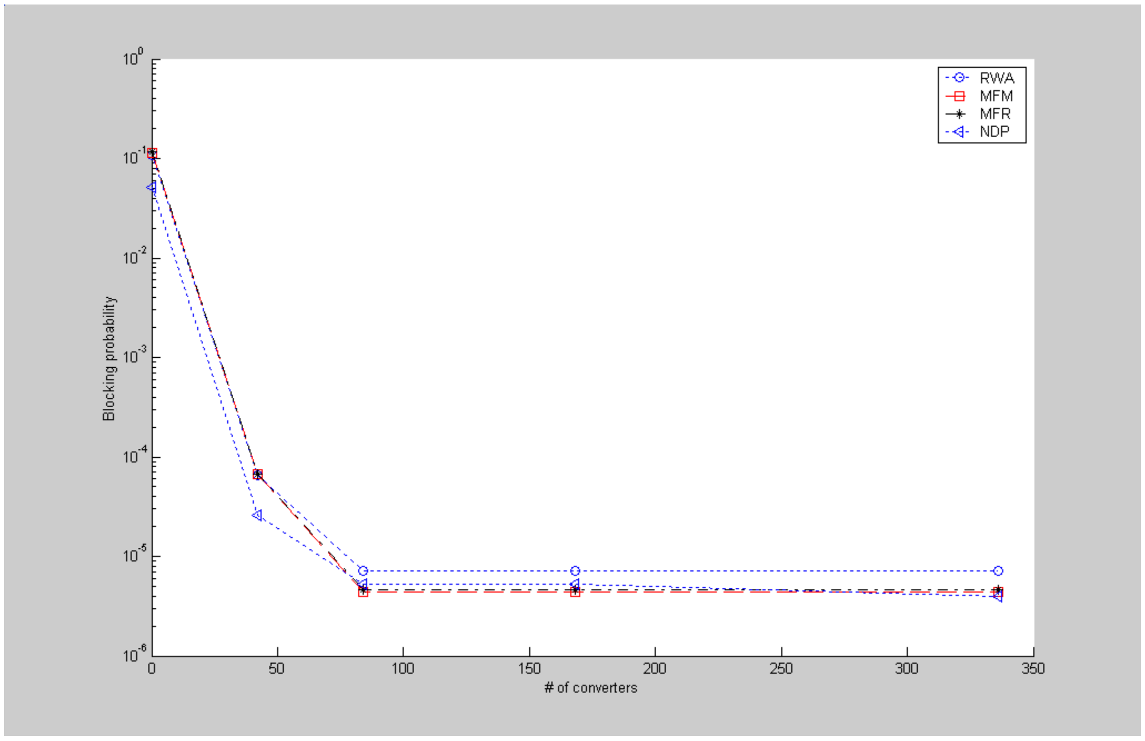


Figure 5.17: Blocking probability of random and proposed wavelength assignment algorithms vs. number of converters uniformly distributed at all nodes. Traffic rate per node is $1*10^6$ packets/sec and a burst is composed of 20 packets

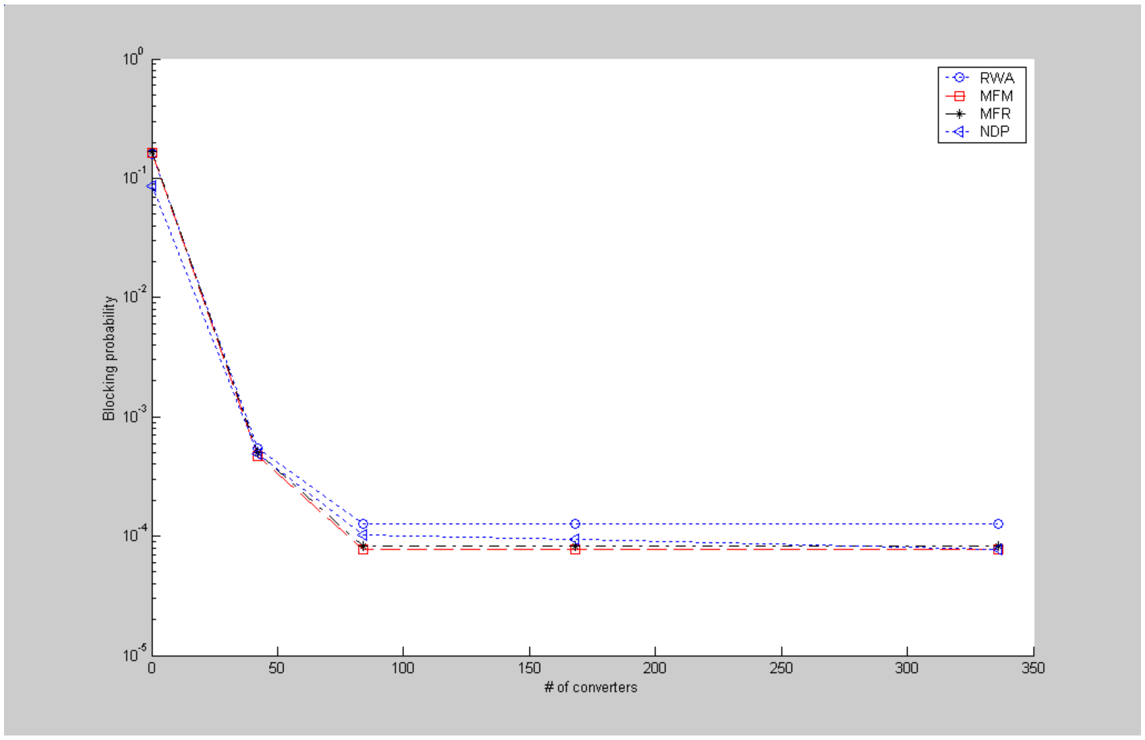


Figure 5.18: Blocking probability of random and proposed wavelength assignment algorithms vs. number of converters uniformly distributed at all nodes. Traffic rate per node is 1.5*10^6 packets/sec and a burst is composed of 20 packets

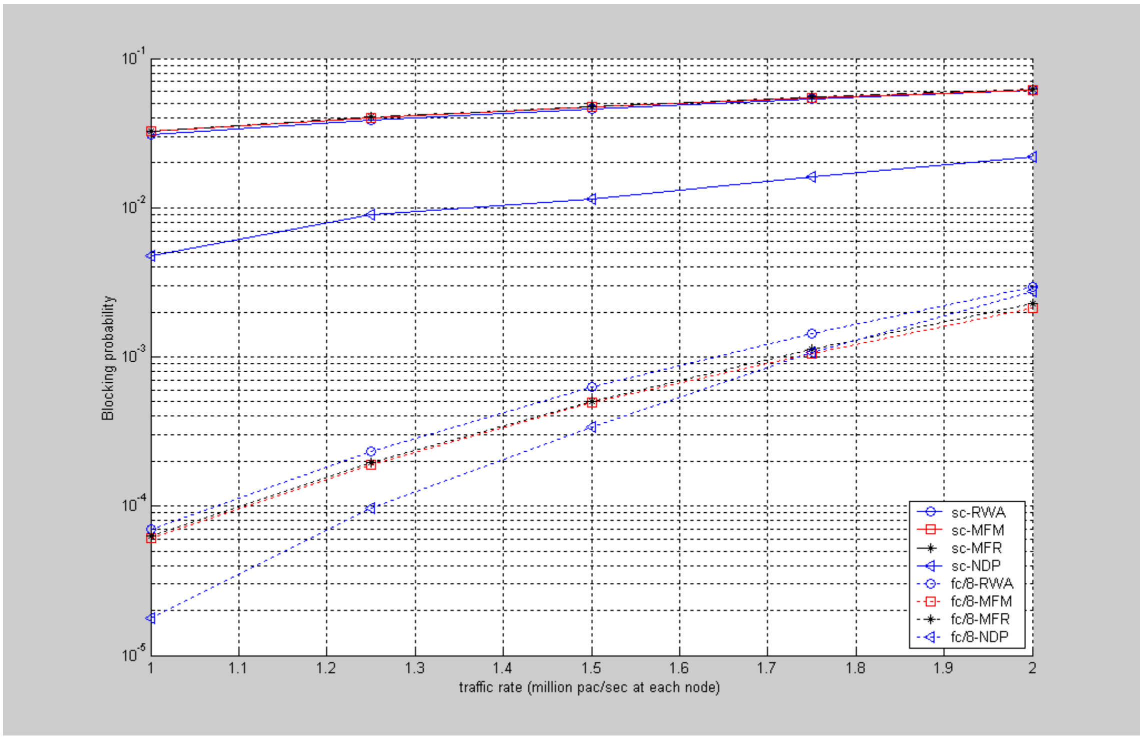


Figure 5.19: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 10 packet bursts and both full sparse conversion (sc) and fc/8 uniform assumption

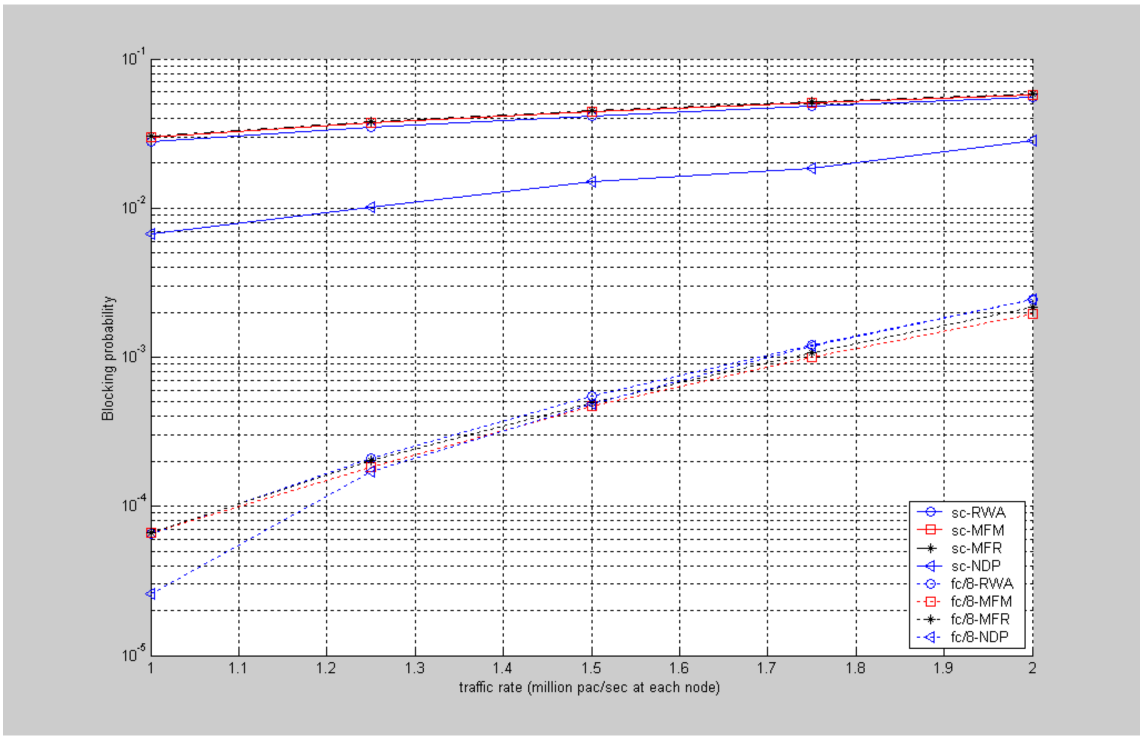


Figure 5.20: Blocking probability of random and proposed wavelength assignment algorithms vs. traffic rate for 20 packet bursts and both full sparse conversion (sc) and fc/8 uniform assumption

Moreover, we also tested performance of the policy obtained at a specific traffic rate also over all other network loads. The simulation is carried out under fc/8 assumption, where we can observe the efficiency of wavelength assignment algorithm clearly than other assumptions with more wavelength converters. Moreover, the results are for 10 packet bursts and F feature vector. Table 5.1 displays average blocking probabilities of the selected policies over all traffic rates. It is obviously observed that policies obtained at high traffic loads improve the performance for lower traffic rates, actually more than their own policies. This is due to the reason that the policy obtained at high traffic loads trains itself for tougher cases, i.e. a better training environment. Conversely, policies trained at lower network loads do not perform well at high traffic rates.

Table 5.1: Average blocking probability of policies obtained at a specific traffic rate simulated over all network loads. Results are for 10 packet bursts and F feature vector

fc/8 (*pac/sec*):	**$1.00*10^6$**	**$1.25*10^6$**	**$1.50*10^6$**	**$1.75*10^6$**	**$2.00*10^6$**
$1.00*10^6$	1,78E-05	1,06E-04	4,18E-04	1,26E-03	3,21E-03
$1.25*10^6$	1,78E-05	9,76E-05	3,78E-04	1,18E-03	3,03E-03
$1.50*10^6$	1,46E-05	8,37E-05	3,40E-04	1,06E-03	2,80E-03
$1.75*10^6$	1,57E-05	9,43E-05	3,53E-04	1,08E-03	2,83E-03
$2.00*10^6$	1,33E-05	8,80E-05	3,45E-04	1,07E-03	2,75E-03

Chapter 6

Conclusions

Several heuristic methods are defined for wavelength assignment algorithms in optical networks. However, most of these heuristics end up with unfair utilization of wavelengths in OBS networks, since they are not designed for link-based wavelength assignment approach of OBS protocols. Interestingly, usually the worst performing of all heuristics, random wavelength selection algorithm, spreads the utilization of the wavelengths uniformly, and results with better performance than the others for optical burst switching. Considering this observation, it is obvious that there is a need for designing intelligent wavelength assignment algorithms for OBS networks. Although many research communities tackle high blocking probability problem of OBS protocols in space and time domain, there is not so much significant study from a wavelength assignment point of view.

At this thesis work, we propose two heuristic algorithms, especially designed for one-way reservation link-based wavelength assignment structure of OBS networks. Heuristic wavelength assignment algorithms significantly improve over conventional random wavelength assignment algorithm.

Furthermore, under the assumptions like memoryless interarrival and fixed holding times, wavelength assignment problem is considered as a Semi-Markov Decision Process (MDP). Therefore, minimizing the average number of bursts blocked per unit time is formulated as a stochastic dynamic programming problem. Due to large size of the problem, neuro-dynamic programming (reinforcement learning) implementation is done for wavelength assignment algorithms in optical burst switching networks. However, the nice picture suffers from a difficulty: introduction of link-based wavelength assignment approach. As a result, decomposition approach which has weak convergence guarantees is used to train the parameter vector θ. Nevertheless, obtained policies result with improved performances.

Simulation results show that NDP techniques can be applied to wavelength assignment problem in OBS networks. It is seen that smaller average blocking probabilities than those of all heuristic algorithms can be obtained. On the other hand, under some circumstances for the extracted feature, NDP method cannot improve beyond proposed heuristics. This is mainly due to the fact that NDP can suffer from learning parameter selection, such as learning constants and cost values. The adjustment of learning parameters to optimal values, and many other relevant feature extractions from network state are promising research subjects.

The performance of proposed wavelength assignment algorithms is investigated over varying converter numbers at network nodes using a pool structure, varying burst lengths and varying traffic load. It is observed that very approximate blocking probabilities of full wavelength conversion can be achieved with less number of converters for proposed heuristic wavelength assignment algorithms. Moreover, the results show that sparse conversion does not improve blocking probability very much, and it is not appropriate for OBS networks. When the dropping rate is taken into consideration over a packet based approach, it is seen that lower dropping rates are obtained for smaller burst lengths.

Performance of the NDP policy obtained at a specific traffic rate is tested also over all other network loads. As expected, the policies obtained at high traffic loads improve the performance for lower traffic rates, actually more than their own policies.

Another important contribution of the study is an event-driven packet-based OBS simulator written in Visual C++ environment. Well-known simulator programs are not capable to implement novel protocols of optical burst switching networks. Due to its resilient structure, the written simulator has many features that make it adjustable for implementation of various wavelength assignment algorithms over varying network conditions.

6.1 Some Interesting Future Directions

The following aspects of the subject I believe are essential and worth exploring in the future:

- Further research on neuro-dynamic programming, especially on feature extraction and on optimizing parameters such that learning constant, cost values, burst length, number of wavelength converters at each node etc. over different network topologies for different traffic characterizations
- Further investigation of neuro-dynamic programming of wavelength assignment algorithms over networks with multi-fiber links, fiber delay lines, and multi/deflection routing capability
- Further study on neuro-dynamic programming in order to introduce adaptive routing methods in wavelength assigment procedure
- Further research on neuro-dynamic programming of routing and wavelength assignment algorithms which takes service differentiation (QoS) into account

References

[1] K. G. Coffman, and A. M. Odlyzko. “The size and growth rate of the internet,” unpublished technical report, available at
http://www.research.att.com/amo/doc/networks.html, 1998

[2] P.E. Green, “Fiber Optic Networks,” Prentice-Hall, 1993

[3] R. Ramaswami, K. Sivaraja, “Optical Networks: A Practical Perspective,” Morgan Kaufmann Publishers Inc., 1998

[4] M. Yoo and C. Qio, “Optical Burst Switching(OBS) – A New Paradigm for an Optical Internet,” J. High Speed Networks (JHSN), vol. 8, no. 1, pp. 69-84, 1999.

[5] C. Qiao, “Labeled Optical Burst Switching for IP-over-WDM Integration,” IEEE Communications Magazine, September 2000, pp. 104-114

[6] M. Yoo and C. Qiao. “Just-enough-time (JET): a high speed protocol for bursty traffic in optical networks,” In IEEE/LEOS Tech. for a Global Information Infrastructure, August 1997

[7] D. P. Bertsekas and J. N. Tsitsiklis, “Neuro-Dynamic Programming,” Belmont, MA: Athena Scientific, 1996.

[8] A. Erramilli, O. Narayan, and W. Willinger, “Experiemental queueing analysis with long-range dependent packet traffic,” IEEE/ACM Transactions on Networking, 4(2):209-223, 1996.

[9] V. Paxon and S. Floyd, “Wide area traffic: the failure of Poisson modeling,” IEEE/ACM Transactions on Networking, 3(3):226--244, 1995.

[10] G.C. Hudek and D.J. Muder. “Signaling analysis for a multi-switch all-optical network,” In Proceedings of Int'l Conf. on Communication (ICC), pages 1206-1210, June 1995.

[11] ITU-T Rec. I.371. “Traffic control and congestion control in B-ISDN.” Perth, U.K. Nov. 6-14, 1995.

[12] C. M. Gauger. “Performance of converter pools for contention resolution in optical burst switching,” Proceedings of the SPIE Optical Networking and Communications Conference (OptiComm 2002), Boston, July 2002

[13] Dimitri P. Bertsekas. “Dynamic Programming and Optimal Control,” Athena Scientific, Belmont, Massachusetts, 1995. Volumes 1 and 2.

[14] Richard Bellman. “Dynamic Programming,” Princeton University Press, Princeton, NJ, 1957.

[15] Dimitri P. Bertsekas. “Dynamic Programming: Deterministic and Stochastic Models,” Prentice-Hall, Englewood Cliffs, NJ, 1987.

[16] Richard S. Sutton. “Learning to predict by the method of temporal differences,” Machine Learning, 3(1):9-44, 1988.

[17] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. “Neuron-like adaptive elements that can solve difficult learning control problems,” IEEE Transactions on Systems, Man, and Cybernetics, SMC-13(5):834-846, 1983.

[18] Ronald J. Williams and Leemon C. Baird, III. “Analysis of some incremental variants of policy iteration: First steps toward understanding actor-critic learning systems,” Technical Report NU-CCS-93-11, Northeastern University, College of Computer Science, Boston, MA, September 1993.

[19] Satinder Pal Singh. “Learning to Solve Markovian Decision Processes,” PhD thesis, Department of Computer Science, University of Massachusetts, 1993. Also, CMPSCI Technical Report 93-77.

[20] Peter Dayan. “The convergence of *TD*(λ) for general λ. Machine Learning, 8(3), 341-362, 1992.

[21] Peter Dayan and Terrence J. Sejnowski. *TD*(λ) converges with probability 1. Machine Learning, 14(3), 1994.

[22] Christopher J. C. H. Watkins. Learning from Delayed Rewards. PhD thesis, King's College, Cambridge, UK, 1989.

[23] Christopher J. C. H. Watkins and Peter Dayan. *Q*-learning. Machine Learning, 8(3):279-292, 1992.

[24] John N. Tsitsiklis. “Asynchronous stochastic approximation and *Q*-learning,” Machine Learning, 16(3), September 1994.

[25] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. “On the convergence of stochastic iterative dynamic programming algorithms,” Neural Computation, 6(6), November 1994.

[26] Jing Peng and Ronald J. Williams. “Incremental multi-step *Q*-learning,” In Proceedings of the Eleventh International Conference on Machine Learning, pages 226--232, San Francisco, CA, 1994. Morgan Kaufmann.

[27] Anton Schwartz. “A reinforcement learning method for maximizing undiscounted rewards,” In Proceedings of the Tenth International Conference on Machine Learning, pages 298-305, Amherst, Massachusetts, 1993. Morgan Kaufmann.

[28] Sridhar Mahadevan. “To discount or not to discount in reinforcement learning: A case study comparing *R*-learning and *Q*-learning,” In Proceedings of the Eleventh International Conference on Machine Learning, pages 164-172, San Francisco, CA, 1994. Morgan Kaufmann.

[29] Sridhar Mahadevan. “Average reward reinforcement learning: Foundations, algorithms, and empirical results,” Machine Learning, 22(1), 1996.

[30] Tommi Jaakkola, Satinder Pal Singh, and Michael I. Jordan. “Monte-carlo reinforcement learning in non-Markovian decision problems,” In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, Advances in Neural Information Processing Systems 7, Cambridge, MA, 1995. The MIT Press.

[31] P. Marbach, O. Mihatsch, and J. N. Tsitsiklis, “Call admission control and routing in integrated services networks using neuro-dynamic programming,” IEEE JSAC, vol. 18, pp. 197-208, February 2000.

[32] J. N. Tsitsiklis, and B. Van Roy, “Average cost temporal-difference learning,” MIT, Cambridge, MA, Lab. Inform. Decision Syst. Report LIDS-P-2390, May 1997.

[33] Serkan Yesildag. “Dynamic routing and wavelength assignment in wavelength assignment in wavelength-division multiplexed (WDM) optical networks using neuro-dynamic programming,” MSc thesis, Department of Electrical and Electronics Engineering, Bilkent University, July 2001.

[34] G. J. Tesauro, “Practical issues in temporal difference learning,” Machine Learning, vol. 8, pp. 257-277, 1992.

[35] S. Singh and D. P. Bertsekas, “Reinforcement learning for dynamic channel allocation in cellular telephone systems,” Advances in Neural Information Processing Systems 9, pp. 974-980, 1997.

[36] J. Tesauro, “Practical issues in temporal difference learning,” Machine Learning, vol. 8, 1988.

[37] W. Zhang and T. G. Dietterich, “High performance job-shop scheduling with a time-delay *TD*(λ) network,” Advances in Neural Information Processing Systems 8, pp. 1024-1030, 1996.

[38] J. N. Tsitsiklis, and B. Van Roy, “On average versus discounted reward temporal-difference learning,” MIT, Cambridge, MA, Lab. Inform. Decision Syst. Report, March 1999.

[39] J. N. Tsitsiklis, and B. Van Roy, “An analysis of temporal-difference learning with function approximation,” IEEE Trans. Automatic Control, vol. 42, pp. 674-690, May 1997.