Improved Learning Automata Applied to Routing in Multi-Service Networks

Philip John Aranzulla

School of Engineering
University of Durham

September 2000

A thesis submitted for the degree of Doctor of Philosophy (Ph.D.)
of the University of Durham

Philip John Aranzulla

Improved Learning Automata Applied to Routing in Multi-Service Networks

Ph.D. 2000.

Abstract

Multi-service communications networks are generally designed, provisioned and configured, based on source-destination user demands expected to occur over a recurring time period. However due to network users' actions being non-deterministic, actual user demands will vary from those expected, potentially causing some network resources to be under-provisioned, with others possibly over-provisioned. As actual user demands vary over the recurring time period from those expected, so the status of the various shared network resources may also vary. This high degree of uncertainty necessitates using adaptive resource allocation mechanisms to share the finite network resources more efficiently so that more of actual user demands may be accommodated onto the network. The overhead for these adaptive resource allocation mechanisms must be low in order to scale for use in large networks carrying many source-destination user demands.

This thesis examines the use of stochastic learning automata for the adaptive routing problem (these being adaptive, distributed and simple in implementation and operation) and seeks to improve their weakness of slow convergence whilst maintaining their strength of subsequent near optimal performance. Firstly, current reinforcement algorithms (the part causing the automaton to learn) are examined for applicability, and contrary to the literature the discretised schemes are found in general to be unsuitable. Two algorithms are chosen (one with fast convergence, the other with good subsequent performance) and are improved through automatically adapting the learning rates and automatically switching between the two algorithms. Both novel methods use local entropy of action probabilities for determining convergence state. However when the convergence speed and blocking probability is compared to a bandwidth-based dynamic link-state shortest-path algorithm, the latter is found to be superior.

A novel re-application of learning automata to the routing problem is therefore proposed: using link utilisation levels instead of call acceptance or packet delay. Learning automata now return a lower blocking probability than the dynamic shortest-path based scheme under realistic loading levels, but still suffer from a significant number of convergence iterations. Therefore the final improvement is to combine both learning automata and shortest-path concepts to form a hybrid algorithm. The resulting blocking probability of this novel routing algorithm is superior to either algorithm, even when using trend user demands.

The main thesis conclusion is that although stochastic learning automata can self-organise their action probabilities to produce good overall routing performance by effectively processing the full set of path possibilities, they thrive when there is a deterministic aid to their action probability convergence.

Declaration

I hereby declare that this thesis is a record of work undertaken by myself, that it has not been the subject of any previous application for a degree, and that all sources of information have been duly acknowledged.

Acknowledgements

I firstly wish to express my thanks to my academic supervisor, Professor Phil Mars, for his work in obtaining the research grants which funded this research, and for his oversight throughout the period of study.

I also acknowledge and am thankful for the funding of this research, a scholarship from St. Aidan's College, and an EPSRC 'CASE' award from Cable & Wireless (formerly Anite Networks). I am very grateful for the guidance provided by my industrial supervisor, Dr. Dennis Nyong, whose contributions I have found very instructive.

Thanks must also go to my fellow researchers during my time in Durham, and in particular Dr. Jonathan Reeve whom I found to be a great help during my time there.

Finally, I am most grateful for the continual support given to me by my mother, especially during the final stages of this undertaking.

Table of contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction to multi-service networks

Historically, networks have been categorised into one of two types: either a voice network or a data network. A company would generally have two separate physical networks connecting its sites together, one carrying its internal voice calls, with the other carrying its data traffic. The voice network would have Private Branch Exchanges (PBXs) located at each company site, with these being connected together by leased lines provided by a Public Services Telecommunications Network (PSTN) company.

Transporting a voice call involves digitising or encoding the analogue voice trace, transmitting the digital information, and decoding the analogue voice waveform at the destination. The basic Coder Decoder (CODEC) used produces a constant 64 kb/s train of information. Each voice call carried by the company voice network would be allocated a specific time slot of the traversed leased line, each time slot comprising a 64 kb/s transmission capacity. It is during the call set-up phase that the required time slot on the leased line is reserved by the call, and if the leased line capacity is being fully consumed by the existing calls then the PBX breaks out that call request on to the PSTN as a standard dial-out voice call. This system ensures that each voice call receives sufficient network resources for its data to be fully carried within the voice traffic delay requirements; and due to the network resource reservation mechanism, no other calls' traffic can interfere with this call's traffic and so degrade its Quality of Service (QoS).

The data network operated differently, being packet switched rather than circuit switched as was the voice network. As data applications are generally bursty in nature rather than constant in their traffic transmission as is a voice CODEC, so their traffic is encapsulated in small packets of information, the data network switching these packets rather than the whole call's traffic. This ensures a higher statistical multiplexing gain, meaning that a higher number of data calls can be carried by the links as one call's silent period can be used for another call's data transmission.

This type of network is termed a best-effort data network as no separation of differing flows' traffic occurs, so that under congestion conditions all flows are affected and all may lose data. This situation is acceptable to data applications which do not require delay guarantees, as the transport layer would detect the data loss after a timer expiry and retransmit the data.

More recent developments in network technology have produced single physical networks which can handle both types of traffic, these being termed integrated or multi-service networks. These networks can provide differing degrees of QoS according to the application requirements, and so make the most efficient use of the expensive Wide Area Network (WAN) link resources, these generally being the greatest component of the total network cost. By providing the required bandwidth and delay characteristics for applications such as voice which need strict QoS from the network, together with little or no QoS for best-effort data traffic, as well as many possible degrees of QoS strictness in between, all application types can be efficiently accommodated and transported by the single physical network infrastructure. The technology used for such networks is generally one of two types: Asynchronous Transfer Mode (ATM) or Internet Protocol (IP) based.

1.2 Multi-service networks using ATM technology

Asynchronous Transfer Mode (ATM) technology arose and was driven mainly from the PSTN carrier perspective. As such it has been used for multi-service PSTN carrier links and networks, as well as being used for corporate WAN links. Previously PSTNs had carried data based on single or multiples of 64 kb/s connections combined together to form larger aggregate pipes. As the PSTN backbone had moved from analogue to digital exchanges, so local Integrated Services Digital Network (ISDN) ports at customer sites would allow that customer to transmit their digital data directly onto the network at 64 kb/s or multiples thereof. Therefore the PSTN network could carry both data and voice application traffic with the constraint that the data channels or flows had to be in multiples of 64 kb/s.

Coming from this circuit-switched perspective, one of the drivers of ATM was to have a network which would allow greater statistical multiplexing of the traffic, whilst still separating individual 'active' flows so that no existing flow's QoS would be degraded by interference from other flows. Like traditional data networks, the application data is packetised (in this case in fixed 53 byte ATM cells [1]) and the various applications' data packets are multiplexed and carried over the physical links. However unlike traditional data networks, the various data flows are logically separated into virtual channels, and by having separate buffers at switch output ports on a virtual channel basis, the flows are segregated and the network's QoS for a flow cannot be degraded by interference from another flow.

As some data flows may be quite small in their WAN link bandwidth requirements, so many more simultaneous data flows may arise than with voice calls. In order to properly manage the network so that it is scaleable up to large PSTNs, there is the need to aggregate these many flows together so that network management functions operate on multiples of

simultaneous traffic flows. ATM provides a two level hierarchical structure for aggregating micro-flows into higher level network flows [2]. At the lower level, end-to-end Virtual Channel Connections (VCCs) are instantiated in the network [3], these being separated from other VCCs by utilising a scheduling buffer unique to the VCC at the output port for each transit node. Micro-flows for the same source to destination are aggregated onto the VCC by the layer 3 IP packets being segmented and encapsulated into ATM cells at the IP to ATM convergence layer, this being termed the ATM Adaptation Layer (AAL). VCCs form the logical network route topology over which the micro-flows can be routed by edge layer 3 IP routers. The higher level aggregation occurs with the formation of Virtual Path Connections (VPCs). These traverse one or more physical links, and form a logical network topology over which the VCCs are set up. The size of these logical network links is specified at set-up time, and is used by the Call Acceptance Control (CAC) function when new VCC set-up requests to traverse this VPC arrive. If there exists sufficient resources over a VPC to accommodate a VCC request across it, the CAC for the VPC accepts the request; otherwise the CAC rejects or blocks the VCC request.

Two main types of VCC are defined: Constant Bit Rate (CBR) and Variable Bit Rate (VBR) VCCs. CBR VCCs are for application types generating either a deterministic data stream with predictable packet sizes and interpacket intervals, or for reserving a pre-defined amount of bandwidth for aggregates of micro-flows. Rather than reserving a pre-specified amount of bandwidth as with CBR, the CAC for VBR VCCs generally reserves an amount of bandwidth between the average and the peak, computed using burst characterisation parameters of the application source. VBR VCCs are used for bursty data applications, where rather than reserving the peak required bandwidth, a smaller value is reserved in order to obtain greater statistical multiplexing in the VPC.

CBR VCCs are ideal for use by voice CODEC applications, as these generally generate a constantly sized application level packet and constant inter-packet time. Other single data source types may be better served by VBR VCCs, thus allowing for more efficient usage of the VPC's bandwidth. Most data sources require only best-effort network service: these sources can be conveniently served by either CBR or VBR VCCs, as the higher level (transport level or layer 4) protocols (e.g. TCP) take care of end-to-end flow control under VCC congestion conditions. Another type of VCC, the Available Bit Rate (ABR) VCC, has also been defined. This less used type seeks to allocate unused VPC bandwidth for best-effort traffic, throttling it back as the spare VPC bandwidth is allocated to new CBR or VBR VCCs. As this concept is similar to the layer 4 functionality, so the benefit of ABR VCCs lies in the more efficient usage of spare VPC capacity. However a similar effect to ABR VCCs occurs when setting up either CBR or VBR VCCs to fully utilise VPC bandwidth together with policing mechanisms to ensure the maximum user demand doesn't exceed the stated VCC

bandwidth requirements. The best-effort traffic sources are allowed to use these VCCs, and the layer 4 flow control mechanisms regulate the traffic sources to use all the available bandwidth within the VCC.

1.3 Multi-service networks using IP technology

Internetworking Protocol (IP) [4] technology arose from the need to allow for data interconnectivity between heterogeneous networking technology types. Its use has become synonymous with the Local Area Network (LAN) arena, as the various layer 2 and 1 technologies are combined together into a seamless networking environment by the IP layer. Being a layer 3 protocol in the OSI model, it performs the routing and congestion control functions [5] with the aid of associated protocols.

Whilst it is true that because the IP layer is local to the user so in general ATM networks carry IP traffic, yet IP networks generally refer to networking technologies which utilise the whole suite of protocols associated with IP, these protocols going up to the application layer (layer 7 in the OSI model), and down to the data-link layer (layer 2 in the OSI model). The reader is referred to appendix A for further details on the IP protocol suite and associated planning and design issues. Due to IP networks providing only best-effort service, the WAN has used other network technologies such as ATM or Frame Relay. However with the advent of IP QoS mechanisms in the IP network equipment, the possibility of pure IP even at the carrier level is fast becoming a reality.

These IP QoS mechanisms utilise certain fields in the IP packet header for the packet's class identification so that QoS differentiation of traffic flows becomes possible. In IP version 4 the 3 precedence bits in the Type of Service field are used, giving a possible 8 different classes of service. The improved IP version 6 [6] has a larger flow label field of 24 bits, so that more than 16 million differing classes of service can be defined.

The best-effort service occurs through the use of a single First In First Out (FIFO) output queue, which stores packets under link congestion conditions, forwarding them on in order of arrival. This basic queuing and scheduling method is unsuitable for use in multi-service networks for whilst packets from bursty traffic sources may not necessarily be dropped, yet the high delay variation possible when traversing the buffer would seriously affect the time-sensitive traffic flows.

By having multiple queues at an output port, served by a scheduling mechanism which empties each queue in turn, the differing classes' traffic can be separated and handled differently. IP packets enter the router and after being routed to the required output port, are

placed into the buffer corresponding to the class indicated in the IP packet header. The servicing of each class queue is handled by the scheduling mechanism, an established method being the Weighted Round Robin (WRR) algorithm [7]. Bandwidth is effectively allocated to each aggregate class buffer flow by configuring WRR to spend a certain percentage of its scheduling time servicing one buffer, another percentage another buffer, and so on. The more recent Weighted Fair Queueing (WFQ) algorithm [8] is an improvement in that it allows the re-use of remaining scheduling time by other class buffers when a class buffer is empty.

Unlike the per-VCC level flow segregation of ATM networks, these IP buffers are aggregate class buffers, with many micro-flows possibly using a particular buffer. Therefore in order to ensure misbehaving micro-flows do not adversely affect the network QoS to other flows of the same class, strict policing mechanisms at the network edges are required. Having done so, soft guarantees can be given to flows (because of the shared buffer resources), rather than hard QoS guarantees which ATM can provide. End-to-end network performance is improved by using the Random Early Detection (RED) mechanism [9]. This mechanism seeks to avoid cases of recurrent network congestion prevalent with TCP traffic flows (see appendix 1 for further details), in order to utilise network resources more efficiently, so allowing the network to provide a more consistent response or QoS to users and applications. The algorithm probabilistically discards IP packets before the onset of chronic congestion in order to cause the TCP engine at the micro-flow source to effectively slow its transmission.

Although utilising shared network resource pools at transit nodes, end-to-end resource reservation is facilitated by the Resource reSerVation Protocol (RSVP) [10] which reserves network resources along a source-destination route according to the source's requirements. True QoS separation of flows also requires the possibility of using one route for a class' traffic and another for another class' traffic for the same source-destination pair. This ability is provided by MultiProtocol Label Switching (MPLS) [11] which can provide route separation down to the micro-flow level, with RSVP reserving the required resources for bundles or single MPLS flows at transit nodes.

All these QoS mechanisms operating together form an architecture for providing multi-service requirements for heterogeneous traffic sources using IP networks.

1.4 Planning, provisioning and allocation of network resources

The planning and design function of multi-service networks, be they IP or ATM based, centres on some form of user demand modelling for each application type using the network. By combining the number of expected concurrent sessions of an application, and the source and destination locations for each traffic flow, together with the expected traffic demand for

each application session instance, an expected aggregate user demand for each required logical link can be derived. When mapping these logical links onto an existing physical topology via routes produced from a routing algorithm (due to resizing an existing network in the case of a network change or upgrade) the required physical link sizes that the logical links traverse are obtained. In cases of a greenfield site (i.e. where the network is to be designed and installed from scratch) the physical network topology generated will be dependent on the node or user population sub-set concentration or location. The resulting physical link sizing is again based on the aggregate requirements of the multiplexed expected traffic logical links.

With the network planning stage being completed, the physical network is provisioned according to the expected user demand-based design. This provisioning process not only involves the installation of the physical network resources such as the switching and router equipment together with the associated connecting links, but their configuration as well. This configuration allows for logical links with heterogeneous QoS requirements to be multiplexed over the same physical links, there being a separating mechanism effective over the link to ensure that misbehaving sources do not compromise the QoS of other multiplexed flows. This separating mechanism may involve separate class buffers and scheduling in the case of either IP or ATM network technology.

Finally there is the actual allocation of these provisioned network resources which may occur on a per-demand basis. There are two associated functions or mechanisms linked with resource allocation: Call Acceptance Control (CAC) and call routing. The CAC function works in conjunction with the routing function in allowing new call requests access to the network resources needed to maintain the call's QoS requirements, or downgrading the call's traffic QoS tag if insufficient network resources are available. Both ATM and IP (using RSVP) technology CACs may also deny a new call or connection request any resources in cases of network congestion, so blocking the call. The routing function routes the call's traffic through the network from the source to the destination, the CAC function having reserved the resources along the route for the call's duration.

However in cases of more static provisioning, the CAC function is more akin to a policing function in that the aggregate bandwidth for an application type's source to destination flow is already reserved or allocated based on expected application demands. The CAC or policing mechanism then ensures that there are not too many such application micro-flows routed over and so using that reserved shared resource, blocking or downgrading the QoS of new requests at the edge of the network. Implemented in an ATM network, such an allocation policy would require pre-established VPCs and VCCs (otherwise known as Permanent Virtual Circuits or PVCs), each VCC carrying multiple application micro-flows. Implemented in an IP network, such an allocation policy would require pre-established transit node class buffer sizing and scheduling algorithm configuration (this allocation policy being

termed Differentiated Services or DiffServ [12]). Such network resource allocation can still be varied according to network conditions with resizing of the reserved aggregate bandwidth sizes, as well as recalculation of the routes through the network if necessary. To simplify this procedure in an ATM network, fairly static SVCs could be used instead of PVCs, these being resized as necessary or even re-routed or reformed over a different route. With an IP network, aggregate class buffer sizes and scheduling algorithm time can be reconfigured using RSVP, this therefore operating over multiple MPLS flows through that transit node [13]. The MPLS routes themselves can also be reconfigured as required.

Weaknesses with the static allocation approach centre on the fact that non-deterministic user actions, which result in demands for network resources, cannot be predicted accurately. This leads to a mismatch between expected user demands for which the network has been provisioned, and actual user demands that the provisioned network attempts to satisfy. Whilst most network designs may over-provision resources in an attempt to solve this problem as well as building in some traffic demand growth expectation, unless there is a high degree of over-provisioning there will generally occur hot-spots in the network, with some areas being under-provisioned and other having spare capacity. As user demands vary with time, so the hot-spot network areas may also move over time, meaning that a large number of higher capacity links may be required to eliminate the problem. Of course, this situation is exacerbated when provisioning for network failure conditions.

By allowing some dynamism in the resource allocation policy, user demands can be redistributed as necessary in order to ease network congestion hot-spots, by consuming some of the spare capacity found at other parts of the network. Therefore all the network resources are used more efficiently, resulting with a lower provisioning requirement. Whilst it is true that differing static allocation policies may be put in place at different times of the day or under network failure conditions to more accurately allocate resources to expected user demands, yet significant mismatches may still occur. Therefore fully dynamic allocation policies are investigated in this thesis, these resulting with the most efficient use of shared network resources. As this area is the more combinatorially complex, it is believed that contributions in this area will be applicable and useful in the less complex and more static allocation policy scenarios.

1.5 Dynamic routing algorithms

A dynamic resource allocation policy requires the dynamic operation of the combined CAC and routing functions. The routing function calculates a route from the source to the destination, with the CAC function accepting the call request if there remain sufficient unused

network resources over the calculated route. This set of dynamic calculations may be performed on an aggregated call basis, or down to the micro-flow level call basis. When operating in an ATM paradigm, this translates to a SVC environment. When using IP technology this maps directly to the Integrated Services (IntServ) [14] operational environment, but also conceptually to the DiffServ paradigm. This is true when there are multiple route possibilities for that flow's class over partitioned and allocated pools of shared network resources for that class. At that point the combined mechanism of CAC and routing functions can decide over which route to send the call request, the feedback for the decision making process being the status of the shared network resources along that downstream path.

Most dynamic routing algorithms currently used (such as IP's OSPF) re-calculate routes on indications that the network topology has changed, with either links or network equipment going out of service. This is in order to preserve network stability. However it is only with bandwidth-based dynamic routing algorithms that the allocation of the network resources can vary from that matching the expected traffic demands to that matching the actual traffic demands. Such algorithms vary the route set calculated according to the current network state or link loading levels, lower loading levels returning higher available bandwidth indications, this predisposing the route calculation to favour usage of such links. By performing the route calculations at the call level based on the reserved aggregate call level bandwidth rather than current flow level, large swings in link state are avoided, so aiding network stability. The most combinatorially complex network scenario type was chosen (that of user individual calls) in order to assess the proposed novel dynamic routing solutions contained in this work within the most challenging context.

As regarding the routing and multiplexing of multi-service traffic on the same links, the preferred policy is, in general, to reserve some resources exclusively for each traffic class whilst sharing the remainder [15]. The performance of routing algorithms within a multi-service context with shared resources is generally given in terms of the bandwidth blocking probability. This is defined as follows:

$$\text{bandwidth blocking rate} = \frac{\sum_{i \in B} \text{bandwidth}(i)}{\sum_{i \in S} \text{bandwidth}(i)}$$

where B is the set of all blocked call requests, and S the set of all call requests. Previous work has shown that a single routing policy returns comparable performance to a set of routing policies in a shared resource multi-service flow context (i.e. one policy for each service type) [19]. However the blocking probability increases as the traffic intensity for a source remains constant but its bandwidth requirement increases for each instance. As the focus of this work is to improve user perceived network performance by improving the dynamic routing algorithm, therefore any possible clouding effects of heterogeneous traffic sources are

removed by using homogeneous or single class types in the experiments. However the experimental framework is provided for evaluating heterogeneous source traffic scenarios, which because of possible reservation policies which include partitions exclusively used by a source type with other partitions shared between source types, require various scenarios in order to be evaluated.

The initial experimental work contained in this thesis explores the performance of current link-state shortest path based algorithms proposed in the literature. Further work ensues in trying to reduce the associated link state advertisement overhead, whilst still trying to maintain good performance in terms of blocking probability under moving and steady network state conditions.

However the main focus of work covered in this thesis lies in the field of the application of stochastic learning automata to the dynamic routing problem in multi-service networks. Learning automata have been shown to produce near optimal performance after convergence in stationary environment applications [16], and have also been previously applied to the routing function in communications networks [17]. The main work of this thesis seeks to improve their weakness of slow convergence speed whilst retaining their strength of good steady-state performance. These improvements are validated by comparing the resulting performance with that obtained from the proposed novel shortest-path based routing algorithm. Unlike previous studies using learning automata [18, 19], experiments are also performed to show that these improvements hold under dynamic user demand conditions in the network.

1.6 Outline of the thesis

Figure 1 shows the thesis contents breakdown in terms of the chapters' information dependencies. Rather than the work having a single strand of thought and being described in a purely sequential manner, there are two main areas of work that although interrelated may be thought of as distinct work areas.

The first and main area of work is the critical examination and subsequent improvement of stochastic learning automata operation when applied to routing in reservation-based networks such as ATM or IP with QoS features. This body of work is contained in chapters 2, 4, 5, 6 and 7. The second area of work deals with obtaining a dynamic routing algorithm that performs well in either ATM or IP QoS networks and is based on currently used routing algorithms, this then being used on a comparative basis with the resulting learning automata based method. This second area of work is presented in chapter 3, its contents feeding into chapter 5 onwards. The proposed and validated algorithms for IP

QoS networks are modified for use in more regular DiffServ IP based networks in the further work section of chapter 8.

Chapter 2 contains a literature review of stochastic learning automata, including their operation and application to the routing function in networks.

Chapter 3 sets the scene for routing in both ATM and IP QoS networks by examining the role of the Call Acceptance Control function for differing traffic types. After detailing an accepted effective bandwidth calculation method for voice connections, a novel method for effective bandwidth calculation of MPEG sources is presented, with initial results indicating its superior accuracy when compared with previous methods. Various link-state routing algorithms are then examined, with a modified existing algorithm being comparatively evaluated via simulation experiments. This results with a link-state routing algorithm whose performance can be used for comparison with learning automata based methods.

Chapter 4 examines the resulting performance of the various reinforcement algorithms, these being an integral part of learning automata operation. Their performance has been characterised for stationary environment applications, which is where the environment state doesn't change. However the communications network environment is of the non-autonomous non-stationary kind, as its state changes according to the learning automata actions performed. Previous work on applying learning automata to networking problems has simply assumed that the relative performance of the differing reinforcement algorithms when learning automata operate in a stationary environment is the same as when operating in non-autonomous non-stationary environments. However the comparative examination presented in chapter 4 shows this to be an erroneous assumption, with a number of the algorithms thought to produce the best performance being clearly unsuitable for use in applications of non-autonomous non-stationary environment kinds.

Chapter 5 takes the best performing learning automata schemes arising from the work of chapter 4, and seeks to improve their performance. Two differing schemes are taken from the work of chapter 4, one having strengths in network steady-state conditions with the other strengths in moving network state conditions. The performance of both schemes is improved with a novel adaptive mechanism, which adapts the learning rate according to a novel method of characterising the local network state. A different novel adaptive mechanism which seeks to combine the strengths of both schemes highlighted in chapter 4 is then proposed and evaluated, this also showing a performance improvement. The performance of both of these improved learning automata schemes are then compared with that resulting from using the dynamic shortest-path algorithm detailed in chapter 3.

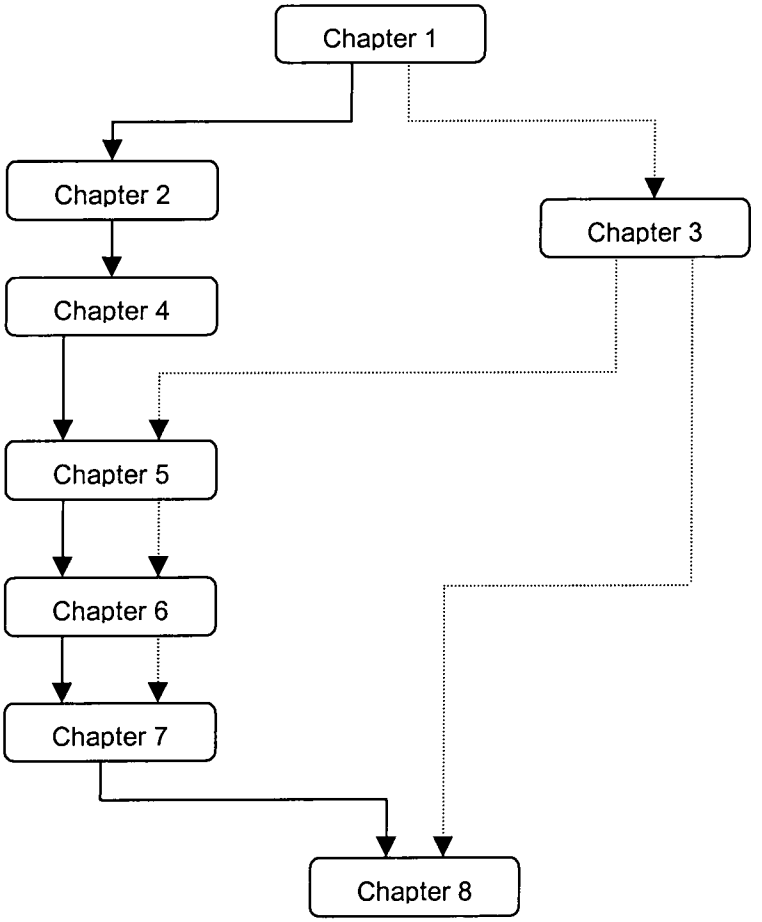


Figure 1: Chapter information dependencies

The work contained in chapter 6 re-examines how learning automata have historically been applied to the routing problem in reservation-based networks, and proposes a novel way of applying them to the problem, the reinforcing feedback being based on link utilisation levels. The resulting performance of the new learning automata routing algorithm is finally compared to that of the dynamic shortest–path algorithm.

Having seen that each of these algorithms is stronger in differing network state circumstances, the work collated in chapter 7 seeks to combine the two algorithms to produce a hybrid routing algorithm. This hybrid includes both a learning automata component, and a shortest-path component, the part that is active at any one time being dependent on the locally perceived network state.

Chapter 8 draws all the strands of findings and conclusions together, also presenting some directions for further work. This further work section proposes modifications of the hybrid algorithm for it to operate in a more regular IP environment, (i.e. a Diffserv paradigm).

2 A Brief Review of Learning Automata

2.1 Introduction

The purpose of the following chapter is to provide the theoretical aspects of learning automata, which will be required when detailing proposed improvements in later chapters. The chapter begins by introducing the basic concepts of an automaton interacting with an environment, and proceeds to detail in a simple and unified manner the various main reinforcement algorithms. It is by using these algorithms that the automaton is able to learn the statistical environment characteristics. The use of different reinforcement algorithms results in different theoretical performance characteristics when operating in stationary random environments, so a tabulation of the various performance groupings is given.

Finally, the means by which learning automata have previously been applied to the routing problem in both circuit and packet-switched networks is briefly shown, together with the observed resulting performance characteristics.

2.2 Basic Concepts

Learning can be defined as any relatively permanent change in behaviour resulting from past experience. Therefore a learning system has the ability to improve its behaviour with time, according to a defined performance measure [20]. A learning automaton can be defined as a decision maker which operates in a random environment, updating its strategy for choosing actions on the basis of the environment's response [16]. The automaton has a finite number of actions and the response of the environment to each action can be either favourable or unfavourable. The automaton's interaction with its environment is shown in Figure 2.

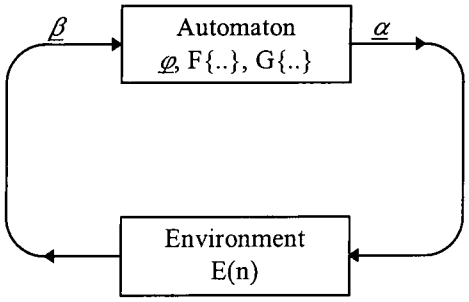


Figure 2: Learning Automata acting in an environment

The automaton is defined by its state set φ, an output or action set $\underline{\alpha}$, an input set $\underline{\beta}$, a transition or updating function F{..}, and an output function G{..}. The automaton can be either stochastic or deterministic; the former's output function G{..} being composed of probabilities based on the environment's response, whilst the latter having a fixed mapping function between the internal state and the function to be performed. Further sub-division of classification occurs when considering the transition or updating function F{..} which determines the next state of the automaton given its current state and the response from the environment. If this is fixed then the result is a fixed structure deterministic or a fixed structure stochastic automaton. However if the updating function is variable, allowing for the transition function to be modified so that choosing the operations or actions changes after each iteration, then the result is a variable structure deterministic or a variable structure stochastic automaton. For this study the particular types used are variable structure stochastic automata, these having the potential of greater flexibility and therefore performance. For such an automaton A at instant n:

$$A(n) = \{ \underline{\alpha}, \underline{\beta}, \underline{p}, T(\alpha, \beta, \underline{p}) \}$$

where we have an action set $\underline{\alpha}$ with r actions, an environment response set $\underline{\beta}$ and a probability set $\underline{p}$ containing r probabilities, each being the probability of performing every action possible in the current internal automaton state. The function T is the reinforcement algorithm which modifies the action probability vector $\underline{p}$ with respect to the performed action and the received response. The new probability vector can therefore be written as:

$$\underline{p}(n + 1) = T\{ \alpha, \beta, \underline{p}(n) \}$$

The environment is defined by E(n):

$$E(n) = \{ \underline{\alpha}, \underline{\beta}, \underline{c} \}$$

where $\underline{c}$ is the penalty set. This is the probability that the action α_i would result in an unfavourable response from the environment, and is defined as:

$$c_i = \Pr\left[\beta(n) = 1 \middle| \alpha(n) = \alpha_i\right]; \quad i = \{ 1, 2, \ldots, r \}$$

This characterises the response of the environment to a performed action, and therefore indicates the desirability of different actions, responding with a penalty signal or penalty weight depending on the environment model. There are three possible environment response models: the P-model responds to an action with a binary signal (either 0 or 1, reward or penalty); the S-model has a continuous response in the region (0, 1) and the Q-model's response is one from a finite set of discrete values in the range of (0, 1). For this study both the P-model and S-model were chosen. The P-model's binary response can be easily generated from the network according to whether a connection attempt had been successful or not, and this is the method that has been traditionally used. However an enhancement presented in later chapters involves the use of the S-model. These matters are further explained in the next main section.

One quantity useful in judging the behaviour of a learning automaton is the average penalty received by the automaton. The average penalty received by the automaton for a given action probability vector is expressed by:

$$M(n) = E\left[\beta(n) \middle| p(n)\right]$$

$$= \sum_{i=1}^{r} p_i(n) c_i$$

If no a priori information is given and the actions are chosen with equal probability, then the average penalty received by the automaton is given by:

$$M_0 = \frac{c_1 + c_2 + \ldots + c_r}{r}$$

The use of the term learning automaton can be justified if the average penalty is made less than M_0 at least asymptotically, such behaviour being called expediency. So an expedient learning automaton performs better than one whose actions are chosen in a purely random manner. An optimal learning automaton would produce the minimum value of $M(n)$. Optimality implies that asymptotically the action associated with the minimum penalty probability is chosen with probability one. Whilst optimality is very desirable it is often not achievable, and so suboptimal performance is aimed for. Such is termed ε-optimality, and implies that the performance of the automaton can be made as close to the optimal as desired.

An absolutely expedient learning automaton has a monotonic decrease in $M(n)$. Absolute expediency implies expediency and ε-optimality in all stationary random environments [21].

Having defined the learning automaton and environment type that will be used, the differing learning automaton classification that occurs in the remainder of this study is based on the reinforcement algorithm employed.

2.3 Reinforcement Algorithms

Classification of such algorithms can be based either on the nature of the function used in the scheme, or on the property exhibited by a learning automaton using the algorithm [20].

The first classification type may be split into two main areas: the linearity of the scheme, and whether the scheme is continuous or discrete in nature. An example of the former area is if $\underline{p}(n + 1)$ is a linear function of $\underline{p}(n)$, then the scheme is termed linear. Schemes involving higher orders of $\underline{p}(n)$ are non-linear, with the final class of algorithm combining aspects of the two and so being a hybrid. As non-linear and hybrid algorithms have given no appreciable improvement over the linear updating schemes [22], only linear algorithms will be used in this study.

As for whether the scheme is discrete or continuous, with a continuous scheme the action probabilities may take any real value between the interval (0, 1), this being limited when implemented only by the floating point precision. A discretised scheme on the other hand discretises the probability space so that the action probabilities may take values only from a finite set in the interval (0, 1).

This second classification type results with broadly two types of learning automaton: those that are absolutely expedient and generally ε-optimal, and those that are ergodic. In general the theoretical proofs given for these properties assume a stationary random environment, so it remains unclear whether these properties hold in non-stationary random environments, a multi-service network corresponding to such an environment, as detailed in chapter 4. However, the properties for each algorithm are given as an aid to classification.

It was thought beneficial to detail the algorithms which follow as the original sources varied in their terminology and method of explanation. The details below are given in a uniform manner and style to ease both their comprehension and the appreciation of differences. All the algorithms have a learning rate parameter, which indicates the possible size of change in the updating of action probabilities.

2.3.1 Standard Algorithms

Standard reinforcement algorithms use the instantaneous environment response β to directly update the action probabilities. According to the algorithm used, either one or both of the environment responses are used in the updating mechanism. The following algorithms follow: LRI, LRP and LRεP, these being the ones mainly used. Other possible algorithms include LIP, LRr, LpP, which are not considered in this study as the former have been shown to be superior [18].

2.3.1.1 Linear Reward Inaction (LRI) Algorithm

The following algorithm only updates the probabilities when receiving a successful response from the environment, keeping the probabilities unchanged for a penalty response. Using either the P-model or S-model response environments, the probabilities are updated in the following manner:

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = [0, 1]$:

$$p_{j\neq i}(n+1) = \big(1 - a\big(1 - \beta(n)\big)\big)p_j(n) \qquad 0 < a < 1$$

$$p_i(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

Therefore when there is a penalty response ($\beta(n) = 1$) on $\alpha(n) = \alpha_i$, the result will be:

$$p_j(n+1) = p_j(n);\ \forall j$$

The LRI scheme is expedient and ε-optimal in stationary random environments, but as the automaton is not ergodic it is possible for it to get stuck in absorbing states. This makes it sensitive to the starting conditions and probabilities, and also to non-stationary environments. Such occurs when action probabilities tend to one, so that if the network state changes, the probability vector may not adapt to the new optimum for a long time.

2.3.1.2 Linear Reward Penalty (LRP) and LRεP Algorithms

This algorithm updates the probabilities for both a successful and unsuccessful response from the environment. Again, using either response environment, the probabilities are updated as follows:

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = [0, 1)$:

$$p_{j\neq i}(n+1) = (1 - a(1 - \beta(n)))p_j(n) \qquad 0 < a < 1$$

$$p_i(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

Penalty on $\alpha(n) = \alpha_i$ $\beta(n) = 1$:

$$p_{j\neq i}(n+1) = \frac{b}{r-1}(1-b)p_j(n) \qquad 0 < b < 1$$

$$p_i(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

where $b = a$.

The LRP scheme is expedient, and the probability vector $\underline{p}(n)$ has been shown to converge in distribution to a normal random variable for small step sizes [16, 23]. The scheme is also ergodic [24], so that this distribution function is independent of the initial probability vector $\underline{p}(0)$. This feature is also advantageous when operating in non-stationary environments such as a communications network, as the automaton does not get stuck in absorbing states and so is better able to track the changing optimum probability vector.

The LRεP scheme is similar to the LRP scheme except that the penalty learning rate is less than the reward learning rate, meaning that (b < a). This changes its behaviour to be both ε-optimal and ergodic. In general the penalty rate is set at one tenth of the reward rate.

2.3.2 Estimator Algorithms

To improve the main limitation of learning automata, that of a slow rate of convergence, a new class of algorithm was proposed [25]. Its novel feature is that it uses history by maintaining estimates of the reward characteristics of the environment, which in turn drive the

updating algorithm to produce a stronger convergence result. This occurs as the probability vector is updated based on both the estimate vector and the current response from the environment. Thus for this class of algorithm, even if a particular action is rewarded, it might be the case that the probability of choosing another action is increased. The family of estimator algorithms has been shown to be ε-optimal in stationary random environments.

2.3.2.1 The Pursuit Algorithm

The pursuit algorithm is a special case of a general estimator algorithm, and is characterised by the fact that it pursues what it reckons to be the optimal action [26]. It is similar in design to the LRI algorithm, except that whereas the LRI algorithm moves the action probability vector in the direction of the most recently rewarded action, the pursuit algorithm moves it towards the action that possesses the highest estimate of reward. Like the LRI algorithm, the pursuit algorithm has been shown to be ε-optimal [27]. Using either response environment, the algorithm is as follows:

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = [0, 1]$:

$$p_{j\neq k}(n+1) = (1 - a(1 - \beta(n)))p_j(n) \qquad 0 < a < 1$$

$$p_k(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

This again causes the following on receiving a penalty indication ($\beta(n) = 1$) on $\alpha(n) = \alpha_i$:

$$p_j(n+1) = p_j(n);\ \forall j$$

For either environment reply, the running estimates are subsequently updated as follows:

$$W_i(n+1) = W_i(n) + (1 - \beta(n))$$

$$Z_i(n+1) = Z_i(n) + 1$$

$$d_i'(n+1) = \frac{W_i(n+1)}{Z_i(n+1)}$$

where p_k is the action probability that has the highest running estimate d_k' of being rewarded; $W_i(n)$ is the number of times the ith action has been rewarded up to n; and $Z_i(n)$ is the number of times the ith action has been selected up to n.

2.3.2.2 The General Estimator Algorithm

The general estimator algorithm is a more complex scheme, with the probabilities being updated as a function of both the reward estimates and the action probability vector [25]. The algorithm is as follows:

For either environment response to $\alpha(n) = \alpha_i$:

$$p_{j\neq i}(n+1) = p_j(n) - a\left[f\left(d_i'(n) - d_j'(n)\right) \cdot \left(S_{ij}(n)p_j(n) + S_{ji}(n)\frac{\left(1-p_j(n)\right)p_i(n)}{r-1} \right) \right]$$

$$p_i(n+1) = p_i(n) + a\sum_{j\neq i}\left[f\left(d_i'(n) - d_j'(n)\right) \cdot \left(S_{ij}(n)p_j(n) + S_{ji}(n)\frac{\left(1-p_j(n)\right)p_i(n)}{r-1} \right) \right]$$

where $S_{ij}(n)$ is an indicator function defined as:

$$S_{ij}(n) = 1 \qquad \text{for } d_i'(n) > d_j'(n)$$

$$= 0 \qquad \text{for } d_i'(n) \leq d_j'(n)$$

and f is a monotonic increasing function, such as 'x' or 'x^3'.

For either environment response, the running estimates are subsequently updated as follows:

$$W_i(n+1) = W_i(n) + (1 - \beta(n))$$

$$Z_i(n+1) = Z_i(n) + 1$$

$$d_i'(n+1) = \frac{W_i(n+1)}{Z_i(n+1)}$$

As can be seen, the updating of the probability vector depends indirectly on the response from the environment as this feedback changes the components of $D(n)$ which affect the sign of $f(....)$ and S_{ij}. If α_i is chosen and $d_i'(n) \leq d_j'(n)$, then an amount proportional to ($p_i(n)$ / (r - 1)) (1 - $p_j(n)$) is added to $p_j(n)$. However if $d_i'(n) > d_j'(n)$, then an amount proportional to $p_j(n)$ is subtracted from $p_j(n)$.

2.3.3 Discretised Reinforcement Algorithms

All algorithms given to date approach the optimal action probability asymptotically, so by discretising the probability space to discrete values in the region [0,1], convergence is speeded up when the optimal action probability is close to unity, as occurs with stationary random environments. This happens by the discretised automaton increasing the probability of choosing that action to the value of unity directly, instead of approaching that value asymptotically as with continuous schemes [28]. The discretisation is termed linear if the allowable values are equally spaced; otherwise it is called non-linear. In general, the discretised version of a continuous algorithm retains the original's property of ε-optimality or ergodicity. Another benefit of discretisation is that the requirements on the system random number generator are reduced as the probability space is now a set of integers, rather than a continuous set. This is due to the contents of the probability space not needing to be stored as real numbers, only their integer index requiring storage. Therefore the random number generator needs to generate an integer within the bounds zero and the number of discrete steps rather than a high precision real number, so that it is as close as can be possible to the continuous case when using digital circuitry.

As with the section on the standard continuous algorithms, other possible algorithms such as DLIP and ADLIP are not given as their performance was found to be less than the ones given below [29].

The details for the following reinforcement algorithms are applicable with the P-model response environment. For the S-model response environment, a number of Δ additions or subtractions may occur according to the magnitude of the reward indicator. However, it should be borne in mind that the reception of a variable environment response really requires the capability of a continuous algorithm to map it to an appropriate reinforcement of the action probabilities. As the size of the minimum granularity is limited when using discretised algorithms, the effective use of an S-model response environment is often not possible. This is not so much the case with the discretised estimator algorithms as the updating function fully takes into consideration the variable nature of the environment response.

2.3.3.1 Discretised Linear Reward Inaction (DLRI) Algorithm

The updating algorithm is fairly similar to that of the continuous case, except that for the two action case the automaton has $(N + 1)$ states, N being an even integer, and associated with the

state s_i is the probability i/N, which represents the probability of the automaton choosing an action [30]. For the r action case the step size Δ, instead of being $1/N$, translates to:

$$\Delta = \frac{1}{rN}$$

The algorithm is as follows for the P-model response environment:

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = 0$:

$$p_{j\neq i}(n+1) = \max\left\{p_j(n) - \Delta, 0\right\}$$

$$p_i(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

Penalty on $\alpha(n) = \alpha_i$ $\beta(n) = 1$:

$$p_j(n+1) = p_j(n);\ \forall j$$

Like its continuous counterpart, the DLRI algorithm has been shown to be ε-optimal in all stationary type environments for the two-action case [30].

2.3.3.2 Discretised Linear Reward Penalty (DLRP) Algorithm

This is similar to the DLRI updating mechanism, except that the penalty environment responses are also utilised, as follows for a P-model response environment:

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = 0$:

$$p_{j\neq i}(n+1) = \max\left\{p_j(n) - \Delta, 0\right\}$$

$$p_i(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

Penalty on $\alpha(n) = \alpha_i$ $\beta(n) = 1$:

$$p_{j\neq i}(n+1) = \min\left\{p_j(n) + \Delta, 1\right\}$$

$$p_i(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

Interestingly, the discretised version of the LRP algorithm has slightly different properties to its continuous counterpart. It has been shown that whilst keeping its ergodic nature, it is also ε-optimal in environments where the lowest penalty probability for an action is less than 0.5 [31]. Moreover, by artificially creating absorbing states (called the ADLRP algorithm) the algorithm loses its ergodicity but becomes ε-optimal in all environments. However it has been noticed at least in non-stationary random environments that absorbing schemes generally perform poorly [17]. Finally, a modification of the DLRP algorithm can be performed (called the MDLRP algorithm) which renders it ergodic and ε-optimal in all random environments. The modification is as follows: if the environment response is a penalty, then there is a 50% chance of decreasing that action probability, and a corresponding 50% chance of increasing it. This modification effectively reduces the penalty response updates, so making it similar to the LRεP algorithm in performance and rationality.

The DLRεP scheme is similar except that a reward results in a multiple number of step changes, and a penalty in a single step change.

2.3.3.3 Discretised Pursuit Algorithm

Like its continuous counterpart, this algorithm has also been shown to be ε-optimal. For either response environment, its updates are as follows [28]:

Reward on $\alpha(n) = \alpha_i$ $\beta(n) = [0, 1)$:

$$p_{j\neq k}(n+1) = \max\left\{p_j(n) - \Delta, 0\right\}$$

$$p_k(n+1) = 1 - \sum_{j\neq i}^{r} p_j(n+1)$$

Penalty on $\alpha(n) = \alpha_i$ $\beta(n) = 1$:

$$p_j(n+1) = p_j(n);\ \forall j$$

with the update of the running estimates being as for the continuous case.

2.3.3.4 Discretised General Estimator Algorithm

The entire family of discretised estimator algorithms have been shown to be ε-optimal as long as the following two properties are met: the property of moderation, and the monotone property [27]. The first property states that the maximum magnitude by which an action probability can decrease per iteration is bounded by $1/rN$. The monotone property indicates absolute expediency, in that an action probability monotonically moves to the optimum one.

There are various modifications of the continuous algorithm to allow its use for the discrete domain. First the value for Δ is slightly modified to make it hold to the property of moderation as follows:

$$\Delta = \frac{1}{rN\theta}$$

where θ represents the largest multiple of Δ that any one component of the probability vector can decrease by in one iteration. It has been stipulated from the change equation below, that θ effectively replaces the term 'a' found in the continuous case [27]. However, if one takes Δ to be the same as for the other discretised schemes which leave out θ, then the analogous continuous case learning rate is actually the full step size ($\theta\Delta$ for this case).

Secondly, the terms $p_i(n)$ and $(1 - p_j(n))\, p_i(n)$ have been dropped completely so that the algorithm approaches its end point directly rather than asymptotically. Thirdly, two new functions are introduced: Rnd(x) rounds x up to an integer, being one of $\{ -\theta, -\theta+1, -\theta+2, \ldots, \theta-1, \theta \}$; and Check takes as inputs the current action probabilities and the allowable change, returning the maximum permissible number of step changes which keeps all action probabilities within the bounds of 0 and 1. It is formally specified as follows:

For any environment response to $\alpha(n) = \alpha_i$, for each action j starting with m:

$$\text{change} = \text{Rnd}\left(\theta f\left(d_i'(n) - d_j'(n)\right) \cdot \left(S_{ij}(n) + S_{ji}(n)\frac{1}{r-1} \right) \right)$$

$$p_j(n+1) = p_j(n) - \Delta\text{Check}\left(p_i(n), p_j(n), \text{change}\right)$$

$$p_i(n+1) = p_i(n) + \Delta\text{Check}\left(p_i(n), p_j(n), \text{change}\right)$$

with the update of the running estimates being as for the continuous case.

2.3.4 Comparisons between Algorithms

The following section attempts to collate the information available for the aforementioned algorithms to provide readily accessible comparisons in type and functionality. Unfortunately, the vast majority of performance indicators available in the literature are for random stationary environments, but these are not relevant in our study as the environment to which the learning automata will be applied is non-stationary, as will be explained in the following section. In fact one of the aims of the following chapter is to give some empirical indications on comparative performance of the various available algorithms for state-dependent non-stationary environments such as a communications network.

Therefore the relevant possible comparative indicators currently available concern the asymptotic properties in stationary environments of these variable structure stochastic automata. These are shown in Table 1.

	ε-optimal	**ergodic**	**ε-optimal and ergodic**
Continuous			
LRI	yes		
LRεP			yes
LRP		yes	
pursuit	yes		
general estimator	yes		
Discrete			
DLRI	yes		
DLRεP			yes
DLRP		yes	yes if $c_{min} < 0.5$
MDLRP			yes
ADLRP	yes		
discretised pursuit	yes		
dis. gen. estimator	yes		

Table 1 - Summary of the asymptotic properties of various reinforcement algorithms

Ideally, since the environment type used will be non-stationary, the property of ergodicity is important. If this can be coupled with ε-optimality and a fast convergence rate, then such an algorithm should be the ideal for use in this environment type.

2.4 Routing using Learning Automata

There follows an overview of how previous studies have applied learning automata to the problem of routing in networks, for both circuit-switched and packet-switched networks. The performance for the reinforcement algorithms that were used is also given, together with the reasons why variations in performance occurred.

2.4.1 The Network as an Environment

An environment is non-stationary if the penalty probabilities c_i (i = 1, 2, ... , r) corresponding to the various actions vary with time. There are three main types of non-stationary environment; periodic environments, Markovian switching environments and state dependent environments [20, 16]. Periodic environments vary the penalty probabilities periodically in time with a common period. Markovian switching environments are similar in that the penalty probabilities vary with time, but the new penalty probability set is chosen from a number probabilistically [32]. Finally state dependent environments vary implicitly or explicitly with the current state. For example implicit dependence may arise if the state transitions are determined by the action of the automaton. This final type of non-stationary environment is also termed a non-autonomous environment [18].

Routing is the process that decides over which physical links the data will be transmitted to eventually reach its destination. The action of routing a call changes the network state as the link utilisations will change. Therefore the environment for the learning automata when they are applied to the problem of routing in a communications network is of the non-stationary non-autonomous kind.

2.4.2 Routing in Networks

Elements that a good routing algorithm should ensure are: robustness, stability, fairness and optimality [5]. Using learning automata as the routing algorithm ensures these criteria are met due to its load balancing properties, its close to optimal performance, and its ability to learn the optimal routing pattern for a new network state. Its stochastic nature also aids stability as wild swings in network loading are avoided.

Learning automata have previously been applied to both circuit switched and best-effort rather than multi-service packet switched networks. Studies have shown that the basic elements that determine network performance are network architecture, call processing effectiveness, the routing method, and the switch structure [33]. The interest in applying learning automata to the routing problem has been partly due to their good performance and adaptability, but also due to their low processing requirement in comparison to other routing algorithms [34]. The following sections briefly describe their implementation in these differing network technologies, and general results obtained.

2.4.2.1 Learning Automata for Routing in Circuit Switched Networks

Telephone networks employ circuit-switching, where a circuit is set up from link to link in a progressive manner and the message is transmitted after the entire circuit has been set up, the transmission resources being reserved for the complete duration of the connection [35]. If there are no outgoing trunks free at a source or transit node then the call is blocked. Network performance is therefore normally measured as the blocking probability, and can be gathered for the whole network or local to each node, these being termed the global and local blocking probabilities respectively.

Using learning automata for the routing function in such a network, requires (N-1) automata at each node for a N node network, there being one automaton providing routes for one destination at a node. The destination address is held in the arriving set-up packet, and so the appropriate automaton for that destination is selected. If the connection request reaches the destination node, then an acknowledgement returns to the source node. If the call is blocked then a release message returns to the source node. Therefore the response from the network is binary, the environment being classed as P-model responsive.

Previous simulation studies have centred on fully connected or hierarchical network topologies, these both mapping directly to current telephone networks. Performance of learning automata have been shown to equal existing fixed rule alternate routing schemes under engineered loads, but outperform the latter in overload conditions where there exists extra capacity elsewhere in the network [36].

2.4.2.2 Learning Automata for Routing in Best-Effort Packet Switched Networks

Learning automata have also been applied to best-effort packet switched networks, in a different form to the circuit switched case. Best-effort packet switched networks can be roughly subdivided into datagram and virtual circuit types. Since each call is now accepted, the performance measure for a routing scheme in these kinds of network is no longer the blocking probability but the mean end-to-end delay of packets arriving at the destination node. Therefore the feedback from the environment is no longer binary, but continuous. This can be discretised, and by using digital technology effectively always is, but the S model response environment is still applicable in this case.

As with the circuit switched case, routing decisions are performed by (N-1) independent automata situated at each node. In addition to these, delay estimate vectors are also held at each node, these containing the average delay between the current node and all the destination nodes for each outgoing link. With the datagram network case, the feedback is returned by a small acknowledgement packet for each data packet received, this including the sum of the delay between the previous node and the current node, and the estimate of the delay from the current node to the destination. This sum is therefore the delay estimate for the destination node for the previous node. The response to the automaton at a node is bounded between zero and one, and is the normalised delay. This normalisation procedure can be calculated in either of two ways, the first being:

$$\text{normalised delay} = \frac{\text{delay}}{\text{max. delay}}$$

However this requires prior knowledge of the maximum delay. The other method doesn't require a priori information, using the minimum recorded delay to date:

$$\text{normalised delay} = 1 - \frac{1}{\sqrt[n]{\frac{\text{delay}}{\text{min. delay}}}}$$

This latter technique has the added attraction of effectively separating low values of delay which are closely spaced and compressing the range of longer delays.

Rather than updating the delay estimate by simply storing the new one, the exponential smoothing technique can be used as follows:

$$\text{normalised delay(new)} = \varepsilon(\text{normalised delay(old)}) + (1 - \varepsilon)(\text{normalised delay(returned)})$$
$$\text{with} \quad 0 < \varepsilon < 1$$

The virtual circuit case is similar in using normalised delay feedback updates, but this occurs for the acknowledgement to the original set-up packet for the virtual circuit only. To minimise switch processing overhead, learning automata have also been used as decision makers at the source node of a set of pre-determined routes to a destination node [37]. According to the delay feedback and reinforcement algorithm employed, the automaton probabilistically chooses a route for the virtual circuit request, this effectively being source routing.

2.4.2.3 Steady State Performance for Routing using Learning Automata

It has been shown that both LRI and LRεP learning automata converge to equalise the penalty probabilities [38], these being representative of ε-optimal schemes in steady-state performance. LRP automata on the other hand, converge to equalise the penalty probability rates [18], this being representative of ergodic schemes. This is why both LRI and LRεP automata have been found to result with a lower blocking probability and so give performance closer to the optimum when compared with LRP automata.

This translates to LRI and LRεP equalising the path blocking probabilities, and the LRP scheme equalising the path blocking probability rates in circuit switched networks. In the best-effort packet switched network case, the average packet delays are equalised by the LRI and LRεP schemes, whilst the LRP scheme equalises the delay rates.

2.4.2.4 Transient Performance for Routing using Learning Automata

It is not so straightforward to explain the dynamic behaviour of automata action probabilities, as the steady state performance does not hold during convergence. A new model of a nonstationary automaton environment was proposed, whose response characteristics are dynamically related to the probabilities of the actions performed on it [39]. This model was shown to give good correspondence with the transient response of the automaton action probabilities when appropriate model parameters are chosen.

However the transient behaviour when multiple automata provide adaptive routing in an environment has not been examined, this requiring a more complex model. The problem is

that other automata in the network might cause more complex phenomena, such as oscillatory behaviour.

2.5 Summary

The chapter has provided an overview of the theoretical aspects of learning automata, with the purpose of giving sufficient background from which to draw when detailing the proposed improvements in the following chapters. As the improvements relate to the reinforcement algorithm used, this being the learning mechanism of the automaton, so this chapter has concentrated on detailing the current standard approaches and algorithms commonly used.

These have been tabulated according to the resulting performance characteristic of the automaton in stationary random environments, together with comments on desirable performance properties for operation in non-stationary environments.

Brief explanation on how learning automata have been previously applied to the routing problem have also been included, together with relative reinforcement algorithm performance.

3 Improving Standard Dynamic Routing Algorithms for Routing in Multi-Service Networks

3.1 Introduction

The purpose of this chapter is to detail a link-state routing algorithm which provides low blocking probability performance when compared to other previously proposed algorithms. The resulting performance from this algorithm may then be compared to the learning automata based routing methods outlined in the chapters which follow. All aspects of the link-state routing method are examined: the algorithm itself, the link-state information to be propagated, and finally also the propagation method.

However before looking at the routing mechanisms, the Call Acceptance Control (CAC) method is examined as this is linked with the routing function. A bandwidth-based CAC mechanism is proposed, with methods and calculations for obtaining the effective bandwidths for different traffic types being given. This section includes a novel method for calculating the effective bandwidth of MPEG streams which provides more accurate results than previous methods.

Having examined and detailed the CAC mechanism, previously proposed algorithms for routing in multi-service networks are outlined. A new algorithm is proposed, and the results from simulation experiments are then given to show its benefits when compared to another algorithm which was detailed in previous work.

Next the type of link-state information is examined, having first summarised previous work in the area. The section continues with simulation results which show the benefits of using the proposed type of link-state information as part of the route calculation.

Finally, mechanisms for reducing the signalling overhead when propagating the link-state information are examined. After summarising some previously proposed mechanisms, a new method which uses existing connection set-up signalling is detailed. This is compared to other methods via further simulation experiments, with conclusions being drawn as to its relative benefits.

3.2 The routing function and call acceptance control

3.2.1 Bandwidth-based CAC

There is a strong relationship between proposed congestion control mechanisms used to guarantee Quality of Service (QoS) in multi-service networks and routing. Congestion control is required in order to ensure that all connections' QoS requirements, such as delay and/or cell loss, are satisfied. There are two aspects of congestion control in multi-service networks: accepting a connection request based on its pre-defined traffic contract, and policing an accepted connection to ensure its compliance to its pre-defined traffic contract. Policing is referred to as Usage Parameter Control (UPC) in ATM networks, and various algorithms have been proposed to perform this function [40, 41].

However, it is at the acceptance of a call's request based on its traffic contract that the strong link with routing occurs. It is the role of the Call Admission Control (CAC) to allow the new connection onto the network if sufficient resources are available to meet its QoS whilst not affecting that of others already accepted. If insufficient network resources are available however, the call request can be rejected or the QoS downgraded. The CAC can use one of two general methods to perform this function: either measure current network utilisation to ascertain whether the new connection can be accepted [42]; or pre-characterise each accepted connection's bandwidth requirement permitting a new connection if the sum of the bandwidth requirements is not above each virtual links' bandwidth (VPC in ATM and class bandwidth in IP with QoS) which comprises the route from the source to the destination [43]. As this study is primarily interested in the routing function, so the simplest CAC method was chosen: that of reserving capacity for an accepted connection based on its effective bandwidth [44].

3.2.2 Effective bandwidth calculations

The effective bandwidth of a connection is generally characterised by a value lying between the peak and mean bit-rates of the call. When a new connection is set up over a logical link, an amount of bandwidth equal to its effective bandwidth is reserved on the logical link for the duration of the call. The CAC function consists of determining whether there is sufficient residual bandwidth to accommodate the effective bandwidth of the new connection request.

3.2.2.1 Effective bandwidth allocation strategy

If a call traverses several logical links, the cell losses or delays accumulate along the connection and the end-to-end QoS achieved by the connection is equal to the sum of the QoS on the logical links. In an ATM network, or an IP network with QoS features, the same effective bandwidth allocations may therefore result in different end-to-end QoS for different connections, depending on the route chosen.

Previous work has highlighted the need to sum the individual VPC's QoS for a route using an ATM network in order to obtain the end-to-end QoS, with both fully and sparsely connected topologies [45, 46]. The resulting allocation strategy derived from previous work is as follows: the required end-to-end QoS is divided by the number of VPCs (or logical links) which form the route, an effective bandwidth being reserved on each for the connection. The effective bandwidth calculated for each logical link will therefore be for a stricter QoS than the end-to-end QoS. It follows that the longer the paths permitted in the network, the higher the effective bandwidth which will be reserved along the route for the same connection request and therefore same end-to-end QoS requirement as with a shorter path. This factor impinges on the routing algorithm, in that any algorithm for routing in multi-service networks should try to limit higher hop count paths more so than with standard dynamic circuit-switched routing algorithms.

3.2.2.2 Computing effective bandwidth values

Effective bandwidths may be calculated according to the fluid model described in [44]. This details equivalent capacity equations which may be applied for either cell loss rate or delay QoS calculations. As delay through a node is upper bounded by each switch manufacturer, it is relatively straightforward to design the physical network ensuring that worst-case end-to-end delay is still conformant with the QoS required by application types. This study therefore uses the equivalent capacity equation to meet cell loss QoS criteria for connections. The resulting equivalent capacity equation is:

$$\hat{c} \approx \frac{\alpha b(1-\rho)R_{\text{peak}} - x + \sqrt{[\alpha b(1-\rho)R_{\text{peak}} - x]^2 + 4x\alpha b\rho(1-\rho)R_{\text{peak}}}}{2\alpha b(1-\rho)}$$

where $\alpha = \ln(1/\varepsilon)$ (ε being the desired QoS), ρ is the utilisation, R_{peak} is the peak rate, b is the mean of the burst period, and x is the available buffer size.

3.2.2.2.1 Voice call calculations

Previous studies have characterised voice traffic according to the number of on-off sources required to model the traffic type [47]. Table 2 gives the resulting modelling parameter values for uncompressed and uncoded traffic sources, where M is the number of on-off sources required to model the traffic type, $1/\lambda$ is the mean 'off' period, and $1/\mu$ is the mean 'on' period.

So for a QoS specifying a cell loss probability of 10^{-9}, the resulting effective bandwidth for the voice call is about 0.026 Mb/s. This figure being lower than the standard 64 kb/s reserved on telephonic networks shows one aspect of the savings that the statistical multiplexing effects of ATM networks can bring. The effect of increasing hop count when choosing a route is shown in Table 3, the results indicating that unlike the reports from a previous study [46], the effect is relatively limited as there occurs a 1.5% increase when using a 5 hop route over a 1 hop route.

Traffic Source	M	λ(1/s)	μ(1/s)	R_{peak}(Mb/s)
voice	1	1/0.65	1/0.352	0.064
videophone	10	1.3078	2.5922	1.163

Table 2: Traffic modelling parameters

Number of hops	Effective bandwidth requirement (Mb/s)
1	0.0266708
2	0.0268359
3	0.0269327
4	0.0270014
5	0.0270547

Table 3: Effect of increasing hop count on voice effective bandwidth

3.2.2.2.2 Video call calculations

Using the values given in Table 2 together with a QoS specifying a cell loss probability of 10^{-9}, the resulting effective bandwidth for the videophone call is 9.917 Mb/s. However most video sources will be MPEG coded, and as yet no satisfactory algorithm for calculating the effective bandwidth exists, with current formulae grossly underestimating the bandwidth required [48]. Therefore there exists a requirement for a more accurate a-priori effective bandwidth calculator for MPEG traffic, as ATM networks with bandwidth-based CACs will require this in order to guarantee the MPEG connection request and other existing connections' QoS requirements.

Some progress has occurred in this area by examining traces from various programmes encoded by MPEG. It has been shown that there is a wide variation of peak and effective bandwidth requirement for these various traces [48, 49], with peaks varying from around 0.3 to 7 Mb/s and most being between 3.5 and 6.5 Mb/s. Whilst the mean bandwidth requirements were much lower, the effective bandwidth required was in the region of half of the peak as this source type is very bursty.

In order to produce an effective bandwidth formula which provides results close to that of real traffic, MPEG traces were examined to obtain a characterisation of its pattern. Three frame types are visible: I, P and B frames, there being two patterns transmitted each second. Each pattern consists of a twelve frame group of pictures (GOP) pattern of IBBPBBPBBPBB [49]. Thus an I frame will occur twice each second, and so on. The following novel solution is proposed: that of characterising each frame type as a separate deterministic 'on-off' source, and summing the required effective bandwidths generated by the equivalent capacity equation to obtain the total bandwidth required for the traffic stream. By knowing the number of GOP patterns that occur each second, the 'on' time where each frame type is transmitted can easily be calculated, the 'off' time being the GOP pattern time minus the calculated 'on' time.

In order to test the proposed method, model parameters were obtained from the 'Star Wars' film trace, these being shown in Table 4. So for a QoS specifying a cell loss probability of 10^{-6}, the resulting effective bandwidth for the I frames is 2.315 Mb/s, for the P frames 0.165 Mb/s and for the B frames 0.064 Mb/s resulting in the MPEG video connection requiring about 2.544 Mb/s. This final result compares extremely favourably with the required bandwidth found for the real trace, which was about half of the peak requirement [48], in this case being 2.12 Mb/s. This result is more impressive after considering that methods to date consistently underestimate the bandwidth required.

The slight overprovisioning of resources which has resulted may be explained by the fact that the examined trace yielded a mean rate of 0.537 Mb/s, whilst the full film results with a mean of 0.36 Mb/s. This implies that the examined trace had a higher degree of traffic than the overall film, and it is therefore possible for this method to yield precise required bandwidth calculations were the I, P and B average frame parameters from the whole film used.

There is no doubt that further empirical results using different traces are required to more fully validate this method. However, for our purposes of utilising a method providing realistic effective bandwidth requirements which increase with longer route hop counts, the proposed novel method is currently the best available.

Using this method, the effect of increasing hop count is shown in Table 5 by the required bandwidth. As can be seen, the relative effect of increasing hop count is of a higher order than with the voice connections, there now being an 8% increase when using a 5 hop route over a 1 hop route. This combined with the higher absolute effective bandwidth required becomes significant in heterogeneous traffic situations where lower bandwidth requiring traffic such as voice may be severely restricted due to unnecessarily long hop counts in the higher bandwidth traffic's routes. This is evidenced by the video traffic's 5 hop route consuming extra bandwidth equivalent to about 8 voice connections when compared to the video's 1 hop route.

Historically routing algorithms have tended to use the shorter paths in order to maximise the number of simultaneous calls being carried by the network. This CAC work shows that this principle must be all the more strictly adhered to in the case of multi-service networks, as choosing a path of twice the length of another possibility, more than doubles the amount of network resources used.

Traffic Source	**Frame Type**	λ(1/s)	μ(1/s)	R_{peak}(Mb/s)
video	I	1/0.4583	1/0.0417	4.24
	P	1/0.09375	1/0.0417	0.48
	B	1/0.0417	1/0.0833	0.095

Table 4: Model parameters for the MPEG coded 'Star Wars' film trace

Number of hops	**Effective bandwidth requirement (Mb/s)**
1	2.544
2	2.641
3	2.694
4	2.730
5	2.757

Table 5: Effect of increasing hop count on MPEG video effective bandwidth

3.3 Standard dynamic routing algorithm selection

Previous work has shown link-state routing schemes to be more flexible and robust [50]. Therefore this type of routing algorithm was chosen as representative of the best performing algorithm type currently used in real networks.

3.3.1 Overview of algorithms

Link-state routing algorithms are dependent on the accuracy of the network state information which each node or decision maker holds. However, frequently propagating network state information in order to maintain database accuracy incurs a heavy overhead in extra signalling and processing. Therefore a trade-off exists between increasing the routing algorithm performance and minimising the extra signalling bandwidth and processing required in order to do so.

Previously proposed algorithms for routing in multi-service networks have in the main been of the link-state type. Comparative investigations of routing algorithms have been previously undertaken [46, 51], with each algorithm being seen to have different strengths and weaknesses. Four main variances on shortest-path type algorithms have been proposed: widest-shortest path, shortest-widest path, dynamic- alternate path, and shortest distance path. There follows a summation of the perceived strengths and weaknesses of each, followed by details of a proposed new algorithm called Alternate Adaptive Minimum Hop (AAMH).

3.3.1.1 Dynamic-alternate path

This algorithm has been proposed due to the success of the dynamic alternate routing algorithm used in telecommunication networks. In its strictest form, such as Least Busy Path (LBP) [45], it is equivalent to that implemented by AT&T in the form of its RTNR algorithm [52].

The LBP algorithm assumes a highly-connected topology, and operates as follows: an attempt is made to route the connection on the direct path first, and failing that it attempts the two hop path with the maximum residual capacity. Paths are not permitted to exceed two

hops in length, and the residual capacity (N_{std}) on a path from source node s to destination node d via intermediate node t is given by:

$$N_{std} = \min[N_{st}, N_{tf}]$$

where N_{st} is the residual capacity on the link between node s and node t, and N_{td} is the residual capacity on the link between node t and node d. Bandwidth may be reserved for 'direct' traffic on links in order to prevent unrestricted use of longer alternative paths leading to a reduction in throughput at heavy loads.

The initial proposal was to emulate a highly-connected circuit switched network topology by using ATM VPCs to provide logical direct links between pairs of nodes [45]. However by so doing, there is a reduction in the level of statistical multiplexing, causing inefficient use of the network resources.

In order to loosen the algorithm's dependency on a highly-connected topology, subsequent work has modified the algorithm. Rather than specifying one hop minimum routes, any route with a minimal number of hops is permitted, the residual capacity for the route being the minimum of all its comprising links. If no feasible minimal hop path is available, minimal hop plus one hop are chosen.

Even without using trunk reservation, this modified algorithm has been shown to result with good performance, returning a comparable if slightly worse blocking probability as other algorithms at low loads, whilst providing comparably superior performance as the loads increased [51]. In addition it has been clearly demonstrated to have greater robustness and insensitivity to inaccurate network state information.

3.3.1.2 Widest-shortest path

This algorithm is similar to the improved version of the previous, apart from only permitting paths with a minimal number of hops [53]. It was termed Adaptive Minimum Hop (AMH) in a previous study [46], and was found to provide superior performance when compared with other algorithms of the shortest-widest and shortest-distance types.

3.3.1.3 Shortest-widest path

This algorithm type chooses paths with the maximum bandwidth, and if there are several such paths the one with the fewest number of hops is selected. If there are several paths with the

same minimum number of hops, one is randomly selected. Minimum Sum of Loads (MSL) is of such a type [54].

As this algorithm type does not place an upper limit on the route hop length, it has been found to produce poor performance in all scenarios except where very light traffic loading was present. In such cases it gave comparable or slightly improved performance over the previous algorithm types.

3.3.1.4 Shortest-distance path

Shortest-distance path algorithms differ from shortest-widest path types in their cost function. Instead of summing the residual bandwidth in a linear manner, the reciprocal of the link residual capacity is added together to form the cost of a route. The route cost therefore becomes:

$$c = \sum_{i=1}^{k} \frac{1}{R_i}$$

where $R_1, \ldots, R_k$ are the bandwidths available on the path with k hops.

Least Loaded Path routing (LLP) [55] is one of a number of this type which has been shown to be effective when selecting routes for high bandwidth connections [56]. It also results with improved performance over shortest-widest schemes as highly loaded links are effectively excluded from the route possibilities since their cost tends towards infinity. When compared to dynamic-alternate type schemes in the main comparative performance has been achieved [51], with slightly improved performance at low traffic loads and slightly poorer performance at high loads. However, it has also been noted that this algorithm type is more susceptible to inaccurate network state information, this being explained by it having the opportunity to select longer paths, inaccurate link information therefore being compounded the greater number of links comprise a route.

3.3.1.5 The proposed new algorithm

It has been shown that in QoS routing there is a need to limit route length as a higher hop route requires a higher effective bandwidth in order to guarantee the end-to-end QoS for the connection. It is for this reason, in addition to that of multiple link consumption, that results in inferior performance of routing algorithms which permit unnecessarily long routes.

Therefore the proposed new algorithm, termed Alternate Adaptive Minimum Hop (AAMH), operates as a superset of AMH and in a similar vein to more recent dynamic-alternate schemes. As with AMH, the algorithm attempts to route on the least-loaded minimum hop route. If all are congested, it attempts to route on the next shortest hop route which is topologically permissible. This differs from dynamic-alternate schemes which permit alternate routes only of minimum hop plus one, even if the shortest alternate route is minimum hop plus two or more.

AMH has been shown to produce good performance when applied to sparsely-connected networks [46]. However it cannot be assumed that most multi-service networks will generally take this topological form, especially with the capability of a logical topology diverse from the physical one when using VPCs in ATM networks or class reservation in IP networks with QoS features. AAMH is therefore proposed as a generically applicable routing algorithm to any network topology, which should still result with leading performance from all the shortest-path type algorithms. Moreover dynamic-alternate path type schemes have been shown to produce good performance under diverse traffic loads whilst being relatively insensitive to inaccurate network state information. All these factors point to AAMH being a strong general purpose routing algorithm, having insensitivity to network scenario extremes. The performance derived from this algorithm may therefore be used as a comparison when evaluating that obtained from learning automata based routing schemes.

Increases in network bandwidth cause transmission times to decrease since the same information transfer occurs more quickly. Therefore the requirement for faster connection set-up times occurs since the signalling to manage the connection becomes a greater percentage of the total connection time on the network. In order to reduce the processing required at intermediate nodes, the use of source based rather than hop-by-hop based routing algorithms has been proposed. These algorithms calculate the whole route at the source node, including the details in the connection set-up packet that then traverses intermediate nodes on its way to the destination node of the route. There incurs a smaller processing delay at intermediate nodes and so faster connection set-up time. As routing algorithms in multi-service networks provide QoS guarantees by interacting with the CAC mechanism, so the required effective bandwidth for a connection request needs to be calculated at the source node, with the length of the route being one of the parameters in the calculation. Therefore multi-service routing algorithms are biased towards being implemented in a source-based fashion as the benefit of intermediate node route recalculation is lost since the effective bandwidth for a new route must be calculated at the source via a crankback mechanism. AAMH therefore lends itself to being implemented so that it operates within a source-based paradigm.

3.3.2 Simulation scenarios

Simulation modelling was chosen rather than analytical modelling for evaluating research ideas in order that the ideas would not be constrained by the necessary simplifying assumptions required to make the problem tractable analytically. The OPNET modelling and simulation tool was chosen after an extensive comparative evaluation with other available commercial simulation tools. As the standard model libraries included with the package are constrained functionally (so that it becomes unfeasible to undertake large dynamic call generation scenarios), it was decided at the commencement of the research to write a new model library in order to fully evaluate any ensuing research direction. This new library consists of more than 13,000 lines of C code, and further elucidation on all these modelling aspects are given in appendix B.

The simulations were performed using three differing logical network topologies. These are shown in Figure 3 to Figure 5, and they range both fully-connected and sparsely connected topologies. The fully-connected topology was kept small so that the performance improvement of using an alternate routing scheme over a minimum hop scheme would not be too dramatic. The seven node topology is representative of a typical corporate WAN, whilst the large sparse topology is a simplified version of the US internet backbone. The results obtained using these differing topologies will indicate the generic applicability of AAMH.

Previous comparative studies have demonstrated that algorithms with a strong preference for minimum-hop routes almost always outperform algorithms that do not consider path length [51, 58]. As AMH was designed by taking this into consideration, and was found to produce favourable results compared to other proposed algorithms at that time [46], it was used as the comparative benchmark for AAMH.

In order to ascertain the relative performance of AAMH over AMH, simplified simulation scenarios were generated. These centred on the pure algorithm performance, and so consisted of using homogeneous traffic sources and perfect link-state knowledge. The former precluded algorithm performance being affected by traffic source types with widely different bandwidth requirements, whilst the latter leaves out the effects of the frequency and type of link-state information propagation, this being addressed in the section that follows.

The link capacities were set to 500 Mb/s, and a data type traffic source was used which proved to have an effective bandwidth of around 2.4 Mb/s, yielding a mean bandwidth of about 0.5% of link capacity in order to have a realistic networking situation. Each simulation run consisted of 400,000 call requests, with statistics for the first 100,000 being discarded to allow the network state to settle into normal operational mode. Due to the long simulation run, it was found that the results obtained varied by no more that 0.15% when

varying the random number generator seeds. It was therefore felt sufficient to perform simulation runs using two different seeds, the average from both being taken as the blocking probability for the connection arrival rate.

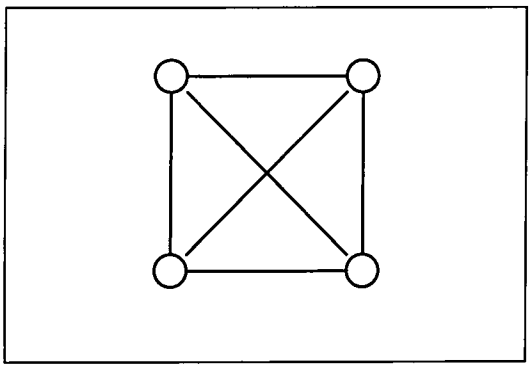


Figure 3: Fully connected logical topology

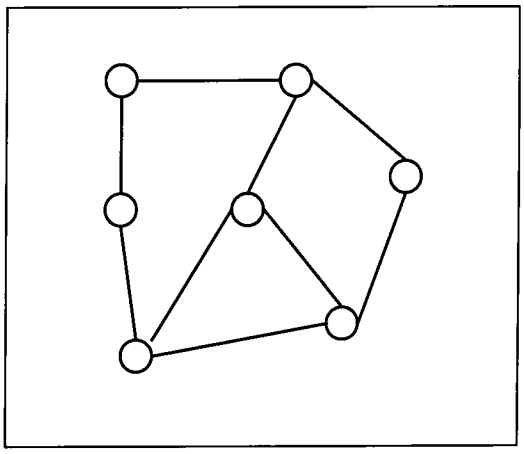


Figure 4: Seven node sparsely connected logical topology

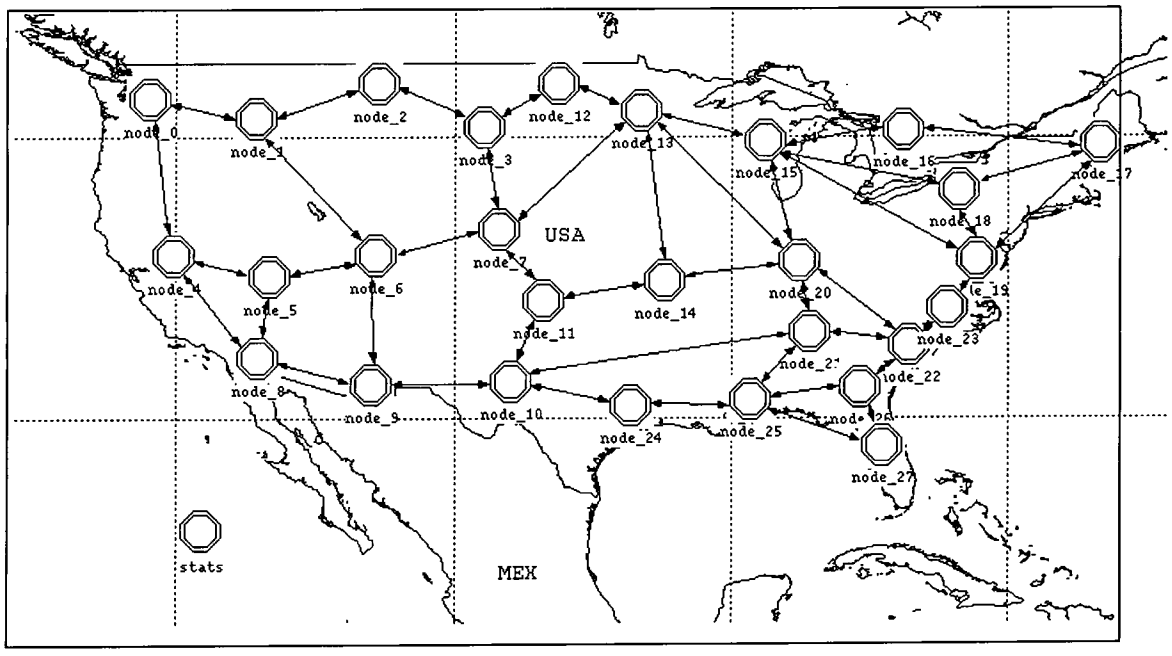


Figure 5: 28 node sparsely connected topology

Some previous studies on routing algorithm performance have included results generated only under symmetrical traffic loading, where source nodes obtain destination addresses for connection requests by using a uniform distribution over all the network nodes bar itself [51, 58]. Whilst it is true that symmetrical loading and homogeneously sized links approximates the situation of the user demand equalling the expected demand for which the network was designed (overloading occurring when the magnitude of the actual demand is higher than the expected, but the source-destination binding probabilities remain the same), yet it is often the case that actual demand varies significantly in type and magnitude than that expected, this being approximated by non-symmetrical traffic loading. Therefore our simulation experiments use both symmetrical and this non-symmetrical network loading configuration.

3.3.3 Results for the fully connected topology

It is in the context of highly-connected topologies that the benefits of the alternate routes available to AAMH should produce significantly better routing performance. The symmetrical network loading case is the worst-case scenario for AAMH, as it is in these conditions that the use of alternate paths will utilise more network resources and so produce a higher blocking probability.

It was found that AAMH with trunk reservation was comparable in performance to AMH under symmetrical traffic loading. However AAMH significantly out-performed AMH under non-symmetrical loading.

Were the topology to have a greater number of nodes, it is expected that the results would be similar under symmetrical loading, and superior for AAMH under non-symmetrical loading due to the increased number of available alternate paths.

3.3.3.1 Results for symmetrical network loading

Teletraffic networks use the trunk reservation mechanism in order to ensure that alternate path routing schemes do not overly use longer routes to the detriment of future connection requests which could have been routed on minimum hop routes [57]. Trunk reservation operates by reserving a certain percentage of the link capacity for minimum hop routes, so that when the link is close to saturation only minimum hop new routes are allowed to traverse it.

The requirement for having the same mechanism with AAMH is evident from Figure 6 where AAMH with no trunk reservation produces significantly higher blocking probability

than AMH which just utilises single hop routes. In general a trunk reservation parameter of 5% is used in teletraffic networks, this causing AAMH to return blocking performance close to AMH as shown in Figure 8. In fact AAMH produces slightly better performance than AMH under low traffic load levels, and slightly worse at higher levels with fixed trunk reservations.

Further experiments were performed by varying the trunk reservation parameter over different network loading rates. Figure 7 shows the trunk reservation parameter which caused AAMH to produce the lowest average blocking probability at each loading rate. As expected, at low traffic arrival rates a lower trunk reservation parameter allows AAMH to choose more alternate paths so resulting in a lower blocking probability. However at higher traffic loadings, the use of alternate paths must be discouraged, so requiring a higher trunk reservation parameter. Were the trunk reservation parameter to be dynamic according to the traffic loading, it is expected that AAMH would in general out-perform or be equal to the blocking probability produced by AMH for highly-connected network topologies.

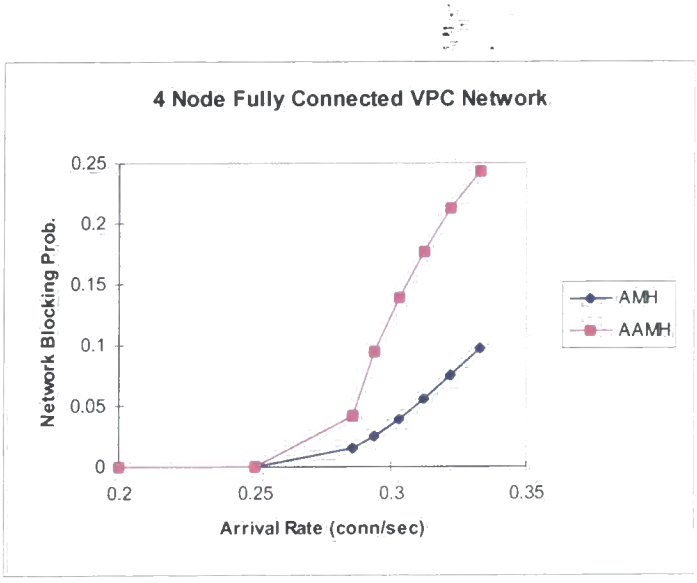


Figure 6: Performance of AMH and AAMH with no trunk reservation for the fully connected topology

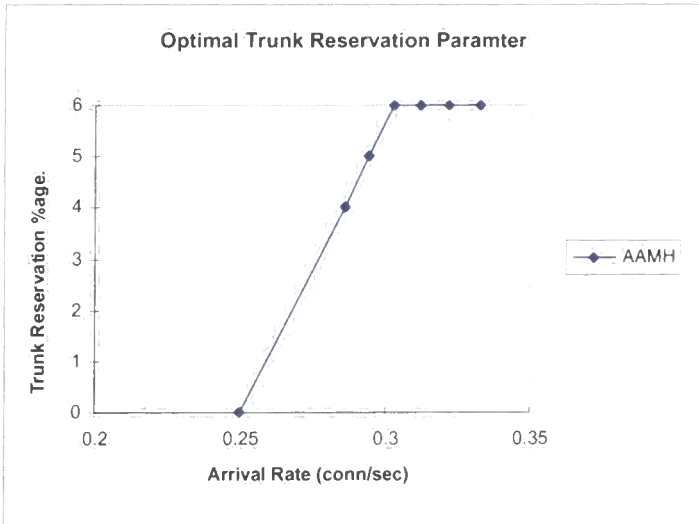


Figure 7: Optimum trunk reservation for AAMH for the fully connected topology

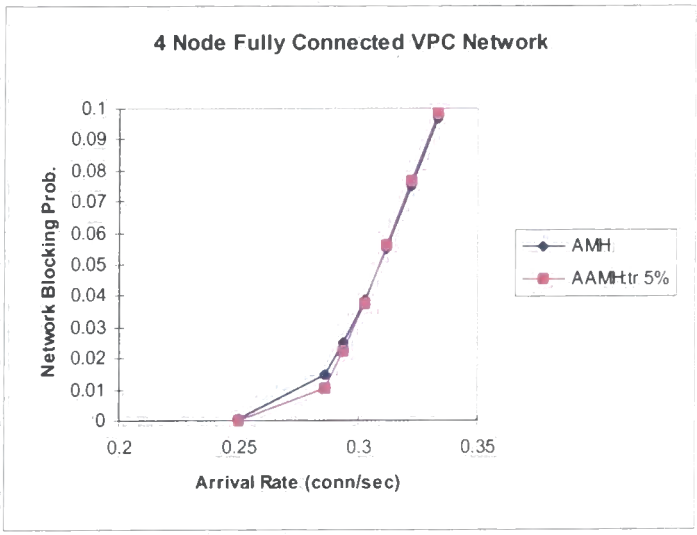


Figure 8: Performance of AMH and AAMH with 5% trunk reservation for the fully connected topology

3.3.3.2 Results for non-symmetrical network loading

The user demand traffic matrix used for this set of experiments was one node transmitting to another fixed destination address, with the other nodes establishing connections to randomly generated destination addresses at an eighth of the arrival rate of the first node.

Figure 9 shows that in these circumstances AAMH significantly outperforms AMH, producing a blocking probability up to 40% lower. This is due to it being able to use two hop

paths when AMH is limited to using the direct one hop route. The use of alternate paths should not be discouraged under these circumstances, Figure 10 showing that when they are there results a light decrease in performance, to the extent of 2.5% higher blocking probability.

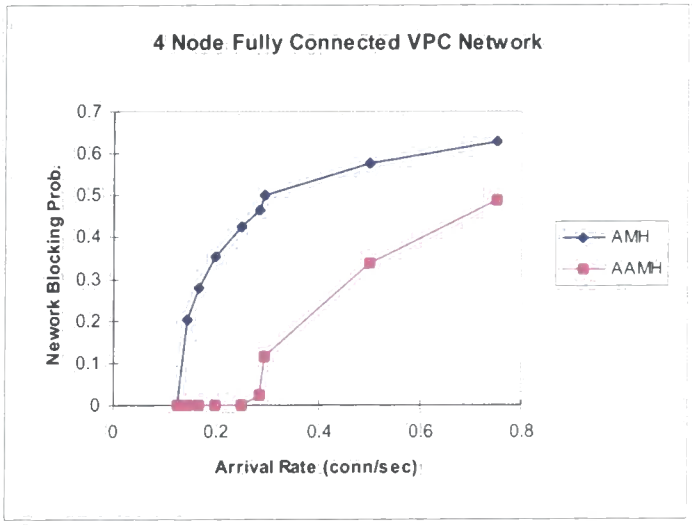


Figure 9: Performance of AMH and AAMH for the fully connected topology under non-symmetrical loading

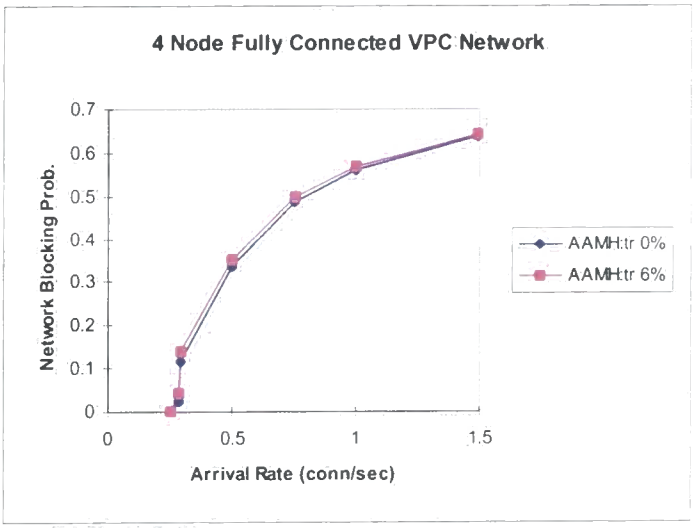


Figure 10: Performance of AAMH with 0 and 6% trunk reservation

3.3.4 Results for sparsely connected topology

AMH was primarily designed for use in sparsely connected network topologies. However, the results below indicate that AAMH still out-performs AMH with non-symmetrical traffic loading, although not to the same degree as with the fully-connected topology.

3.3.4.1 Results for symmetrical network loading

Figure 11 shows the results for AMH and AAMH with no trunk reservation and, as expected, AMH returns superior results. However the difference between the two is not as great as with the fully connected topology scenario. The reason for this is that rather than consuming a little over twice the network resources as previously, alternate paths now consume 1/3 to 1/2 as much depending on the route. This factor is also evident in Figure 12 which shows in general that a much lower trunk reservation parameter is required for optimum AAMH performance since alternate routes should not be discouraged as much as in the previous case.

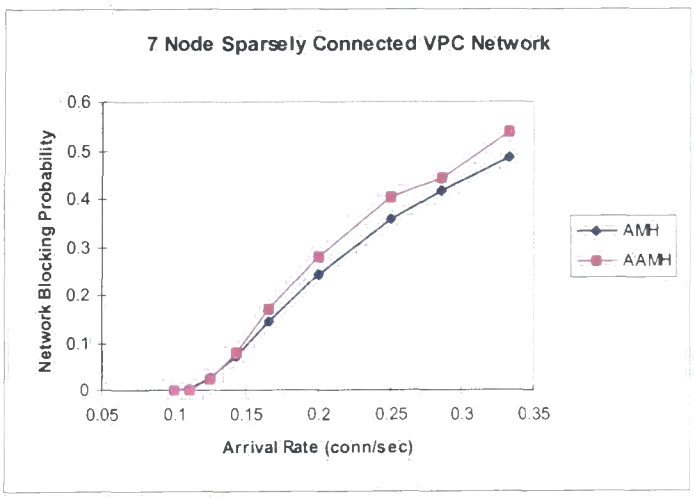


Figure 11: Performance of AMH and AAMH for the sparsely connected topology

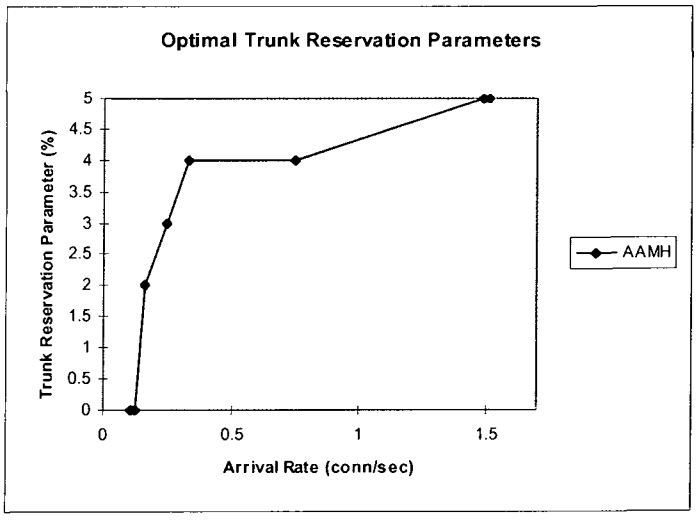


Figure 12: Optimum AAMH trunk reservation parameter for the sparsely connected topology

3.3.4.2 Results for non-symmetrical network loading

As with the fully connected network topology, the best case realistic scenario was simulated. This consisted of the top left node transmitting to the far right node, with the other nodes establishing connections to randomly generated destination addresses at an eighth of the arrival rate of the first node.

Figure 13 clearly shows AAMH significantly out-performing AMH due to the extra paths it has at its disposal as alternate routes. The performance improvement is not as great as with the fully-connected topology due to AMH now having a number of paths at its disposal for routing to the destination nodes rather than the one it previously had. This means that the increase in paths which AAMH has over AMH is not as great as before, and so not as much extra traffic can be routed on the alternate paths as previously.

Considering that AMH was designed for sparsely-connected topologies, this is a strong result in favour of using AAMH as a general routing algorithm for routing in multi-service networks.

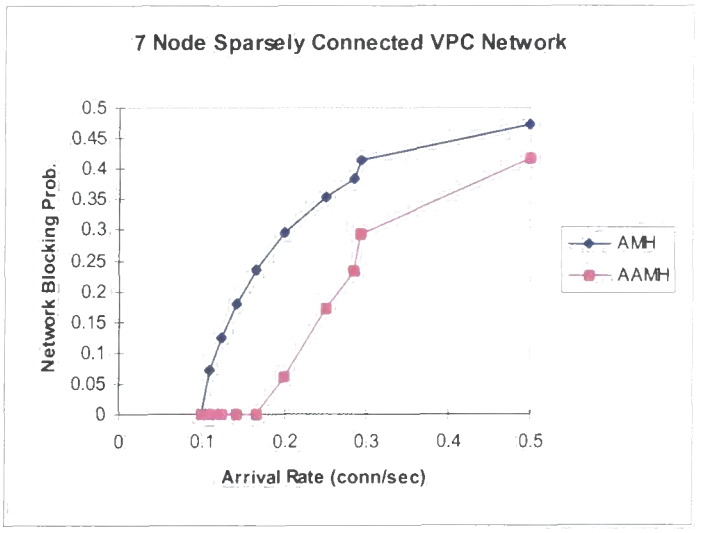


Figure 13: Performance of AMH and AAMH under non-symmetrical loading for the sparsely connected topology

3.4 Realistic routing algorithm performance

In order to remove performance affecting factors so that the performance due to pure algorithm selection could be shown, the results given in the previous section were based on the source node having perfect network state information at its disposal. This is not possible in a real-world situation, but can be approximated by periodic updates of link-state information which are propagated throughout the network and so to every source node. A minimum time interval might be imposed to avoid overloading of network bandwidth and processing resources, but large periods result in out of date link-state information which can cause a switch to select a sub-optimal or even unfeasible route. Hence tuning the frequency of link-state update messages requires a careful understanding of the tension between network overheads and the accuracy of routing decisions.

3.4.1 Route selection with partial information

The link-state information can be propagated in a periodic fashion or in response to a significant change in the link-state metric; for example its utilisation. By updating link load information in response to a change in available bandwidth, triggered updates respond to

smaller changes in utilisation as the link nears capacity. In contrast to periodic updates however, triggered messages complicate the provisioning of network resources since rapid fluctuations in available capacity can generate a large number of link-state updates, unless a reasonable hold-down timer is used.

With a periodic update policy, large periods substantially increase connection blocking, ultimately outweighing the benefits of QoS routing. In fact, under uniform loading static routing becomes competitive with QoS routing once the update period grows beyond 60 times the average connection interarrival time [58], the reason being that the fluctuations in link state begin to exceed the random variations in traffic load. Under non-uniform loading QoS routing does indeed continue to outperform static routing. Periodic updates with large periods also cause dramatic fluctuations in link state between successive update messages, therefore meaning that the routing algorithm is now causing oscillatory and less stable network behaviour. This phenomenon occurs by the network reacting to an update message that a link has low utilisation by routing more traffic through that link. Blocking remains low until saturation occurs, and is then constant until the next link-state message update occurs which can cause another dramatic change in the link utilisation as no further connections are routed along it and the present ones disconnect. However some form of periodic update is required in order to detect link or equipment failures in the network. It is proposed that such updates form part of a separate local system, which upon detection of a local link failure propagates a link fully utilised (or link unavailable) Link State Advertisement (LSA) throughout the network using full flooding.

Triggered updates may be implemented with dynamically calculated events, or pre-determined event threshold levels, and should result in more stable network operation as large differences between the actual and advertised link utilisation are no longer possible. Dynamically calculated triggers occur upon detection of a significant change in the available capacity since the last update message, responding to smaller changes in utilisation as the link nears saturation. In contrast to periodic updates, coarse-grain triggers do not have a significant impact on the overall blocking probability [58].

Event trigger thresholds may also be pre-determined throughout the network. This involves the mapping of load measures into discrete categories, and generating a message update when the link utilisation crosses into another category [59]. Benefits of using this method include simplicity: less processing at nodes is required, and only the integer category value is required in the updating message rather than the utilisation itself. Another benefit is that the thresholds could then be set according to expected traffic loads, causing message updates only when actual traffic patterns differ from the expected ones. This latter reason fits well with the case for dynamic routing algorithms, for if the network designer and planner was able to predict the traffic demands accurately, then the network could be perfectly

dimensioned and static routing be used. For this study pre-defined event trigger thresholds were used.

3.4.2 Simulation results of using event trigger thresholds

A previous study which used category bands for link-state updates concluded that link-state information at higher load levels is useful, but is not required at low loads as no loading information is necessary for a routing algorithm to produce good performance [60]. Indeed, it found that even though there may be fewer categories in a set, if it divided the load levels with a finer granularity when the link was close to saturation, this produced better results from the dynamic routing algorithm. The load categories chosen for use in our experiments were therefore as follows:

[0%, 50%), [50%, 80%), [80%, 90%), [90%, 95%), [95%, 100%].

In order to assess the impact of effectively discretising the load space into categories and just updating the categories when a transition occurred into a new one, the following experiments were undertaken on the 28 node sparsely connected logical network topology. This provided the greatest diversity of route options, and so would best indicate the performance improvements of using load information to guide the route calculation.

The performance baseline chosen was a routing algorithm which had the option of using the full set of alternate routes, but had no load information to guide it in its choices. It attempted to route on all the minimum hop routes and then on all the alternate routes in turn, using crankback in order to choose another route. Whilst this algorithm would in general produce multiple routing attempts and so have a significant network processing overhead, it does result with the lowest blocking probability possible for a shortest-path routing algorithm not utilising load information in its calculations. This algorithm is termed Alternate Routing (AR) in the graphs below. The other two algorithms are as follows: 'AAMH' is AAMH operating with exact link-state knowledge, and 'AAMH(LB)' is AAMH operating with the load bands specified above. Symmetrical network loading was used for these experiments as it was felt that it would more clearly show the benefits of directing route calculation with the aid of link-state information. Otherwise the performance difference between AR and AAMH with full link-state knowledge might not be so noticeable.

Figure 14 shows that this difference is in the region of a 2% lower blocking probability for AAMH with full network knowledge. As expected, the performance of AAMH with load bands is found to be between the two. What is of interest however is that at

low traffic loads its performance is close to that of AAMH with full link-state knowledge, whilst at higher loads its performance degrades to become closer to that of AR. From this it seems that at low load levels the two final load categories make a significant difference in path selection. When the traffic load increases, most links will be close to saturation and so in the last category, path selection then becoming more random and so closer to the performance produced by AR.

When examining the connection set-up time shown in Figure 15, we find that AAMH with full link-state knowledge gives a fairly constant value regardless of the loading rate. These results are reasonable for if the network cannot support a new call request it is blocked before any network signalling occurs. Conversely, if the network can support the call, then it is accepted and the set-up signalling occurs. The time taken to do so remaining unchanged indicates that the average route length does not alter as loading increases, as would be expected.

AR on the other hand, returns a higher connection set-up time which increases still further as the loading rate increases, whilst AAMH with load bands produces times that lie between the two. The increasing set-up time of AR makes sense when considering that a higher traffic loading would produce more highly utilised links, forcing a greater number of crank-backs as the routing algorithm attempts to choose other routes to reach the destination address. The matter of load bands discretising the link utilisation levels has the effect of causing crank-backs, but not to the same degree as with AR. This indicates that whilst the blocking probability performance might be similar under high loads, there remains the considerable benefit of a lower connection set-up time when comparing AR to AAMH with load bands.

The use of AAMH with load bands has been shown to produce superior results to AR which does not use link-state information in its decision making process. The issue now is on how to propagate the required link-state information to the decision making nodes.

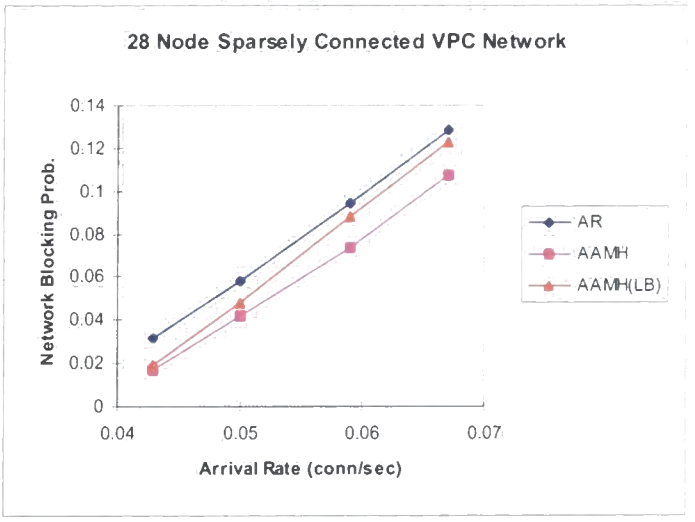


Figure 14: AAMH performance with and without load bands

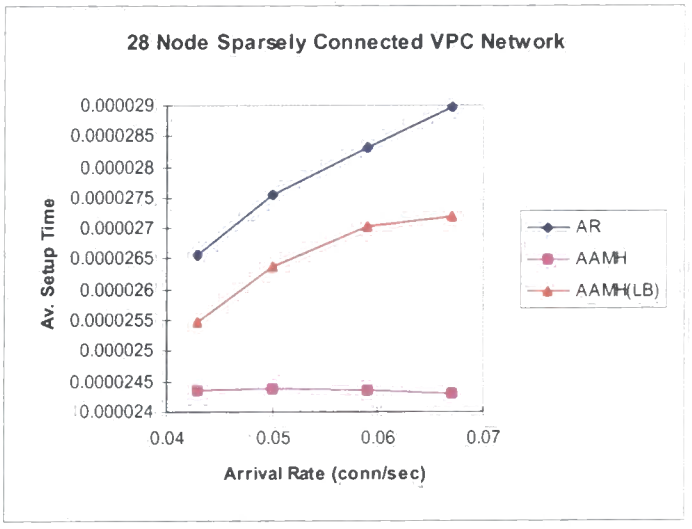


Figure 15: AAMH set-up times with and without load bands

3.5 Mechanisms for signalling overhead reduction

In large networks or internets, flooding dynamic information may not be possible because of the network processing and bandwidth involved. Therefore in this section ways to take advantage of the benefits of adaptive routing without relying on global flooding of dynamic updates are investigated.

3.5.1 Limited update distribution methods

Conventional hop-by-hop link-state routing protocols, such as OSPF [61], flood updates to all network nodes since all nodes are required to maintain consistent information to avoid routing loops. However a source-based routing scheme such as AAMH does not require this as loop-free routes are precomputed with the dynamic information being used to select from among the available routes and to improve routing decisions. Two different methods have been proposed in a previous study [60], and are outlined here.

As AAMH is a source-based routing algorithm and so guarantees loop-free routing even if updates are lost, so mechanisms for ensuring the correct delivery of the updates are not required as would be with conventional link-state routing protocols such as OSPF [61].

3.5.1.1 Hop-count limited flooding

Hop-count limited flooding is a simple mechanism by which routing updates are distributed within some fixed hop count, R, of the node initiating the update. When a node initiates an update, it sets a Time-to-Live (TTL) field in the update packet to R. Each node that processes the update decrements the TTL. If the TTL is greater than zero, the node continues the flooding process. Otherwise it records the update information but halts the flooding process. Using this distribution mechanism, each node learns dynamic information about those nodes within R hops of it.

3.5.1.2 Reverse path update

The previous method provides nodes with information about nearby nodes in the network. Reverse Path Update (RPU) is an alternative mechanism which provides nodes with dynamic information about some nodes further away in the network. RPU operates by forwarding update messages in the reverse direction of currently installed connections in the network. When a node initiates an update, it consults its forwarding table for currently active connections and forwards an update in the reverse direction of all active routes. In this way, each source node that currently has an active route through the initiating node will receive a copy of the update.

This method provides a reduction in update messages as it takes advantage of overlap routes, sending a single copy of the update message along a link shared by multiple routes.

3.5.2 Using locally available information

The same previous study also proposed the following two methods: caching reject information, and local link status. In addition to these, another method is newly proposed in this study: using the existing connection set-up signalling.

3.5.2.1 Caching reject information

With caching rejects, when a node attempts to establish a route and receives a reject notification from one of the nodes along the route, it caches this information. The notification will indicate which link lacked sufficient resources to admit the new sessions. The source is then able to avoid routes that traverse links having a high likelihood of being unavailable, and can try other routes which might have a better chance of admitting the session.

The issue with this strategy is how long to keep the reject information at the source. If it is held for too short a time it may be of limited use, and if it is held for too long it may no longer be accurate. Whilst the proper time-out for this information is a function of the traffic patterns in the network, a previous study used an interval of four times the average session length [60].

3.5.2.2 Local link status

This simple method involves nodes periodically measuring the status of their adjacent links, storing their current loading but not generating updates to other nodes. Therefore when a node makes a routing decision, it does so on the status of its adjacent links. Indeed this method has been commonly used elsewhere, such as with the classical hot-potato routing [62].

3.5.2.3 Using existing connection set-up signalling

This proposed method does not require extra signalling to propagate update messages throughout the network as it uses the existing call set-up signalling for this purpose. When attempting to set-up a call, a route is calculated at the source node and the signalling packet traverses the network according to this calculated route. Whether or not the call request is accepted or rejected by intermediate nodes, some sort of acknowledgement packet returns to

the source node which initiated the call request. This method 'piggy-backs' onto the returning acknowledgement packet the information concerning the link with the lowest available bandwidth found along the route. This information is subsequently stored at all upstream nodes from the link as the acknowledgement packet returns to the source node. The method will be termed Route Accepted plus Blocked (RA+B).

The ATM Forum specifies that the setting up and tearing down of VCCs be performed using defined signalling packets encapsulated within the payload of the standard ATM cell [41]. The IETF specifies a similar mechanism for RSVP, the call set-up and tear down messages being encapsulated in IP packets [10]. It is therefore proposed to include this loading information in both the 'connect', or call set-up, and 'release', or call tear down, signalling packet types.

3.5.3 Performance of limited distribution mechanisms

The previous study which evaluated these methods did not result with conclusions of any one method being better than another [60]. This was due to a higher signalling overhead being generated by methods which resulted in superior routing performance. What is of interest in this study is the performance of the new method which utilises existing signalling for link-state propagation compared to those previously proposed. As our proposed method includes as a subset of its total data input that which might be derived from using reverse path update, caching reject information, and the local link status, these were not simulated to be included in the comparison. Therefore there remains the general hop-count limited flooding method with which to make comparisons.

Figure 16 shows the blocking probability resulting from AAMH when using hop-count limited flooding and the proposed method with various route lengths for the 28 node logical topology. This network topology has a diameter of seven hops, and during simulation it was noted that when using AAMH an average route length of just under four hops was returned. As may be seen from the diagram, the performance increases when the hop-count limit is increased. What is of particular interest however is that the performance achieved when using a hop-count limit equivalent to the average route length is the same as that obtained when utilising full flooding. This result implies that a simple way of reducing the network bandwidth and processing required with link-state routing methods, whilst not affecting overly the routing performance, is to limit the update information propagation to that of the average path length. Were the traffic loading on a network to be more non-symmetrical

in nature, then perhaps certain source nodes which utilised certain longer routes might require extra link-state information.

When comparing the result achieved for our proposed method, shown as the RA+B line, it is encouraging to note that it is comparable to that returned when using full flooding. Whilst the RA+B method returns information on the most highly utilised links, the performance of the routing algorithm might improve on knowing the utilisation of links close by. With this in mind, a further experiment was undertaken by combining both the RA+B and the hop-count limited flood update methods. The blocking performance results obtained from this approach however are little different to that resulting from the use of RA+B singly. This result confirms previous work which indicated that information pertaining to highly utilised links is more important than that of other links [58].

As shown in Figure 17, the average set-up times resulting from the link-state methods are all fairly comparable and are lower than that of AR. This again indicates the fact that link-state information, however incomplete, will direct the routing process and so result with lower connection set-up times.

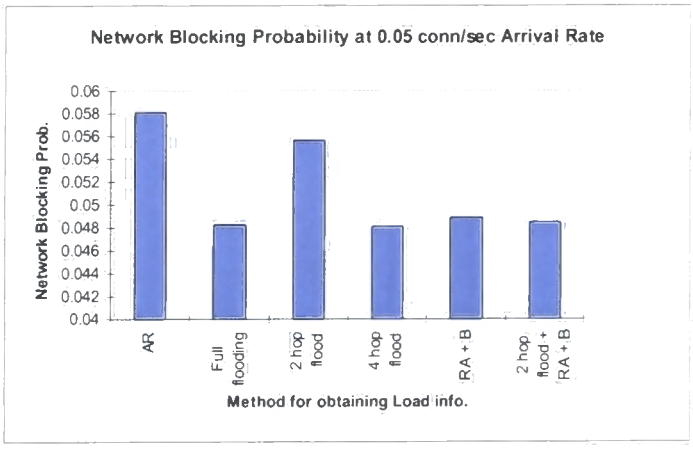


Figure 16: Resulting blocking probability performance of link-state updating methods

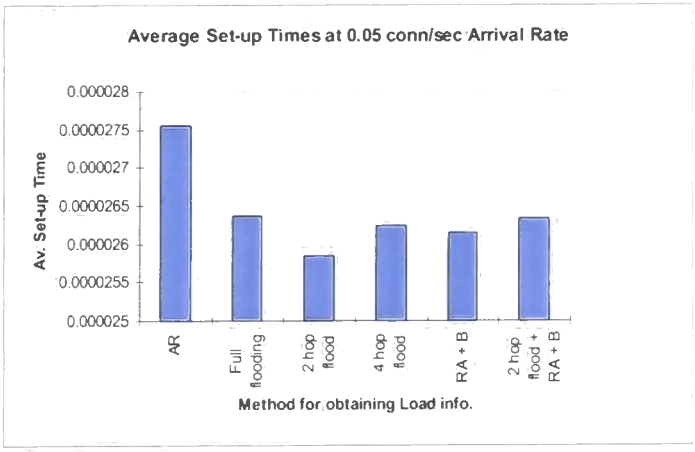


Figure 17: Resulting set-up time performance of link-sate updating methods

3.6 Summary

This chapter has examined the necessary issues linked with proposing a current-generation routing algorithm that performs well. Its performance can then be compared in subsequent chapters to that obtained from learning automata based schemes.

In order to permit QoS routing, the CAC mechanism was initially examined. For simplicity, a bandwidth-based mechanism was proposed. Effective bandwidth calculations are used in order to reserve the required network resources so that the connection's QoS can be guaranteed. Whilst methods for accurately calculating the effective bandwidth a priori of voice or data connections are available, those for MPEG video streams grossly underestimate the actual bandwidth required. A new method for calculating the effective bandwidth of MPEG streams is proposed, and is found to result in accurate characterisation of bandwidth requirements.

Next, current dynamic routing algorithms which are link-state based are evaluated based on previous studies. Four main types are outlined: widest-shortest path, shortest-widest path, dynamic-alternate path, and shortest distance path. The strengths and weaknesses of each type are given, the section ending with the proposed algorithm (AAMH) being outlined. The ensuing simulation experiments compare AAMH with an algorithm from the widest-shortest path variety (AMH). The simulation results, gathered from both fully-connected and sparsely connected logical topologies, show the benefits of AAMH over AMH. With symmetrical traffic loading patterns, trunk reservation is required for AAMH to match AMH performance under higher loads. Otherwise AAMH consistently out-performs AMH due to the extra paths available to its route decision-making process.

In order to make dynamic routing decisions, the link-state information must be propagated throughout the network. Each node having perfect network state information can therefore be approximated using frequent link-state message updates. However tuning the frequency of link-state update messages requires a careful understanding of the tension between network overheads and the accuracy of routing decisions. In order to reduce the amount of network processing and bandwidth related to the update messages, the link-state information propagation method is examined with the view of reducing this overhead whilst retaining good routing algorithm performance. The strengths and weaknesses of both periodic and triggered updates are summarised from previous work, and pre-planned event triggers are proposed for use in this study as it is felt, amongst other factors, that such a method would be more likely to be utilised in real network due to it being linked with pre-planning of both user demands and the network capacity and configuration. Results from simulation experiments of the 28 node sparsely connected topology are then given, in order to

show the effect of discretising the utilisation space into load bands. Whilst it is evident that the routing performance is not as high as with perfect network state information, yet it is higher than the best possible shortest-path algorithm which does not use link-state in its routing calculations. This therefore shows that the number of message updates can be greatly reduced by using load bands, whilst still retaining some blocking probability improvements resulting from using the link-state information in the routing decision-making process. Also, throughout the traffic loading range it is evident that the connection set-up time is greatly reduced by using link-state information to guide the routing process. Therefore the use of load bands is validated in that it allows AAMH to return superior routing performance than the best shortest-path routing algorithm which does not use utilisation as a variable in its route calculation.

Finally, explicit methods are examined for reducing the network processing and bandwidth overhead incurred by propagating the link-state update messages. Previous work has highlighted both limited update distribution methods and use of locally available information. Examples of the former include hop-count limited flooding and reverse path updates; whilst of the latter are caching reject information and use of local link status. A new method is proposed which by using existing connection set-up signalling incurs no bandwidth overhead and little processing overhead. By including reverse path update, caching reject information and local link status ideas, the proposed update method was compared in simulation experiments solely with hop-count limited flooding. The results from these simulations indicate that the proposed method results with routing performance close to that obtained when using full flooding. What was also of note was that according to the traffic loading type, a hop-count limited flood equal to the average route length will return a routing performance equivalent to that obtained when using full flooding.

The outcome of this chapter's study is an implementable algorithm which returns good routing performance, both in terms of low blocking probability and acceptable connection set-up times.

4 Performance analysis of various learning automata reinforcement algorithms

4.1 Introduction

The purpose of the following chapter is to highlight the currently used reinforcement algorithms that have superior performance for learning automata interacting with non-autonomous environments, the function of routing in a communications network being of such a type. These best performing algorithms can then be used as the baseline comparison to those learning automata based methods with the proposed improvements detailed in the later chapters.

Previous studies have given performance indicators for the various reinforcement algorithms when used with stationary and switching environments. The conclusions drawn from these studies have then been assumed to hold true for learning automata interacting with non-autonomous environments. The aim of the work detailed in this chapter is to give a framework for rigorously assessing the performance of the currently used reinforcement algorithms for learning automata interacting with non-autonomous environments.

The analytical basis for the framework is initially given, from which the converged action probabilities and blocking rates are obtained for a relatively simple network scenario. The remainder of the chapter deals with the experimental analysis to obtain the performance of the various reinforcement algorithms, the performance indicators of interest being both speed of action probability convergence and the subsequent steady-state accuracy.

Both standard and estimator type reinforcement algorithms are examined within this framework. Within these types of algorithm, the results for both continuous and discretised schemes are detailed with conclusions being drawn for each one as to its applicability for use in non-autonomous environments. Finally, the various results are brought together in the summary, with the best performing reinforcement algorithms being highlighted.

4.2 Learning automata for routing in multi-service networks

Multi-service networks using per-call reservation are a mixture of both circuit switched and packet switched network technologies. They can in some measure be thought of as virtual

circuits over a logical topology (i.e. VCCs over a VPC topology in the case of ATM, or RSVP over class reservations in IP with QoS). However, unlike previous virtual circuits in packet switched networks where the optimising factor was delay, multi-service networks can guarantee end-to-end delay if the user requests so in the specified QoS. Assuming the CAC used is bandwidth based, an effective bandwidth is assigned for a connection to meet the specified QoS, causing the call routing function to occur as in the circuit switched case. The difference is that since multi-service networks allow different reservations of bandwidth for different types of call, such as voice or video, so they are analogous to a multirate circuit switched network.

Learning automata may be applied to the problem of call routing in multi-service networks in a similar way to the circuit switched case. By using the P-model response environment, the network response to a connection request is either 0 for a successful routing attempt or 1 for a blocked connection. Enhancements to this method will be proposed in a later section. Using the P-model response environment, ε-optimal type reinforcement algorithms will therefore equalise the blocking probability, whilst ergodic type algorithms will equalise the blocking probability rates.

Two main traffic types will be used for this study: both voice and video traffic. The effective bandwidth for either can be calculated by the equivalent capacity equation given in [44], the MPEG calculation requiring the use of the method outlined in chapter 3.

4.3 A framework for obtaining relative reinforcement algorithm performance

Generally speaking, all the standard reinforcement algorithms are presented in the literature with accompanying theoretical and simulation analysis for performance characteristics when operating in stationary random environments. Performance analysis for non-stationary time-varying environments, including Markovian switching environments, has also been undertaken in other studies [32]. The results from these studies have in general been accepted as valid for learning automata operating in non-stationary non-autonomous environments, most recently causing the prevalent use of discretised schemes [17, 34]. However, the validity of this assumption remains unclear, and it is therefore thought important to undertake a study comparing relative reinforcement algorithm performance for learning automata operating in non-autonomous environments.

4.3.1 Performance metrics

Previous studies that have applied learning automata to the network routing problem have obtained a characterisation of the action probability values after convergence for the two performance related reinforcement algorithm classification [18]. ε-optimal schemes tend to equalise the blocking probabilities of their various actions, whilst ergodic schemes tend to equalise the blocking probability rates. These characterisations enable the calculation of the average penalty rate and action probabilities of a single automaton after convergence. However, the speed of convergence and ensuing steady state accuracy has not been characterised in any way. It is these two performance metrics which are of most interest in communication network situations as the network state can rapidly change due to multiple varying traffic sources and dynamic routing algorithms, and unvarying steady state accuracy improves user perceived network performance under steady-state situations.

4.3.2 Framework outline

Whilst no analytical analysis currently exists for learning automata interacting with a non-autonomous environment, analytical techniques are available for the network routing problem. Therefore when learning automata are applied to such an environment scenario, the same analytical techniques may be used to gain steady-state expected performance characteristics, these therefore occurring after convergence.

Erlang's Loss formula characterises the average blocking probability for a link, taking as inputs the connection arrival rate and mean holding times, and the size of the link. This may be modified by including the action probability effects, as shown in appendix C. By so doing, the expected average blocking and action probabilities may be analytically derived for both ε-optimal and ergodic schemes.

Having obtained analytically the action probability to which a reinforcement algorithm will converge, both the speed of convergence to that value and the steady-state accuracy thereafter may be gained experimentally. Doing so for all the main reinforcement algorithms currently found in the literature results in their relative performance indices for learning automata operating in non-autonomous environments.

The resulting best performing reinforcement algorithm of those currently used will then be utilised as the baseline when comparing the proposed improved algorithms.

4.3.3 Analytical results

A simple network routing scenario which has been used in the literature to show the benefits of learning automata routing over fixed rule routing [18] is shown in Figure 18. As may be seen, there is one traffic source on node 1 generating traffic for node 3. The link size units are given in multiples of voice bandwidth connections (each connection being 0.026 Mb/s). Using a mean connection arrival rate of 10 per minute, and a mean holding time of 6 minutes per call, we would intuitively expect the optimum call blocking probability to be around 0.5 as there are 30 free units of voice call bandwidth and about 60 call requests in a call time period.

Figure 19 shows analytically what should occur using learning automata and an arrival rate of 10 calls per minute. The c1 and c2 traces were calculated using Erlang's Loss formula, as per appendix 3. cT is the total or overall blocking probability based on the action probability p1 (p2 being 1 - p1) and is calculated as the addition of the two traces p1c1 and p2c2. It shows that the result of using the LRI or LRεP reinforcement algorithms will be p1 converging to 0.676, since ε-optimal schemes tend to equalise the blocking probability which is the intersection between traces c1 and c2. When using LRP p1 should converge to 0.578 as ergodic schemes tend to equalise the penalty rates, which is the intersection between the traces p1c1 and p2c2. At this arrival rate we see that the cT trace is fairly flat for a range of p1, so that even though the two different schemes converge to slightly different p1s, yet their overall blocking probability is very similar at around 0.53 .

Figure 20 shows the same convergence diagram for different arrival rates resulting in under 10% to about 80% blocking probability. The concern here is to see whether the arrival rate affects the convergence of the p1 action probability, and the effect on cT.

We find that at low arrival rates, the graph for cT is no longer flat but a pronounced curve. This would imply a noticeable difference in asymptotic performance between LRI/LRεP and LRP schemes, as p1 converges to different values according to the scheme.

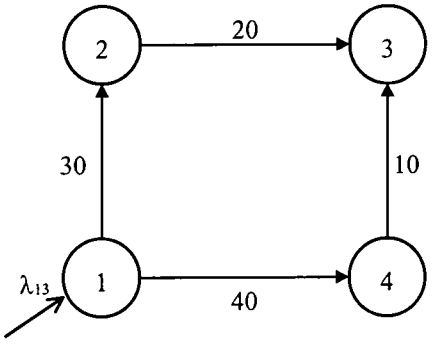


Figure 18: Four node network

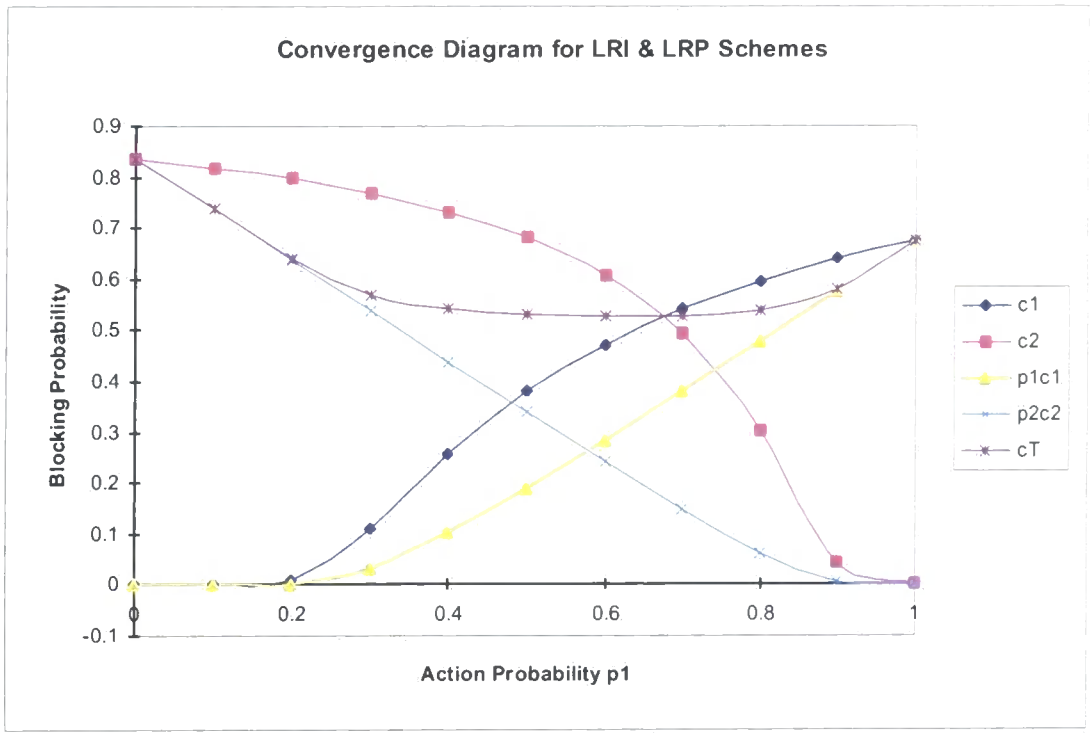


Figure 19: Learning Automata Convergence Under λ = 10 calls/min

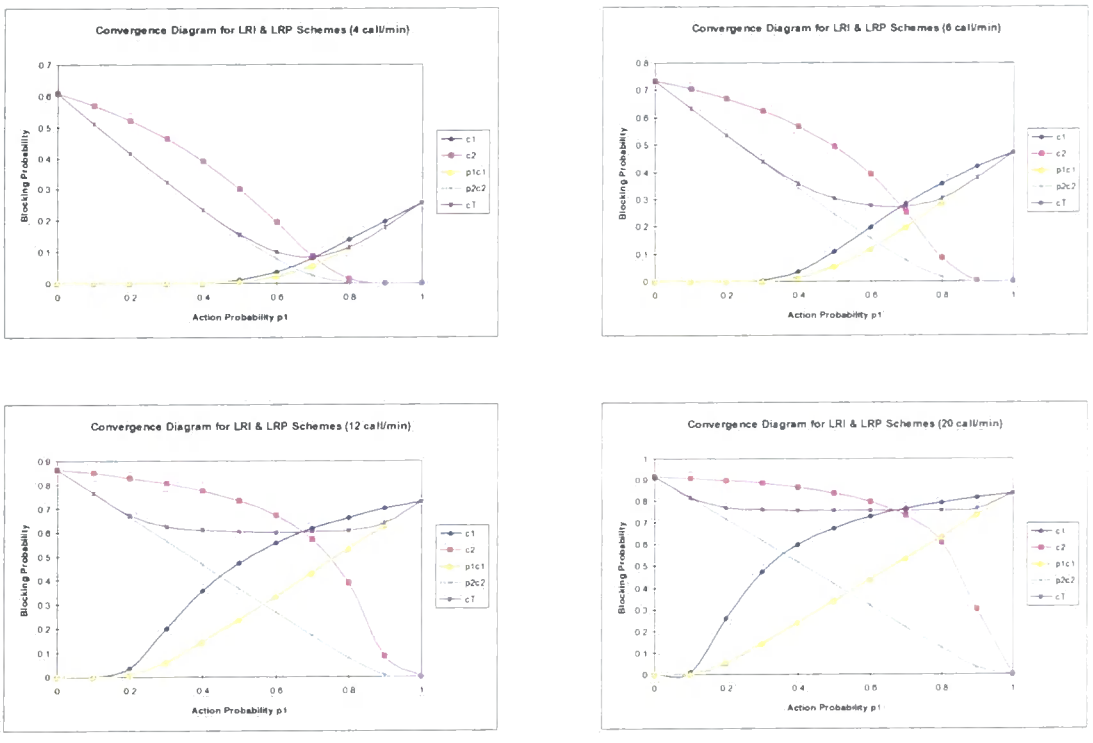


Figure 20: Influence of Arrival Rate on Convergence

As the arrival rates increase so cT 'flattens out' causing the blocking probabilities resulting from either scheme to be fairly similar. As regarding the convergence of p1, we find that for the LRI scheme it remains fairly constant over all arrival rates, at around 0.67, increasing slightly at low arrival rates. For the LRP scheme however, p1 is close to 0.67 under low arrival rates, but decreases as the arrival rates increase to 0.53 under around 77% blocking probability. This equates to it being closer to that for LRI under low arrival rates, with it being more distant under higher arrival rates.

We conclude that the ε-optimal class schemes should be fairly unaffected by arrival rate, but the ergodic class schemes have a noticeable variance in p1 according to the arrival rate. Combining these observations with the effects of arrival rate on cT, we expect that this variance in p1 between the two types of schemes should not result with variance in performance under higher arrival rates but only under low arrival rates. Telecommunications networks are generally dimensioned to a blocking probability of up to 10% [46], so it is reasonable to assume future multi-service networks will operate with a similar loading as they will carry voice and other traffic types with stringent QoS requirements. Therefore realistically it is envisaged that ε-optimal class schemes would produce a noticeable performance improvement were learning automata to be used for the routing function.

4.4 Experimental results

The following experimental results were obtained using one hundred simulation runs with varying random number generator seeds, subsequently obtaining the mean and 90% confidence intervals for each connection attempt from all the simulation runs' results. Multiple simulations are required to obtain each graph due to the stochastic nature of the learning automata, so that smooth behavioural traces are an average of multiple runs.

As a number of reinforcement algorithms are available for learning automata, there is the requirement to select one that might be most suitable for this application or environment type. Performance in this case is the number of feedback messages required for convergence to an analytically calculated action probability, and the level of variance from that value once convergence has occurred.

4.4.1 Basic algorithms

The following basic reinforcement algorithms have commonly been used in various environments: LRI and LRP, with LRεP also sometimes being used due to its joint property

of ε-optimality and ergodicity. However, the discretisation of such algorithms has been applied in more recent studies, often taking the analytical stationary environment conclusions as valid for the non-stationary case.

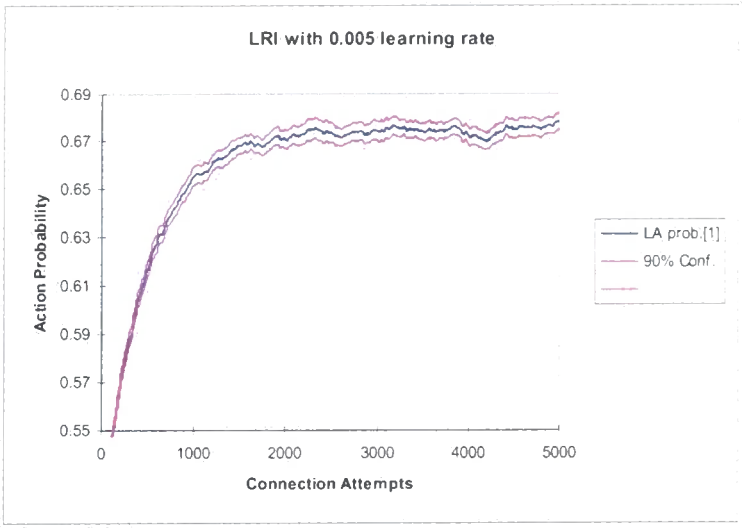
4.4.1.1 Continuous algorithms

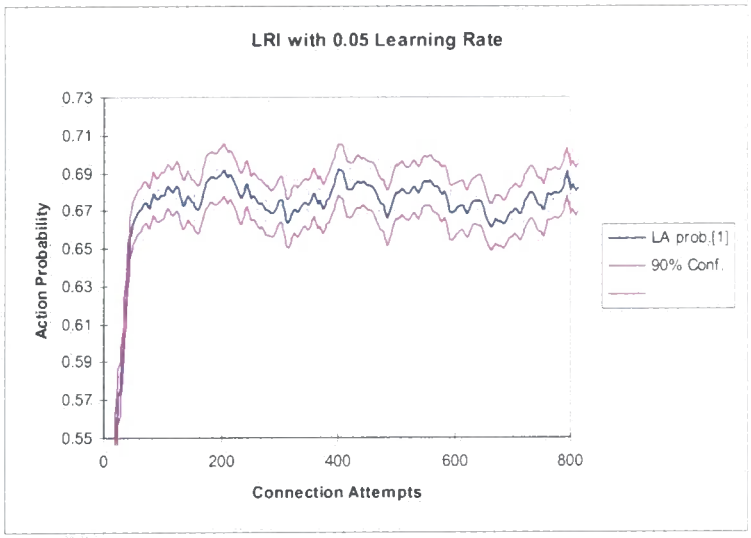
The main factor in differing speed of convergence performance between LRI and LRP reinforcement algorithms is that in addition to utilising the reward feedbacks, the LRP algorithm also utilises the penalty feedbacks. For scenarios where each member of the penalty probability set has a high value, it would therefore be expected that the LRP algorithm would converge within a much smaller number of overall feedback responses, when using the same learning rate. This scenario has a blocking probability and thus penalty probability of 0.53 after convergence, so it is expected that the LRP algorithm will converge at around half of the time taken by the LRI algorithm.

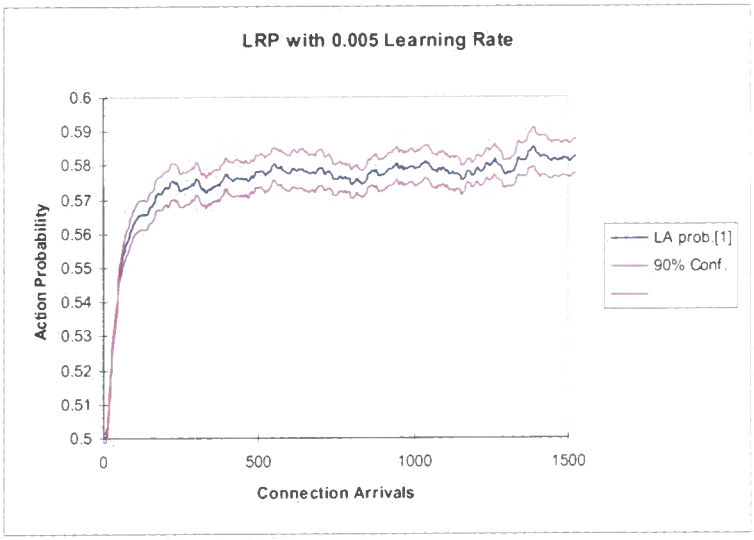
Figure 21 and Figure 22 show how the action probabilities vary with the number of connections attempted for both algorithms using different learning rates. As might be expected, the higher the learning rate the faster the convergence and the higher the variance for both algorithms. This is shown more clearly in Figure 22 where the spread of values with 90% confidence after convergence is plotted.

The LRP has faster convergence than expected, and should converge faster than LRI even when the penalty probabilities are low. This faster convergence is not just due to the lower action probability value after convergence, as is shown by Figure 23. Here the minimum blocking rate was 5%, and gives LRI converging after around 2710 connection attempts with LRP doing so after about 1620 connection attempts, even though the converged action probabilities are similar. Rather than the 5% speed increase, LRP is seen to have a 40% speed increase. This initially counter-intuitive statement is explained by recalling that the penalty rate is 5% only after convergence, but higher previous to that point.

LRP also has higher variance than LRI when using the same learning rate, with the trace for 0.05 learning rate being similar to a standard control trace which has the gain too high causing overshoot. This result implies that the learning rate for the LRP algorithm should be lower than that for the LRI algorithm to obtain similar performance of low variance after convergence.


LRI with 0.005 learning rate
Action Probability
0.69
0.67
0.65
0.63
0.61
0.59
0.57
0.55
0
1000
2000
3000
4000
5000
Connection Attempts
LA prob.[1]
90% Conf.


LRI with 0.05 Learning Rate
Action Probability
0.73
0.71
0.69
0.67
0.65
0.63
0.61
0.59
0.57
0.55
0
200
400
600
800
Connection Attempts
LA prob.[1]
90% Conf.


LRP with 0.005 Learning Rate
Action Probability
0.6
0.59
0.58
0.57
0.56
0.55
0.54
0.53
0.52
0.51
0.5
0
500
1000
1500
Connection Arrivals
LA prob.[1]
90% Conf.

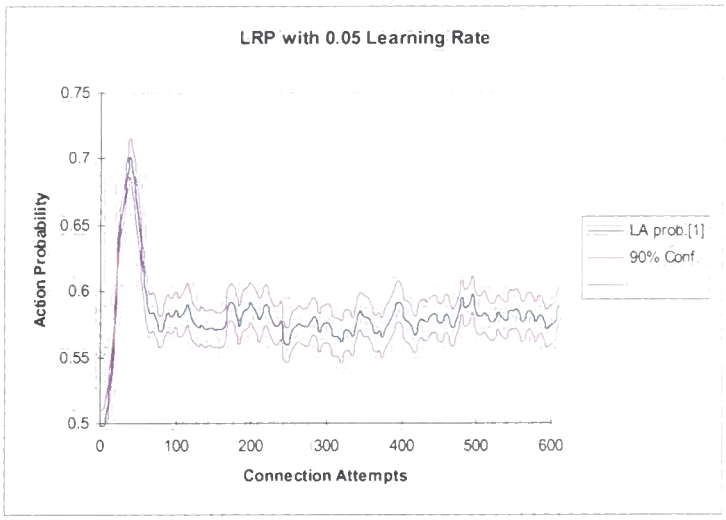


Figure 21: Convergence for LRI and LRP

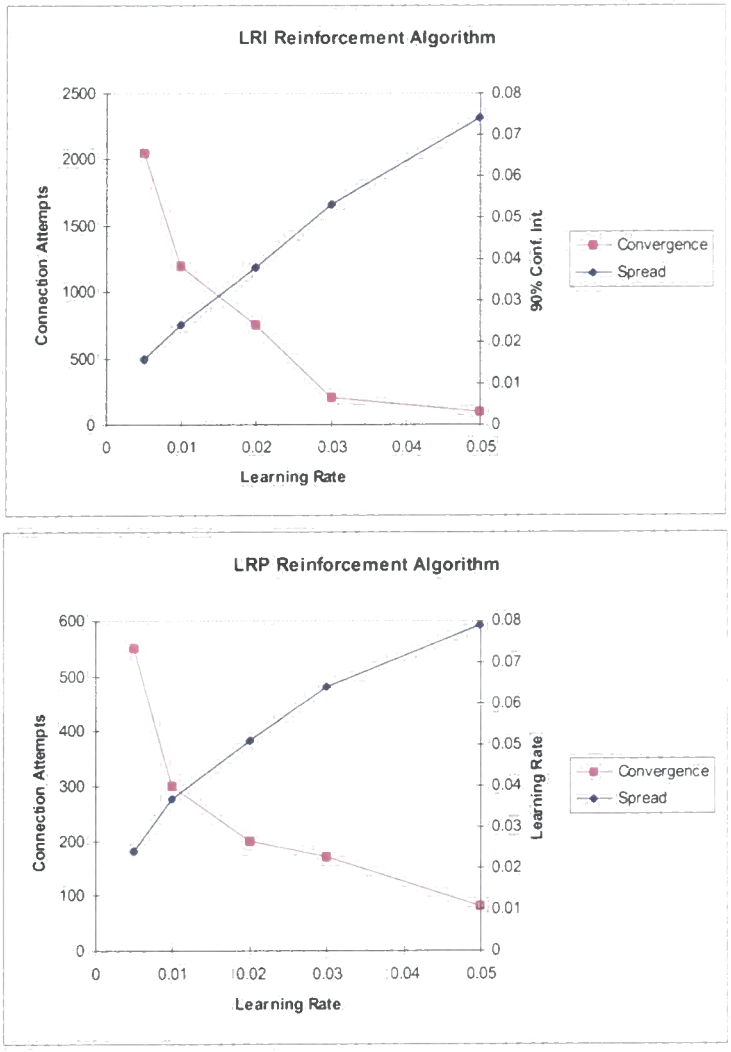


Figure 22: Convergence Properties for LRI and LRP Algorithms

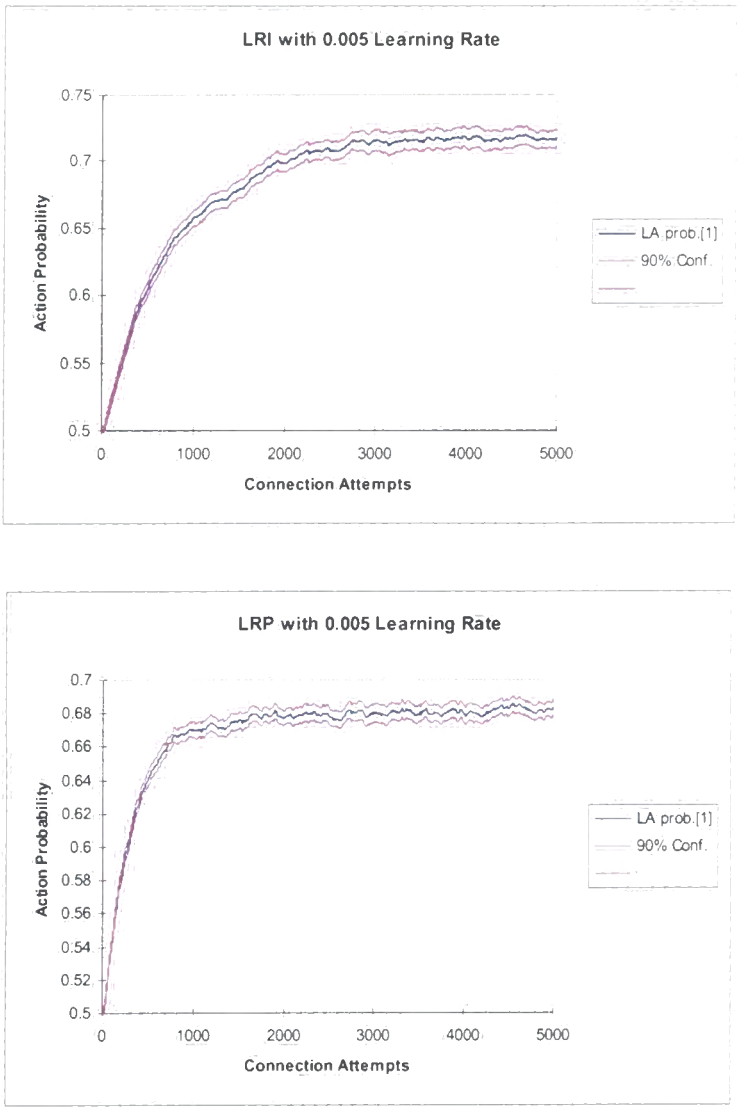
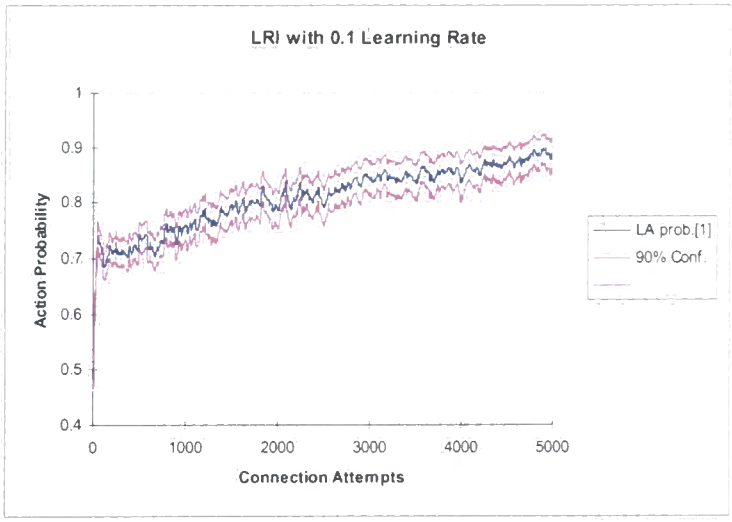
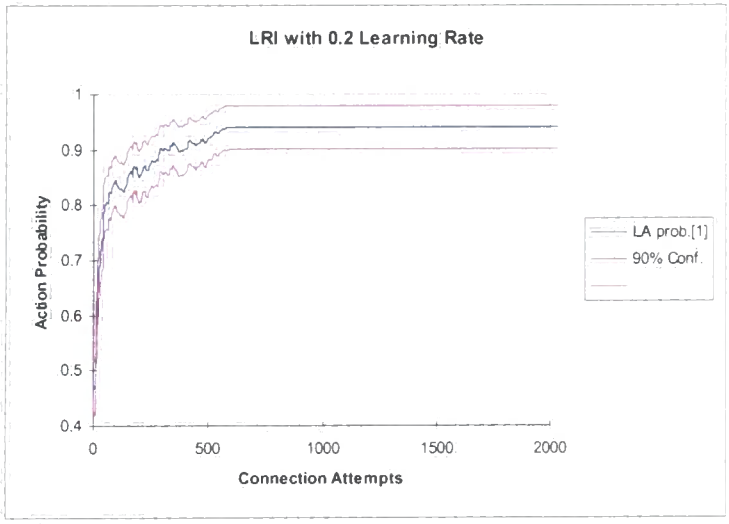


Figure 23: Convergence for LRI and LRP with low Penalty Rates

Intuitively it might be thought that by increasing the learning rate of LRI in order to produce the same degree of overshoot seen with LRP would result in a convergence rate of twice that of LRP for this scenario. However, when attempting to increase the learning rate for LRI to match the initial overshoot properties before convergence of LRP, unexpected results ensued. Figure 24 shows LRI with learning rates of 0.1, 0.2 and 0.3, and indicates that there occurs a rapid movement to the expected action probability, but rather than converging the action probability continues moving towards unity. With the very high learning rate of 30%, convergence occurs to a fixed value lower than with the previous case, due to the high granularity effects. This implies that high learning rates may not be used when employing LRI in non-autonomous environments.





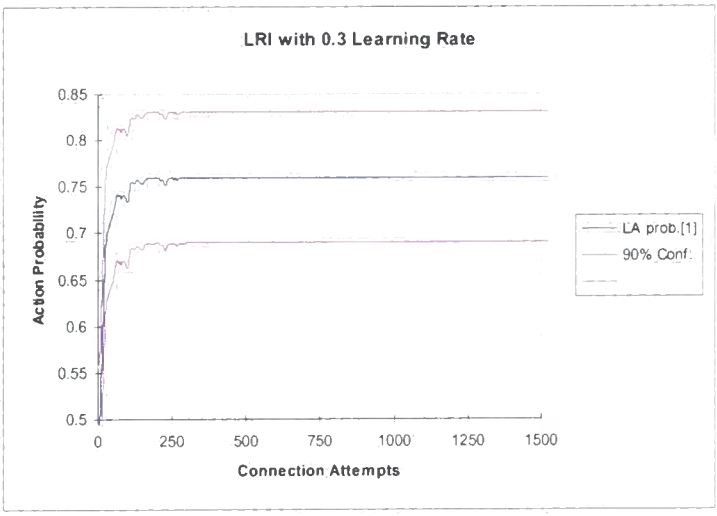
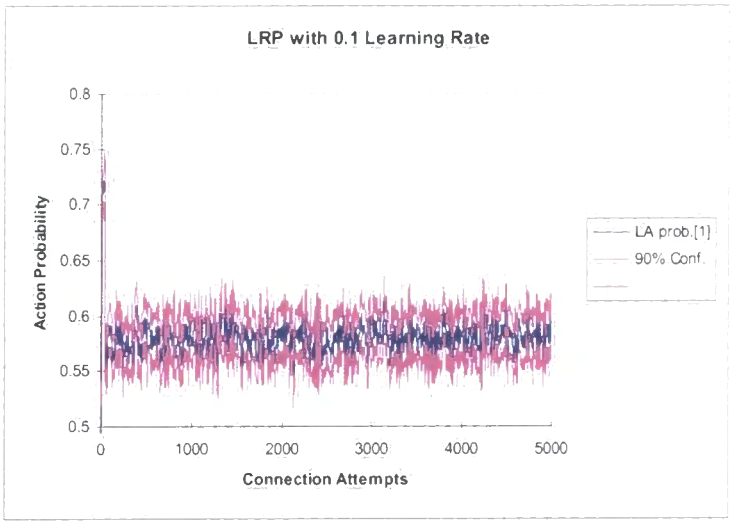


Figure 24: Convergence for LRI using high Learning Rates



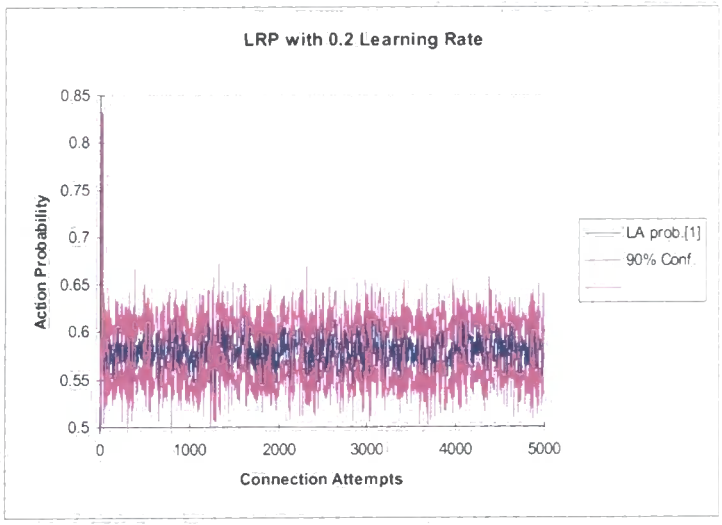


Figure 25: Convergence for LRP using high Learning Rates

Previous studies have commended the use of the LRI algorithm for such environments [38, 39] but it is evident that its poor steady-state performance with higher learning rates was not known. The LRI algorithm is therefore not recommended for general use in such environments.

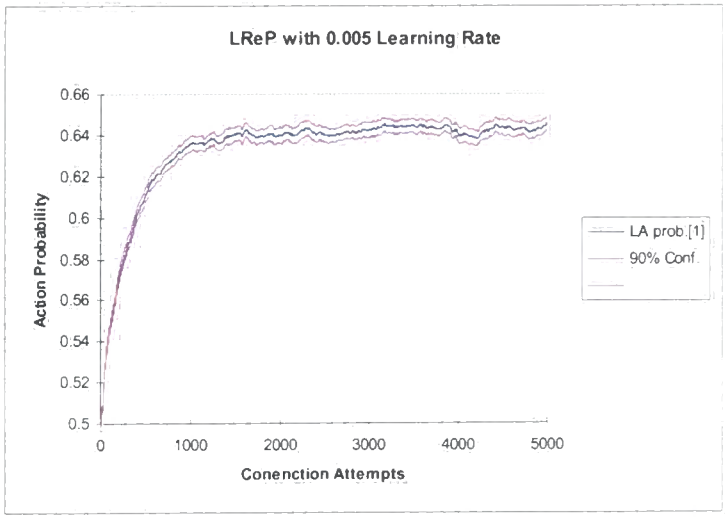
The LRP reinforcement algorithm does not suffer from this effect when using very high learning rates; Figure 25 showing that the variance simply increases once convergence has occurred.

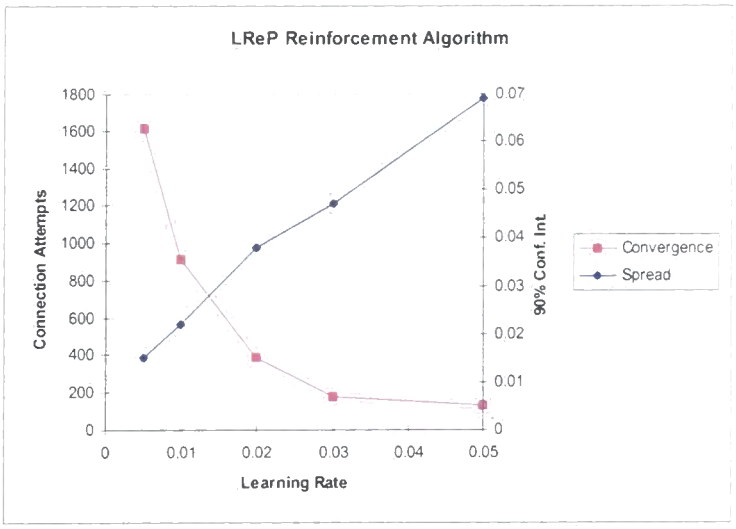
Figure 26 shows the convergence properties of the LRεP algorithm. The results indicate a faster convergence than the LRI algorithm, but slower than the LRP algorithm. Of greater interest is the low overshoot and variance that LRεP exhibits, being lower than even the LRI algorithm. Using very high learning rates, the LRεP algorithm does not exhibit the

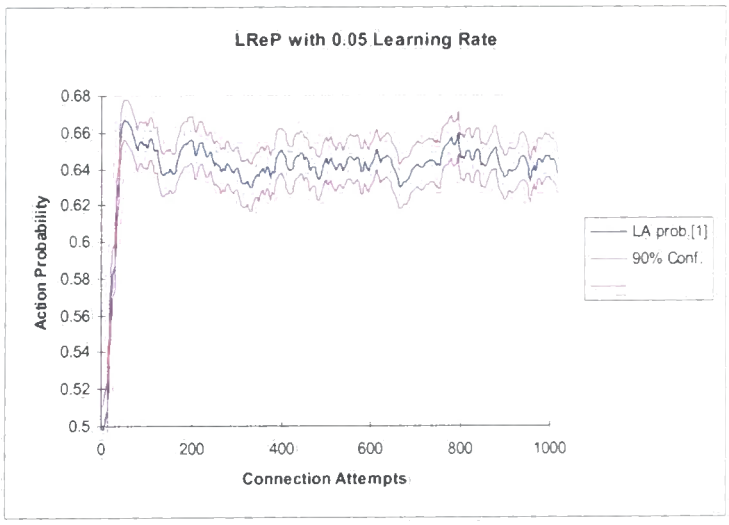
same loss of convergence behaviour of LRI, imitating rather the LRP algorithm that simply increases the variance after convergence.

The action probability converged to is 0.643 for this scenario, having a minimum penalty probability of 0.53. A previous analytical study showed that the two-action DLRP algorithm is both ergodic and ε-optimal in stationary random environments where the minimum penalty probability is less than 0.5 [31]. However, as shown in the next section, with scenarios of increasing minimum penalty probability the converged action probability tends towards that of equalising the penalty rates rather than the penalty probabilities. Whilst the literature for the LRεP algorithm does not mention the same, yet these empirical results indicate that this algorithm seems to bear the same property since with an increasing minimum penalty probability the converged action probability moves towards that for equalising the penalty rates. Further work is required to derive an analytical verification of this.

These results indicate that both the LRP and LRεP algorithms would be suitable for use in non-autonomous environments.







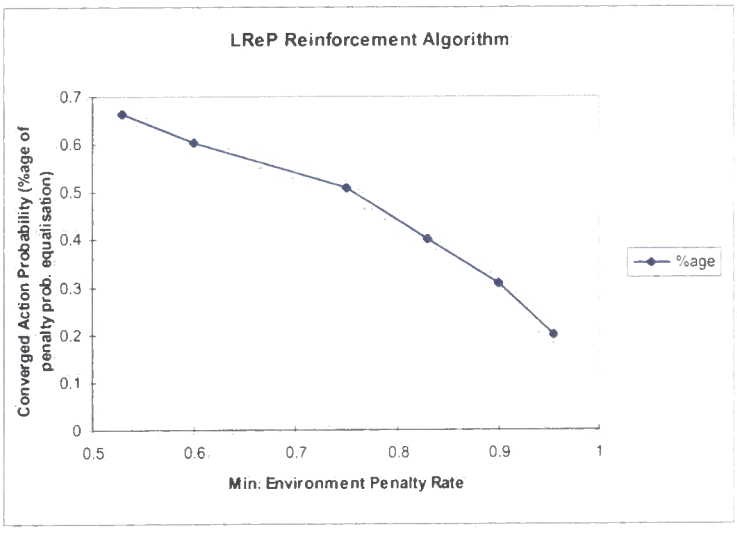


Figure 26: Convergence Properties for LRεP

4.4.1.2 Discretised algorithms

Discretised versions of continuous reinforcement algorithms have been shown to converge much quicker in stationary random environments, due to their linear rather than asymptotic convergence properties. They have also been used in non-stationary environments, even of the non-autonomous kind [17]. However the validity of so doing remains unclear, as their convergence performance has not been examined in the literature.

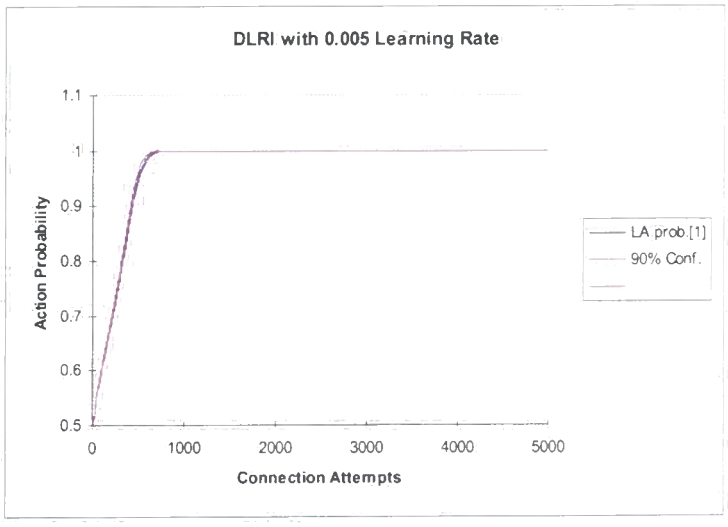
Figure 27 and Figure 28 show the results for the DLRI and DLRP algorithms, each having unexpected characteristics. For low learning rates (up to 0.02), the action probability converges to 1 instead of 0.676, the speed of convergence varying according to the learning rate. Once converged, the action probabilities remain fixed as penalty environment responses are not acted upon by the reinforcement algorithm. As the learning rate is increased, so the value to which the action probability converges decreases to below the analytically derived optimum of 0.676 . The unvarying mean and confidence intervals imply an oscillation in the action probabilities between a couple of values. These results indicate that the DLRI reinforcement algorithm should not be used for such environments as the action probabilities fail to converge.

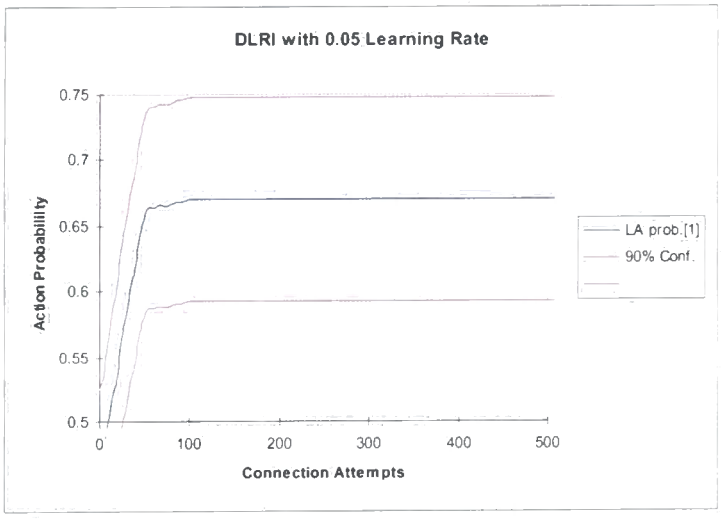
A previous study's blocking probability results [17] for the discretised algorithms do not vary from the generally accepted thinking on discretised performance because a fairly high learning rate of 0.1 was used for all the different algorithms. These results show this high learning rate to result with better steady-state performance than lower and more usual learning rates. Were a more usual learning rate of under 0.05 chosen the results would have shown a marked difference, with performance being much lower than the other algorithms.

The DLRP results do not show such unexpected behaviour in convergence, but do so in the final action probability steady-state value converged to. A previous analytical study showed that the two-action DLRP algorithm is both ergodic and ε-optimal in stationary random environments where the minimum penalty probability is less than 0.5 [31]. The results shown, where the penalty probability is 0.53 after convergence, indicate that for non-autonomous environments DLRP causes the action probabilities to converge to between those arising from equalising the penalty probabilities, and those resulting from equalising the penalty probability rates.

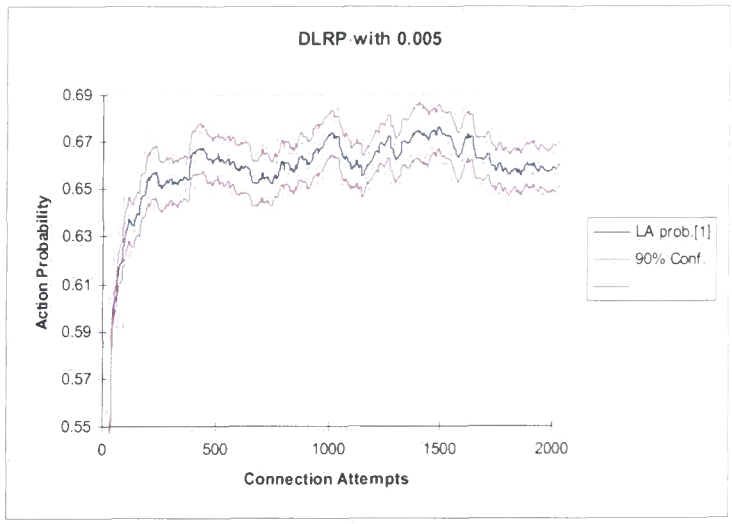
This is clear from Figure 29 where the effect of increasing minimum penalty probability is seen on the converged action probability. As expected, the mean action probability after convergence decreases with increasing minimum penalty probability from that obtained when equalising the penalty probabilities to that resulting from equalising the penalty rates. As it is reasonable to assume that multi-service networks will be dimensioned

to operate with a blocking probability of up to 10%, so the DLRP reinforcement algorithm is very suitable to use, ensuring ε-optimal performance. Moreover, it has faster convergence than its continuous counterpart but suffers from a higher variance after that point, evidenced by comparing Figure 28 with Figure 22.





(a): Convergence for DLRI



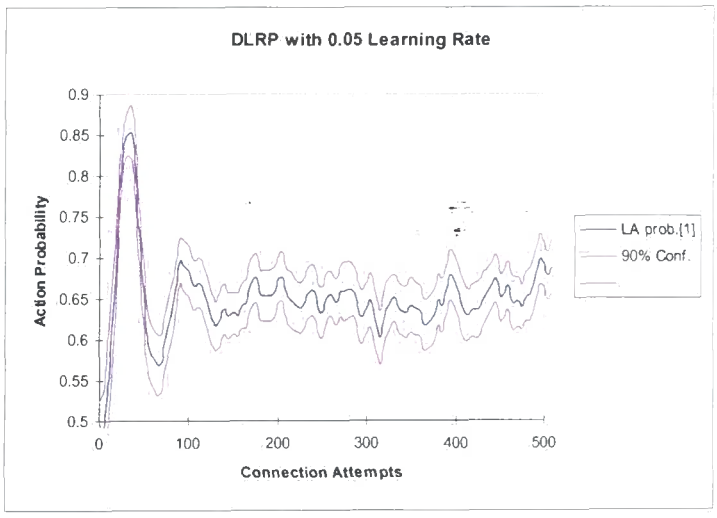


Figure 27: Convergence for DLRI and DLRP

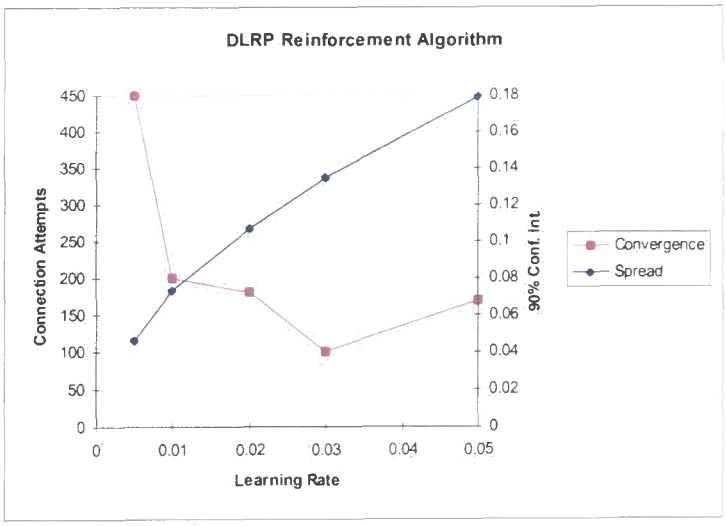


Figure 28: Convergence Properties for DLRP Algorithm

The graphs of Figure 27 also indicate that the effect of discretisation is to increase the control gain which causes the faster convergence, so that for the same learning rate as the continuous case, there is a greater degree of overshoot and number of oscillation before convergence occurs. Therefore for good steady-state performance, meaning low variance after convergence, a smaller discretisation value is required than the learning rate for the continuous case.

Figure 30 shows that the DLRεP algorithm takes after the DLRI rather than DLRP algorithm in performance. It fails to converge to the expected value, the action probability converging to unity for low learning rates, and progressively lower values as the learning rate is increased. It is therefore not recommended for use in non-autonomous environments.

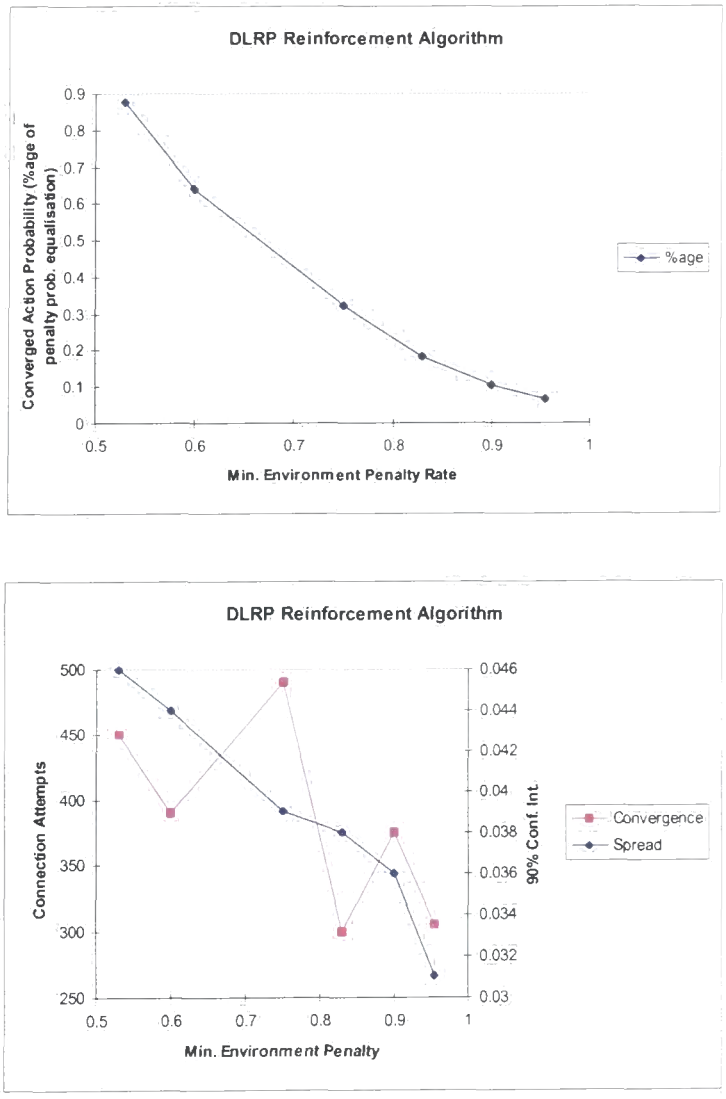


Figure 29: Convergence Properties of DLRP Algorithm under Increasing Minimum Penalty Probability Scenarios

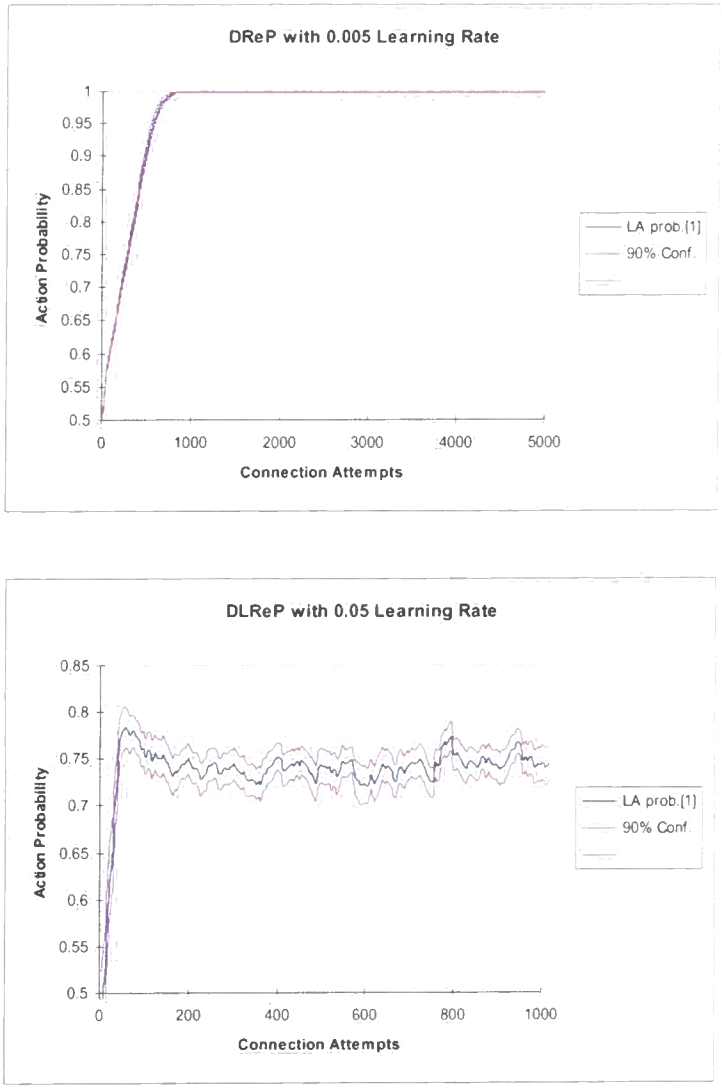


Figure 30: Convergence Properties for DLRεP Algorithm

4.4.2 Estimator algorithms

By reinforcing the action probabilities based on both the current environment response and stored history, estimator algorithms would be expected to outperform other types of algorithm. This has been shown to be the case for stationary and switching environments [25, 26, 63], but no known detailed example exists to date for non-autonomous environments, with studies such as [34] simply assuming its better performance as regarding convergence speed compared with other reinforcement algorithms.

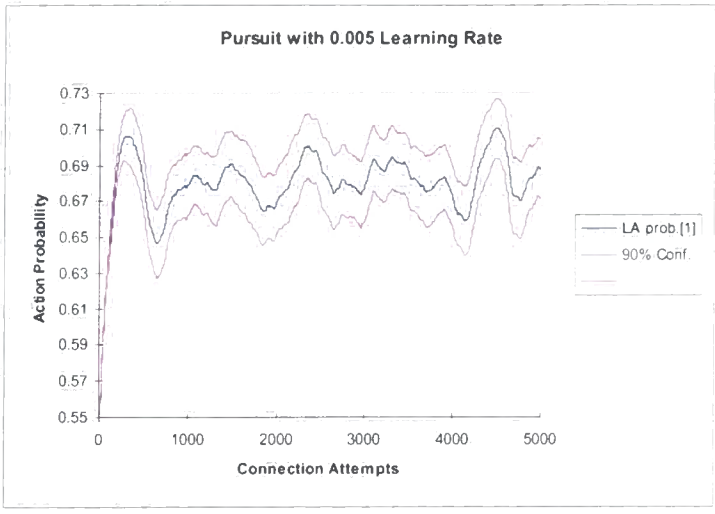
Two estimator reinforcement algorithms are currently used: the pursuit and the general estimator algorithms. As with the basic algorithms, the discretisation of these

estimator algorithms has occurred, with studies such as [34] assuming their steady-state performance in non-autonomous environments rather than validating it.

4.4.2.1 Continuous algorithms

The pursuit algorithm is the simpler of the two estimator algorithms, and its performance is shown in Figure 31 and Figure 32. It can be seen that even with a low learning rate the algorithm does not converge to its theoretical value of 0.676, instead overshooting its mean value as it takes some iterations before it can pursue another action. Interestingly, it is seen that the mean or converged value after initial overshoots is higher than the theoretic one, this increasing with the learning rate as shown by Figure 32. The variance or maximum spread with 90% confidence in possible values after convergence also increases with the learning rate, indicating that the gain in 'overshoots' is increased, as might be expected. As the pursuit algorithm does not cause the action probability to converge to the analytically calculated value, it is not recommended for use in non-autonomous environments.

The general estimator algorithm not only takes the learning rate as input parameter, but also the monotonic function type which forms part of the reinforcement function. For this experiment three different functions were used: 'x', 'x^3' and 'x^5' with various learning rates for each.



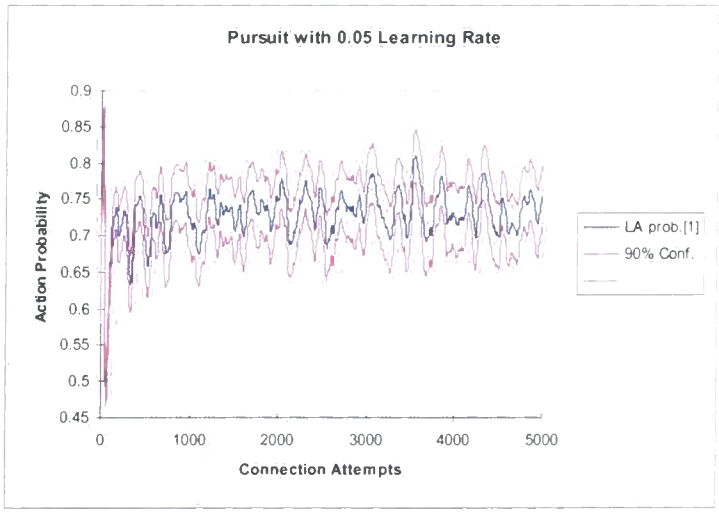


Figure 31: Convergence for Pursuit Algorithm

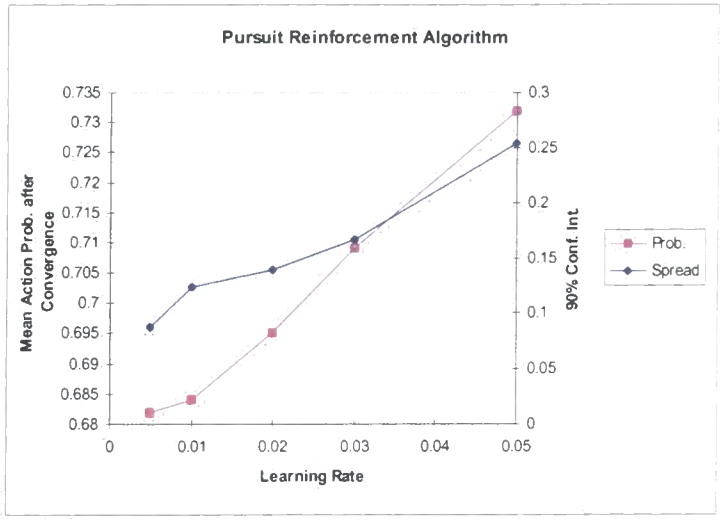


Figure 32: Convergence properties for Pursuit Algorithm

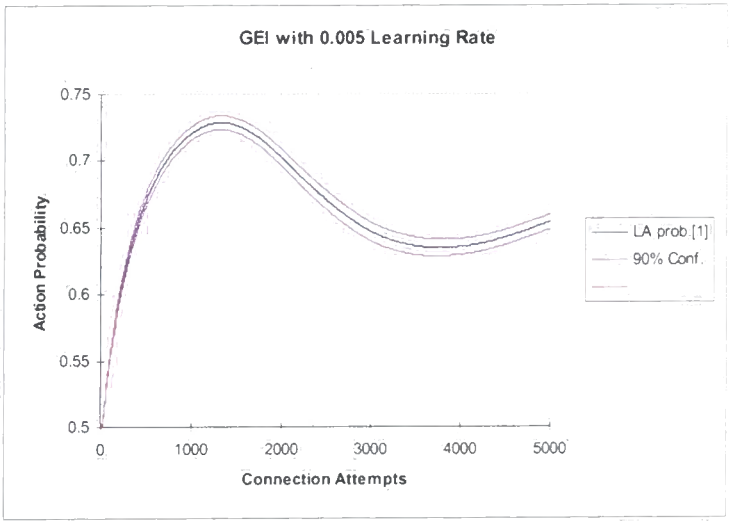
Figure 33 shows results for the general estimator algorithm for both the linear and 'x^3' updating functions. Both function types result with a long pre-convergence period, requiring more than 5000 connection attempts for both cases. Using the linear function results in overshoot of convergence after a relatively long period of time (more than 1000 connection attempts when using a learning rate of 0.005). By using very low learning rates, for example 0.001, overshoot still occurs to the same degree (0.73) but with a slower rate, requiring a relatively long interval of 7500 connection attempts. Changing to very high learning rates, such as 0.3, causes the action probability to converge fairly quickly but with a high variance, but as the number of connection attempts increase it drifts lower from its converged value.

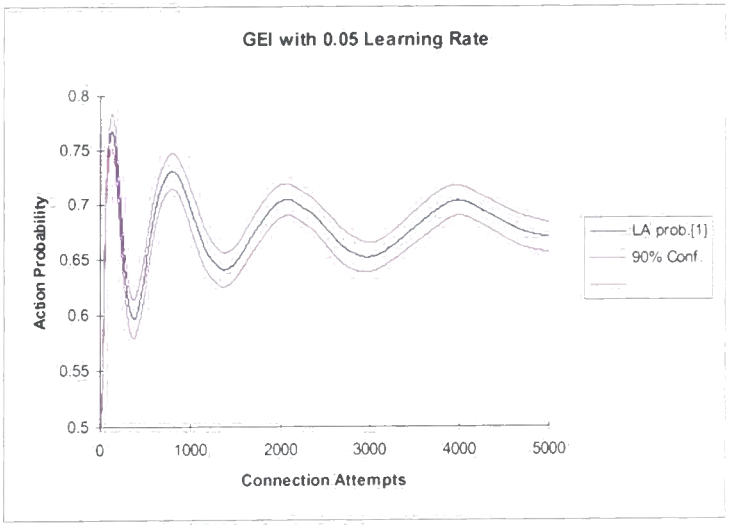
On the other hand, using the 'x^3' function results with little overshoot and a gradual convergence to the analytically derived action probability. As shown by Figure 34, by using

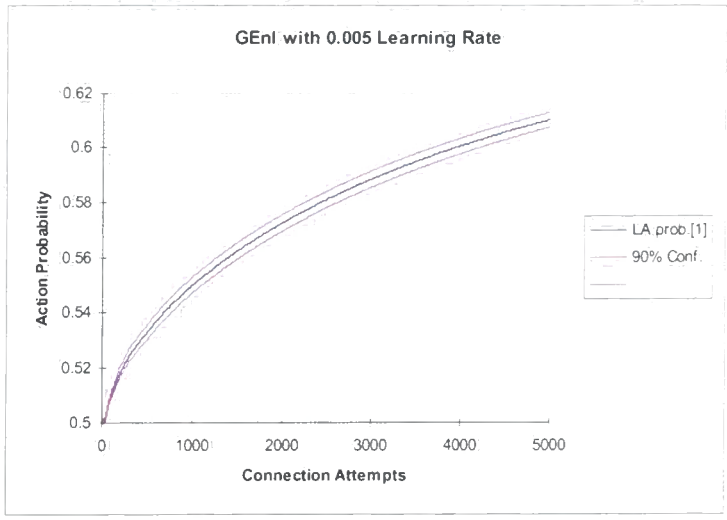
the much higher learning rate of 0.3, overshoot is increased but convergence occurs sooner but still relatively slowly, after 2430 connection attempts. For both updating functions, the possible spread of values with 90% confidence after convergence is seen to be small.

The effect of increasing the updating function to 'x^5' is shown in Figure 35. These preliminary results indicate that increasing the power of the updating function decreases oscillatory and overshoot behaviour as well as the time to converge, whilst causing the converged action probability to move slightly from its analytically derived expected value.

From these results it is seen that the general estimator reinforcement algorithm with a 'x^3' updating function coupled with high learning rate is suitable for use in non-autonomous environments, for although the convergence speed is slow the variance is low after convergence.







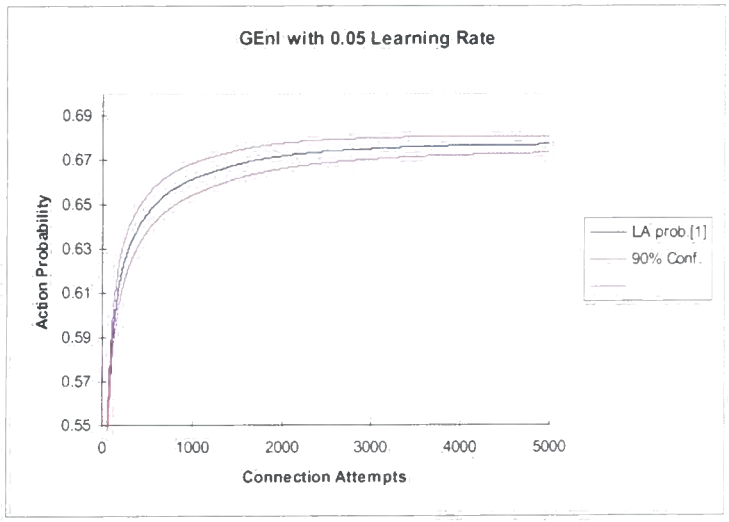
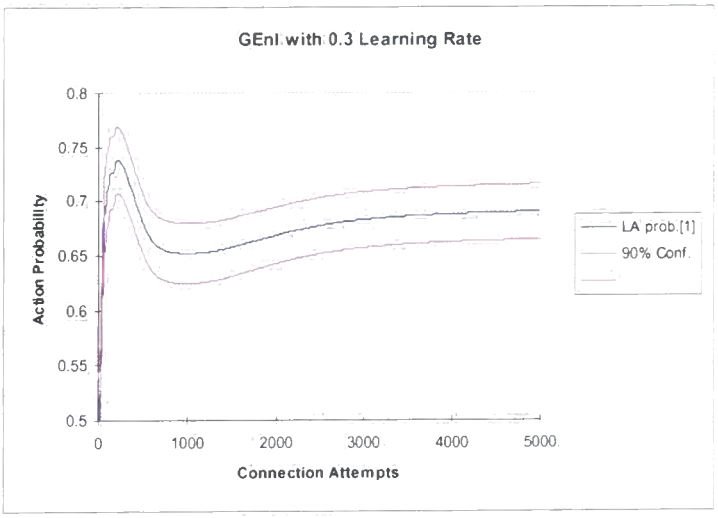


Figure 33: Convergence for GE Algorithm for both Linear and 'x^3' Non-linear Updating Function



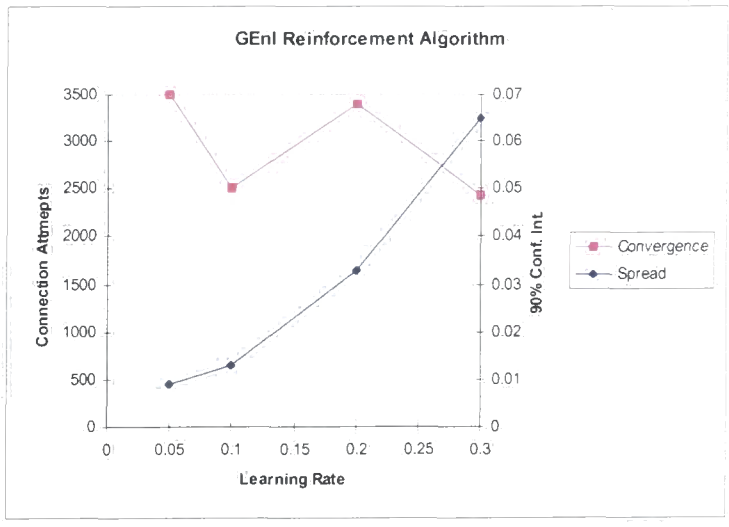


Figure 34: Convergence Properties of GE Algorithm for 'x^3' Updating Function and High Learning Rates

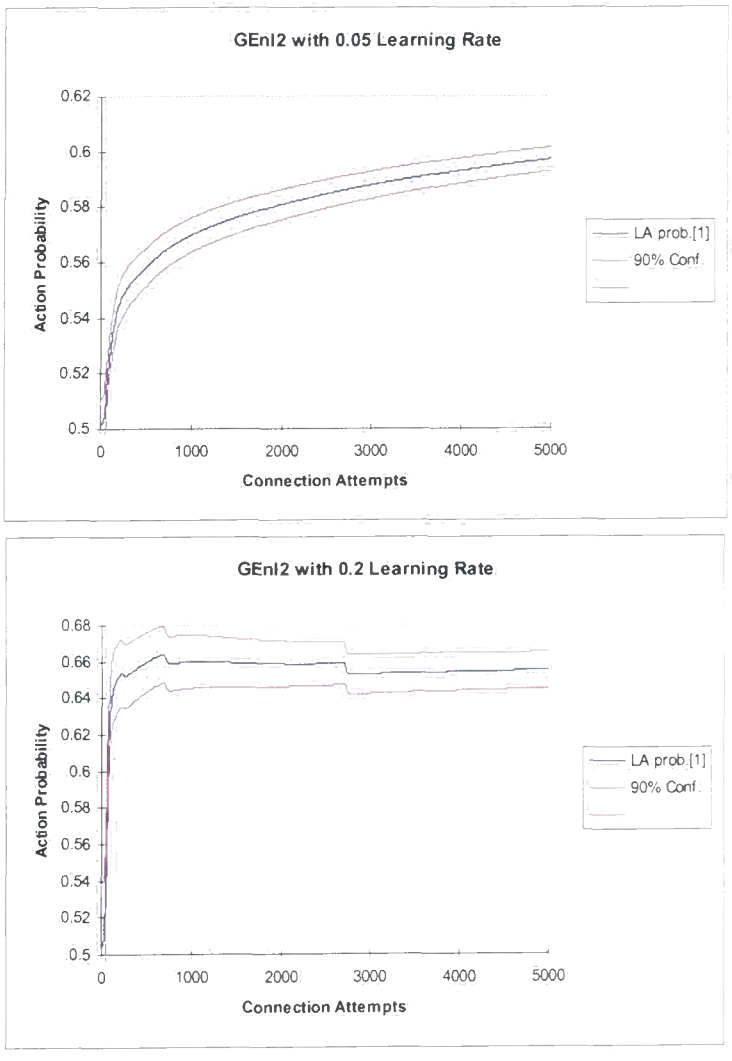


Figure 35: Convergence for GE Algorithm with 'x^5' Non-linear Updating Function

4.4.2.2 Discretised algorithms

As always, the rationale for discretising continuous estimator algorithms is to obtain a linear rather than asymptotic convergence of an action probability, resulting with much faster convergence in stationary or switching environments [27].

As with the continuous case, Figure 36 and Figure 37 show that the discretised pursuit algorithm does not converge to its expected analytical value. In fact it overshoots to a greater degree than with the continuous case, so that with a moderately high learning rate of 0.05 the action probability converges close to unity. Therefore as with the continuous variant, the use of the discretised pursuit algorithm is not recommended for non-autonomous environments.

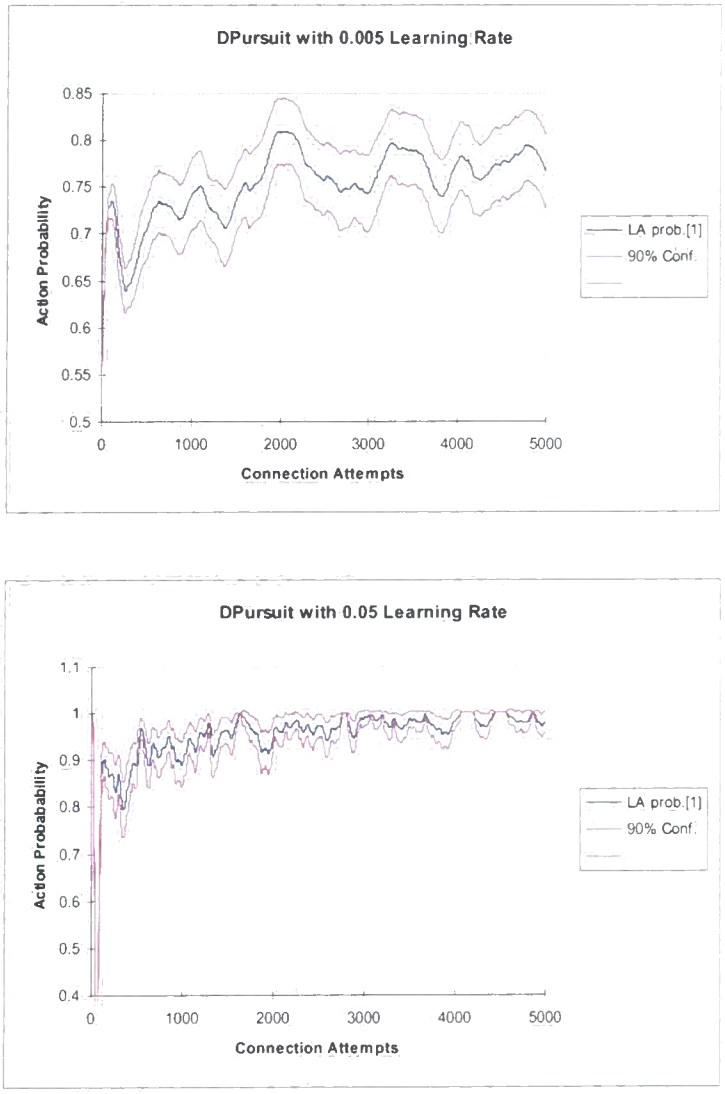


Figure 36: Convergence for Discretised Pursuit Algorithm

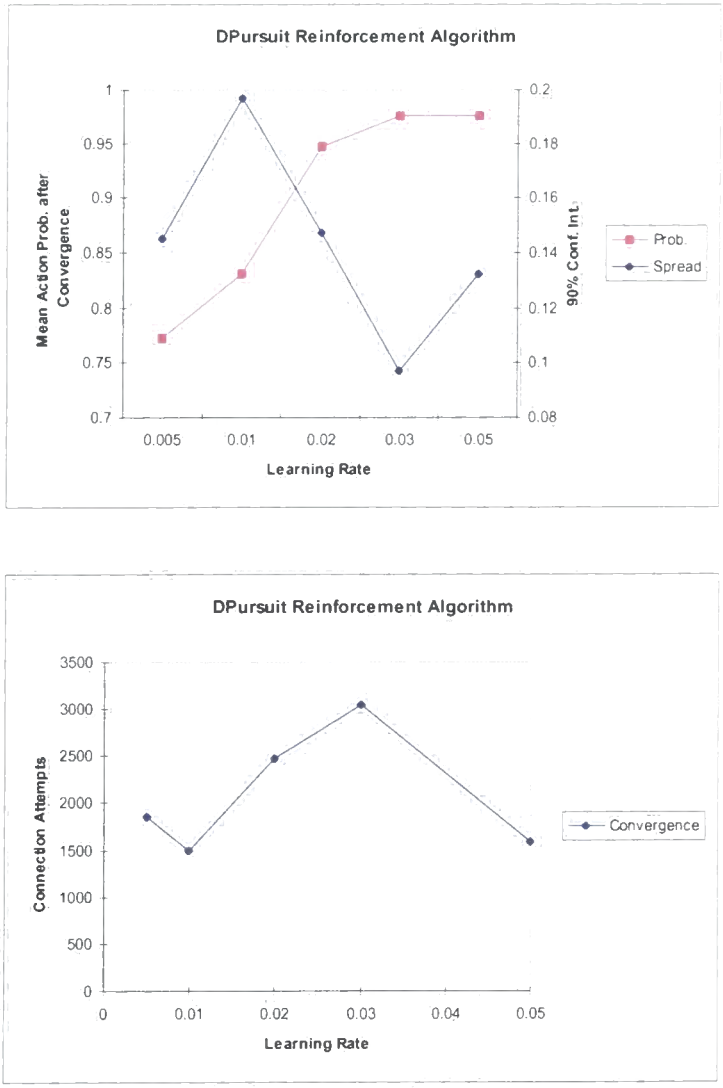
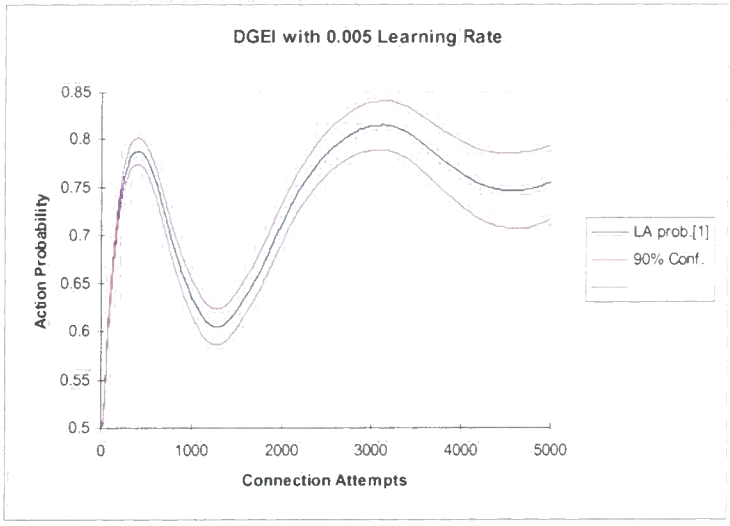
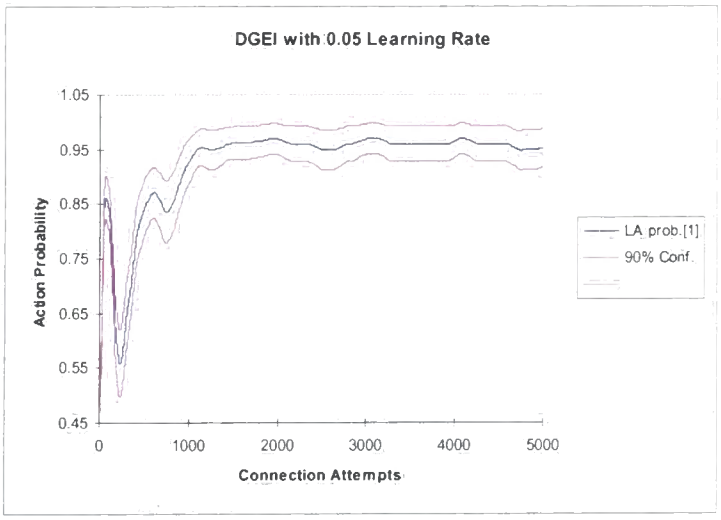


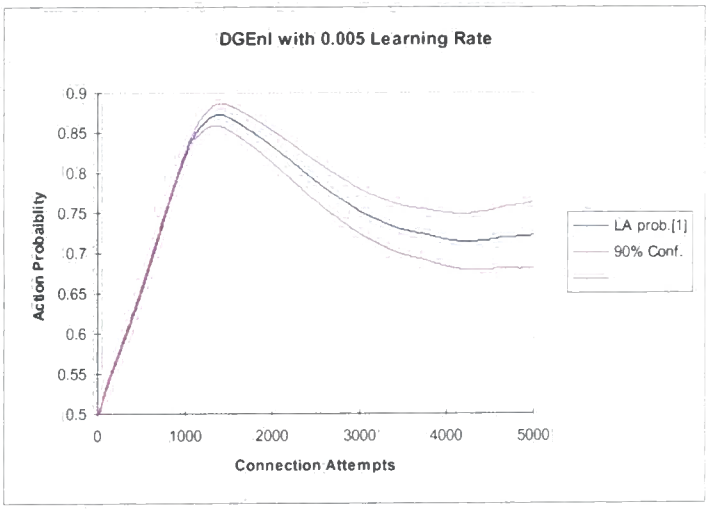
Figure 37: Convergence properties for Discretised Pursuit Algorithm

Figure 38 shows the results when using the discretised general estimator algorithm with both linear and 'x^3' updating functions. It is evident that discretising the general estimator algorithm causes it to lose the property to converge to the analytically derived value, instead overshooting towards unity as the learning rate increases. The effect of greater non-linearity with the 'x^5' updating function is to increase the overshoot away from the analytically derived value.

Therefore the use of the discrete general estimator algorithm for non-autonomous environments is not recommended.


DGEI with 0.005 Learning Rate
Action Probability
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
1000
2000
3000
4000
5000
Connection Attempts
LA prob.[1]
90% Conf.


DGEI with 0.05 Learning Rate
Action Probability
1.05
0.95
0.85
0.75
0.65
0.55
0.45
0
1000
2000
3000
4000
5000
Connection Attempts
LA prob.[1]
90% Conf.


DGEnI with 0.005 Learning Rate
Action Probability
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
1000
2000
3000
4000
5000
Connection Attempts
LA prob.[1]
90% Conf.

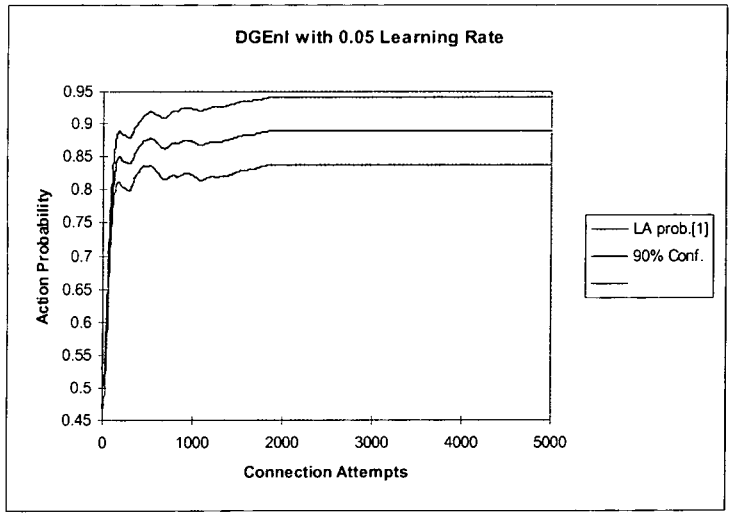


Figure 38: Convergence for DGE Algorithm for both Linear and 'x^3' Non-Linear Updating Function

4.5 Summary

The chapter began by highlighting the need for examining the performance of currently used reinforcement algorithms when learning automata interact with a non-autonomous environment. This is due to previous studies having analysed their performance with stationary and switching environments, and assuming the conclusions drawn as valid for non-autonomous environments. The results obtained with this study, however, are in opposition to this generally held assumption.

A well performing reinforcement algorithm in non-autonomous environments should produce fast convergence of action probabilities with low variation afterwards, and ideally converge to equalise the penalty probabilities rather than the penalty rates so producing lower penalty probability performance in the steady-state. Having obtained the converged action probabilities analytically for a simple network scenario, the convergence speed and steady state variance was obtained experimentally. The analytical work also resulted in the conclusion that at low blocking rates there would arise a noticeable difference in network blocking performance between ε-optimal and ergodic schemes, this not being noticeable at higher penalty rates.

Of the basic continuous algorithms, both the LRP and LRεP algorithms have been shown to converge faster than the LRI algorithm, although the variance afterwards is higher

using LRP. As the LRI algorithm fails to converge once the learning rate is increased to higher levels, it is not recommended for general use in such environments. This is an important result considering that this algorithm has been used in most previous studies involving learning automata for routing in networks. Another interesting contribution of this work has been to show that the LRεP algorithm seems to display the same property as the DLRP algorithm: that of tending towards equalising the penalty rates rather than the penalty probabilities as the minimum penalty rate for the scenario increases significantly above 0.5 .

It has been consistently noticed in this study that the effect of discretising continuous algorithms is to increase the gain so that convergence times are decreased and variance increased. However, due to it causing an action probability of unity to be approached linearly rather than asymptotically, convergence to the analytical value sometimes does not occur. Discretising the LRI and LRεP algorithms causes them to consistently fail to converge to the analytical value, and so they are not deemed suitable. The DLRP algorithm has increased variance compared to its continuous version, and converges slightly quicker. It also equalises the probabilities rather than the probability rates of its continuous counterpart. This algorithm is therefore suitable for use in such environments, provided that a low enough learning rate is used after convergence has occurred.

Of the two continuous estimators, the pursuit algorithm fails to converge to the expected value, the overshoot increasing with the learning rate. The general estimator with a linear updating function works fairly well with low learning rates but always overshoots, and although the convergence time is slow it produces very low variance once convergence to equalise the penalty probabilities has occurred. Using the non-linear updating function of 'x^3', low variance is evident although convergence takes a long time.

Discretisation encourages failure of convergence for the pursuit algorithm, the overshoot being still higher than with the continuous case. The same effect is observable with the discretised general estimator algorithm, for any updating function and learning rate. It fails to converge, constantly overshooting its target.

Of the range of reinforcement algorithms which are generally used, the LRP, LRεP, DLRP and general estimator with the 'x^3' non-linear updating functions were found to perform the best this non-autonomous environment. Of these three, LRP is the only one to equalise the penalty probability rates, resulting with a higher penalty probability in the steady-state. The LRεP and DLRP algorithms produce a lower penalty probability when the minimum is less than 0.5, and the general estimator always produces the lower penalty probability. As the general estimator also has a much lower variance after convergence, it is the one to be preferred as long as its relatively slow convergence time is acceptable.

With current understanding assuming the applicability of stationary and switching environment results to non-autonomous environments, so discretised schemes have been more recently favoured for use in such environment scenarios. The importance of this study is to show the general non-applicability of discretised reinforcement schemes, and the superior performance after convergence of the continuous general estimator algorithm using the 'x^3' non-linear updating function, the LRεP, and the DLRP schemes.

5 Improved learning automata applied to routing in multi-service networks

5.1 Introduction

The purpose of the work contained in the following chapter is to increase the' learning automata performance to be higher than that obtained when using the best performing reinforcement algorithms detailed in the previous chapter; namely DLRP and LRεP. The improved learning automata performance can then be compared with that resulting from the use of the improved dynamic shortest-path based mechanism detailed in chapter 3.

Improving a reinforcement algorithm's convergence speed, for example by increasing the learning rate, degrades its steady-state performance. Contrariwise, increasing its steady-state accuracy slows down its rate of convergence. Therefore the improvement methods studied in this chapter are based on increasing convergence speed under changing environment conditions, and increasing accuracy under environment steady-state situations.

A novel mechanism for detecting the environment state is detailed. Rather than requiring centralised operation, it is applied in a local manner, so retaining a benefit of learning automata operation; that of local feedbacks. This mechanism, based on action probability entropy, is used by both novel learning automata performance improvement methods outlined in this chapter. They are adaptive learning rates, and automatic reinforcement algorithm selection.

Next, the resulting best performing learning automata based method is compared with the AAMH algorithm outlined in chapter 3. A new network scenario is used for performing the comparison, this scenario mimicking real networking situations more closely. This is based on trend user demands rather than the statistically constant simulations which are generally used in the literature.

Finally the chapter's findings are summarised, and reasons given as to why the further learning automata improvement work of chapter 6 is required.

5.2 Reinforcement algorithm selection

The results from the previous chapter highlighted a number of reinforcement algorithms which produce good learning automata performance when interacting with non-autonomous environments. They are as follows: DLRP, LRεP, and GE with the 'x^3' non-linear updating function.

DLRP produced a faster convergence rate than LRεP, but with a higher steady-state variation thereafter. The GE with 'x^3' updating function produced the lowest steady-state variation of all the algorithms, but required the longest number of iterations before convergence occurred. It therefore seems that each of these reinforcement algorithms has its own particular strengths and weaknesses relative to each other.

A particular reinforcement algorithm might be selected according to the application type. For example if the environment for the application was generally in steady-state behaviour and rate of convergence for the learning automata is not of great importance, then using the GE reinforcement algorithm is an apt choice. On the other hand, if the environment exhibits dynamic behaviour due to factors other than the automata actions then the DLRP algorithm might be used, in order to track the moving environment state most quickly. Were neither of these performance factors of overriding importance, then the LRεP reinforcement algorithm might be used. The application of routing in multi-service networks has elements of both dynamic and steady-state network behaviour, according to the user demands throughout the day. Therefore it is difficult to propose the use of just one for this application type.

However, rather than having to characterise the application type a priori and then choose the most appropriate reinforcement algorithm, it is proposed to improve the performance of these three algorithms in order to make them more generically applicable to applications which exhibit environment types of both steady and moving states, such as routing in networks. The two proposed methods are: adaptive learning rates, and automatic reinforcement algorithm selection.

5.2.1 Adaptive learning rates

Each of the three reinforcement algorithms has a configurable parameter: the learning rate. Variations in this variable produce variations in the rate of convergence and subsequent steady-state accuracy. A higher value causes the learning automaton to converge within a fewer number of iterations, but it also causes a higher subsequent steady-state action

probability variation. The use of a lower learning rate produces opposite effects in both performance indicators.

Increasing the learning rate has a positive effect on the rate of convergence, whilst decreasing it has a positive effect on the subsequent steady-state variation. Were one to dynamically vary the learning rate according to the environment state and associated conditions, then it might be possible to obtain both a higher convergence speed and subsequent lower steady-state variance than is currently the case with fixed reinforcement algorithms and learning rates. The use of such a mechanism in this application would increase the learning rate in cases of moving network state, and decrease it in more steady-state network conditions when the action probabilities are close to convergence.

As the use of learning automata is beneficial in cases of environment uncertainty, due to them automatically converging to produce near optimal performance, so ideally the mechanism should be able to automatically detect whether the environment, which in our case is a multi-service network, is in a steady or moving state. The following section details a method which allows the learning automaton mechanism to automatically detect the environment status.

5.2.1.1 Automatic adaptive mechanism

The method or mechanism for deciding whether convergence of the action probabilities has taken place requires a numerical indicator of the status of the network and action probabilities. This might be based on the average network blocking probability, or the average path length of routes selected. As convergence of the action probabilities takes place, both the average network blocking probability and path length should decrease and finally plateau at a minimum value when convergence has occurred. According to whether these numerical indicators are increasing, decreasing or stationary, so the learning rate might be increased, held steady, or decreased respectively. Another indicator that might be used is the entropy of the system. An experiment was undertaken to ascertain which of these indicators is the most suitable to perform this function. The average path length was not included for consideration as this is heavily influenced by the changing traffic matrix, and so might often report an increasing or decreasing value even when the action probabilities had converged.

5.2.1.1.1 Using entropy measures

The entropy of a system is a measure of its disordedness, and it has been shown that as learning automata which perform the routing function in a telecommunications network

converge, so the disorganisation of the overall system decreases [18]. The system entropy in terms of the learning automata action probabilities may be characterised by the following equation:

$$H(n) = \sum_{i \in N} \sum_{j \in D} \sum_{k \in R} p_j^{ik} \log p_j^{ik} \quad \text{bits}$$

where N is the set of nodes, D is the set of destinations, R are the allowable actions at each automaton in the network, and p is the probability of performing an action. It is therefore a function of the action probabilities. As the action probabilities converge, so the disorderedness of the system decreases causing the entropy to also decrease. It is not necessarily the case however that this reduction is monotonic, for the action probabilities are updated based on stochastic events, and so might cause the system disorderedness to increase in the short term.

In order to ascertain which indicator might be best suited for the automatic adaptive mechanism proposed, the following experiment was undertaken. It consisted of the 28 node network topology with symmetrical network loading. Using different learning rates with the basic reinforcement algorithms recommended in the literature, namely LRI and LRP, Figure 39 is produced. Adaptation in this case was taken as the point where the decreasing network blocking probability began to plateau, and so is overly optimistic on the minimum number of iterations required for convergence. As expected, increasing the learning rate decreases the number of iterations required for convergence but increases the average network blocking probability due to the higher steady-state action probability variance. This is true for both LRI and LRP, indicating its validity for both ε-optimal and ergodic schemes. The aim is therefore to adapt the learning rate to give the fast convergence rate seen when using a fixed 5%, and the low steady-state variance (and so network blocking probability) seen when using a fixed 1% learning rate.

Figure 40 shows how the network blocking probability and entropy of the system changes during convergence of the action probabilities when using the LRI reinforcement algorithm. The network blocking probability values given are the average over the 1000 connection period. It can be seen that as the number of iterations passed increases, so the entropy and network blocking probability decrease. However, what is of interest is that the network blocking probability reaches a minimal plateau whilst the entropy measure is still decreasing. Were the indicator for whether the action probabilities had converged to be the blocking probability, then the onset of convergence would be reported too early and so inaccurately. It is therefore proposed to use the entropy measure as the indicator as to whether convergence has taken place.

This was done using this global entropy measure, and by automatically adapting the learning rate, a combined performance of the fast 5% learning rate convergence rate and the good 1% learning rate steady-state accuracy was obtained [64]. However, in order to have an automatic mechanism for convergence detection and so learning rate adaptation, it is not possible to use a system-wide indicator such as global entropy due to the network resources it would consume in information transmission and processing. A compromise is to use the local entropy measure, this being calculated as follows:

$$H = \sum_{i=1}^{n} p_i \log p_i \qquad \text{bits}$$

Whilst it is expected that the use of this method will be more problematic, due to the stochastic nature of the action determination and so action probability updating, it appears feasible that its use will provide a usable automatic learning rate adjustment mechanism.

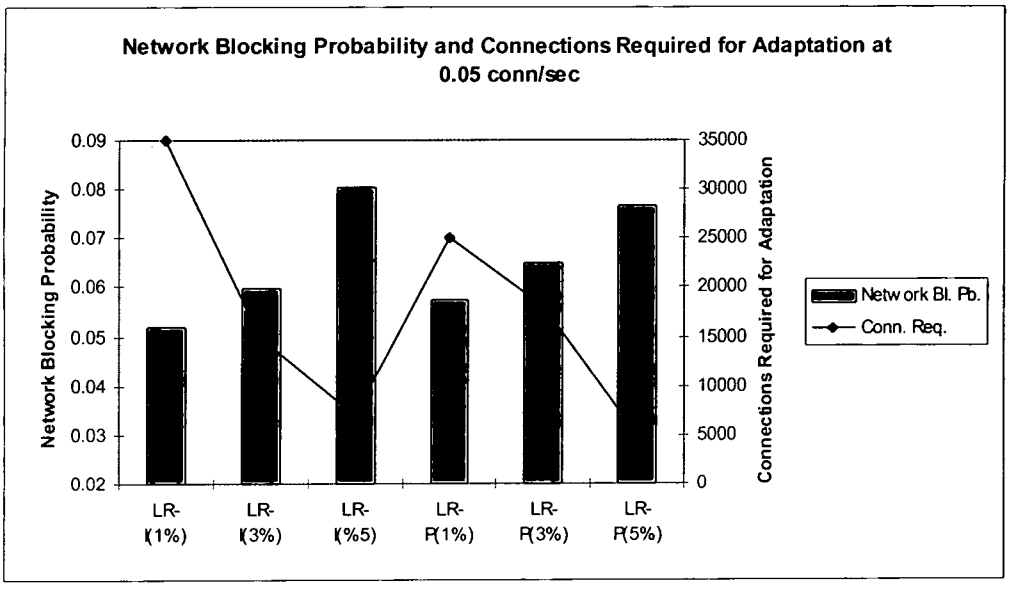


Figure 39: Network blocking probability and iterations required for convergence using LRI and LRP

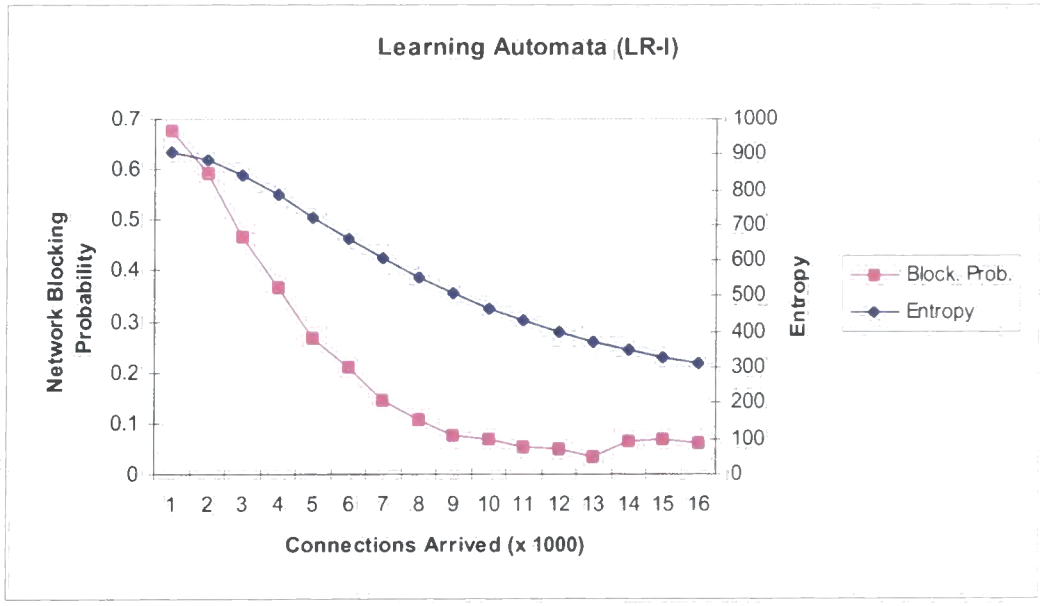


Figure 40: Entropy and network blocking probability during convergence using LRI

5.2.1.2 Entropy threshold calculation

The maximum local entropy occurs when the action probabilities are equidistant, and as they change values from that point the local entropy decreases. For the four node network example, the maximum entropy value at node three is 3.46 as there are eight action probabilities in all when including the two for destination node 3. The minimum entropy value possible is when half of the action probabilities are close to unity (the other half therefore being close to zero), the calculated value when probabilities are at 0.999 and 0.001 being 0.04, which is arbitrarily close to 0. However the spread in possible values is much smaller in realistic scenarios as the entropy is 2.5 when action probabilities are at 0.8 and 0.2 .

The four node network example was again used, as it is possible to determine analytically the converged action probability values. These were 0.67 and 0.33 respectively for the experiment's loading rate. Figure 41 shows the local entropy value for the node having the traffic source, when using a learning rate of 5% with the LRεP reinforcement algorithm. This trace shows significant variation around the value of 3.35 which is what it should converge to, indicating that the high learning rate is causing it to consistently overshoot.

Figure 42 shows the local entropy value for the node having the traffic source, when using a learning rate of 1% with the LRεP reinforcement algorithm. As expected, the entropy

decrease is much more progressive, with little oscillatory behaviour evidenced. Towards the end of this initial convergence period, a slight increase in entropy is noticed, as the action probabilities temporarily move away from their final converged values.

Both these figures show runs of 270 iterations, and indicate that individual learning automata traces can vary somewhat from the smoothed average traces of various simulation runs. Therefore there exists the requirement to smooth the varying individual entropy traces, so that a true indication on whether the entropy is changing significantly (meaning that the environment is in a non-steady state condition) may be obtained. To this end, ten point samples are taken and averaged, these values also being represented on the graphs.

From these the following threshold values are obtained. When using a 5% learning rate, a change in entropy less than 0.05 between averaged 10 point samples should cause a change to a 1% learning rate. When using a 1% learning rate, an entropy change greater than 0.01 should cause the learning rate to change to 5%. When first initiated however, the mechanism should first allow for an entropy change of 0.05 or greater, before permitting the learning rate to be dropped to 1%. To avoid the learning rate remaining at 5% with cases whose converged action probabilities are around 0.5, the mechanism should change to a learning rate of 1% after ten consecutive 10 point samples of entropy changes less than 0.05 . These threshold values are valid for LRεP, as other reinforcement algorithms have different convergence and variance characteristics. When the number of possible actions is increased, the maximum entropy value also increases, therefore also possibly affecting the threshold values. In this application however, an increase in the number of possible actions should not overly affect the threshold values as some of the actions would rarely be used, causing the entropy spread to be much less than otherwise. Therefore for this application there is only required a re-characterisation of the thresholds for the other reinforcement algorithms.

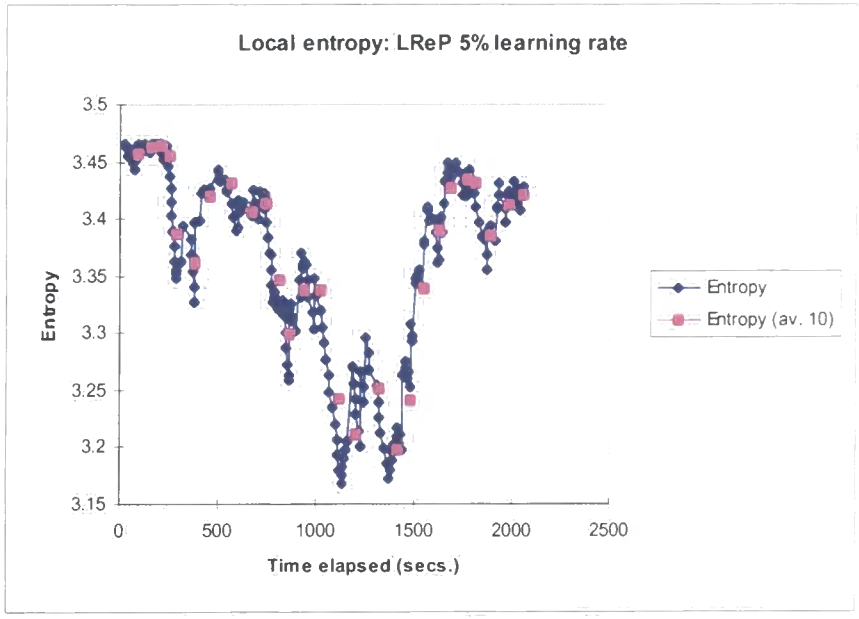


Figure 41: Local entropy using 4-node network with LRεP and 5% learning rate

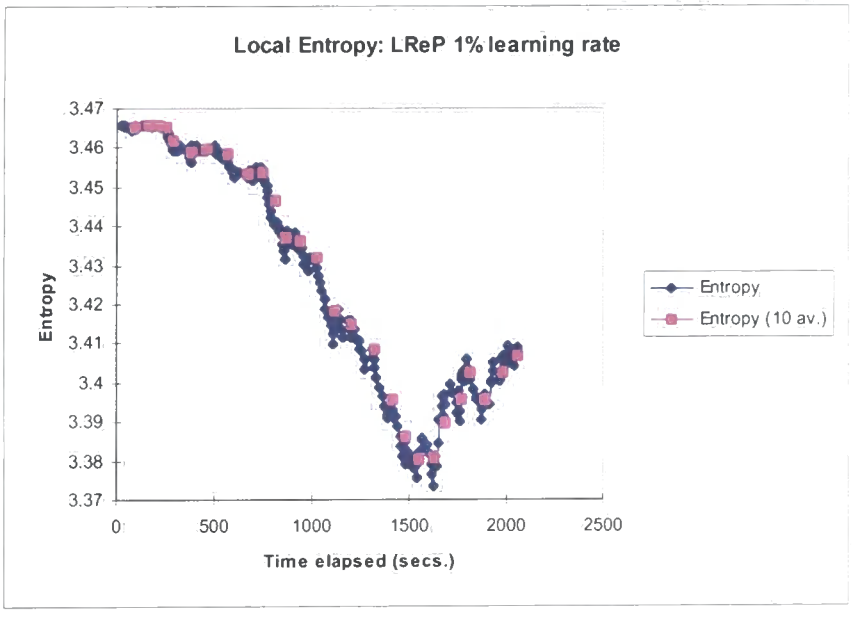


Figure 42: Local entropy using 4-node network with LRεP and 1% learning rate

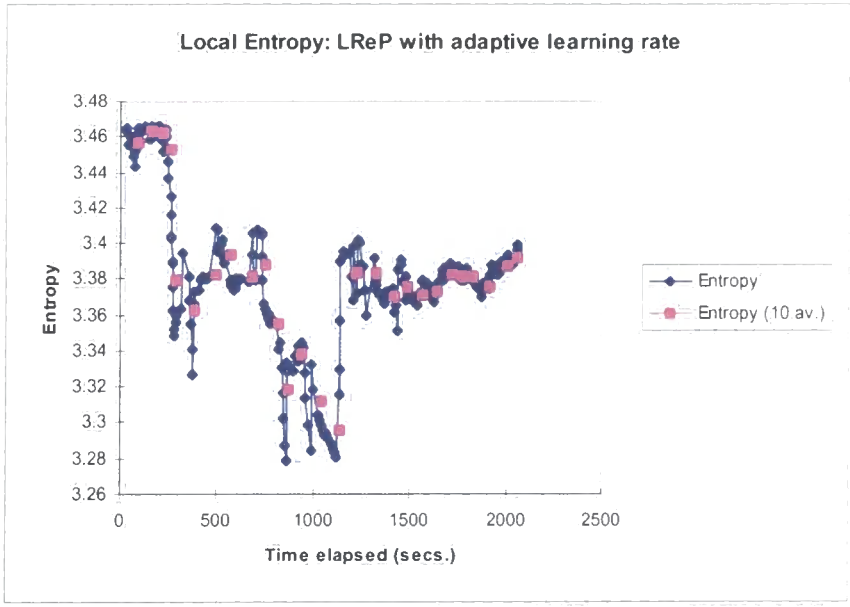


Figure 43: Local entropy using 4-node network and LRεP with adaptive learning rate

5.2.1.3 Experimental results

This section seeks to validate the benefits of automatic adaptation of learning rates. To this end, only the DLRP and LRεP reinforcement algorithms are used, as the GE algorithm has an additional parameter which affects performance, namely the updating function. Also it requires a large number of iterations before convergence occurs, and it has been shown that even high learning rates do not reduce the number to that comparable with the other reinforcement algorithms.

5.2.1.3.1 Results for LRεP

The characterisation threshold values for the adaptive learning rate scheme used with the LRεP algorithm were the same as those calculated in the previous section.

Beginning with the 4-node network scenario, Figure 44 shows the convergence results for LRεP algorithm when using a fixed 5% learning rate, with Figure 45 showing the same when using a fixed learning rate of 1%. As may be seen, the latter has a much slower convergence rate than when using the 5% learning rate, but its subsequent steady-state

variation is much lower. The 90% confidence interval spread in the action probability is around 2% when using the 5% learning rate, and 1% with the 1% learning rate.

Figure 46 shows the convergence of the action probabilities when using the adaptive learning rate scheme. As may be seen, the rate of convergence is very similar to that when using the fixed 5% learning rate, but the subsequent steady-state variation is just a little higher than that when using the fixed 1% learning rate, and certainly far lower than that evidenced with the 5% learning rate.

The point of adaptation for the 28 node network was taken to be when the decreasing network blocking probability rather than the global entropy plateaued. This was because the continuing convergence of the action probabilities, and so decrease in the global entropy, did not significantly affect the performance measure for this application; namely the network blocking probability.

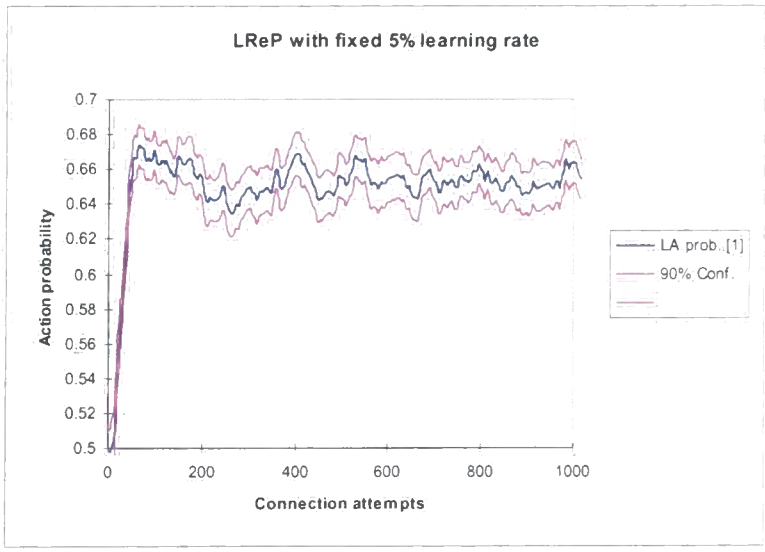


Figure 44: 4-node network and LRεP with fixed 5% learning rate

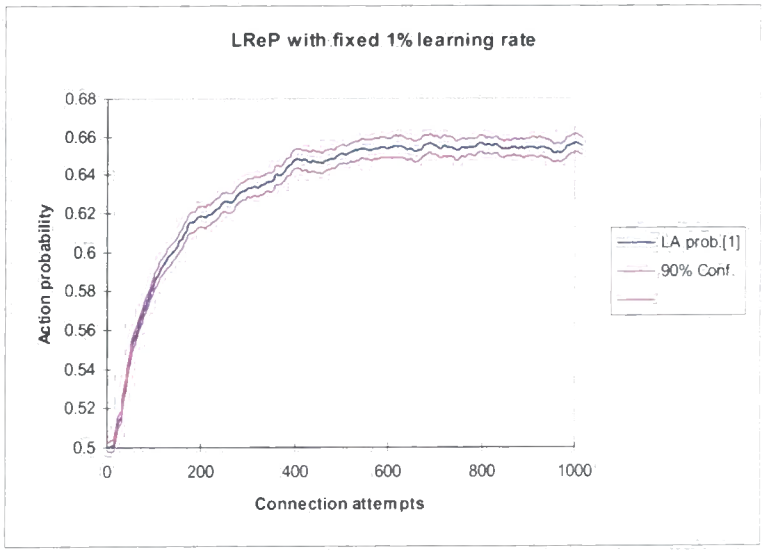


Figure 45: 4-node network and LRεP with fixed 1% learning rate

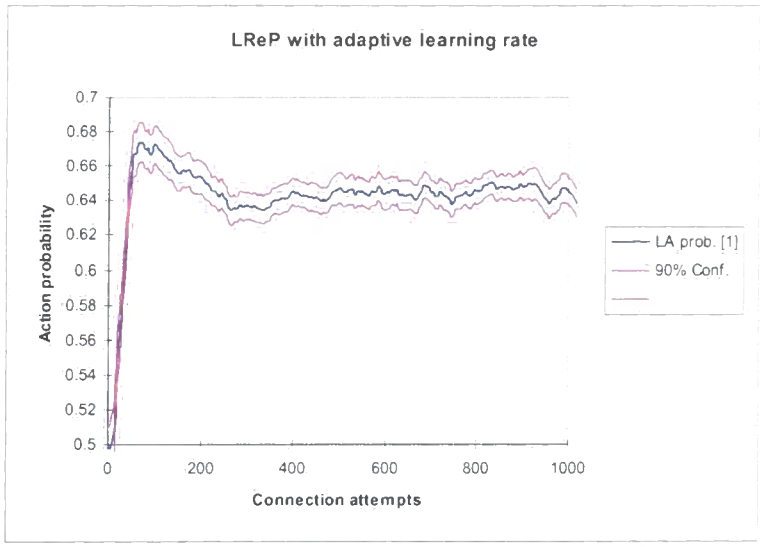


Figure 46: 4-node network and LRεP with adaptive learning rate

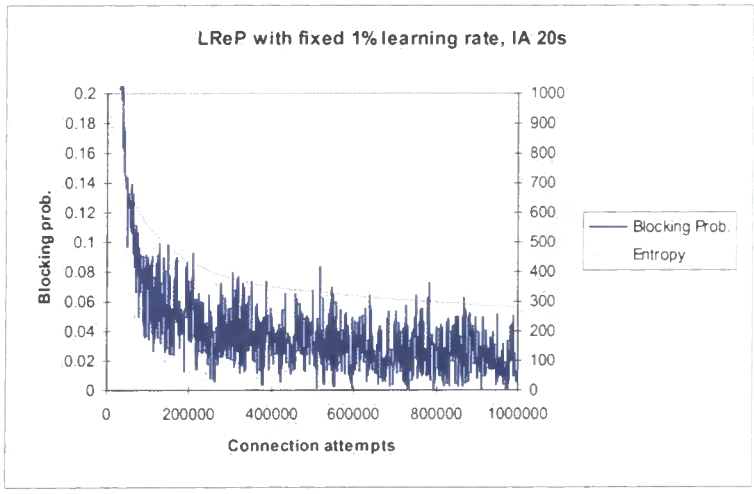


Figure 47: 28-node network and LRεP with fixed 1% learning rate

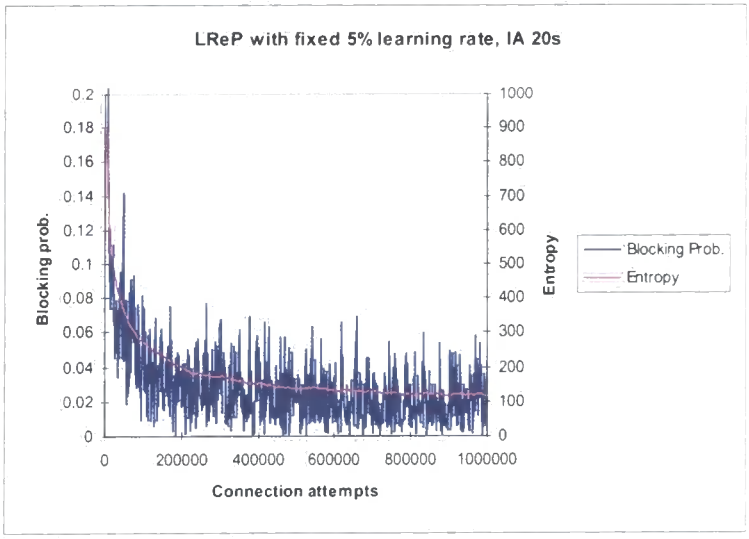


Figure 48: 28-node network and LRεP with fixed 5% learning rate

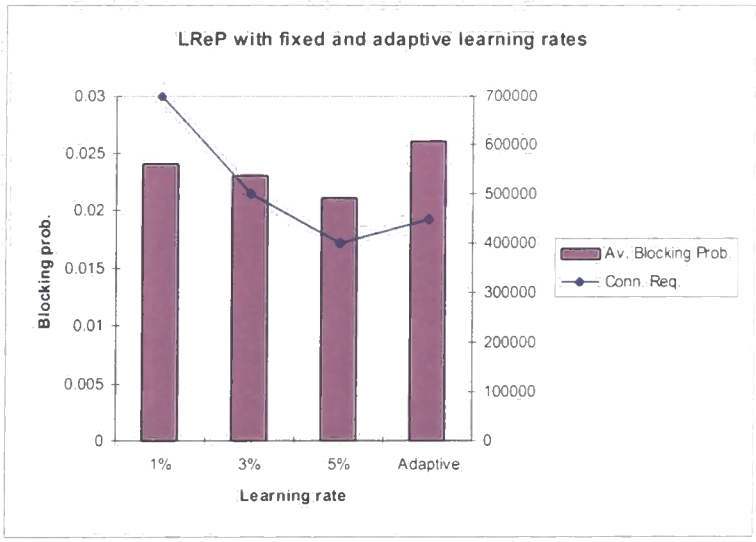


Figure 49: Network blocking probability and iterations for convergence using LRεP and 28-node network

However rather than the case with Figure 39, a more conservative estimate of when the plateauing began was chosen. This explains the apparent increase in the iterations required for convergence, when compared with Figure 39.

Figure 47 for example shows the process for the fixed 1% learning rate case, whilst Figure 48 shows that for the fixed 5% case. As may be seen from Figure 47, the action probabilities are still converging by the end of 1,000,000 connection attempts when using the fixed 1% learning rate, the entropy being more than twice that of Figure 48 at the end of its simulation run. The effect of this is seen by the decreasing trend of blocking probability in Figure 47, whilst that of Figure 48 is fairly static after 400,000 connection attempts. In this case, we take convergence to have occurred after 700,000 connection attempts when using a fixed 1% learning rate, as the blocking probability is fairly stationary subsequent to that point. These results are summarised together with those for the fixed 3% and adaptive learning rate schemes in Figure 49. In each case, the average network blocking probability was calculated from the point of convergence onwards.

The iterations before convergence results indicate that increasing the learning rate decreases the number of iterations required for convergence of the action probabilities, which is as expected. It can be seen that the number of iterations required is a little above that of when using a fixed 5% learning rate.

What is unexpected however, are the blocking probability results which seem to suggest that the average blocking probability is relatively unaffected by the learning rate parameter, the value being around 2.3% blocking probability. This is especially surprising considering that the 4-node network scenario results show a visible affect on the action probability of changing the learning rate. A possible explanation for this is the relatively

small variation in action probability values after convergence when using the highest learning rate, Figure 44 showing the range being within an action probability band of 5% (63-68%). This implies that even with the highest learning rate, the action probabilities in the 28-node scenario do not vary greatly from their optimum, so returning a similar average blocking probability whatever the learning rate. This explanation is supported by the DLRP results which show a larger action probability band together with a higher blocking probability.

LRεP with fixed 5% and adaptive learning rates are also compared with each other over various traffic loadings as they exhibit a similar number of iterations before convergence. Table 6 shows that the adaptive scheme returns equal or slightly poorer average network blocking probability to the fixed 5% scheme; returning poorer results at low blocking probability levels. Similarly, Table 7 shows the adaptive scheme having a higher steady-state variation at the lower traffic loads, but is comparable to the fixed 5% scheme at higher loads. Finally Table 8 shows the adaptive scheme generally requiring a slightly higher number of iterations before convergence. These results are not of great use in themselves, but are important for comparative purposes with the following which are obtained using other reinforcement algorithms.

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Fixed 5%	0.021	0.117	0.28	0.351	0.423	0.53	0.666
Adaptive	0.026	0.12	0.28	0.351	0.424	0.532	0.665

Table 6: Average network blocking probability for 28-node network with LRεP using fixed 5% and adaptive learning rates

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Fixed 5%	0.0135	0.0239	0.0258	0.0254	0.0233	0.0222	0.018
Adaptive	0.0148	0.0238	0.0253	0.0251	0.024	0.0223	0.0186

Table 7: Standard deviation on network blocking probability for 28-node network with LRεP using fixed 5% and adaptive learning rates

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Fixed 5%	400,000	80,000	60,000	75,000	25,000	35,000	60,000
Adaptive	450,000	100,000	60,000	75,000	50,000	45,000	40,000

Table 8: Global connection attempts for convergence for 28-node network with LRεP using fixed 5% and adaptive learning rates

5.2.1.3.2 Results for DLRP

The use of the DLRP reinforcement algorithm requires a re-characterisation of the adaptation threshold entropy values, as per section 5.2.1.2 . As the DLRP exhibits a higher gain for the same learning rate, so the fixed rates were reduced to 1.5% and 0.04%. As may be seen from Figure 50 and Figure 51, the characterised entropy threshold values were higher than 0.015 to switch from 0.4% to 1.5%, and lower than 0.1 to change from 1.5% to 0.4% learning rate. The resulting DLRP performance when using the adaptive learning rate is shown in Figure 52.

The results for the 4-node network scenario are given first. As with the case of using LRεP, Figure 53 shows DLRP producing a faster convergence than Figure 54, but with higher subsequent steady-state variation. The benefits of using the adaptive learning rate scheme with the DLRP algorithm are shown by Figure 55, which exhibits the benefits of both the high convergence speed of Figure 53 and the low subsequent steady-state variation of Figure 54. Using the DLRP rather than LRεP algorithm causes the action probability to have a larger possible value spread after convergence, this being up from 5% to 8% (67%-75%). Also the converged action probability is different to the LRεP one, from 67% to 71%. This seems to indicate that the established body of theoretical work needs revision, as current thinking indicates that the converged action probabilities should be the same for both reinforcement algorithms.

The 28-node network results are closer to those expected than the LRεP ones. Figure 57 clearly shows that DLRP exhibits a faster convergence but a higher blocking probability and steady-state variation when the fixed learning rate is increased from that of Figure 56. These results are summarised in Figure 58, which clearly shows the adaptive learning rate scheme being only slightly slower in convergence to the fixed 1.5% learning rate. Moreover the subsequent average steady-state probability for the adaptive scheme is only slightly higher than that when using the fixed 0.4% learning rate, and certainly lower than that resulting with either of the fixed 1% or 1.5% learning rates.

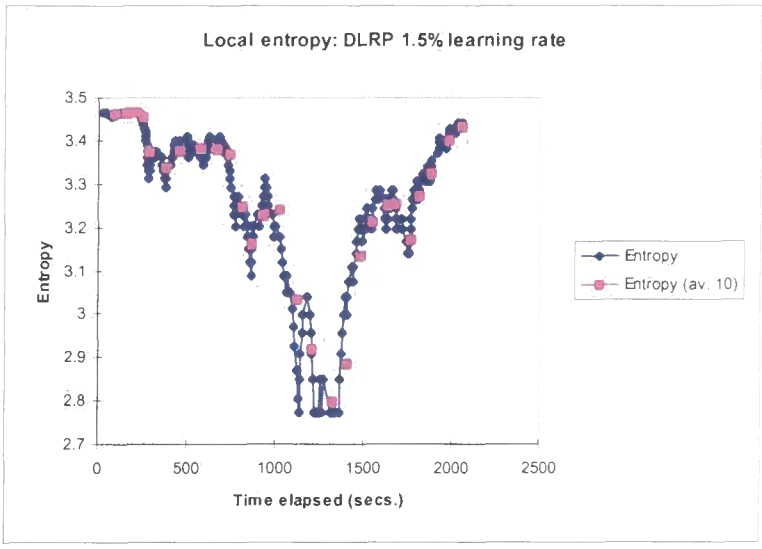


Figure 50: Local entropy using 4-node network with DLRP and 1.5% learning rate

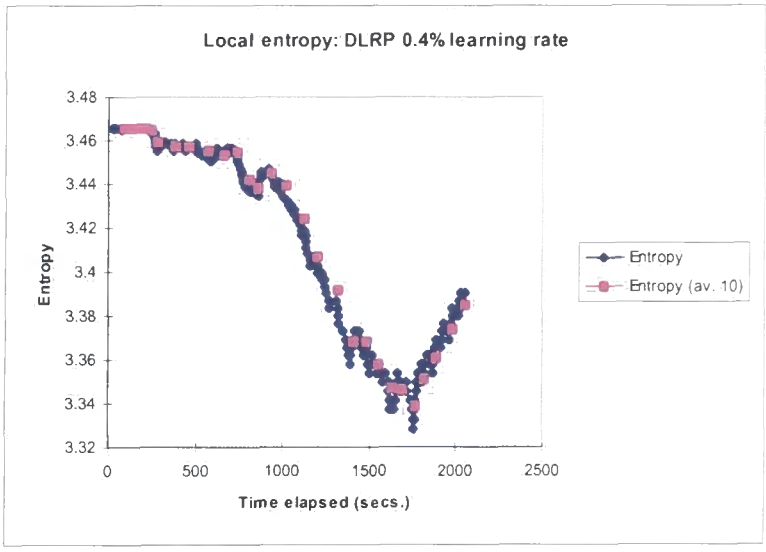


Figure 51: Local entropy using 4-node network and DLRP with 0.04% learning rate

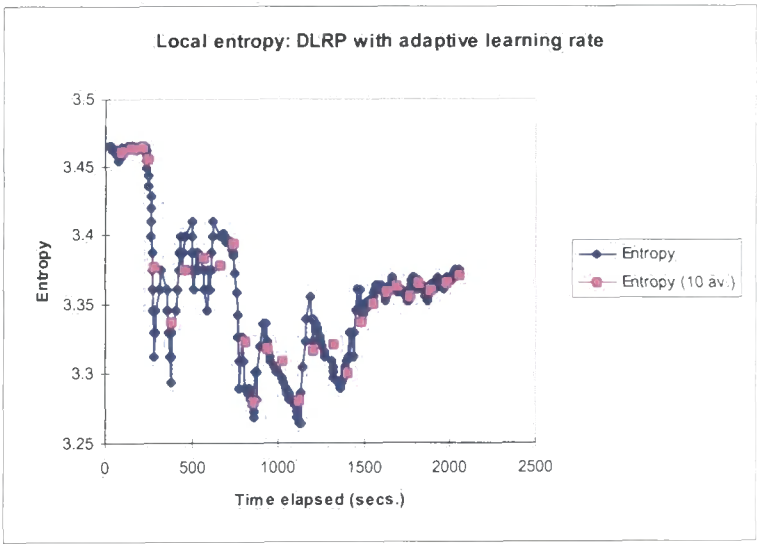


Figure 52: Local entropy using 4-node network and DLRP with adaptive learning rate

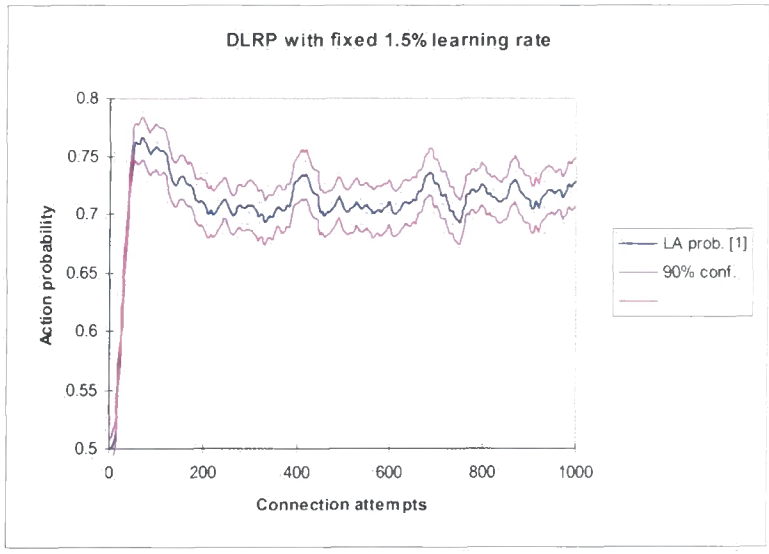


Figure 53: 4-node network and DLRP with fixed 1.5% learning rate

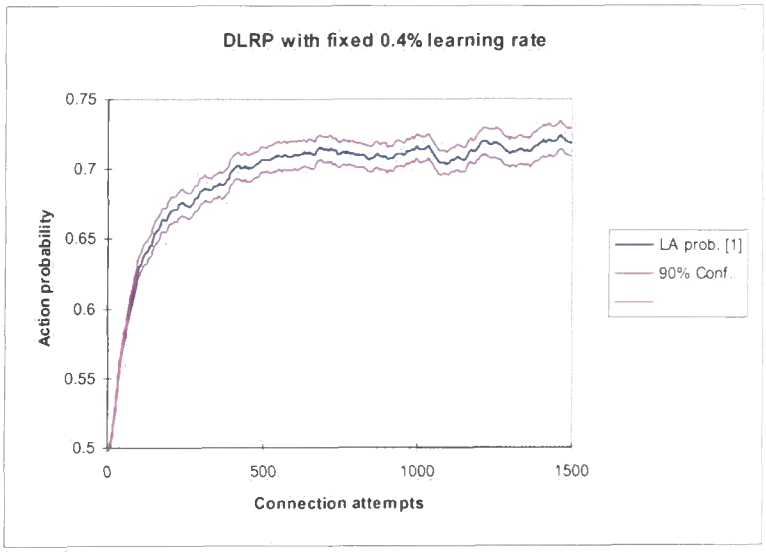


Figure 54: 4-node network and DLRP with fixed 0.4% learning rate

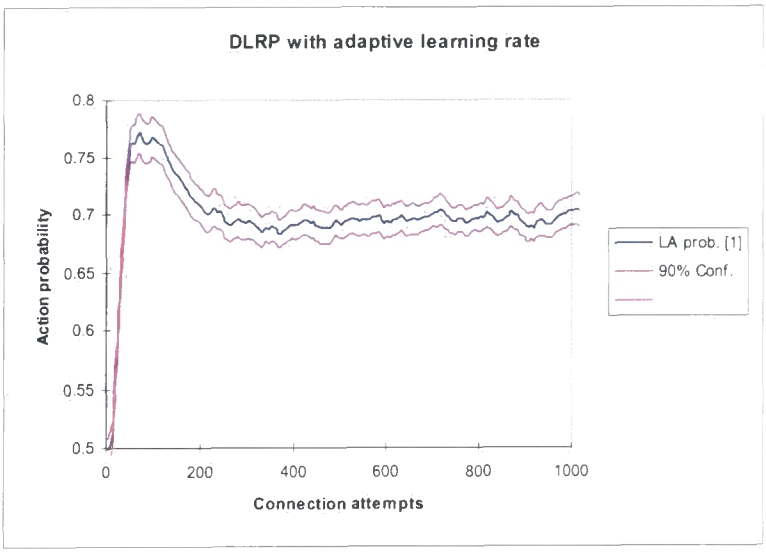


Figure 55: 4-node network and DLRP with adaptive learning rate

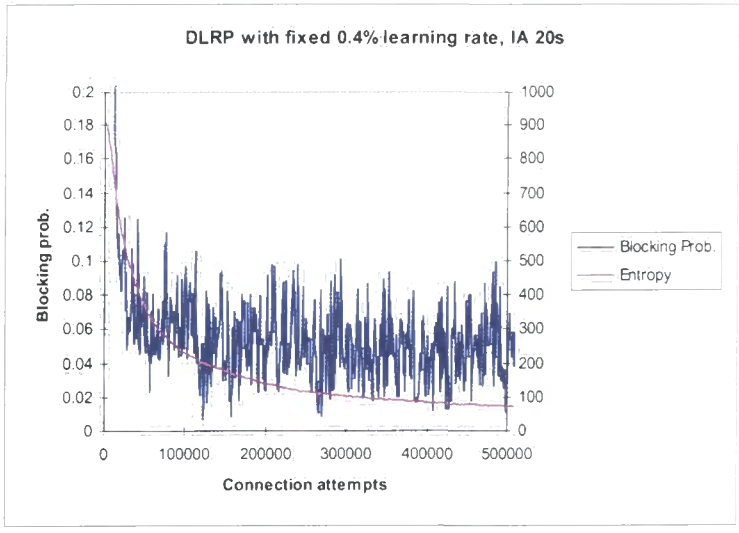


Figure 56: 28 node-network and DLRP with fixed 0.4% learning rate

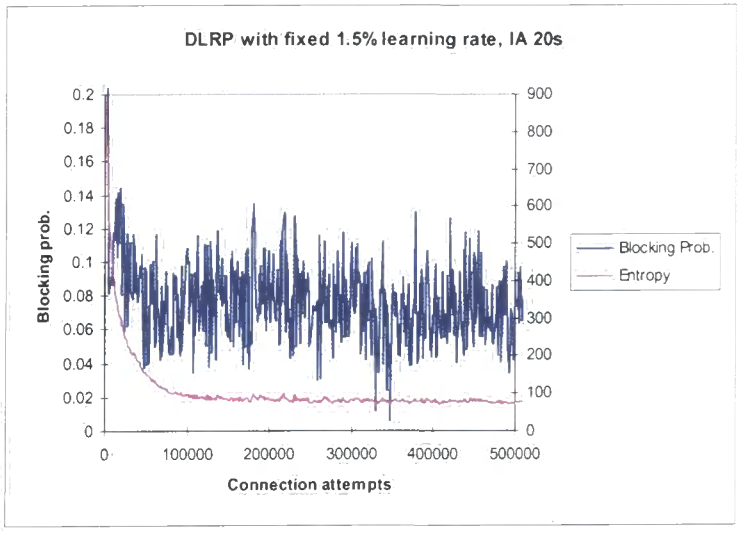


Figure 57: 28-node network and DLRP with fixed 1.5% learning rate

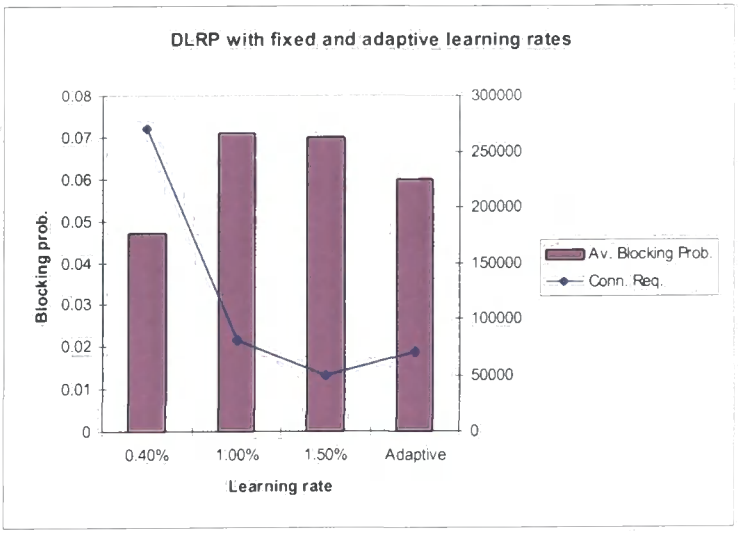


Figure 58: Network blocking probability and iterations for convergence using DLRP and 28-node network

Table 9 shows that in general the adaptive scheme returns a lower blocking probability to the fixed 1.5% learning rate, together with a comparable or lower standard deviation in the action probability after convergence. As was the case with the LRεP convergence results, the adaptive scheme also generally requires more iterations in order to converge than the fixed scheme with the high learning rate.

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Fixed 1.5%	0.07	0.139	0.288	0.349	0.422	0.531	0.664
Adaptive	0.06	0.156	0.281	0.347	0.422	0.531	0.663

Table 9: Average network blocking probability for 28-node network with DLRP using fixed 1.5% and adaptive learning rates

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Fixed 1.5%	0.021	0.025	0.026	0.0256	0.0174	0.0226	0.0185
Adaptive	0.019	0.026	0.026	0.0256	0.0173	0.0226	0.0182

Table 10: Standard deviation on network blocking probability for 28-node network with DLRP using fixed 1.5% and adaptive learning rates

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Fixed 1.5%	50,000	40,000	9,000	10,000	14,000	9,000	15,000
Adaptive	70,000	40,000	9,000	15,000	16,000	9,000	15,000

Table 11: Global connection attempts for convergence for 28-node network with DLRP using fixed 1.5% and adaptive learning rates

It is difficult to make direct comparisons of these results with those of the LRεP algorithm as both convergence speed and subsequent steady-state behaviour are affected by the learning rate parameter. This means that the resulting performance of either reinforcement algorithm might be improved by modifying the learning rates according to the environment scenario. However general characteristics are visible in each set of results, and these can be compared and conclusions drawn.

Although the LRεP reinforcement algorithm does not seem to be greatly susceptible to learning rate parameter changes when operating in more complex environments, it does seem to consistently return a lower average blocking probability than when using the DLRP algorithm. The reasons for the poorer DLRP performance seem to be twofold: different converged action probability values, and a higher subsequent variation and possible spread in these values. The latter reason is validated by Figure 58 which clearly shows an increase in the blocking probability when the DLRP fixed learning rate is increased.

The DLRP algorithm on the other hand clearly displays a higher convergence speed than the LRεP algorithm. By comparing Table 8 with Table 11, it can be seen that the DLRP requires between half to a sixth of the number of iterations than LRεP requires in order to converge.

It therefore seems that the results on algorithm strengths and weaknesses brought out in chapter 4 still hold even when using differing fixed learning rates, the range used in this study being 0.4 to 1.5% for DLRP and 1 to 5% for LRεP. This conclusion is surprising when considering simply the 4-node network scenario results, as these show similar traces in both convergence speed and subsequent steady-state accuracy for both algorithms at both their learning rate parameter extremities. It therefore seems important to perform more complex environment interaction experiments before drawing conclusions on the learning automata performance when using a specific updating method.

The adaptive learning rate scheme has been shown to be beneficial for both reinforcement algorithms, in general resulting with a slightly poorer convergence speed than when using the highest fixed learning rate, and a slightly poorer subsequent blocking probability to that obtained from the lowest fixed learning rate. These improvements however do not change the essential performance characteristic of either of these algorithms: the LRεP algorithm produces lower blocking probability after convergence, and the DLRP algorithm requires significantly fewer iterations in order to converge. It may be concluded that even when using the adaptive learning rate scheme, one algorithm might be better for a specific non-autonomous environment application than the other, but neither may be generically recommended to produce good learning automata performance.

5.2.2 Automatic reinforcement algorithm selection

The previous section has outlined a scheme for improving learning automata performance based on setting the learning rate to that which produces the best performance for the current

network and action probability convergence states. This scheme was automated using a mechanism which detected the state of convergence of the learning automaton action probabilities.

This section proposes to harness the same idea to automatically select the most appropriate reinforcement algorithm for the network and action probability convergence states. Results given in chapter 4 showed that the DLRP, LRεP, and GE reinforcement algorithms produced the best performance for learning automata interacting with non-autonomous environments. It was also shown that GE with the 'x^3' updating function produces the best performance once convergence has occurred, and DLRP produces the best under moving network and convergence state conditions. By utilising the most appropriate reinforcement algorithm for the environment and learning automata action probabilities, superior performance both in convergence speed and steady-state behaviour should follow.

For this application type, namely routing in communication networks, the environment is rarely in a steady-state condition, and if so only for short periods of time. This factor precludes the use of the GE reinforcement algorithm as it requires a large number of iterations in order to converge. It is therefore proposed to switch from the DLRP to the LRεP algorithm in steady-state conditions. The validity for doing so is shown when comparing Figure 49 with Figure 58, the network blocking probability being lower for LRεP than DLRP.

The entropy thresholds for automatically switching between reinforcement algorithms are a combination of both sets of previous experiments. When using the DLRP algorithm, if the entropy change is less than 0.1 then the algorithm should be changed to the LRεP algorithm. If the current algorithm is LRεP and the entropy change is greater than 0.01, the reinforcement algorithm should be changed to DLRP.

As this method seeks to improve convergence and steady-state performance by changing the reinforcement algorithm dynamically, so fixed learning rates were used. The values chosen were learning rates of 1.5% for each algorithm, this being a high learning rate for the DLRP algorithm which is used under converging conditions, and a low learning rate for the LRεP algorithm which is used under steady-state conditions. This learning rate should therefore produce good performance for each algorithm, as each will be utilised under conditions most suited to its strength.

5.2.2.1 Experimental results

Figure 59 shows the performance curve for the automatic reinforcement algorithm selection method when operating in the 4-node network scenario. Comparing this curve with those in

Figure 46 and Figure 55 is inconclusive as these latter two display similar characteristics and yet the algorithms return significantly diverse performance results when operating in more complex environments. What can be gleaned however is that the action probability converged to seems to be between that of the DLRP and LRεP, as might be expected. With the algorithm currently operational moving the action probability to its convergence value and the action probability value being subsequently moved back when the other algorithm is switched in, the variation in the steady-state is higher.

Figure 60 shows the performance for the algorithm when operating in the 28-node network scenario with an average interarrival rate of 20 seconds for the user demand models. When comparing these graphs with those in Figure 47 and Figure 57, it seems that the automatic algorithm selection exhibits similar convergence characteristics to the DLRP algorithm, together with a subsequent steady-state blocking probability approaching that of the LRεP algorithm. However it does exhibit a higher steady-state variation than either of the two algorithms by themselves.

Looking at the algorithm's blocking probability performance over differing traffic loads in Table 12, it can be seen that it is between the LRεP and DLRP algorithm singly used, and so can be thought as being superior to either as sometimes one outperforms the other according to the traffic loading. Table 13 shows that the subsequent steady-state variation is as high or higher than either singly used, whilst Table 14 indicates the automatic algorithm selection method returns close to or better convergence speed when compared to that for the DLRP algorithm.

To conclude, it has been shown that the automatic reinforcement algorithm selection provides better generic performance than the adaptive learning rate mechanism applied to either of the two algorithms by themselves. This updating mechanism for the action probabilities can therefore be recommended for most non-autonomous environment applications, as it provides both relatively good convergence speed and subsequent environment penalty probability.

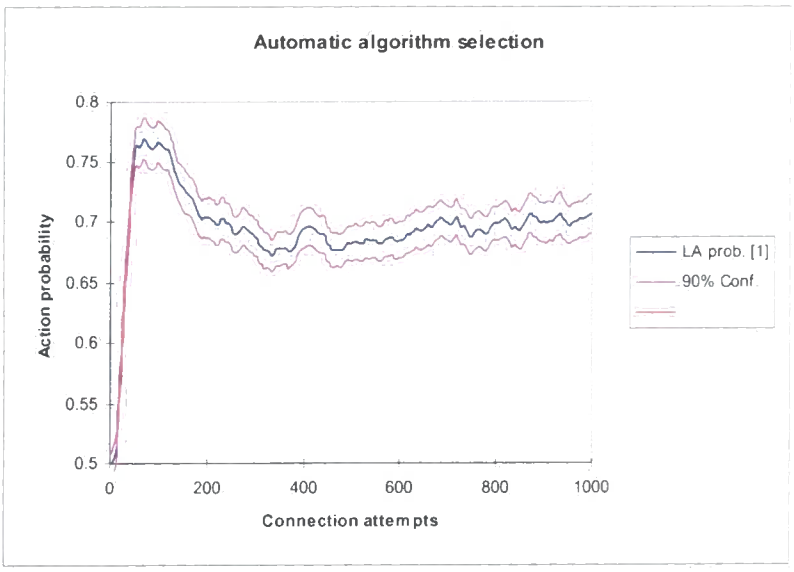


Figure 59: 4-node network and automatic LRεP or DLRP selection with 1.5% learning rate

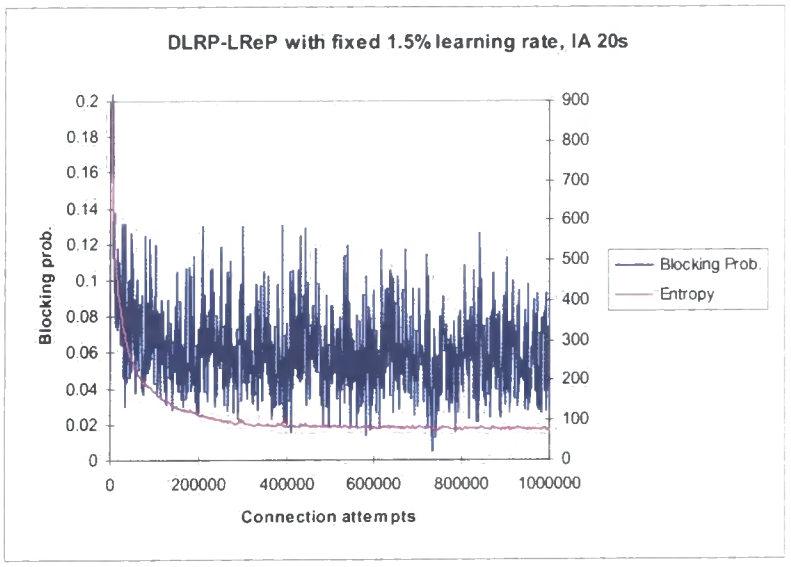


Figure 60: 28-node network and automatic algorithm selection

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Automatic	0.059	0.138	0.287	0.353	0.423	0.532	0.663

Table 12: Average network blocking probability for 28-node network with automatic algorithm selection

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Automatic	0.021	0.025	0.026	0.025	0.023	0.022	0.018

Table 13: Standard deviation on network blocking probability for 28-node network with automatic algorithm selection

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Automatic	100,000	33,000	8,000	11,000	15,000	12,000	16,000

Table 14: Global connection attempts for convergence for 28-node network with automatic algorithm selection

5.3 Comparisons with standard routing method

As was written in chapter 3, the better performing group of routing algorithms which are currently used in real networks are based around shortest-path principles. This work took one of the better performing algorithms from this group, and improved its method of operation for use in realistic network scenarios. These improvements encompassed both its algorithm performance, the type of link-state information to be propagated, and the method its update throughout the network. This has resulted with a standard routing method which is thought to provide good network performance, whilst requiring no extra signalling for propagation of link state information.

The improved standard routing method is compared with the automatic reinforcement algorithm selection scheme, which is the best of the improved learning automata routing methods detailed in this chapter for this type of application. The main experimental comparison occurs using a more realistic network simulation scenario than previously, the rationale for which follows in section 5.3.2.

5.3.1 Initial algorithm comparison

Figure 61 shows the blocking probability performance for the 28 node network when using AAMH with existing signalling for link-state propagation. When compared to Figure 60 which is that for the learning automata with automatic algorithm selection, it can be seen that the AAMH algorithm seems to return on average a lower blocking probability. This is

confirmed by comparing Table 12 with Table 15, for AAMH returns a lower blocking probability at all loading rates. A comparison of Table 13 with Table 16 also shows AAMH blocking probability performance having a lower or equal standard deviation at loads causing blocking probabilities up to 27%, with a higher one at greater loads. This indicates that AAMH might produce a more consistent network performance than the learning automata based method under realistic network loads. A table on iterations for convergence is not included as the AAMH algorithm does not require a period in order to converge.

These results are a little unexpected as learning automata based methods should produce superior performance after convergence at lower loads due to the accessibility of a greater number of paths. The reason for the poorer performance must therefore centre on the failure to fully converge due to the statistical variations in the user demand models.

However this experimental scenario does not match real-world network scenarios very closely. The following section therefore details a more realistic experiment, from which results are gathered and observations drawn.

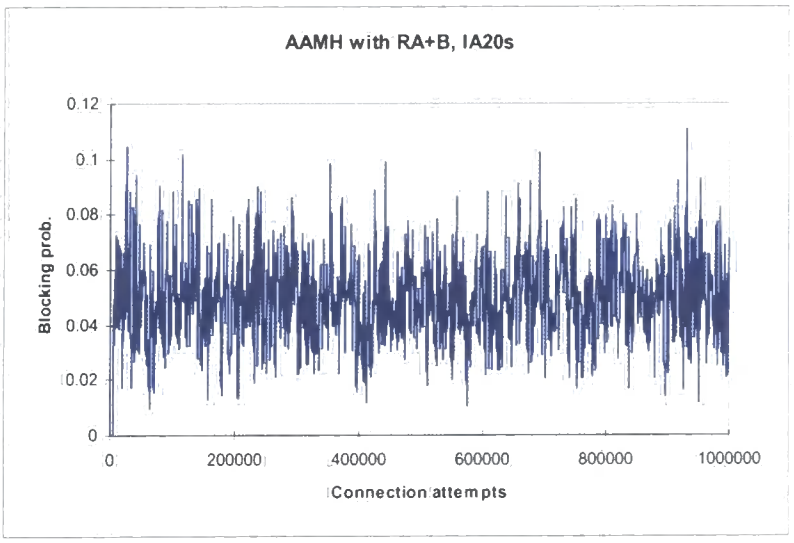


Figure 61: 28-node network and AAMH with RA+B

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
AAMH, RA+B	0.050	0.128	0.276	0.344	0.421	0.529	0.662

Table 15: Average network probability for 28-node network with AAMH and RA+B

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
AAMH, RA+B	0.017	0.021	0.025	0.026	0.026	0.026	0.018

Table 16: Standard deviation on network blocking probability for 28-node network with AAMH and RA+B

5.3.2 A more realistic network scenario

The simulation results to date have used statistical interarrival and holding times for the user demand model traffic generation. This results with a dynamic variation in the traffic arrivals and so network resource usage at any one instant, but a fixed or static statistical variation in the user demands. By running separate simulations using different average interarrival times, the user demand traffic is altered so causing the network resource usage to be changed. These experiments have been used to give an indication of network behaviour at different loading rates, the effect of resource control algorithms being different according to the user demand traffic loading.

Whilst such simulation methods have historically been used to ascertain network behaviour, real networks do not exhibit such scenarios in practice. User demand traffic is dynamic, but rarely statistically constant as user demand characteristics change over different measurement periods. Simulation work to date has tried to take this factor into account by generally undertaking peak busy-hour experiments in order to characterise the network performance under the worst-case traffic loading. What such simulation experiments fail to capture however, is that in real networks user demand traffic does not perform a large step response in size, but gradually and statistically increases to the peak level. The effect of this is for certain network resources to be consumed so that the network is in a certain state before the peak busy-hour period occurs, the network moving from one state to another as the user demand traffic changes. Using a step response change in the user demand traffic from zero to that of the peak busy-hour period causes the network to begin in a different state than in real-life, so that there is a distinct possibility of it ending up in a state different to that of the real network situation. Another weakness of such experiments is that much of the dynamics of network behaviour is lost because the user demand traffic being statistically constant.

Recent work has highlighted these failings [65], and subsequent simulation experiments have used trends in user demand traffic in order to match real-life network scenarios more closely [66]. These trends were composed of increasing, decreasing and steady-state user demands, the level of demand being deterministically calculated according to the current simulation time and start of the currently valid straight-line trend. The work in this section goes a natural step forward by using statistical rather than deterministic trends for user demand traffic generation, as was outlined in [67]. Due to the increased resulting dynamism of the network state, it is expected that learning automata performance relative to that of AAMH may be poorer still.

5.3.3 Experimental results

Figure 62 shows the mean trend used for each 24 hour recurrent time periods. Due to limitations in the statistical variations of the modelling environment, step sizes were used instead of true trends. However this method convincingly approximates trend generation, as is shown by the typical user demand trace at each source node given in Figure 63. As may be seen, the mean trend is replicated eleven times and a statistical variation applied to it. The multiplicity of recurrent time periods allows for a large enough sample space from which to derive statistical conclusions.

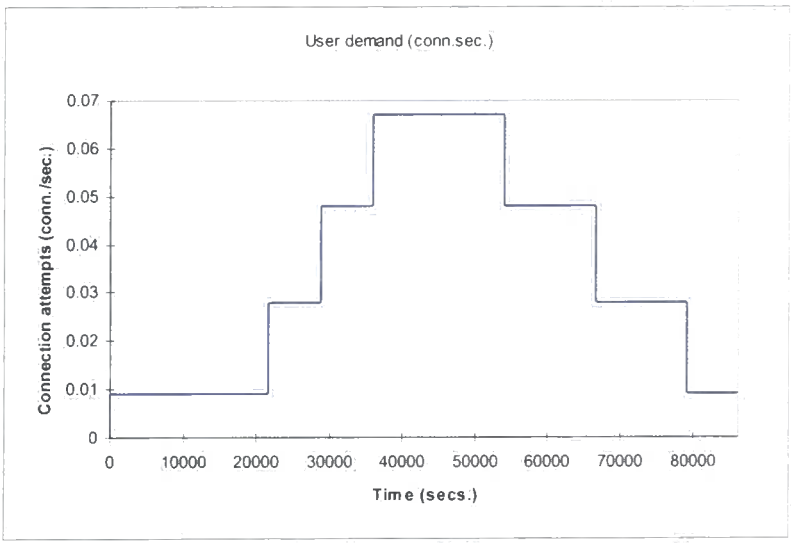


Figure 62: Mean trace used for a 24 hour period

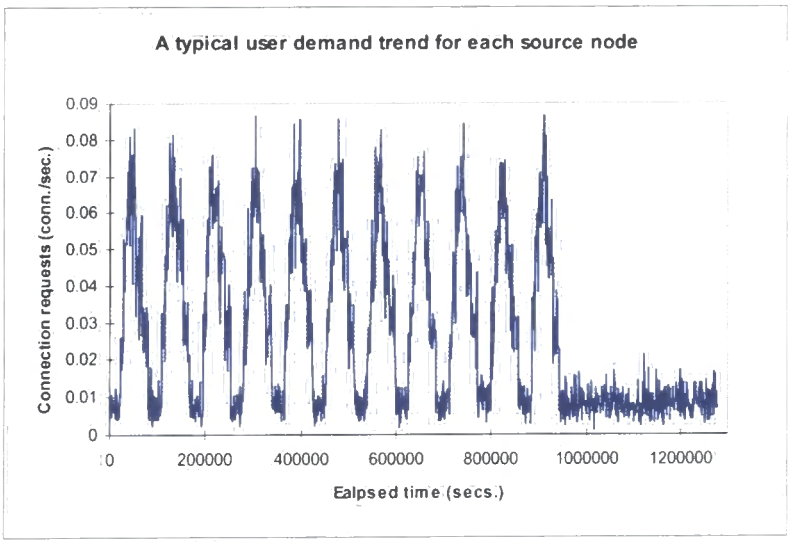


Figure 63: A typical resulting user demand trace at a source node

When using the AAMH algorithm combined with the existing signalling method of link-state propagation, the recurrent time periods are clearly indicated in the resulting blocking probability performance, this being shown in Figure 64. Peak demand levels result with blocking probabilities of up to 18%, but generally up to 16%. Comparing these results with those of the learning automata with the automatic reinforcement algorithm selection which are shown in Figure 65, confirms our expectation of poorer performance. This graph shows peak blocking probabilities of up to 22%, but generally up to 19%. Examining the entropy curve shows that as previously the reason for this poorer performance is due to lack of action probability convergence. This is shown by the curve evidencing continuous change, indicating that the action probabilities are continually changing.

There therefore seems to be the requirement for schemes to improve the convergence of the action probabilities when learning automata interact with non-autonomous environments. A problem certainly apparent with this application revolves around the binary environment feedback mechanism. The network does not differentiate between a route which is almost saturated, and one having low utilisation, returning a positive feedback response for the reinforcement algorithm in either case. The improvements proposed in the following chapter are linked with learning automata operating with link utilisation based environment feedbacks, these being a richer and more informative basis with which to reinforce the action probabilities. The use of this mechanism should aid convergence and so result with lower blocking probabilities.

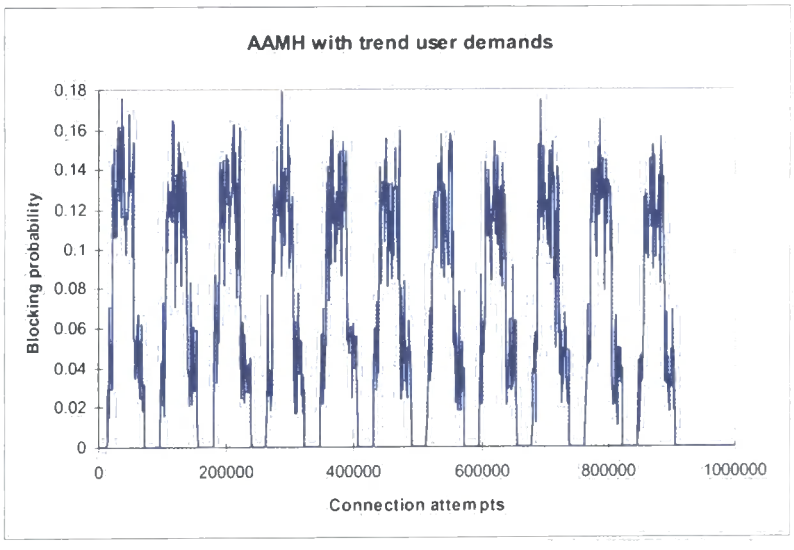


Figure 64: Performance of AAMH with trend user demands

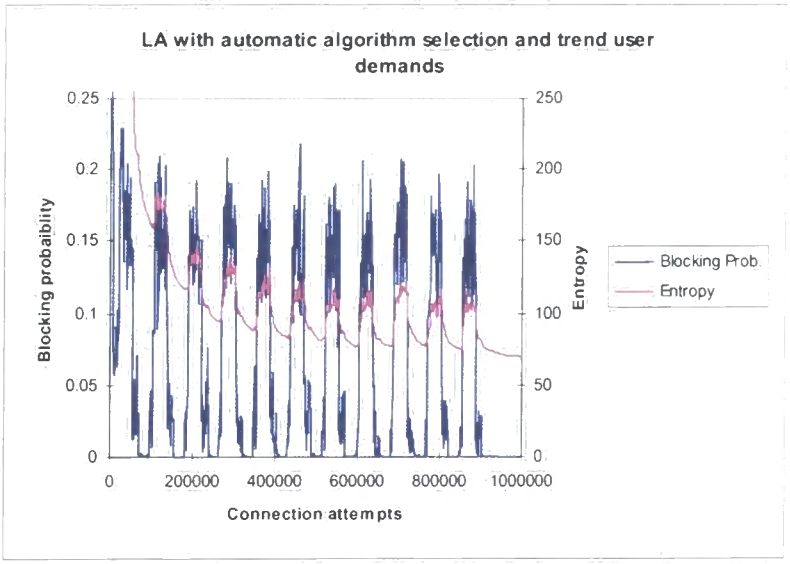


Figure 65: Performance of LA with automatic algorithm selection with trend user demands

5.4 Summary

The results presented in the previous chapter highlighted a number of reinforcement algorithms which produce good learning automata performance when interacting with non-autonomous environments. The DLRP algorithm results with fast convergence but a high subsequent steady-state variation. The LRεP algorithm on the other hand, produced the contrary in both performance metrics. It seems that improving convergence speed degrades steady-state performance, whilst increasing steady-state accuracy slows convergence. The work presented in this chapter has sought to improve the performance of both algorithms by seeking to alter their convergence speed or steady-state accuracy according to the environment state. This has been done in order to produce an algorithm which might be generically applicable to non-autonomous environment applications such as routing in communication networks. The two methods that have been examined in this chapter are adaptive learning rates and automatic reinforcement algorithm selection.

Each mechanism seeks to change a parameter or method according to the converged state of the action probabilities. If the action probabilities have converged, then the reinforcement algorithm parameters or method is changed in order to reduce the steady-state variation. Contrariwise, if the environment state is moving, then the reinforcement algorithm is altered in order to produce a faster convergence of the action probabilities. These alterations or parameter changes therefore require a mechanism for detecting the state of convergence of the action probabilities. Rather than trying to characterise the environment states a-priori to determine when the action probabilities have converged (as with the previous

chapter), a novel mechanism is detailed which calculates the local entropy of the action probabilities for determining whether they are relatively stationary or still moving. Experimentally derived entropy thresholds have been calculated for DLRP and LRεP in turn, where a low mean entropy change after ten iterations would cause the reinforcement algorithm to be set to favour low steady-state variation rather than high convergence speed. If on the other hand there occurs a relatively large mean entropy change after ten iterations, the reinforcement algorithm is altered to favour faster convergence rather than lower steady-state variation. This novel action probability convergence state detection mechanism has been successfully applied in this study for both the adaptive learning rates and automatic reinforcement algorithm selection methods.

The novel adaptive learning rate mechanism seeks to alter the reinforcement algorithm learning rates. The rate is set high when requiring fast convergence speed, and low when trying to increase steady-state accuracy. The mechanism was successfully applied to both the DLRP and LRεP reinforcement algorithms using different entropy change thresholds. The LRεP algorithm seems relatively unaffected by fixed learning rate parameter changes, so a significant improvement was not evidenced when using the adaptive learning rate mechanism. The use of the mechanism with the DLRP algorithm on the other hand, did produce a noticeable improvement, especially under lower network loading.

However this improvement was not large enough to remove the conclusion that using the DLRP algorithm produces better convergence speed, whilst using the LRεP algorithm results with a lower steady-state variation. This observation led to the formation of the novel automatic algorithm selection scheme, where instead of adapting the learning rates to produce better performance, the DLRP algorithm is used under action probability convergence conditions, and the LRεP algorithm is switched in under steady-state conditions. This novel scheme was found to produce significantly superior results in terms of both iterations required for convergence and subsequent blocking probability. It therefore may be considered as a suitable reinforcement method for learning automata interacting with a non-autonomous environment.

Finally the resulting improved learning automata performance was compared to the improved dynamic shortest-path based algorithm proposed in chapter 3. In order to assess the performance benefits of one algorithm versus the other, a new network experiment type was defined, based on user demand trends which more closely resemble real networking situations.

The initial experimental results which used the standard statistically constant user demands showed that the AAMH algorithm consistently returned a lower blocking probability of close to 1%, and did not require a convergence period. It was expected that the relative

AAMH performance would be even more superior when used in the more realistic trend scenario, and this indeed proved to be the case. AAMH was seen to peak at a blocking probability of 18%, whilst the learning automata with automatic algorithm selection scheme peaked at a blocking probability of 22%.

It was seen that the poorer learning automata performance was due to the failure of the action probabilities to fully converge, this being evidenced by the entropy trace constantly changing. The main reason for the lack of convergence was noted as the fact that the network does not differentiate between a route which is almost saturated and one having low utilisation, returning a positive feedback response in either case. Therefore to improve the learning automata performance, there is the requirement to make the network feedback responses linked in some way with the spare capacity on the route. The work detailed in the next chapter seeks to do just this: to modify the reinforcement algorithm and environment so that the learning automaton operates on utilisation based network responses. It is expected that the use of this mechanism will aid convergence and so result with lower blocking probabilities.

6 Using an S-model response environment for a novel learning automata based routing algorithm

6.1 Introduction

The purpose of the following chapter is to re-apply learning automata to the routing problem in networks in a novel way which is related more closely with network performance indicators. Previous reservation-based work has been linked with the acceptance or rejection of connection requests, but this work links the environment feedback with the actual link utilisation levels which cause a connection request to be either accepted or rejected. By so doing, better performance is expected due to the greater information content returned in the environment feedback.

Rather than using the binary P-model response environment as previously, the S-model response environment is used. The relative available bandwidth on the route is normalised to the range of 0 to 1, and then smoothed to erase short-term fluctuations using the exponential smoothing technique.

Experiments are then performed to obtain the best performing learning rate for the reinforcement algorithm chosen. Finally the performance of this novel algorithm is compared to that of AAMH and the P-model automatic reinforcement algorithm selection method detailed in the previous chapter.

6.2 Reasons why a new learning automata method for routing in networks is required

As was outlined in chapter 2, learning automata based routing applied to reservation-based connection-oriented networks has utilised the P-model response environment in all the literature surveyed. Reservation-based connection acceptance is based on whether there remains sufficient bandwidth along the source to destination path to allow the connection request QoS to be met whilst not violating the QoS of the connections already present along the path. Therefore there exists a simple mapping from the network response to a connection

request (this being either an acceptance or rejection) to the binary feedback of the P-model response environment.

However a number of weaknesses are apparent with the use of this method. The main one was highlighted by the results in chapter 4, in that a significant number of the reinforcement algorithms failed to converge. Those particularly affected were discretised schemes, where nearly all types failed to converge. The main difference in operation between discretised and continuous schemes is that the former approach an action probability of unity directly, whilst the latter do so asymptotically. This means that with continuous linear schemes, when the action probability is close to unity a reward environment response causes the probability to be increased slightly whilst a penalty response causes it to be decreased by a higher amount. Discretised linear schemes on the other hand would increase and decrease the action probability by the same amount wherever its value was currently found.

It therefore seems to be the case that penalty responses are an integral part of the convergence process. A sufficient number of them are required for convergence to take place, or in order to stop the action probability continuing to increase to a value of unity as occurs with many discretised schemes. For applications such as the routing function in networks, where minimum environment penalty probabilities are generally low and in the order of up to 10%, the requirement for a significant number of penalty responses from the environment becomes unacceptable. This indicates that learning automata methods in their present form may not be realistically used for the routing function of reservation-based connection-oriented networks such as multi-service networks.

Another weakness of the presently used method is the lack of richness in the information passed with the environment feedback response. The binary response of the connection request being accepted or not includes no indication as to whether the network resources along the path chosen are almost fully or only slightly utilised. In cases of low capacity utilisation it is expected that load balancing would therefore not occur, this being a usual benefit of using learning automata for the routing function. Using the available bandwidth feedback information should result with much faster convergence of action probabilities as their updating will vary in granularity according to the size of the remaining free network resources along the route, so causing the probability value to move more quickly towards using paths of lower utilisation. A load balancing effect should then also occur, whatever the level of utilisation of the network resources. Indeed, the network mechanism currently utilises available bandwidth as its control indicator, accepting a new connection request if there remains sufficient free bandwidth along the route. It therefore seems reasonable to use the same control indicator as the updating mechanism for the learning automata action probabilities. The use of this method should also ensure that environment penalties, which in this case are connection requests blocked, are no longer required for

convergence to take place. It is therefore realistic to use this new method of applying learning automata to the reservation-based connection-oriented routing problem in real networks.

6.3 Using an S-model response environment paradigm

In order to have a richer feedback response capability, the S-model response environment must be substituted for the P-model response environment which has been used up to this point. The S-model response environment allows the feedback response to take any value in the region (0, 1) rather than be limited to either 0 or 1 as is the P-model feedback.

6.3.1 Normalising the available bandwidth

The issue is now to convert the available bandwidth value for a path into a value that lies within the region (0, 1). This is a similar problem to that posed when applying learning automata to routing in packet-switched networks: how to map the variable delay values to the region (0, 1). The difference between these two cases is that delay values are effectively unbounded at the maximum, whilst the available bandwidth is bounded at the upper end by the link capacities. The bandwidth conversion problem is therefore simpler, and so the following formula is sufficient:

$$\text{normalised bandwidth} = 1 - \frac{\text{available bandwidth}}{\text{minimum link capacity along path}}$$

The available bandwidth for a path is the minimum available bandwidth on all the logical links which form the path from the source to the destination node. The formula will return values close to 0 under low utilisation, and close to 1 under high utilisation conditions. As with delay feedbacks, it is beneficial to smooth updates of available bandwidth in order to smooth out short-term fluctuations in available bandwidth. The exponential smoothing technique can be used as follows:

$$\text{normalised bw(new)} = \varepsilon(\text{normalised bw(old)}) + (1 - \varepsilon)(\text{normalised bw(returned)})$$

with $0 < \varepsilon < 1$

A high value of ε causes the normalised bandwidth to react slowly to bandwidth changes, whilst too low a value of ε will cause the smoothing process to fail as the value returned will match short-term fluctuations too closely. Values used in real network scenarios vary according to application. For example Cisco's Weighted Random Early Detection (WRED) mechanism uses a normalised value for the average buffer depth in order to ascertain whether to probabilistically discard IP packets [70]. As buffer depth in IP routers can vary significantly over short periods of time, the weighting factor is set to 0.998 so that a 'longer term' average value is returned. However with our application reserving bandwidth on the network for the duration of a call, it is not expected that link utilisation values will vary greatly in the short-term. Therefore a lower value of 0.7 is used for the ensuing experiments.

6.3.2 Reinforcement algorithm selection

The benefits of using the S-model response environment revolve around the variable environment response which can be generated. To fully utilise the feedback information, the reinforcement algorithm ideally requires the capability of having an updating function which can operate at the same level of granularity as the environment feedback.

This requirement effectively precludes the use of discretised schemes as their benefit concerns the coarse granularity of their action probability updates when these are close to unity. This leaves the two continuous schemes which were recommended from the investigation undertaken using the P-model response environment: LRεP and GE with the 'x^3' updating function. As with chapter 5, because this application type's environment is rarely in a steady-state condition and the GE algorithm requires a large number of iterations for it to converge, it is not used in this experiment. Therefore only the LRεP reinforcement algorithm is used for this new application of learning automata to the connection-oriented reservation-based routing problem. It should be noted however, that it is possible that other reinforcement algorithms are suitable for use with non-autonomous S-model environments such as this, since the performance experiments undertaken in chapter 4 utilised the P-model response environment.

The varying feedback response returned by the environment according to the amount of available bandwidth on the route chosen, will cause a varying adaptation rate of the action probabilities. Therefore the use of automatic adaptation of the reinforcement algorithm learning rate is not utilised in these experiments, as their purpose is to show the effectiveness

of the new method of applying learning automata to the routing problem in networks. Were the learning rate adaptation method utilised in these experiments, further convergence speed gains with higher steady-state accuracy might be expected to result. However its usefulness might be limited to scenarios of high resource utilisation, where the available bandwidth feedback response is relatively low which causes a small change in the action probabilities and so a small entropy change.

6.4 Experimental results

This section is composed of two parts. The first shows the effects of changing the fixed learning rate on the resulting algorithm performance, the purpose being to use a learning rate parameter which returns an acceptable medium between high convergence speed and low subsequent steady-state accuracy.

6.4.1 Learning rate effects

As a fixed learning rate is required, so high learning rates are selected for study. This is valid because the variable feedback response is bounded at the minimum and maximum by values of 0 and 1, which are those returned with the P-model response environment with reward and penalty responses respectively. Therefore as the environment reward response would very rarely be close to 0, this being true only in cases of little or no utilisation, so the updating effect on the action probabilities would be significantly less than the full high learning rate when using the P-model response environment.

The highest learning rate used for the P-model LRεP algorithm was 5%, so this was chosen for the initial experimental evaluations. Three experiments were performed for each learning rate tested: one producing a low blocking probability (around 5%), another producing a higher one (around 35%), and a final one producing one still higher (around 65%). The rationale for this was to see if a learning rate producing good relative performance in a certain scenario would produce poorer relative performance in differing ones.

Initial algorithm results showed promise, with a lower blocking probability (3.8%) and subsequent variation returned under low network loading than either AAMH (4.9%) or the P-model learning automata method with automatic algorithm selection (5.9%). However it took a larger number of iterations for the algorithm to converge, from 100,000 with the P-

model automatic algorithm selection to 400,000 with the S-model LRεP. Another weakness evidenced was that a higher blocking probability compared with the other algorithms was returned for the higher loading rates. In order to improve on the slow convergence rate, the learning rate for the S-model LRεP was increased in 2% increments until poorer blocking probability performance ensued.

As expected, Table 19 shows that as the learning rate was increased, so the connection attempts required for convergence generally decreased. However an unexpected conclusion is drawn from Table 17, in that the resulting blocking probability actually decreases as the learning rate is increased. This initially counter-intuitive result can be explained by examining the entropy of the action probabilities for the various learning rates. It was seen that after convergence the global entropy value was 389 when using the 5% learning rate, and 345 with the 17% learning rate. This indicated that the action probabilities had converged to significantly different values with the 17% learning rate, implying by the better network performance that the action probabilities had been able to converge more accurately, and not get stuck in local minima as with the 5% learning rate. Remaining in local minima might easily occur with S-model reinforcement algorithms since when the environment feedback is close to 1 the action probability update is very small. Therefore it is possible when using lower learning rates for the action probabilities not to fully converge before link utilisation levels are close to capacity, so that the action probabilities do not vary significantly from then on when using the statistically constant user demands of these experiments. As the effect of raising the loading rate is to cause link saturation to occur in fewer connection attempts, since the holding time remains constant, so the action probabilities have less iterations to converge resulting with the higher blocking probability when compared with the other algorithms.

By comparing the resulting blocking probability (Table 17) with the iterations required for convergence (Table 19), it was decided to use a learning rate of 17% as this produced both a low blocking probability and low number of iterations before convergence. However even at this high learning rate, the resulting network blocking probability under high loads is worse than with the previous algorithms. It therefore seems to be the case that a significant portion of the overall action probability convergence for the P-model automatic algorithm selection occurs when the links are already at or close to saturation. This explains why increasing the learning rate of the S-model LRεP still further does not reduce the blocking probabilities, but rather causes them to increase slightly as the pre-link saturation convergence is now less accurate.

Learning rate	Traffic load (conn./sec.)		
	0.05	0.12	0.33
5 %	0.038	0.364	0.677
7 %	0.034	0.364	0.676
9 %	0.036	0.363	0.676
11 %	0.036	0.363	0.677
13 %	0.035	0.363	0.676
15 %	0.036	0.363	0.677
17 %	0.035	0.364	0.677
19 %	0.036	0.363	0.677
25 %	0.039	0.365	0.676

Table 17: Average network blocking probability for 28-node network with S-model LR&P and various learning rates

Learning rate	Traffic load (conn./sec.)		
	0.05	0.12	0.33
5 %	0.009	0.023	0.018
7 %	0.01	0.022	0.017
9 %	0.009	0.023	0.017
11 %	0.009	0.022	0.018
13 %	0.010	0.023	0.017
15 %	0.009	0.022	0.017
17 %	0.009	0.023	0.018
19 %	0.010	0.023	0.018
25 %	0.010	0.022	0.018

Table 18: Standard deviation on network blocking probability for 28-node network using S-model LR&P and various learning rates

Learning rate	Traffic load (conn./sec.)		
	0.05	0.12	0.33
5 %	400,000	60,000	20,000
7 %	400,000	40,000	16,000
9 %	300,000	37,000	20,000
11 %	105,000	12,000	16,000
13 %	150,000	20,000	12,000
15 %	150,000	10,000	17,000
17 %	140,000	10,000	15,000
19 %	135,000	11,000	13,000
25 %	80,000	17,000	20,000

Table 19: Global connection attempts for convergence for 28-node network using S-model LR&P and various learning rates

6.4.2 Comparative algorithm results

Figure 66 shows the convergence of the S-model LRεP algorithm with the fixed 17% learning rate under a statistically constant loading rate with an interarrival time of 20 seconds. Comparing this with that for AAMH in Figure 67 and the P-model learning automata with automatic algorithm selection in Figure 68, it is seen that the S-model LRεP algorithm produces both lower average blocking probability and steady-state variation. This algorithm outperforms the other two at this loading level because it can choose under-utilised routes longer than the shortest or next shortest. This is also true of the P-model automatic algorithm, but its coarse granularity environment feedbacks seem to preclude it from doing so effectively. This is shown by the entropy traces for the two, the one for the P-model automatic algorithm clearly converging to a significantly lower value, finishing the simulation at 80 instead of 250 for the S-model LRεP. This lower value for the P-model automatic algorithm indicates that it is more prone to action probability extremes (i.e. close to 0 or 1), precluding the occasional use of longer paths which the S-model LRεP algorithm uses. It is under these relatively lower levels of loading that the algorithm can occasionally use longer paths for beneficial effect, for at higher loading rates the extra network capacity used for a longer path cannot be used by arriving connection requests, leading to a higher blocking probability and so worse comparative performance. This is shown by the comparative results in Table 20, indicating that at loading levels of 0.1 connections per second and higher, poorer results than the other two algorithms are returned.

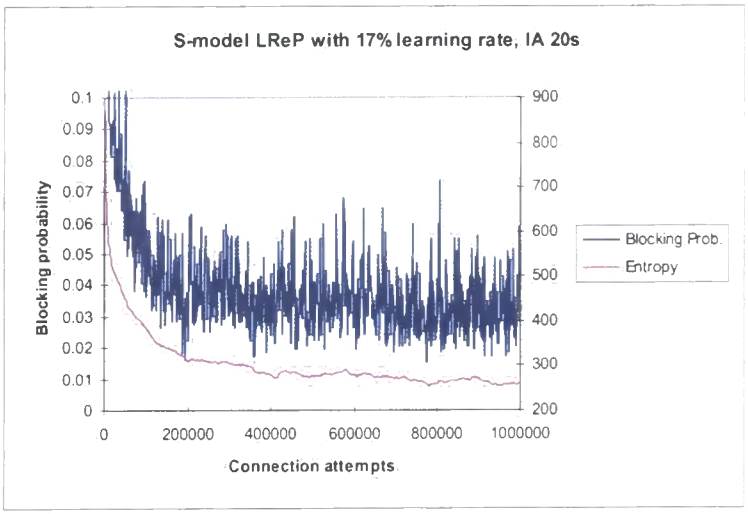


Figure 66: 28-node network with S-model and LRεP with 17% learning rate, IA 20s

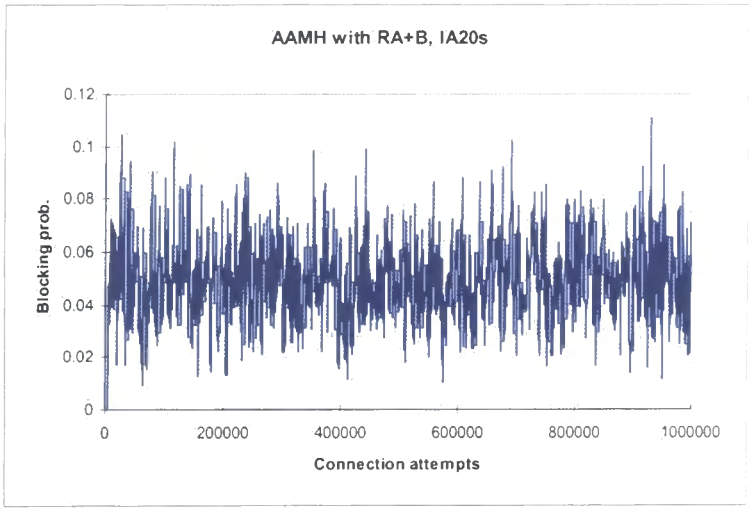


Figure 67: 28-node network with AAMH and RA+B

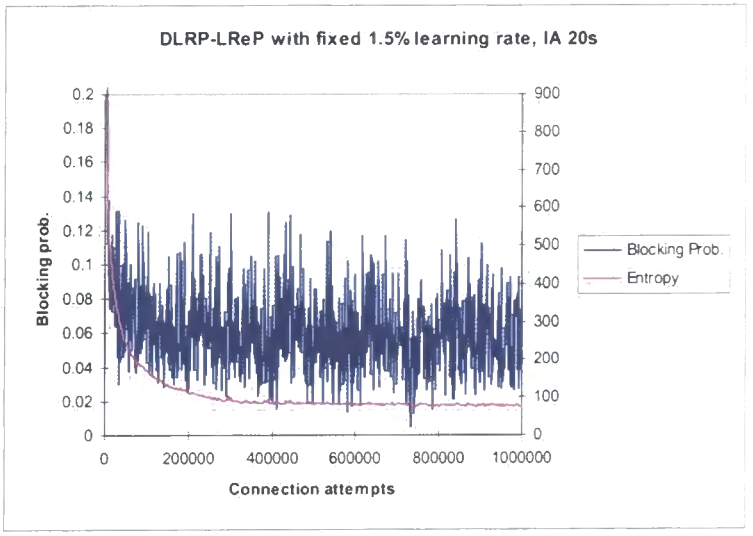


Figure 68: 28 node network with P-model and automatic algorithm selection

Table 21 shows that the S-model LRεP algorithm results with lower or equal steady-state variation than the other two algorithms. This must be due to the relative stability of the action probabilities under saturated or close to saturated link conditions, for the AAMH and the P-model environment responses might have significant variations of actions and feedbacks respectively (which in turn leads to different probabilistic actions for the P-models automatic algorithm).

Table 22 shows that the S-model LRεP algorithm needs a slightly higher number of iterations to converge compared to the P-model automatic algorithm under the relatively lower loadings, the converse being true for the higher loadings. This is acceptable given the view that convergence speed was sufficient for the P-model automatic algorithm.

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
S-model ReP	0.035	0.130	0.297	0.364	0.438	0.546	0.677
AAMH	0.050	0.128	0.276	0.344	0.421	0.529	0.662
Automatic	0.059	0.138	0.287	0.353	0.423	0.532	0.663

Table 20: Average network blocking probability for 28-node network with S-model and LRεP with 17% learning rate

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
S-model ReP	0.009	0.018	0.024	0.023	0.023	0.020	0.018
AAMH	0.017	0.021	0.025	0.026	0.026	0.026	0.018
Automatic	0.021	0.025	0.026	0.025	0.023	0.022	0.018

Table 21: Standard deviation on network blocking probability for 28-node network using S-model and LRεP with 17% learning rate

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
S-model ReP	140,000	35,000	11,000	10,000	14,000	10,000	15,000
Automatic	100,000	33,000	8,000	11,000	15,000	12,000	16,000

Table 22: Global connection attempts for convergence for 28-node network using S-model and LRεP with 17% learning rate

The following three figures show results for the more realistic network scenario, that using the trend user demand models. Figure 69 shows the graphs for the S-model LRεP algorithm, the minimum peak blocking probability in any of the simulated days being just under 19%. This compares extremely favourably with the P-model automatic algorithm which peaked at 22% (Figure 71). A comparison of the two algorithms' entropy traces indicates the reason for the superior performance of the S-model LRεP algorithm. As was the case previously, the entropy trace values are significantly higher for the algorithm, showing that it is using a greater number of paths since action probability extremes are used less often.

However the S-model LRεP algorithm is similar or slightly worse than the AAMH algorithm performance (Figure 70). Since the LRεP algorithm outperformed the AAMH algorithm using statistically constant load user demands, so this slightly inferior performance is due to the required iterations before convergence. With its ability of utilising a wider set of paths however, it is expected that the S-model LRεP algorithm will outperform the AAMH

algorithm in most network scenarios, as these will generally evidence varying degrees of non-symmetry in the user demands. Therefore the conclusion drawn from comparing Figure 69 with Figure 70 may be thought of as the worst-case comparative S-model LRεP performance.

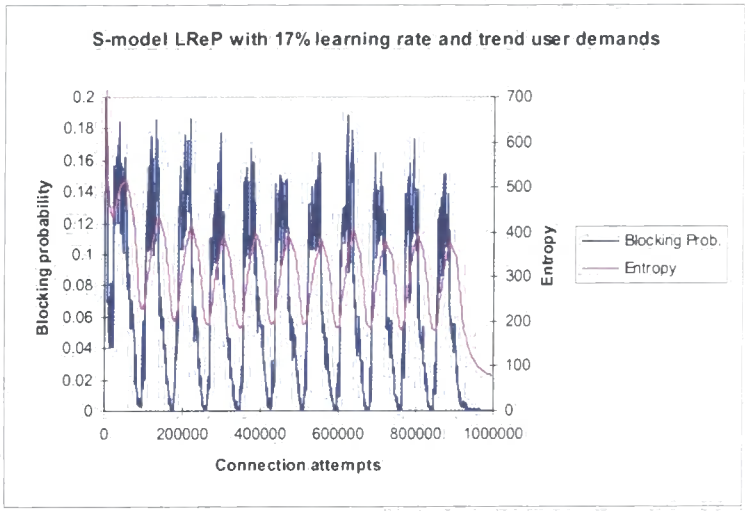


Figure 69: Performance of S-model LRεP LA with trend user demands

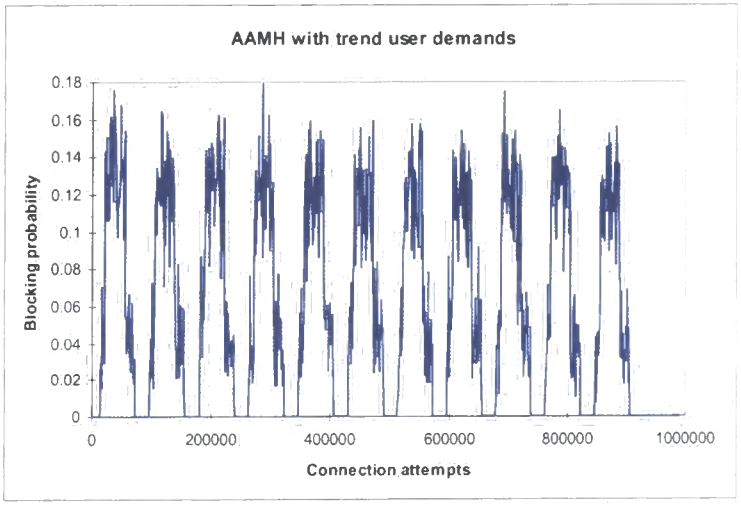


Figure 70: Performance of AAMH with trend user demands

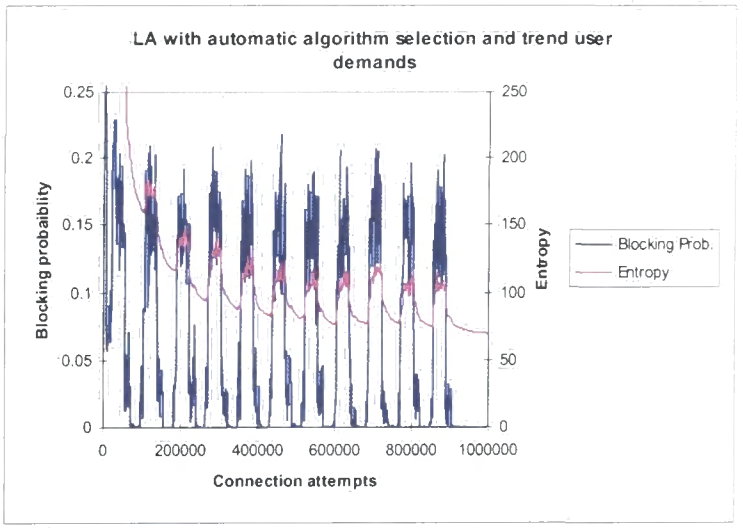


Figure 71: Performance of P-model LA with automatic algorithm selection and trend user demands

6.5 Summary

The previous chapter detailed two improvements to the standard learning automata application to routing in reservation-based networks: adaptive learning rates and automatic reinforcement algorithm selection. Of these two the automatic algorithm selection method produced the greater performance improvement, but this was still short of the superior performance of the AAMH algorithm of chapter 3, both in terms of blocking probability and iterations required for convergence.

The main reason noted for this inferior performance is the P-model response environment, in that no differentiation is possible between routes almost saturated and others with low utilisation, both returning an environment feedback of zero. By using an S-model response environment with a utilisation level based feedback, an improved load balancing would ensue (especially at lower loading rates), leading to an expected reduced blocking probability.

Normalising the available bandwidth into the range 0 to 1 required a relatively simple formula as the maximum value is bounded by the link capacities. The formula is as follows:

$$\text{normalised bandwidth} = 1 - \frac{\text{available bandwidth}}{\text{minimum link capacity along path}}$$

In order to smooth out short-term fluctuations, the exponential smoothing technique is used.

As using the S-model response environment precludes the use of discretised reinforcement schemes, so the fixed LRεP algorithm was used. In order to have the best

learning rate, various experiments were undertaken using differing loading rates, whilst progressively increasing the learning rate until poorer performance ensued. A high learning rate of 17% was found to be superior to others, both in terms of blocking probability and iterations for convergence. Whilst the latter is expected, the former was initially surprising, but can be explained by the fact that a high learning rate will cause the action probabilities to converge quickly before saturation of links occurs, the environment feedbacks being much smaller thereafter.

Comparing the algorithm's results for statistically constant user demands with those of AAMH and the P-model automatic algorithm selection, showed that the former outperformed the two latter under low loading levels, the converse occurring as the loading increased. Using the trend user demands showed that the S-model LRεP and AAMH results were fairly comparable, both returning a comfortably lower blocking probability than the P-model automatic algorithm.

The superior performance of the S-model LRεP algorithm under relatively low loads was shown to be due to its ability to use a larger set of possible routes. Whilst these network scenarios have used symmetrical network loading, non-determinism of user demands in real networking situations would generally cause non-symmetry of loading. It is therefore expected that this algorithm would improve its relative performance compared with the others still further under real network scenarios.

7 A hybrid routing algorithm utilising both shortest-path and learning automata concepts

7.1 Introduction

The work presented in the previous chapters has sought to improve the performance of learning automata methods, when applied to the routing problem in reservation-based networks. However, despite the various improvements, weaknesses with the use of learning automata are still evident; namely the large number of iterations required before convergence, and therefore the poorer blocking probability under certain circumstances when compared with the AAMH algorithm. In order to overcome these weaknesses, the work contained in this chapter seeks to combine the AAMH algorithm with the S-model LRεP learning automata method to produce a hybrid algorithm. The Specification Description Language (SDL) is used to detail the algorithm's functionality.

As with the automatic switching systems proposed in chapter 5, the average local entropy change is used in the decision making aspects of whether to switch from the AAMH to the S-model LRεP algorithm and vice versa. Once the relevant switching thresholds are found via the use of the four node experiment, the algorithm is applied to the 28 node network in various experiments with both statistically constant and trend user demands. Finally the results of the hybrid algorithm are compared to those of the AAMH and S-model LRεP algorithms when used alone, and conclusions on overall performance differences drawn.

7.2 Using learning automata for routing in real networks

The following sub-sections begin with a description of the reasons why a re-application of learning automata based network routing was undertaken. Having provided the rationale, the proposed new routing algorithm is presented in detail, including a flowchart representation of the algorithm. Finally this main section ends with the experimental details for calculating the necessary switching thresholds, based on the average local entropy changes.

7.2.1 Rationale for a novel learning automata based routing algorithm

Real networks are initially designed, provisioned and configured, based on the expected user demand traffic pattern or matrix. This is required so that the designers can reasonably choose the aggregate finite network resources needed together with an initial configuration and policies for sharing them fairly between different streams of user demand traffic. Were the designers to have perfect knowledge of the user demands that will be present on the network, then an optimal provisioning and route configuration might be put in place, requiring just single routes from each source to its destinations. However as it is never possible to have perfect knowledge of user demands, so the expected traffic demand pattern will never exactly match that seen on the actual network. In order to allow variations in the user demand to still have access to the network, alternate routes from source to destination nodes can be configured, so that under-utilised network resources on one main route may be utilised by an alternate path. The ordering of these alternate route attempts may be either static or dynamic. In dynamic routing the ordering of alternate path selection would be changed according to the equipment status and possibly also the network loading at the time. Static routing would not vary the order of alternate path selection. Using either means, the routes are chosen deterministically, based on both expected or historical user demands and possibly current traffic patterns.

Stochastic learning automata based routing, on the other hand, performs its decisions using a stochastic rather than deterministic paradigm. The action probabilities are determined, but the choice of a specific action occurs stochastically. The justification for this mechanism is that the non-autonomous environment response to a node's routing actions is stochastic because of the routing interactions occurring at other parts of the network. Therefore the learning automata can adjust their action probabilities in order to produce a close to optimal stochastic environment response. This stochastic environment response is not just a feature of applying learning automata to the routing problem in this way, but is also present with deterministic routing methods so that performance measures for a such routing algorithms are often given in terms of blocking probability. However it is not the case that the stochastic environment response seen with traditional routing methods is caused by the routing mechanism as this is deterministic. It is therefore solely a factor of the user demands which are non-deterministic, and how they are allocated to the network resources by the routing mechanism. Stochastic learning automata applied to the routing function in this manner are tracking the non-deterministic network load by characterising the combined non-deterministic user demands and demand to network allocation method. This paradigm arose out the observation that network load is non-deterministic, even if the demand allocation

mechanism is deterministic. It seemed reasonable to combine the characterisation of the user demands and network resource reservation function into one as their combined effect is non-deterministic, and learning automata essentially characterise a non-deterministic environment. However improved results from the application of learning automata may arise by applying them in a different way, so that they characterise purely the non-deterministic user demands, and these results are used in deterministically calculating the most applicable resource allocation. Recent work has indicated the need to apply learning automata to the routing problem in this different manner; unbundling the non-deterministic environment response by the learning automata characterising the non-deterministic user demands themselves rather than their effect on the network response [65].

The work collated in this chapter does not go fully down this route, but may be thought of as an intermediate step as it brings together the use of both standard deterministic methods together with a learning automata component. Such a method may be used with any non-autonomous environment application, in order to improve the performance of a learning automata mechanism, and therefore also aid the acceptance of stochastic learning automata methods. A hybrid solution for the routing problem is given: the proposed mechanism being composed of both a dynamic shortest-path component and a stochastic learning automata component, both of which are used to make the routing decisions. The initial motivation for designing this hybrid algorithm was to overcome performance weaknesses of existing applications of learning automata. Whilst learning automata methods have been shown to produce near optimal steady-state performance, their main weakness is in the slow convergence of their action probabilities when tracking an environment to a new steady-state position. Traditional methods, such as shortest-path network routing schemes, normally have good initial performance due to their deterministic action choices, but suffer somewhat in steady-state conditions due to their limited number of possible choices. Therefore the proposed hybrid algorithm solution to interacting with non-autonomous environments is to use traditional deterministic algorithms under moving environment state conditions, whilst using the environment response to continue to update the learning automata action probabilities. When the environment is close to a steady-state condition, the learning automata part may then be used to perform the actions, optimising the action probabilities still further in order to produce a better performance response from the environment. Such should occur due to the reinforcement algorithm updating the action probabilities so that not only are shortest-paths chosen, but occasionally also slightly longer ones [19], so providing a greater degree of load balancing and use of spare network bandwidth.

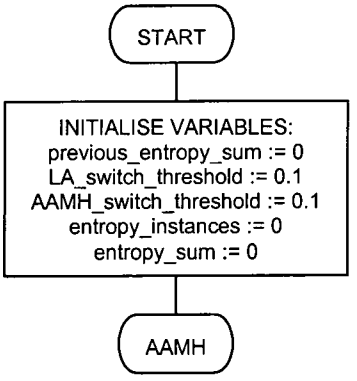
Not only should this hybrid method produce better performance than the use of either component on its own for the routing function, but the deterministic component in the algorithm causes the updating of the learning automata action probabilities to follow paths

engineered in the network because of expected user demands, network topology and configuration. The action probabilities then become set close to the expected user demand traffic pattern under moving network state conditions, adapting to the actual and unexpected traffic pattern when the network is close to steady-state conditions.

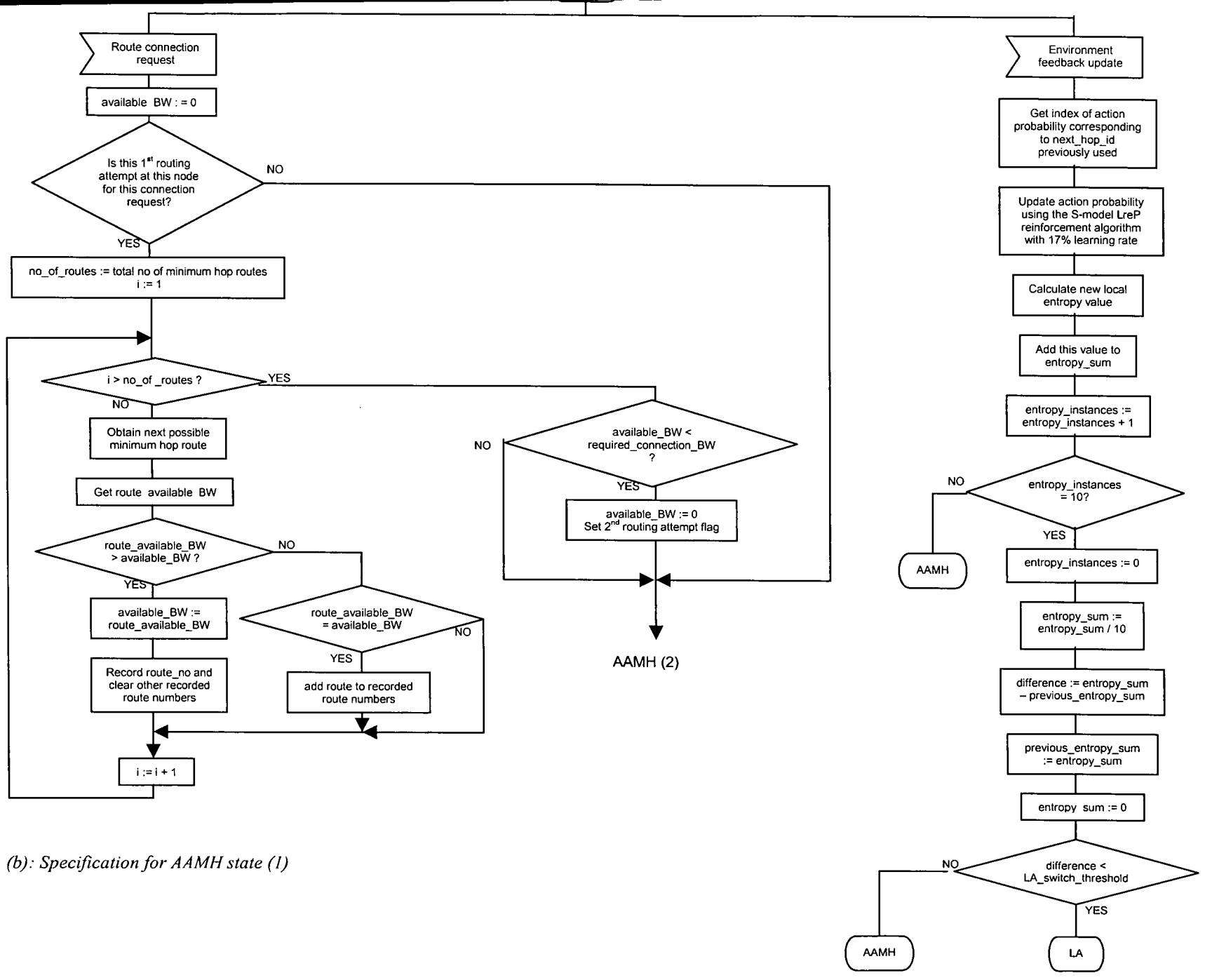
7.2.2 Hybrid algorithm details

The following section describes the proposed hybrid algorithm in detail, utilising the CCITT-Specification and Description Language (SDL) [68]. The algorithm has two main states: the AAMH and the LA states, these describing whether the algorithm process is performing the functionality of the AAMH or the LA algorithm for its routing decisions. Part (a) of Figure 72 shows that after initialisation, the routing algorithm process commences in state AAMH.

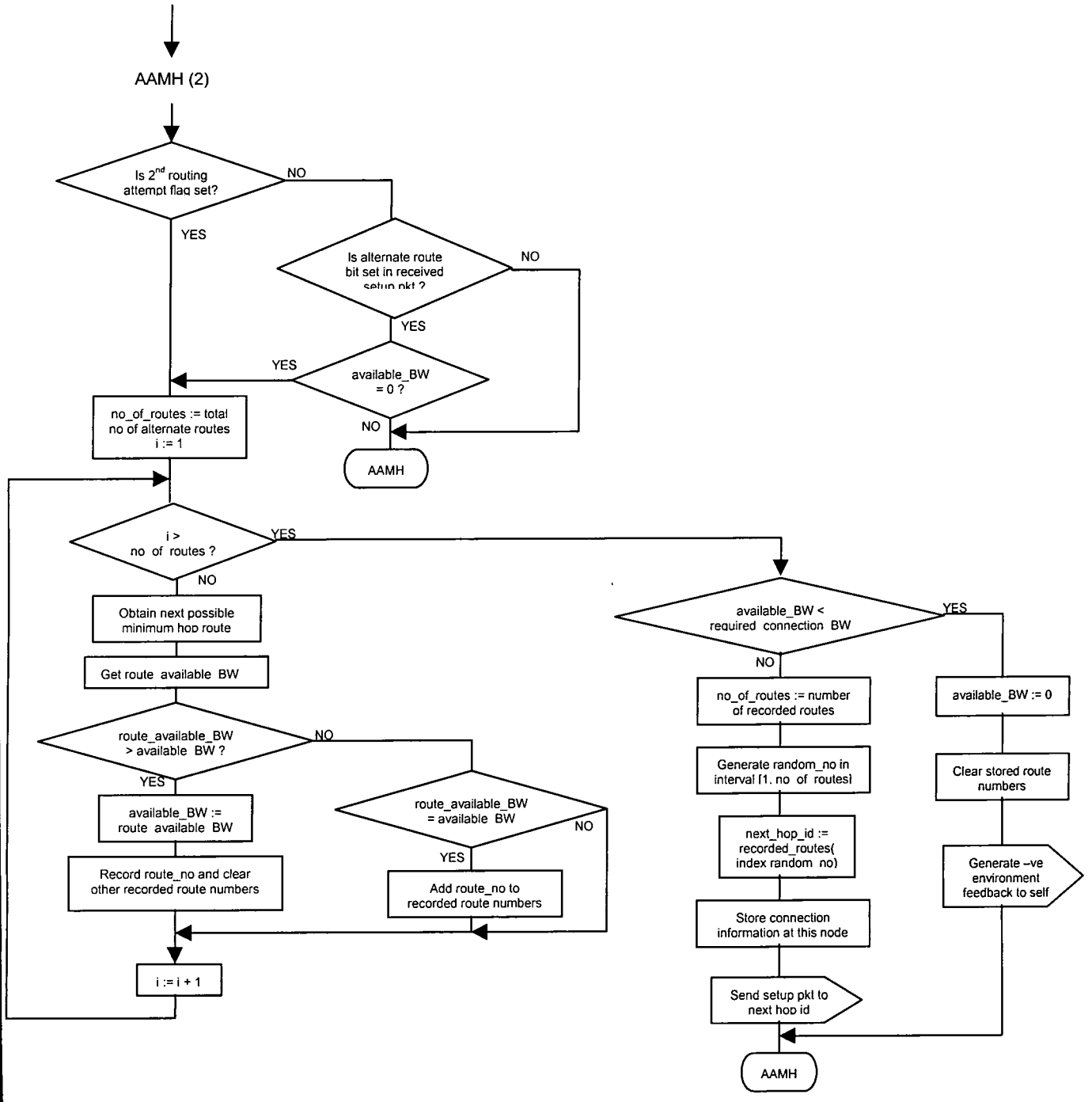
As may be seen in parts (b) and (d), either state can receive and operate on the same two events: a route connection request, and an environment feedback update. The operations performed on reception of the environment feedback update when in either state are very similar. In both cases, the index of the action probability corresponding to the routing attempt previously made is obtained. Next it is updated using the LRεP reinforcement algorithm with a 17% learning rate, with the other action probabilities being adjusted accordingly. The entropy of the new action probabilities is then calculated, and after ten environment feedbacks the average entropy over the set of feedbacks is compared with that for the previous set. If the current state is 'AAMH' and the difference is less than the 'LA switching threshold', a transition to state 'LA' occurs. If the current state is LA and the difference is greater than the AAMH switching threshold, then a transition to state 'AAMH' occurs. If no state transition occurs, the process remains in its current state.



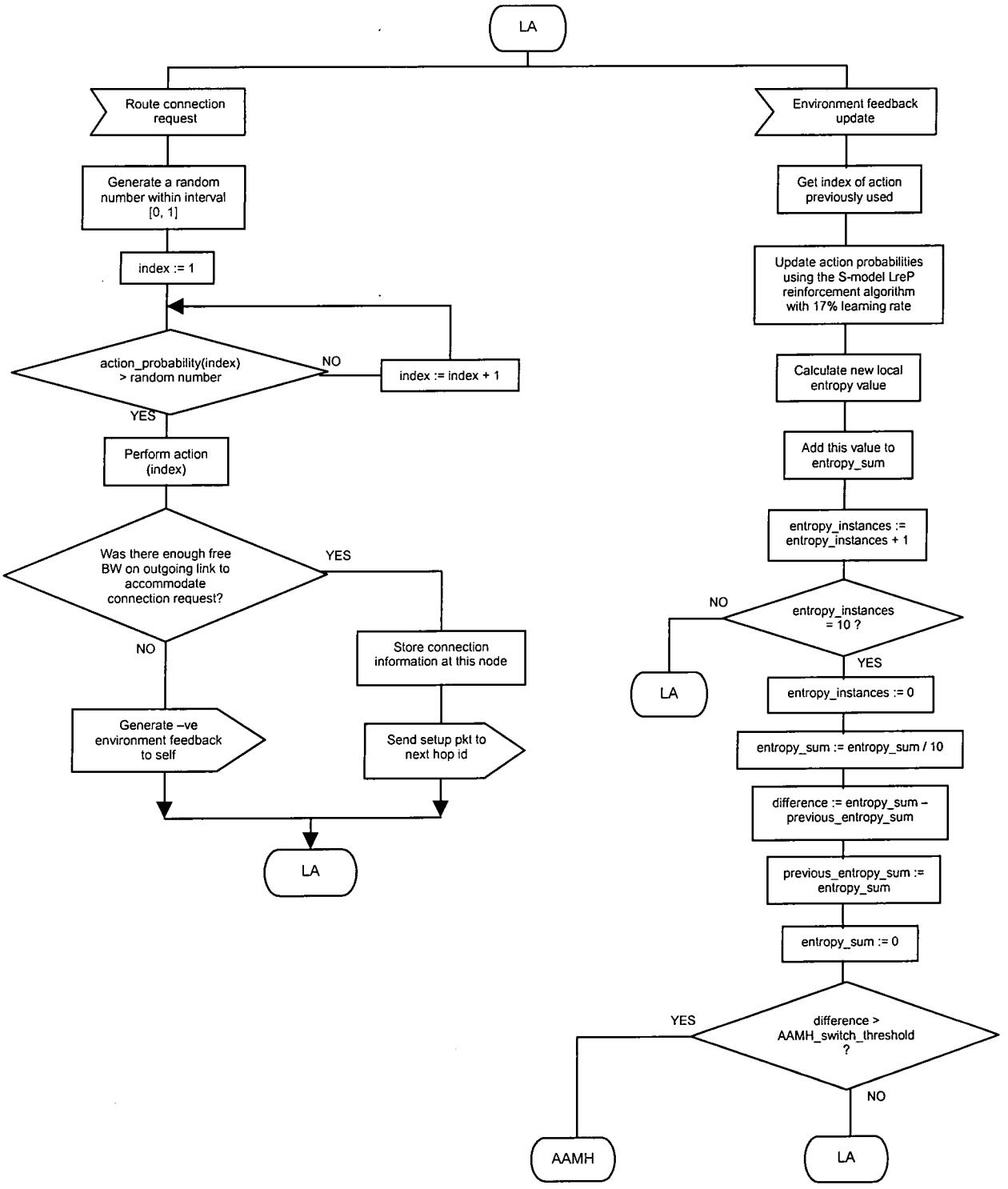
(a): Specification at start



(b): Specification for AAMH state (1)



(c): Specification for AAMH state (2)



(d): Specification for LA state

Figure 72: SDL representation of proposed hybrid routing algorithm

Comparing the difference in functionality between the two states on reception of a 'route connection request' event shows the simplicity of the learning automata implementation to that required for the AAMH algorithm. When in the 'LA' state, the algorithm generates a random number within the interval of [0, 1], and chooses the corresponding action (which in this case is the outgoing link) as shown in part (d). If the outgoing link cannot support the bandwidth required for the connection request due to it being close to full utilisation, a negative environment feedback is generated for the process instance. If the outgoing link can support the connection request, then the connection information is recorded at the node and the setup packet sent on to the next node via the chosen outgoing link.

The 'AAMH' state functionality, on the other hand, evaluates all possible minimum hop routes to the destination, and if none of those can support the connection request's bandwidth requirement all the alternate routes are evaluated. The detailed specification shown in parts (b) and (c) indicates the significantly higher level of complexity of this procedure when compared with that of state 'LA' shown in part (d).

Some of the routing algorithm's associated functions have been left out of the specification presented. For example, this includes the automatic generation of a 'positive environment feedback to self' on reception of a set-up acknowledgement packet returning towards the source node of the connection request.

7.2.3 Switching threshold calculation

The proposed method for determining when to switch from shortest-path to learning automata based routing and back again is the local entropy measure, as used in the previous chapters. The hybrid algorithm is therefore a combination of the algorithm solutions outlined in both chapters 3 and 6. The shortest path component uses the AAMH algorithm with RA+B link state update method, storing and acting on load band link state information. Although only load band information is stored for AAMH, actual minimum available bandwidth on a route is returned and used to continuously update the learning automata action probabilities.

The threshold calibrations are done using the four node network, as the converged action probabilities can then be calculated analytically beforehand. Having done so, the resulting local entropy value after convergence can be derived, this being around 3.35 under the experiment's loading rate. Figure 73 shows the local entropy for the node with the feeder traffic source when using the AAMH algorithm, whilst still updating the learning automata action probabilities. This algorithm results with entropy at around 3.42 which is generally higher than that for learning automata, as shown by the comparison with Figure 74 which

towards the end of the graph shows consistent entropy values around 3.35 . This effect is explained by recalling that the AAMH algorithm is determining and performing the actions whilst updating the learning automata action probabilities with the environment responses. Therefore the action probabilities generally will not converge to the expected values under AAMH, probably requiring further convergence once the learning automata are switched in.

From these two graphs, the threshold values are obtained. When using the AAMH algorithm, a change in the entropy less than 0.1 between averaged ten point samples should cause a change to the learning automata method. When using the learning automata, an entropy change greater than 0.1 should cause the AAMH algorithm to be switched in. The benefits of this hybrid method is shown in Figure 75 which indicates a faster convergence than the learning automata method alone, and a lower variation in entropy and so action probability values than when using the AAMH algorithm alone. Also the entropy values are generally closer to 3.35 than when just using AAMH, showing that the hybrid algorithm should produce lower blocking probability results than pure AAMH.

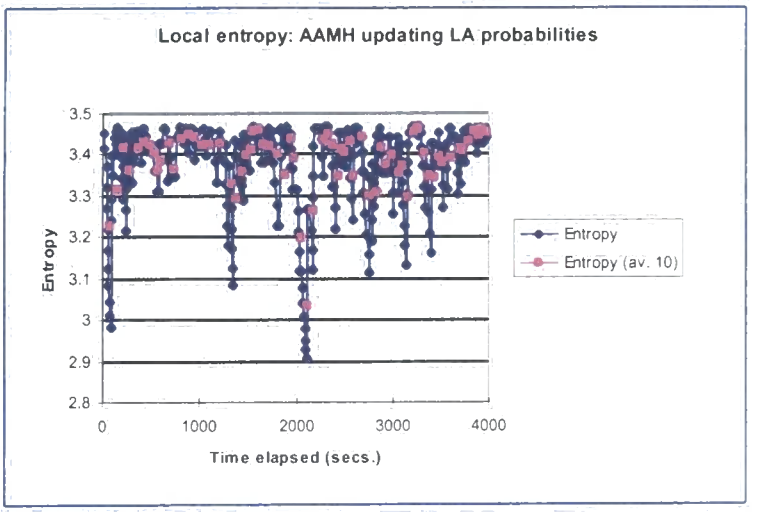


Figure 73: Local entropy using 4-node network with AAMH and action probability updates

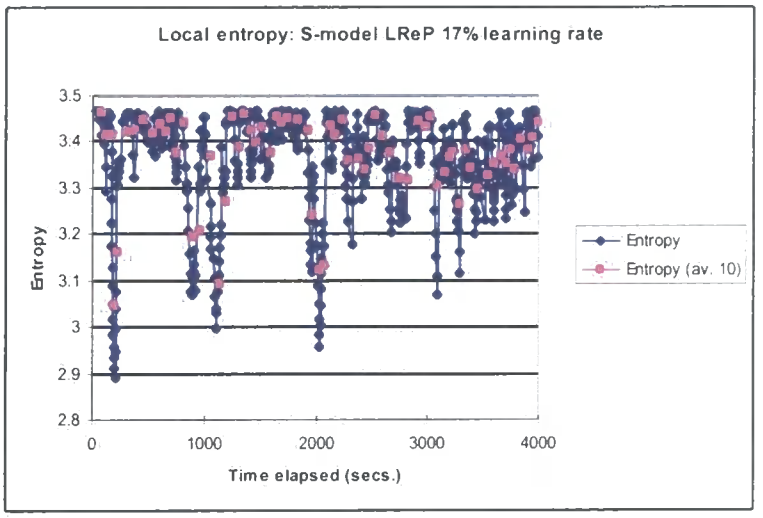


Figure 74: Local entropy using 4-node network with S-model LRεP and 17% learning rate

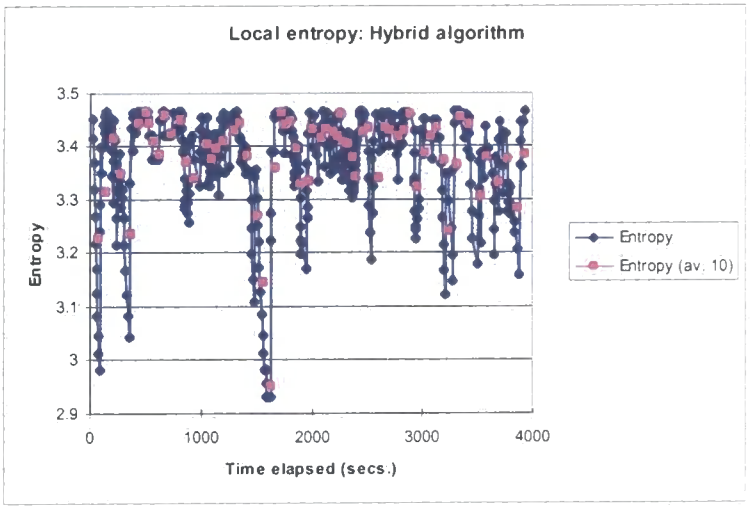


Figure 75: Local entropy using 4-node network and hybrid algorithm

7.3 Experimental results

Figure 76 shows the resulting blocking probability and entropy of the hybrid algorithm under a statistically constant loading rate with an interarrival time of 20 seconds. As may be seen when compared with Figure 78, the blocking probability graph follows that of AAMH fairly

closely, having similar peaks and variations. Like the case when using AAMH, the hybrid algorithm initially produces zero blocking probability, the value then increasing as connections arrive and the network becomes congested. This is in contrast with that for the S-model RεP algorithm of Figure 77, where convergence of the action probabilities is initially required to bring the blocking probability down from an unacceptably high value, many connection attempts being initially blocked because of routing loops or the maximum number of permissible hops being reached.

The similarity in initial performance characteristics is explained by the fact that AAMH is initially used by the hybrid algorithm, the S-model LRεP algorithm being switched in once the action probability variations have stabilised. Therefore the initial blocking probability for the hybrid algorithm is zero, and the action probabilities are updated due to the AAMH actions, therefore initially converging to both the shortest paths and those alternate routes of closest hop distance. This explains the lower entropy value converged to by the hybrid algorithm, evidenced when comparing Figure 76 with Figure 77. A much quicker convergence in entropy occurs using the hybrid algorithm, with the lower final value indicating a fewer number of possible route options for the action probabilities are closer to extremities.

Table 23 compares the blocking probability performance of the hybrid algorithm with AAMH and the S-model LRεP algorithms over various statistically constant loading rates. At the low loading rate of a 20 second average connection request interarrival time, the hybrid algorithm returns a lower blocking probability than AAMH, but higher than the S-model LRεP algorithm. However at all other loading rates, the hybrid algorithm consistently out-performs both other algorithms significantly, generally returning a lower blocking probability of at least 3 percentage units in value. This considerable improvement is achieved with no

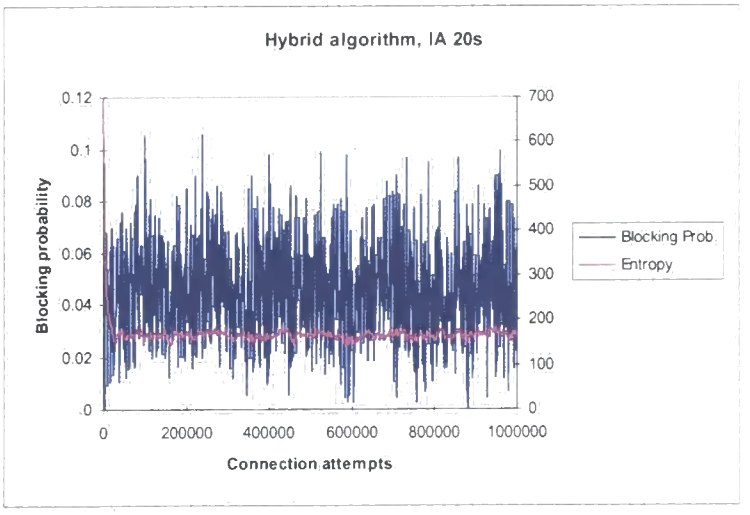


Figure 76: 28-node network with hybrid algorithm, IA 20s

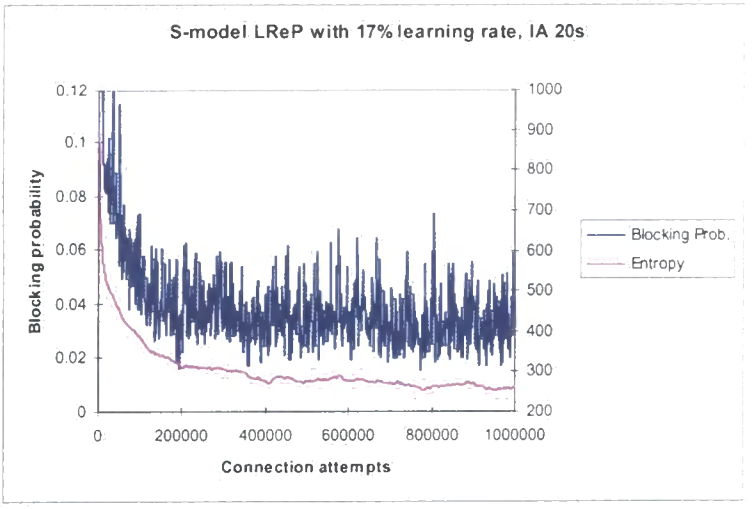


Figure 77: 28-node network with S-model and LRεP with 17% learning rate, IA 20s

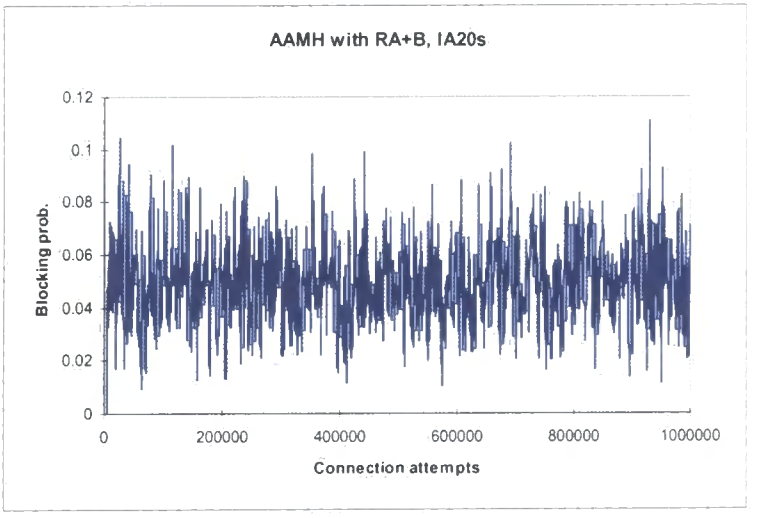


Figure 78: 28-node network with AAMH and RA+B, IA 20s

start-up convergence period, as shown by Table 25. However rather than taking the average blocking probability from zero iterations, calculations are performed from when the blocking probability has stabilised, this figure being indicated in brackets in the table. As was also shown by the faster entropy trace convergence of Figure 76 over Figure 77, these values are significantly lower than those of the S-model LRεP algorithm, over all loading rates.

The reason behind the better hybrid algorithm performance is indicated by Table 24, where the hybrid algorithm returns a consistently higher value of variation in blocking probability compared with the other two algorithms. It is seen that once converged, the S-model LRεP algorithm still generally has a greater choice of possible routes available to a destination than the AAMH algorithm, so returning a lower blocking probability under low

loading levels, and a higher one as the loading rate increases. The hybrid algorithm switches between the two algorithms, so switching between a possibly smaller set of paths to a larger set and vice versa. The switch between one set of paths and another will generally produce a corresponding step change in blocking probability, so resulting with a higher variation in blocking probability over a long time period. The reason why the hybrid algorithm returns superior blocking probability performance to the S-model LRεP algorithm is that by sometimes using the AAMH algorithm whilst still updating the action probabilities, the possible paths to choose from is limited to paths close to the minimum hop length. Whilst this hinders performance under low loading levels, as the loading rate increases so the avoidance of longer paths results with a significantly lower blocking probability. The hybrid results are consistently lower than the AAMH algorithm blocking probability for the opposite reason; in that the hybrid algorithm generally will have a greater number of possible routes to choose from, all these being close to the minimum hop route in length. It therefore seems that the initial guidance of the learning automata action probabilities by an application specific method whilst allowing the learning automata to diverge from that guidance if beneficial, results with superior overall performance than either of the methods singly used. The reason for this is that the use of learning automata by themselves require an infinite number of iterations before convergence to an ε-optimal value is achieved. This is also seen experimentally by Figure 77, where the entropy value is still decreasing after the end of 1,000,000 iterations, meaning that the action probabilities are still changing.

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Hybrid	0.045	0.114	0.266	0.334	0.406	0.511	0.643
S-model ReP	0.035	0.130	0.297	0.364	0.438	0.546	0.677
AAMH	0.050	0.128	0.276	0.344	0.421	0.529	0.662

Table 23: Average network blocking probability for 28-node network with S-model and LRεP with 17% learning rate

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Hybrid	0.019	0.021	0.027	0.027	0.026	0.025	0.019
S-model ReP	0.009	0.018	0.024	0.023	0.023	0.020	0.018
AAMH	0.017	0.021	0.025	0.026	0.026	0.026	0.018

Table 24: Standard deviation on network blocking probability for 28-node network using S-model and LRεP with 17% learning rate

	Traffic load (conn./sec.)						
	0.05	**0.067**	**0.1**	**0.12**	**0.14**	**0.2**	**0.33**
Hybrid	0 (10,000)	0 (19,000)	0 (8,000)	0 (9,000)	0 (7,000)	0 (9,000)	0 (13,000)
S-model ReP	140,000	35,000	11,000	10,000	14,000	10,000	15,000

Table 25: Global connection attempts for convergence for 28-node network using hybrid and pure S-model LRεP with 17% learning rate

Figure 79 shows the resulting blocking probability for the hybrid algorithm when using user demands with trends. Comparing this with Figure 80 which shows the resulting blocking probability for the S-model LRεP algorithm, indicates that the hybrid algorithm out-performs the S-model algorithm, consistently returning a lower blocking probability. The reason for this is shown by the entropy curves, that for the hybrid algorithm being lower than that for the S-model algorithm, indicating the main use of a smaller set of path possibilities, these being constrained when switching in the AAMH sub-algorithm to minimum or close to minimum hop routes.

The maximum peak of the hybrid algorithm is equal to that of the AAMH algorithm shown in Figure 81, but in general the peaks are lower when using the hybrid algorithm. Therefore the conclusion drawn from these experiments is that the hybrid algorithm generally out-performs both the AAMH and S-model LRεP algorithms singly used, under both static and dynamic statistical user demands.

The only proviso to this rule centres around the use of low loading situations. The blocking probability trace of the hybrid algorithm in Figure 79 evidences similar 'slow decay' of value at around 250,000 and 400,000 connection attempts, as the S-model LRεP algorithm always shows (Figure 80). This is in contrast to the sharp change in blocking probability to zero of the AAMH algorithm shown in Figure 81. Therefore even when using a utilisation based feedback, the action probability convergence worsens under low utilisation levels. When applying this observation to the general case, it seems that in low penalty rate application environments, where the environment state changes, less reliance on learning automata adaptation and greater leaning on the deterministic application control method would be beneficial.

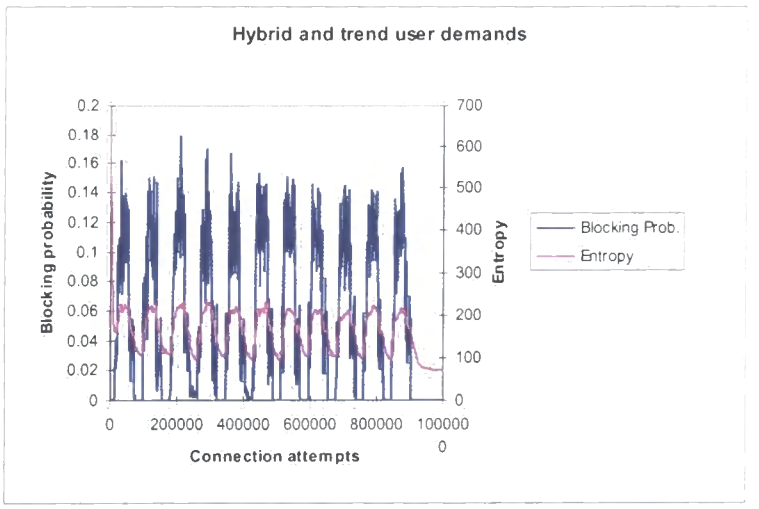


Figure 79: Performance of hybrid algorithm with trend user demands

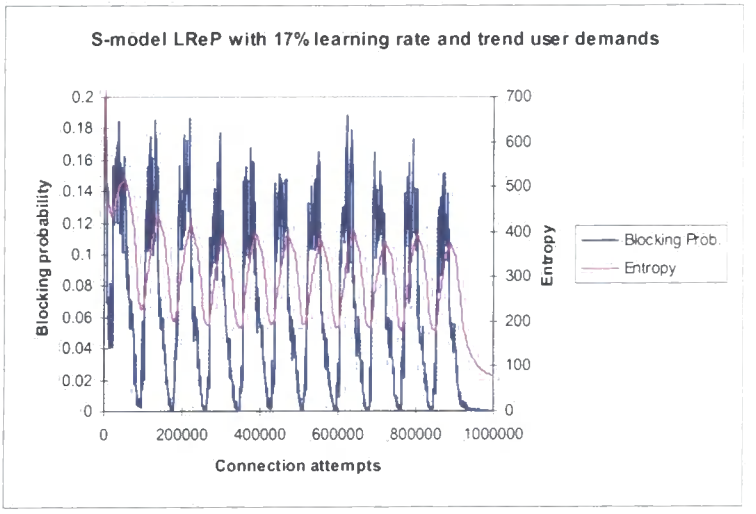


Figure 80: Performance of S-model LRεP LA with trend user demands

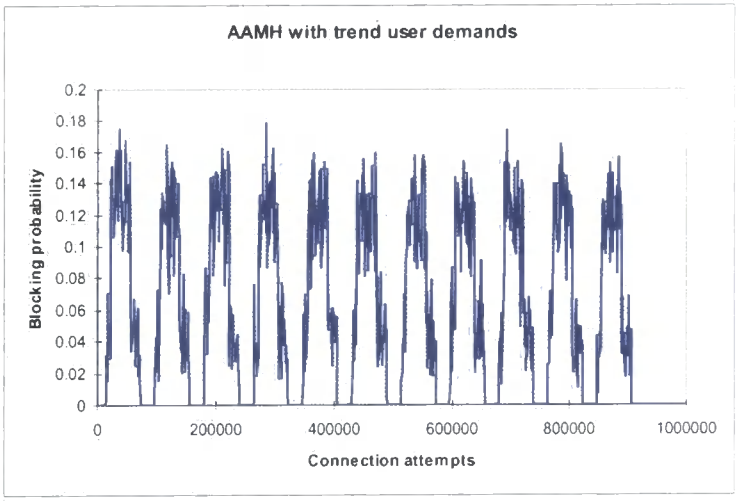


Figure 81: Performance of AAMH with trend user demands

7.4 Summary

The work presented in the previous chapters has sought to improve the performance of learning automata when interacting with a non-autonomous environment. These improvements have been demonstrated by applying learning automata to the routing problem in reservation-based networks. The learning automata performance was compared to that of one of the better performing of the commonly used methods for this problem, and concluded that weaknesses with the use of learning automata are still evident despite various improvements. The main weakness is the large number of iterations required before convergence, and therefore the poorer blocking probability under certain circumstances. The work contained in this chapter has sought to combine currently used deterministic algorithms for problem solving with the learning automata method. The experimental validation of the proposed hybrid algorithm solution is again the routing problem in reservation-based networks.

The strengths of the dynamic shortest-path algorithm called AAMH is that no a-priori convergence period is required, the algorithm immediately choosing routes deterministically whatever the dynamics of the network state. The strength of the learning automata algorithm on the other hand is that once fully converged, the resulting action probabilities should produce superior blocking probability performance due to the evaluation of a greater number of paths. The proposed hybrid algorithm seeks to combine these separate strengths and obviate the weaknesses by using the AAMH algorithm under moving network state conditions, and using the S-model LRεP algorithm under steady-state conditions. When the AAMH algorithm is used, the learning automata action probabilities are still reinforced, based on the environment response which is the minimum available bandwidth along the attempted route. When the learning automata algorithm is used, the local copy at each source node of the link states is still updated, this being used by the AAMH algorithm when it is switched in once the network state is seen to significantly move again.

As with the previous chapters, the proposed method for determining when to switch from shortest-path to learning automata based routing and back again is the local entropy measure. The four node network was used to determine the threshold values required for switching from one algorithm to the other. Once these thresholds were incorporated into the hybrid algorithm, it was applied to the 28 node network scenario under various loading rates.

When using statistically constant user demands the results indicated that in general the hybrid algorithm significantly outperformed both the AAMH and S-model LRεP algorithms singly used. Like the AAMH algorithm, the hybrid method requires no iterations for the blocking probability to reduce to acceptable levels, but the evaluation of a greater

number of path possibilities allows it to return a lower blocking probability. The results confirm these observations when using the trend user demand models, the hybrid algorithm generally returning a lower blocking probability than either of the two methods used alone.

It was seen that the main reason for this result is that when using the AAMH algorithm the learning automata action probabilities are strongly guided towards paths of minimum or close to minimum hop count. When the learning automata method is switched in, the main paths are therefore those of low hop count, but it still has the occasional use of longer under-utilised paths. These suppositions were confirmed by the lower entropy trace of the hybrid algorithm, this fact indicating that the action probabilities were in general closer to the extremes of 0 and 1 when compared with those of the S-model LRεP algorithm singly used. With an action having a probability close to unity, the other actions therefore are very rarely used, these having probabilities close to zero.

Therefore it seems that learning automata methods thrive when there is a deterministic aid to their action probability convergence. From these experiments it is concluded that it is highly beneficial to combine learning automata methods with standard control algorithms in order to produce hybrid algorithms, the expectation being that these will return superior performance than either method singly used.

8 Conclusions and further work

8.1 Conclusions

In this thesis, the practical use of stochastic learning automata for routing in multi-service networks has been examined. After having evaluated currently used routing algorithms and the best performing of the previously proposed learning automata methods, the learning automata based schemes were improved using various novel methods. The resultant routing performance was found to be superior to the proposed shortest-path dynamic routing algorithm, but still required a considerable number of iterations for convergence. By combining the two algorithm types to form a novel hybrid, superior performance ensued than either singly used, with no separate iterations required before convergence.

In chapter 1 multi-service networks were introduced, commencing with networking history that led to their requirement and so creation. Both ATM and IP networks with QoS features were highlighted as the main network technologies that support multi-service networking, as both can provide differing QoS for separate traffic flows. Methodologies and mechanisms for planning, provisioning and allocating network resources were identified, these being based on expected user demand generation together with an associated economic benefit model whose purpose may be to maximise network performance or minimise network cost, or strike a balance between the two. The importance of dynamic resource allocation policies (especially dynamism in the routing mechanism) was highlighted as twofold: coping with actual user demands not matching the expected user demands; as well as increasing resilience under network failure conditions. After noting that most presently used dynamic routing algorithms are of the link-state shortest-path variety, stochastic learning automata were identified as promising for application to the routing problem as their use has been previously shown to result with ε-optimal performance.

Chapter 2 contained a literature review of stochastic learning automata, the outlined theoretical aspects of their operation being required in later chapters when detailing the improvements to their operation. The learning mechanism of an automaton was highlighted as the reinforcement algorithm, there being a number of different algorithms resulting with differing performance characteristics from the automaton when operating in stationary random environments. The historical application of stochastic learning automata to the routing problem in both circuit-switched and best-effort packet-switched networks was also

detailed, with multi-service networks being thought of as similar to the multi-rate circuit-switched case.

Chapter 3 sought to examine the issues linked with proposing a link-state dynamic routing algorithm that would perform well, to use as a comparison to the performance derived from the learning automata based routing method. It began by showing the link between the CAC and routing functions in multi-service networks, and went on to describe a method for calculating effective bandwidth requirements for applications requiring QoS from the network, such as voice and video. These were based on a multiple on-off source model, and a novel application of the method to the MPEG traces was found to produce significantly more accurate bandwidth requirements than methods seen in the surveyed literature. Having looked at the CAC and associated functions, previously proposed algorithms for routing in multi-service networks were outlined and a modified version was proposed (AAMH) as a representative 'good' link-state algorithm. The associated link-state information required for making the routing decisions was then examined, with the possible means for propagating information throughout the network in the literature being found to be both periodic and triggered update methods. As these simulations assumed no link failures, event triggered updates were chosen for propagating link-state information, events being caused when the link utilisation level crossed into another band (the link bandwidth having been divided into the following classes or bands: 0-50%, 50-80%, 80-90%, 90-95% and 95-100%). The effect of this discretisation of the utilisation level was found to result with a higher blocking probability than when using the full link-state information, but lower than when using no link-state information. Finally methods for reducing the signalling overhead incurred by propagating the link-state updates were examined. The literature surveyed highlighted both limited update distribution methods, and the use of locally available information. From these, a new method was proposed: that of using existing call set-up signalling together with local schemes. This method was compared experimentally with hop-count limited flooding and was found to result with performance close to that obtained when using full flooding. It was also found that a hop-count limited flood equal to the average route length returned a blocking probability equivalent to that when using full flooding. Whilst this observation is dependent on the network topology and user demands, it does indicate that by using these concepts for reducing the signalling overhead and configuring them according to the network topology and expected user demands, accurate network state information can be propagated throughout the network whilst having greatly reduced signalling and processing overheads.

Chapter 4 contained experiments whose purpose was to validate which of the currently used learning automata reinforcement algorithms result with superior performance when interacting with a non-autonomous non-stationary environment such as routing in multi-service networks. The requirement for this work existed due to previous studies analysing

learning automata performance with stationary and switching environments, and assuming the conclusions drawn as valid for non-autonomous environments. The chapter presented a framework for rigorously assessing the performance of currently used reinforcement algorithms, this framework being partly analytical and partly experimentally based. By using a simple network scenario, converged action probabilities were analytically derived using Erlang's formula. The number of iterations required for the various reinforcement algorithms to update the action probabilities to their converged state were then experimentally derived, together with their subsequent variation. Of the basic continuous schemes, both the LRP and LRεP reinforcement algorithms were shown to converge faster than LRI. Also it was noted that using LRI with a high learning rate causes the action probabilities to fail to converge, indicating that unlike previous literature its general usage is not recommended. Of the class of estimator reinforcement algorithms, both the pursuit algorithm and the general estimator with linear updating function failed to converge. However using the GE with the x^3 updating function caused slow convergence but with very low action probability variance thereafter. Discretisation of continuous schemes had been shown in the literature to produce faster convergence when interacting with stationary random environments, and so such schemes have been previously applied to non-autonomous environment applications such as routing in networks. However the work contained in this chapter showed that discretisation generally caused failure of convergence, as evidenced by the following schemes: DLRI, DLRεP, DPursuit and DGE. Discretising the LRP scheme caused the action probabilities to converge quicker than with the continuous LRP scheme, but with increased subsequent variance. However by equalising the action probabilities rather than the action probability rates (as does the continuous LRP scheme) a lower blocking probability ensues. Therefore DLRP was preferred over LRP, with LRεP and GE with x^3 updating function also being recommended for general usage in non-autonomous application environments such as routing in multi-service networks. However as such an application's environment state continually changes due to changing user demands, so only the DLRP and LRεP reinforcement algorithms were recommended for general use (as GE evidenced a very slow convergence relative to the others), the DLRP algorithm evidencing fast convergence and the LRεP algorithm good steady-state performance.

Having validated the best performing reinforcement algorithms for use with learning automata applied to routing in multi-service networks, chapter 5 sought to improve learning automata performance from that obtained using either DLRP or LRεP reinforcement algorithms. Two mechanisms were proposed in this chapter: automatically adapting the reinforcement algorithm learning rates, and automatic reinforcement algorithm selection. Both novel methods utilised the fact that improving convergence speed degraded steady-state

performance, whilst increasing steady-state accuracy slowed convergence of the action probabilities. Each method sought to improve the learning automata performance by seeking to alter the convergence speed or steady-state accuracy according to the environment state. In order to determine the current environment state, a novel local mechanism was proposed based on the entropy of the action probabilities for the learning automata based at a node. This method was found to be more accurate than using other criteria such as change in blocking probability or average path length, etc.. The first novel improvement sought to alter the reinforcement learning rates, the rate being set to high when requiring fast convergence speed, and low when trying to increase steady-state accuracy. This method was successfully applied to both the DLRP and LRεP reinforcement algorithms using different entropy change thresholds. Although using this method with the DLRP algorithm produced a noticeable improvement in performance (especially under lower network loading conditions), the LRεP algorithm performance was relatively unaffected (due to fixed learning rate parameter changes having little effect). However as the LRεP algorithm still evidenced superior steady-state performance, the second novel improvement was proposed: using the DLRP algorithm under convergence conditions, and the LRεP algorithm under steady-state conditions. This novel scheme was found to produce superior results to previous learning automata schemes in terms of both convergence speed and subsequent blocking probability. This improved learning automata scheme was compared to the dynamic shortest-path based algorithm proposed in chapter 3 (called AAMH) using the more realistic networking scenario of trend user demands. These comparisons showed that the dynamic shortest-path based scheme consistently outperformed the learning automata based scheme under the topology and traffic loading characteristics of the network scenario used for the evaluation.

The work in chapter 6 therefore sought to improve the learning automata performance still further, by proposing a novel re-application of learning automata to the routing problem in multi-service networks. With the work up to date having used the P-model response environment, the feedback received could not distinguish between highly and lightly loaded routes. By using the S-model response environment and changing the feedback to represent link utilisation levels, it was believed that this novel paradigm would produce better results as it was more closely linked with the actual problem the learning automata were being applied to. After having normalised the instantaneous utilisation level to the region [0, 1] and used the exponential smoothing technique to smooth out short-term fluctuations, the reinforcement algorithm updated the action probabilities. The LRεP with a high fixed learning rate of 17% was used as the S-model response environment generally caused action probability updates of much lower learning rates (especially when links were close to saturation). The DLRP algorithm was not used as the S-model response environment precluded the use of discretised

schemes. Under lower loading levels this improved learning automata based routing scheme was found to outperform both the AAMH and previously proposed learning automata methods. Using the trend user demands showed the S-model LRεP and AAMH results to be fairly comparable, both returning a comfortably lower blocking probability than the P-model automatic reinforcement algorithm selection scheme. The superior performance of the S-model LRεP algorithm under relatively low loads was shown to be due to its ability to evaluate a larger set of possible routes. Whilst the network scenarios evaluated used symmetrical network loading, non-determinism of user demands in actual networking situations would generally cause greater non-symmetry in loading. It is therefore expected that this algorithm would improve its relative performance compared with the others still further under real network scenarios. However these results still show the main weakness of learning automata methods: that of the requirement for a large number of iterations before convergence occurs.

The work contained in chapter 7 sought to address this fundamental weakness by combining both the AAMH and S-model LRεP learning automata algorithms into a single hybrid routing algorithm. The strength of the AAMH algorithm was that no a-priori convergence period was required, the algorithm immediately choosing routes deterministically whatever the dynamics of the network state. The strength of the learning automata algorithm was that once the action probabilities had converged, a superior blocking probability ensued due to the evaluation of a greater number of paths. By using the AAMH algorithm under moving network state conditions (whilst still updating the learning automata action probabilities) and the S-model LRεP algorithm under steady-state conditions, the separate strengths were combined. Using either statistically constant or trend user demands, the hybrid algorithm consistently outperformed either AAHM or the S-model LRεP algorithm used alone, always returning a lower or comparable blocking probability. It was seen that the main reason for this improvement is that the learning automata action probabilities were strongly guided by the deterministic AAMH algorithm, causing other longer paths to be used less frequently than previously.

Therefore it seems that stochastic learning automata methods thrive when there is a deterministic aid to their action probability convergence. From these experiments it is concluded that it is highly beneficial to combine stochastic learning automata methods with standard control algorithms in order to produce hybrid algorithms, the expectation being that these will return superior performance than either method singly used.

8.2 Further work

The work contained in the previous chapters has sought to improve the routing performance within a connection-oriented reservation-based network environment such as ATM or IP with QoS and RSVP. The characterisation and optimisation method called stochastic learning automata has been applied to the routing function, and its performance has been improved. The resulting routing performance has been seen to be superior to more traditional dynamic shortest-path link-state based routing algorithms. There is a need to apply the same novel learning automata concepts to routing within a connectionless network environment, such as an IP network with DiffServ [12] implementation. Such network technology is becoming more and more important in high speed multi-service networking environments, with companies such as Cisco having gigabit-switch routers at the top of their product lines [69]. With these fast routers concentrating on switching and packet forwarding, it is not possible for them to also sustain many RSVP requests making and releasing network resource reservations for calls requiring QoS from the network. Therefore the DiffServ IP network design paradigm is becoming increasingly important as it has superior scaling properties to the IntServ [14] paradigm. In order to use the hybrid algorithm in IP networks with DiffServ, modifications to the algorithm are required in the following areas: calculating link utilisation levels on a time averaged basis, modifying the AAMH part of the algorithm to utilise multiple routes so as to avoid oscillations, and using a limited flooding mechanism for updating link states across the network.

8.2.1 Average utilisation calculation as information input

In a connection-oriented reservation-based network, an accepted connection consumes bandwidth of the links along the path according to its expected traffic demand requirement, this being termed the effective bandwidth for the connection. Thus the utilisation level of links reported may not be the actual utilisation level at that time, but is the level of bandwidth currently reserved on the link, this remaining constant for the duration of the connections traversing the link at that time. Variations in link utilisation therefore occur relatively slowly, being bound by variations at the connection or call level. Connectionless networking environments on the other hand have rapidly varying instantaneous utilisation levels, the value being representative of the link utilisation due to actual packet transmission at that instant in time (or more correctly the average packet transmission over a short time window).

Making routing decisions on a requested connection basis is applicable since connection or call duration is generally much longer than individual packet transmission, effectively encompassing an average of traffic demand requirement a-priori. In connectionless environments however, an averaging of the constantly varying utilisation level is required, in order to make both reasonable routing decisions and so that routes are not recalculated unnecessarily often. An average of the current link utilisation levels can therefore form the basis of information input for both the utilisation based learning automata and the AAMH routing algorithms.

For this to occur there is the requirement to obtain a reasonable averaging method. A dynamic mechanism in the IP world which requires the averaging of instantaneous information for it to perform reasonable operations is the Random Early Detection mechanism. The Cisco implementation (Weighted RED or WRED) [70] uses an exponentially weighted moving average of the class buffer level in order for it to probabilistically discard packets. The packets in these class buffers are served by a scheduling algorithm into the link meaning that, amongst other things, the class buffers are indirectly acting as playout buffers for that link. This indicates that the variation in the buffers will in general be much higher than that of the link utilisation, so that the same averaging method configuration would not be applicable for use in the link utilisation, as it will react too slowly to trend changes in the utilisation level.

Perhaps a more promising route is to examine currently used routing algorithms. Dynamic link-state routing algorithms generally use a hold-down timer [58], meaning that the algorithm is not allowed to recalculate routes more quickly than the pre-set timer expiry. This function bounds the maximum frequency of route set recalculation. Current IP routing algorithms such as Routing Information Protocol (RIP) [73] and Open Shortest Path First (OSPF) [61] are dynamic in the sense that route recalculation occurs on physical network failures such as links, interfaces or nodes. The hold-down timer for RIP is effectively 180 seconds, for whilst refresh packets are sent every 30 seconds, routes are invalidated by the 'timeout' timer. OSPF has a similar hold-down timer of either 40 seconds for broadcast networks or 120 seconds for non-broadcast networks [71], due to the 'hello' packets being send to neighbouring nodes once every 10 or 30 seconds, the 'dead' timer being set to four times the 'hello' timer interval.

Whilst these algorithms will in general be rarely invoked (network failures occurring relatively infrequently) they still have hold-down timers to ensure that situations where weak electrical connections that cause equipment to repeatedly go down and come back up again do not cause repeated routing information flooding and recalculation of the routes. So while these type of events should occur very infrequently, when they do occur there may be a significant number of similar events in a short space of time. When considering the

instantaneous utilisation level, it is found that this will generally show highly varying values within a time period of tens of seconds. The frequency of variation being therefore higher than under equipment failure conditions, it is reasonable to use a hold-down timer with smaller duration for averaging instantaneous utilisation levels. It is proposed that the experiments to be undertaken in future work are to use an averaging period of 30 seconds before updating the link state database at the node.

8.2.2 Flow-splitting modifications

Connection-oriented network environments calculate a route between a source and destination pair, keeping that route in place for the duration of the data transmission. Connectionless-oriented networks, on the other hand, may have multiple routes present in the routing tables for a source-destination pair (such as when the load balancing option in OSPF is used) the set of data packets being carried over various routes in order to reach the destination. The data packets for the connection-oriented network therefore arrive in order, whilst those of the connectionless network may occasionally arrive out of order, the re-ordering occurring both at the network (IP) layer and also at the transport layer, for which Transmission Control Protocol (TCP) [74] is normally used. The flow-splitting ability of connectionless networks is beneficial in order to balance network loading. This feature is approximated in connection-oriented networks by the possibility of multiple concurrent connections with the same source-destination pair being routed over different paths, as a separate route calculation may have occurred for each connection request. This is a close approximation to the connectionless case, as IP routers generally route packets of the same TCP connection over the same route in order not to overload the layer 3 and 4 re-assembly engines by many out of order packets or segments arriving.

In order to modify the AAMH part of the hybrid algorithm to exhibit flow-splitting properties when operating in a connectionless networking environment, some significant changes are required. While previously the algorithm calculated a single path for a source-destination pair, now multiple active routes are required. This is to avoid oscillatory network loading behaviour occurring when single path routes are changed throughout the network and then possibly back again as the network saturation 'hot-spots' move locations due to the route changes. The AAMH algorithm is modified in order to produce multiple routes by allowing all the possible shortest path routes, each with a certain weighting or probability of usage according to the minimum amount of unused bandwidth available on the route. The weighting may be calculated as follows:

$$\frac{\text{min link free bandwidth along route}}{\sum_{\substack{\text{all shortest}\\ \text{path routes}}} \text{min link free bandwidth along route}}$$

As AAMH aggregates link utilisations into class bands, so route recalculation should generally occur less frequently than link state recalculation. One point to note however is that the average link utilisation will generally never reach 100%, so that AAMH in its present form never uses alternate routes. Therefore the further modification is required so that alternate routes are permissible when all shortest path routes have the utilisation level in the highest aggregate class. So in such a case, any alternate routes with minimum bandwidth along the route being greater than the minimum are also included in the set of permissible routes, the associated weightings being:

$$\frac{\text{min link free bandwidth along route}}{\sum_{\substack{\text{all shortest and some}\\ \text{alternate path routes}}} \text{min link free bandwidth along route}}$$

If any routes share the next hop, then their respective weightings are added together to give the true weighting for choosing that link at this node.

When using the learning automata part of the algorithm, route calculation is not deterministic but stochastic, according to the action probabilities. These action probabilities automatically perform a flow-splitting function over time, and so require no modification of operation (with the exception of requiring a mechanism for ensuring that all packets belonging to a TCP connection are routed over the same path).

For both the algorithms however, there is a change to how the paths are set up. In the connection oriented-case, the path is directly set up from the source node to the destination node by the connection set-up signalling. In the connectionless networking technology case, the paths are indirectly set up via the flow splitting percentages present at each node. A packet will therefore utilise a particular path from a source to a destination node with a certain probability; this being calculated as the sum of the probabilities of choosing each of the links in turn which comprise the end to end route.

8.2.3 Link state update propagation

Chapter 3 detailed experimental work linked with reducing the extra signalling required to propagate link-state updates so that all the nodes' link state databases are representative of the network. The recommended method was to use the existing connection set-up signalling,

piggy-backing the link state information of that with the least remaining unused bandwidth along the route. Both the learning automata and AAMH components of the hybrid algorithm have used this method of obtaining the current network state. However with connectionless networks not requiring signalling for connection set-up, another link-state information propagation method is required. The comparative results of other possible methods highlighted in chapter 3 indicated that using a hop count limited flood, limited to the average path length, resulted with almost the same blocking probability as a full flood. Whilst this experiment was performed using symmetrical network loading, the result has general applicability according to the source-destination user demands and dimensioned links present in the network. Therefore this method seems applicable to explore for link state information propagation in the IP networking experiments for future work.

Appendix A: IP technology and protocols, and IP network planning and design

A.1 Overview of current IP technology and protocols

This section seeks to give a brief overview of the historical and current use of IP technology and associated protocols. First the main IP stack mechanisms and protocols are outlined. Next more recent scheduling mechanisms for providing quality of service differentiation of flows are outlined. Finally end-to-end IP related mechanisms are briefly discussed.

A.1.1 Basic IP networking with reference to the OSI layer model

The Internetworking Protocol (IP) [4] is a network layer protocol for packet transmission from source to destination nodes. Being a layer 3 protocol in the OSI model, it performs the routing and congestion control functions [5] with the aid of associated protocols. Its original conception arose out of a need to connect differing network technologies, with end-to-end operation occurring in a seamless way. Higher layer protocol payloads are therefore segmented and encapsulated within IP packets, and presented to the layer 3/2 technology running at that point in the network for further encapsulation before transmission. At the next IP node, the IP packet would be re-assembled, for the convergence sublayer to which it was presented may have segmented it. Further routing decisions are performed before it is again presented to that node's convergence sublayer, which may use a different networking and transmission technology. Therefore seamless operation occurs at the IP layer and higher, whatever the heterogeneous mix of networking and transmission technologies used below the IP layer.

IP networks generally refer to networking technologies which utilise the whole suite of protocols associated with the Internet Protocol, these protocols going up to the application layer (layer 7 in the OSI model), and down to the data-link layer (layer 2 in the OSI model). The layer 2 protocol generally used is the Point to Point Protocol (PPP) [72] which provides a standard method for transporting multi-protocol datagrams over point-to-point links.

However IP network technologies are generally associated with protocols down to layer 3, such hardware or equipment in the network being termed a router.

Two routing protocols are mainly used in the IP world: Routing Information Protocol (RIP) [73] and the Open Shortest Path First (OSPF) [61] protocols. RIP is a distance vector routing algorithm, and therefore evidences slow convergence properties of routing tables. Faster convergence occurs with OSPF as this is a link-state routing algorithm, so this algorithm is superseding the use of the RIP algorithm. With the routing protocol generating the routing tables at each node, the IP packets are then transmitted from node to node, the hop-by-hop route chosen based on the destination address for that packet held in the IP header.

The layer 3 congestion control function has historically been rather limited in IP routers, in general a single queue being used to buffer packets waiting for transmission on a certain link. Congestion control in IP networks has therefore historically relied on other layers' functionality. For example, the layer 2 protocol ensures reliable link transmission, whilst the layer 4 protocols such as Transmission Control Protocol (TCP) [74] ensure end-to-end reliable transmission and rate limiting under congestion conditions. Applications requiring reliable transport therefore use the TCP transport protocol which guarantees reliable end-to-end transmission of application payload packets. It operates by segmenting application packets into TCP segments, which are in turn delivered to the IP layer for further segmentation and encapsulation as required. As an IP network is connectionless in operation, so IP packets can arrive at the destination out of order. TCP therefore re-orders the arrived segments as necessary, passing their payload up to the application layer. Rate limiting under congestion conditions automatically occurs by the source node TCP requiring acknowledgements of TCP segments sent previously, these acknowledgements coming from the destination node's TCP engine.

Other application types place speed of packet delivery as a priority above reliable transmission of the data. For example real-time applications such as voice or video are not overly affected by occasional data loss, but require as small an end-to-end delay as possible. Such applications use the Real-time Transport Protocol (RTP) [75] which is further encapsulated into User Datagram Protocol (UDP) [76] segments, both these protocols being at the layer 4 or transport layer. By not guaranteeing reliable segment transmission, UDP does not re-transmit dropped packets or rate-limit the RTP source, so that application packets arrive at the destination application host with as little delay as possible.

The transport layer of the IP protocol stack generally interfaces directly with the application layer or layer 7 of the OSI model. This layer is for protocols which applications may use for their data transmissions. For example, e-mail applications generally use either the Post Office Protocol version 3 (POP3) [77] or Internet Message Access Protocol version 4

(IMAP4) [78], and the Simple Mail transfer Protocol (SMTP) [79]. POP3 is used by mail clients to download stored e-mails from their mailbox held on a mail server. IMAP4 has additional functionality to POP3 in that remote message folders may be manipulated in a functionally equivalent way to local mailboxes. SMTP is used to transmit messages between mail servers, in order for them to reach the message's destination server where there resides the recipient's mailbox. Another example application is a web browser, this using the Hypertext Transfer Protocol (HTTP) [80]. HTTP allows the web browser to obtain web pages from remote internet servers, automatically reconstructing a page from its various heterogeneous component objects.

Some of the generally used layers 3, 4 and 7 protocols have been outlined, these forming the IP protocol stack. Whilst it is feasible to interchange other lower and higher layer protocols, this being a main benefit of a layered network architecture, the whole set is generally used in its entirety, at least down to the IP layer. The main reason for this is due to IP penetration at network end-points, this being highlighted in the following section.

A.1.2 Quality of Service (QoS) in IP networks

Historically, two significant benefits of ATM network have been their speed and guaranteed QoS for each accepted connection request. IP routers have historically been of lower speed, requiring higher processing than an ATM switch at each node, as packet routing decisions occur from hop to hop in the IP case, whilst the ATM packets traverse a single pre-defined and set-up route in the connection-oriented ATM case. The inherent reduced routing complexity of ATM during packet transmission meant that switching and transmission speed could be increased, as less transit processing is required. The higher switch throughput, combined with an initially higher equipment cost, meant that ATM technology was historically confined to the WAN, whilst the lower costing and slower IP technology remained in the LAN arena. So because ATM was rarely found at the desktop, convergence of the IP packets to the ATM layers occurred at the equipment serving the WAN interface. As regarding QoS, IP routers have historically offered only 'best-effort' service capabilities, with no guarantees for traffic flows being possible.

However more recently various mechanisms have been introduced at the IP layer so that QoS differentiation of traffic flows becomes possible. The mechanisms generally use the precedence field of the IP packet header in order to categorise an IP packet into a certain class or flow. At present most routers operate with IP version 4, whose precedence field is 3 bits long, meaning that up to eight different classes of flows are available (with one being reserved

for network operational use). However the more recently proposed IP version 6 [6] has a larger flow label field of 24 bits, so that more than 16 million differing classes of service or application types can be defined. Once the packet's class has been established, it can be scheduled for link transmission according to the scheduling mechanism's configuration and functionality.

A.1.2.1 Basic mechanisms

Historically IP routers have performed First In, First Out (FIFO) queueing [81], with a single buffer storing packets under link congestion conditions and forwarding them on in order of arrival. Whilst this method meant that the packets from bursty traffic sources would not necessarily be dropped, it also meant that such sources also caused high delays in other time-sensitive traffic flows. Various basic queueing and scheduling schemes have been implemented in IP routers, seeking to overcome this shortcoming.

One of the first mechanisms implemented by Cisco to improve the FIFO queueing situation was Priority Queueing (PQ) [7]. This mechanism can allocate up to four priority queues, these being high, medium, normal and low priorities. The buffers are then scheduled in turn, the algorithm moving to the next lower priority buffer when there are no more packets waiting to be scheduled at the higher priority buffer. This means that one type of traffic (such as applications crucial to the business functions, termed mission-critical applications) is ensured transmission, possibly at the expense of all others.

Custom Queueing (CQ) (otherwise known as Weighted Round Robin or WRR) was implemented to avoid this unfair situation. This method guarantees some level of service to all traffic because bandwidth can be allocated to each class of service. Up to sixteen queues can be thus configured for scheduling, the mechanism ensuring that a class obtains the configured bandwidth, even under congestion conditions.

By differentiating between types of traffic flow via the precedence bits and scheduling differing class flows in a different manner, so QoS is achieved on the IP network. The ATM Constant Bit Rate (CBR) and Variable Bit Rate (VBR) services provide hard QoS, guaranteeing service by reserving network resources specifically for the connection along its path. This function can be emulated by using the CQ feature, as bandwidth is reserved for specific flows or classes, with no other class traffic able to utilise this reserved bandwidth.

PQ on the other hand is an example of soft QoS, which is where some traffic is treated better than the rest, there being statistical preference rather than a hard guarantee.

ATM technology variants of this type of service are Unspecified Bit Rate (UBR) and Available Bit Rate (ABR) services.

A.1.2.2 More complex scheduling

The weakness of PQ is that no guarantees of QoS are possible under congestion conditions, whilst that of CQ is that the bandwidth for a class remains unused and so is wasted under congestion conditions when there are no packets of that class requiring transmission. Weighted Fair Queueing (WFQ) [8] seeks to combine the two algorithms' strengths whilst avoiding each one's weakness. Individual class or flow buffers are configured on each router, these being emptied by the WFQ scheduling mechanism. It then empties packets from each buffer according to the configured scheduling weight for that buffer. However if there are not enough packets held in a buffer to 'use up' its configured scheduling weight, that excess is distributed to the remaining buffers with unscheduled packets still present. It therefore provides both hard QoS with bandwidth guarantees in congestion conditions, but also the most efficient use of bandwidth, redistributing any unused to lower class buffers. This is the scheduling mechanism generally used in most IP networking situations today.

A.1.2.3 End-to-end congestion control

Whilst the above scheduling mechanisms operate at the local node level, there is also end-to-end congestion control occurring at the transport layer by the TCP engines. In cases of network congestion, the TCP engines throttle back their data transmissions, so that the congestion is eased downstream. When there is chronic network congestion and packets are dropped, the TCP source halts transmission and after a timeout period retransmits the information, increasing its transmission rate exponentially. The effect of this slow-start feature can be to cause oscillatory behaviour in the network, network congestion being followed by underutilisation of links, being followed by congestion as the TCP engines increase their transmission rates again.

The Random Early Detection (RED) [9] mechanism was designed to avoid this recurring network congestion by seeking to limit TCP transmissions before the onset of chronic congestion. It operates on the class queues or buffers that have been set up for the QoS scheduling mechanism. A moving average of the instantaneous buffer level is

calculated, and on its crossing a configured threshold a probabilistic discard of differing source TCP engines' IP packets occurs, the probability of discard increasing as the average buffer level goes up.

The effect of occasionally dropping packets is to cause those TCP sources to stop transmissions and re-transmit the data after their timeouts occur. Therefore the aggregated flow though the buffer is reduced but not halted, as the flow normally consists of many TCP sources. This reduction normally means that hard discard (which is where the class buffer is filled and all extra packets are dropped) is avoided so that a few rather than many TCP engines retransmit. The effect of this function is for most users on the network to perceive better network dependent application response as their TCP transmissions remain largely unaffected or at least affected less often. From a network resource perspective, such occurs due to the link bandwidth being more efficiently used, with full or close to full utilisation occurring, rather than oscillations between full and partial utilisation as multiple TCP re-transmission timers expire.

A.1.2.4 End-to-end QoS

IP QoS mechanisms guarantee QoS for an aggregate class flow through a node but not for a specific flow through the network, as does ATM technology. Therefore IP QoS technology provides soft QoS, for if other traffic flows of the same class misbehave a flow's QoS can be degraded.

Hard QoS can be approximated however, with the use of edge policing. For example Cisco's Committed Access Rate (CAR) mechanism [82] can be configured to measure incoming traffic flows and either drop packets exceeding the configured flow 'contract' or downgrade the excess traffic's class type. The mechanism is intelligent enough to allow bursts of the flow's traffic through after periods of flow underutilisation using a token bucket mechanism, but unlike standard leaky bucket implementations no traffic shaping is performed.

Using this or other edge policing mechanisms, end-to-end QoS can be guaranteed for a flow. The mechanism ensures the specified upper bound for a traffic flow is not exceeded, and the scheduling mechanisms present at local points in the network are configured based on the aggregate flows' contracts, which traverse each local point. Therefore having ensured all flows' access to the network are limited to their contracts and the network can support all the contracts' requirements, the end-to-end QoS requirements for each flow is guaranteed (the flow's QoS requirements being an integral part of the flow's contract specification).

A.2 IP network planning and design

With IP networks historically providing just best-effort service, the planning and design function centred purely with link sizing and routes, together with equipment throughput dimensioning. With the more recent QoS features and mechanisms becoming available however, the planning function has become more complex, with provisioning of logical links now being possible.

A.2.1 Network design for best-effort IP networks

The design of both greenfield site networks and modifications of existing networks is based on some sort of user demand modelling. This procedure seeks to estimate the amount of source-destination traffic demand that users might place on the network once it is fully commissioned and operational.

Using the example of a corporate network, the simplest form of user demand modelling is to place the application servers on the network, and then link this or these nodes with the others using transmission pipes thought sufficiently large for the user set accessing these server applications. The linking may occur on a point-to-point basis or via a meshed network topology, according to the number of sites to be linked together, and the cost-performance balance to be achieved. Routes are then assigned in the network, according to the expected application traffic flows on the network. For resiliancy a dynamic routing algorithm may be used, causing the pre-assigned routes to be changed in network failure conditions.

A.2.2 Network design for IP networks providing QoS

The requirements for IP networks providing QoS become essential when the network is used to carry certain types of traffic. For example, applications such as voice over IP (VoIP) and videoconferencing require strict end-to-end delay guarantees from the network in order to operate properly. Therefore if congestion occurs at nodes which the traffic flow traverses, policies must operate locally at the congestion point to allow the traffic from the critical flows to pass whilst buffering that of less important flows. The operation of such local policies ensures that the network can consistently meet a flow's end-to-end delay requirement.

The presence of such mechanisms is also important to 'mission-critical' traffic flows: applications which may not have strict QoS requirements, but whose traffic is important in value to the customer or company. Under chronic congestion situations such traffic is passed through whilst other less important traffic is discarded, meaning that data loss for applications critical to the company's operation is minimised or eliminated altogether.

A.2.3 The place for bandwidth-based dynamic routing algorithms in IP networks

As the level of connectivity in a network increases, so the number of alternate paths from a source to a destination node increases. It is under such circumstances that the possibility of choosing a different path for the same source-destination pair, due to its lower loading, becomes preferable to always choosing the same route whatever the state of the network loading.

The requirement for QoS in IP networks arises from the use of applications important to the business function (termed 'mission-critical' applications), or applications requiring end-to-end delay guarantees from the network in order to operate correctly. The first kind generally do not add connectivity to the network, as they may be used by a few or many people in the company, but are generally server-based and so centralised. However the second kind of application's destination bindings are normally distributed throughout the network, such applications normally being audio and / or video based, connecting a pair of users rather than a user client to a server. It is with the use of such applications therefore that connectivity in the network might expect to increase. As the requirement for such applications increase in corporate networks, so utilisation-based dynamic routing schemes will increase in importance.

These discussions have centred on corporate networks. Carrier networks are the networks which telecommunications providers use to carry the traffic of many corporate networks. Individual corporate networks are designed, with the network's links being overlaid over the carrier network's links and nodes. The carrier network therefore requires a large number of nodes so that all the areas have a relatively close point of presence, coupled with a good connectivity to ensure network resilience. Therefore carrier IP networks, whether best-effort or QoS based, in general would benefit from the use of utilisation-based dynamic route calculation.

Appendix B: A tool and models for simulation analysis

B.1 Introduction

The main aim of this appendix is to describe the important issues relating to the simulation models used to test the various routing algorithms.

The appendix begins by explaining why simulation modelling was used for this particular problem rather than analytical techniques. There follows an overview of the evaluation of commercial simulation packages which was undertaken at the start of the research. This includes references to the full reports which documented the evaluation. Having chosen OPNET as the modelling and simulation environment, different simulation method possibilities are examined and the one giving the required level of detail for the problem under investigation is selected.

When examining the models included with the OPNET environment, it was found that due to certain limitations these were unusable for the research experiments. These limitations are highlighted as are the high-level details of the new model library which was constructed to permit the research experiments. Finally, rather than using the analysis and displaying functions within the OPNET environment itself, reasons are given for the benefit of exporting the raw results data to Excel for analysis and display.

B.2 Modelling technique selection

There are two techniques available for evaluating the blocking probability of routing algorithms: analytical and simulation modelling. Analytical modelling approaches have been favoured in the past for two main reasons. Firstly, the resulting closed-form equations can be used to easily produce the upper and lower bounds together with the mean for the network performance. Secondly, computing power used to be much lower, resulting with much more limited possibilities for simulation modelling.

A large body of analytical modelling material is thus available for circuit-switched networks [35]. This includes techniques for calculating the average blocking probability when operating a particular routing algorithm. Due to the similarities between routing and

call admission in multi-service networks and circuit-switched networks, it has been suggested that some of the techniques used for analysing circuit-switched networks might be used for ATM networks for example [45]. The weakness of analytical methods is, however, that they require simplifying assumptions to be made about the traffic, topology or routing algorithm. For example, fully connected, symmetrical topologies and traffic distributions are a-priori assumptions in order to make the analysis tractable. With circuit-switched networks the use of these assumptions is in the main part reasonable. However the same is not true for multi-service networks due to the more complex traffic, topology and routing algorithm possibilities available with such networks.

The analytical approach was therefore rejected from the outset, and discrete-event computer simulation [83] chosen instead. There next followed an extensive evaluation of the two leading commercial communications simulation packages in order to ascertain the preferable system for use in this research.

B.3 Simulation modelling tool selection

Having decided on using computer simulation techniques for multi-service network performance analysis, the two leading commercial communications simulation packages were evaluated for suitability in this research. These were Optimised Network Engineering Tools Modeller (OPNET) and Block Oriented Network Simulator Designer (BONeS).

BONeS came with two campus type network examples, having an FDDI ring and ATM backbone respectively [84, 85]. The original network scenario consisted of four token ring LANs together with three ethernet LANs connected to a FDDI backbone ring via three bridges. The total number of workstations individually modelled over the whole of the heterogeneous network is 23, each workstation generating up to 1.2 Mbps of traffic. The second network scenario is an evolution of the first, replacing the FDDI backbone ring with a four switch ATM network interconnected with SONET STS-1 links (these being 49.5 Mbps user or payload rate [86]). Furthermore the number of workstations on the LANs is doubled, with the two 4 Mbps token rings also being upgraded to 16 Mbps. Various results and simulation speed comparisons are included in the documentation, with conclusions on network performance and suitability being drawn for each scenario.

For an accurate comparison of environments, including standard libraries and simulation speeds, it was thought beneficial to model the same scenarios in OPNET. However, this was found to be difficult as BONeS lacked the transport layer models which the standard OPNET user demand models required. On the other hand, OPNET lacked a

bridge for the token ring protocol to FDDI, or a router for token ring to ATM. Rather than expend significant effort in modelling the same scenario using both environments in order to obtain accurate simulation speed comparisons, a compromise was reached by modelling FDDI rings instead of token ring LANs. By limiting the traffic to the equivalent token-ring network parameters and by modifying the traffic generators, an equivalent traffic load was simulated so that simulation speed comparisons could be undertaken.

The outcome of the evaluation was that OPNET was chosen as the simulation package with which to perform the experiments. Further details of the evaluation may be found in [87], with the simulation results for the campus network modelled using the OPNET environment being detailed in [88].

B.4 Network model

Having chosen OPNET as the computer modelling and simulation environment with which to perform network performance analysis experiments, a multi-service network model was required in order to simulate different network topologies, traffic demands and routing algorithms.

The following sections begin with detailing the various simulation methods available, and the reasons why session level event-driven simulation models were used in simulations. Rather than utilising the existing models that are included with OPNET new ones were produced, the reasons being shown. Finally the design and functionality of the new model library are outlined.

B.4.1 Simulation method

Ideally a simulation program should run at speeds comparable to the real network in order to enable results to be gathered rapidly [89]. Three levels of simulation detail are possible when constructing network models; the more detailed the level the longer the simulation time required.

The first is cell or packet level simulation, where the basic unit of traffic is the ATM cell or IP packet, so that all the signalling and data carrying cells/packets are simulated, each cell/packet arrival being a simulation event. A coarser level of detail is the burst which is defined as a group of cells or packets with constant arrival rate, an event being a change in this cell/packet rate, so causing far less traffic events than cell/packet level simulation. Finally there exists session level simulation which is where individual traffic events consist of

the setting up, modification and tearing-down of a session, with the data transmission part of the session not being simulated.

With our CAC model being based on pre-calculated effective bandwidths rather than measured statistics (see chapter 3), the data modelling is encapsulated at a coarse level by the effective bandwidth calculations themselves. Therefore for speed of simulation purposes, it was decided to use session level simulation models. In order to obtain simulation results on connection set-up times, a multiple packet-based signalling has been modelled rather than having just the single set-up packet indication.

Two differing methods of simulation are available, both discrete-event simulation and time-driven simulation. Discrete-event simulation is driven by separate events being enqueued in a time-ordered event list and the simulator kernel consuming and acting on these time-ordered events [83]. On the other hand, time-driven simulation progresses in discrete time steps, with the simulator kernel performing the event actions that are scheduled to occur at that particular time instant. Time-driven simulation provides a speed improvement when many events occur during each progressed time instant, otherwise event-driven simulation is faster and so generally preferable. As session level simulation causes significant periods of time with no events and then a number of events on the same time instances, it was decided to use discrete-event simulations. The OPNET simulation kernel operates in a discrete-event fashion, and so naturally complements this decision. OPNET could still have been used if performing time-driven simulations, but each network element would have had to generate its own time-driven events.

B.4.2 Model design

The following section does not deal with the intricacies of the network model design, but some of the high-level concepts only. These include showing the weaknesses in the standard model libraries that are included with OPNET, and the design and functionality of the newly developed library. Details of the network topologies used and the connection characteristics for the various traffic types are given in chapter 3 and so are not included below.

B.4.2.1 Weaknesses of the standard model libraries

The OPNET modelling environment comes complete with certain model libraries; for example the ATM models being compliant to the ATM Forum UNI signalling protocol [41]. These libraries was evaluated for suitability in this research, but was unfortunately found to

be unsuitable due to several significant limitations. These in the main revolved around the fact that the modelling methodology that had been used precluded large topology and concurrent call simulation experiments.

The main limitations in the ATM library range as follows: VPCs can only be one link in length; VCCs are statistically set up at the start of the simulation and cannot be reassigned, and source and sink models can handle just one VCC each. As the research direction was unclear at the beginning of the project, it was thought important to have full ATM functionality available. This was in case strands of the research focused on all three levels in the technology that affected routing decisions: the VPC topology management, the VPC bandwidth allocation, and the VCC route management.

However the main general limitation is due to an object-orientated modelling philosophy having been used in constructing the model libraries. This causes the instantiation of a new process model on each node traversed by a route when setting up a new call on the network. Whilst this method results in clear and easily maintainable models, the weakness is that as the network topology and number of concurrent calls grows, so the run-time simulation program increases in memory size. This precludes the possibility of simulating large network topologies and many concurrent calls as the run-time process size becomes too large for the available computer memory resources.

Due to these limitations, it was decided to construct a new model library. The OPNET environment facilitates the coding of new models by using C code within Finite State Machines, resulting in a pictorial representation of the process' functionality. Currently the constructed library of models amounts to over 13,000 lines of C code.

B.4.2.2 The new model library

The functionality of the new model library currently supports logical link set-up via a simple signalling protocol, and call set-up and tear-down via the ATM Forum UNI signalling protocol [41] and PNNI document [90]. This is similar to using RSVP with the IntServ paradigm [14]. This allows for accurate simulations which also indicate the connection set-up time for different algorithms. The standard can also be tested as to whether it provides for the requirements of various algorithms. As detailed within the body of the thesis, various algorithms to perform unicast call routing have also been implemented, with new algorithm implementation requiring relatively simple modifications as the basic node models are consistent with all the algorithms.

The modelling philosophy used for the standard model libraries has not been followed. Rather than having a separate process which stored the information pertaining to

one call traversing a node, with multiple calls requiring multiple process instantiations at that node, the new design is based on having a single process on each node with an associated list of data holding the information for all the calls traversing the node.

The ATM Forum specifies the setting up and tearing-down of VCCs be performed using defined signalling packets encapsulated within the payload of the standard ATM cell. The full recommended procedure has been followed in order to obtain accurate session set-up times for different routing algorithms. Figure 82 shows the process that is undertaken when setting up a VCC over three VPCs. The time axis is vertical, increasing as one travels down the figure. The various different signalling packets are indicated on the diagram, these being generated by the ATM layer. The application above the source node's AAL layer is allowed to commence transmission of data after the reception of the 'begin ack pdu'.

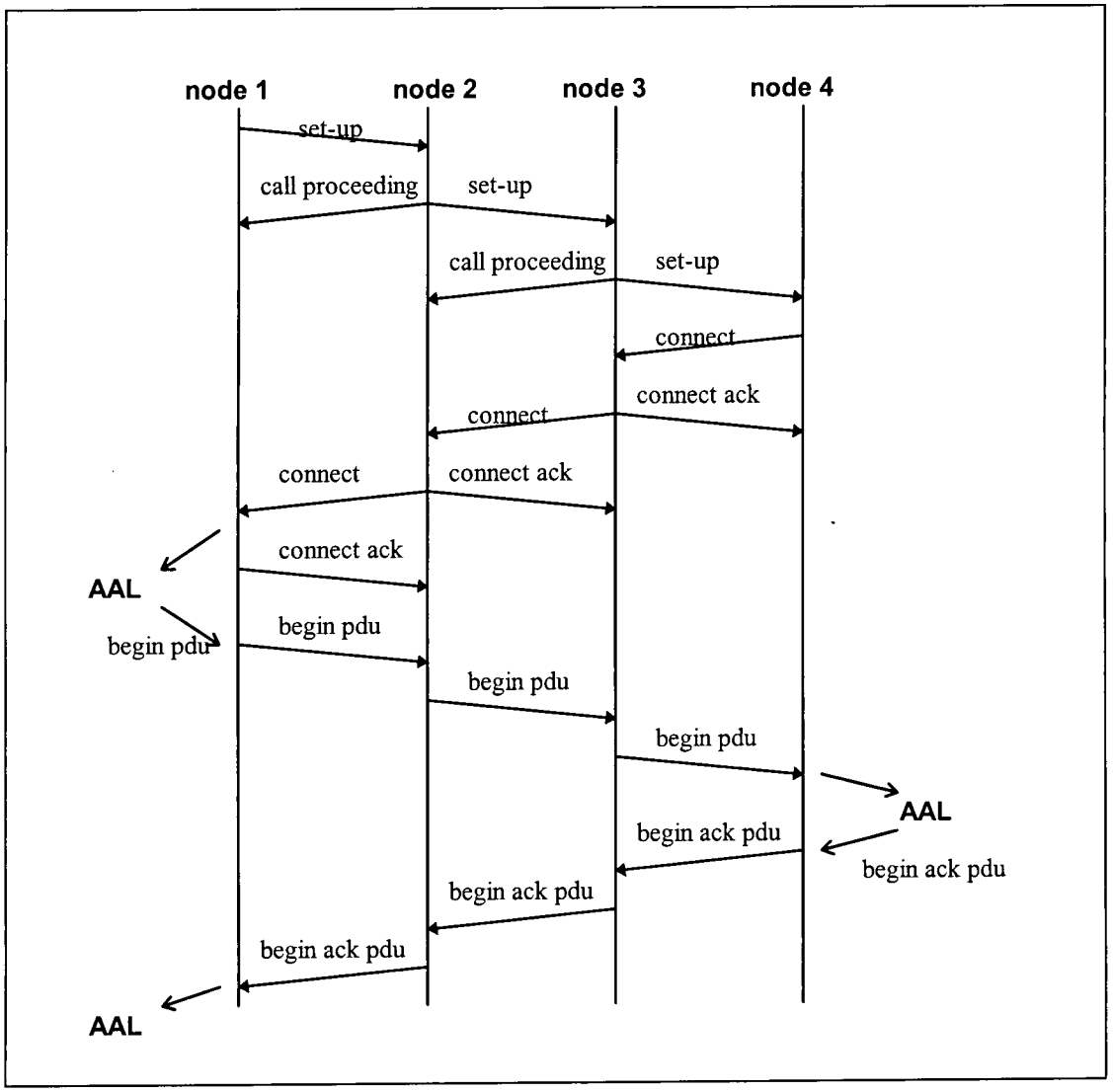


Figure 82: The signalling process required to set-up a VCC traversing 3 VPCs

The 'set-up', 'call proceeding', 'connect' and 'connect ack' packets are all generated by the ATM layer to set up the VCC, with the source AAL being informed of the VCC when the 'connect' packet arrives from the destination node ATM layer. The AAL layer then generates a 'begin pdu' which is encapsulated in an AAL packet which is in its turn encapsulated into an ATM cell and sent to the destination node AAL layer for AAL session instantiation. At the reception of the 'being ack pdu', the source node AAL informs the application above it that the connection is set-up and that transmission of data packets can now begin. In this example, the routing tables at four nodes will be updated, even though the route might traverse many more nodes due to each VPC being more than one physical link in length. It should be noted that if the VCC traverses only one VPC, then the transmission of 'call proceeding' signalling packets will not occur. The equivalent IP based RSVP is similar [10].

Figure 83 shows the signalling process required when tearing down a VCC. The source node application informs the AAL layer that it no longer requires the connection, and the AAL layer commences the tear-down of it by sending an 'end pdu' to the destination node AAL layer. On the reception of the returning 'end ack pdu', the AAL connection is closed, and the ATM layer can then tear-down the VCC. At the reception of the 'release complete' packet, each node clears its routing table of the VCC entry. The equivalent IP based RSVP is again similar [10].

B.5 Analysis of results

The OPNET environment includes the facility for specifying probes that gather results during simulation runs [91]. These results may then be displayed in an analysis window, there also being the possibility of analysis using the built-in analysis function blocks.

Due to perceived limitations in the graphing and analysis functions included with the package however, it was decided to output the pre-processed results to a file as simulation occurred. This raw results data file could then be imported into a standard data analysis package for post-processing and displaying of the analyses.

Microsoft Excel [92] was used as the post-processing package as the spreadsheet analysis functions are familiar and the resulting graphs ported to a document can still be linked with the analysis data.

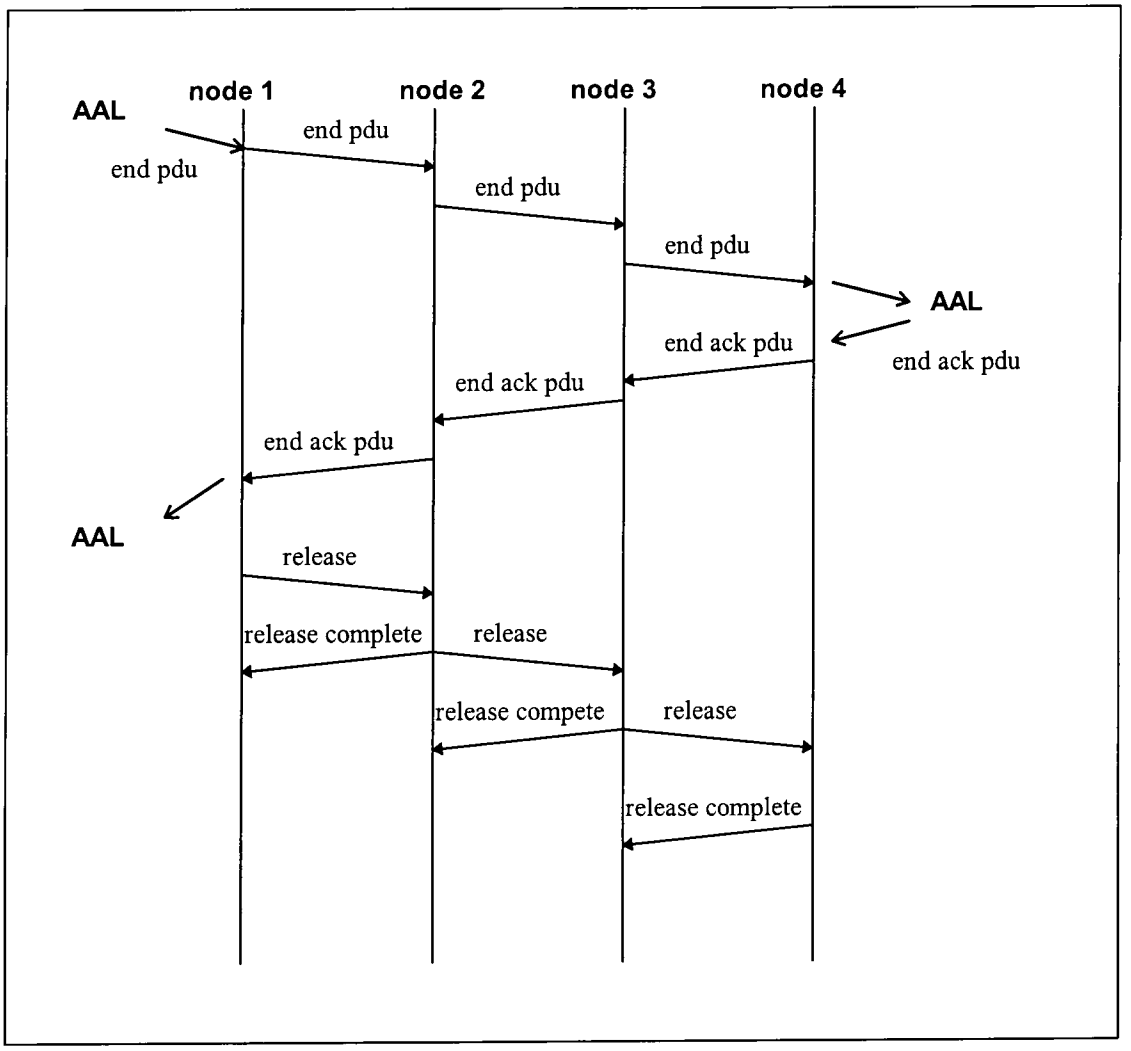


Figure 83: The signalling process required to tear-down a VCC which traverses 3 VPCs

B.6 Summary

The main aim of this appendix has been to describe the important issues relating to the simulation models used to test the various routing algorithms. After giving the reasons for choosing simulation modelling rather than analytical techniques, there follows an overview of an evaluation of commercial simulation packages which was undertaken at the start of the research. Having chosen OPNET as the modelling and simulation environment, different simulation method possibilities were examined and the one giving the required level of detail for the problem under investigation was selected.

Due to limitations in the standard models included with the OPNET environment, it was thought that these were unusable for the envisaged research experiments. These limitations were highlighted along with the high-level details of the new model library which was therefore constructed. Finally, rather than using the analysis and displaying functions within the OPNET environment itself, reasons were given for the benefit of exporting the raw results data to Excel for analysis and display.

Appendix C: Erlang loss formula calculations

In the Erlang Loss Formula, the probability that a call requesting use of a line is blocked is given by [18]:

$$b = \frac{\rho^l / l!}{\sum_{k=0}^{l} \rho^k / k!}$$

where $\rho = \lambda / \mu$, λ being the call arrival rate, $1/\mu$ the mean call time, and l the number of lines in the trunk group. For the four node network shown in Figure 18, the path blocking and penalty probabilities may be written as:

$$c_1 = \frac{(p_1\rho)^{l_2} / l_2!}{\sum_{k=0}^{l_2} (p_1\rho)^k / k!}$$

$$c_2 = \frac{(p_2\rho)^{l_3} / l_3!}{\sum_{k=0}^{l_3} (p_2\rho)^k / k!}$$

Appendix D: Related publications

1. Aranzulla P., Mellor J., Mars P., "Dynamic routing in ATM networks", 3rd. Communication Networks Symposium, July 1996, pp. 159-162.

2. Aranzulla P., "Using OPNET for Investigating Dynamic Routing in ATM Networks", OPNETWORK 1997.

3. Aranzulla P., Mellor J., "Comparing two routing algorithms requiring reduced signalling when applied to ATM networks", 14th. IEE UK Teletraffic Symposium, March 1997, pp. 6/1 – 6/7.

4. Aranzulla P., Reeve J., Mellor J., Mars P., "Improved stochastic learning automata for routing in ISDNs", Proceedings of NOC '97, Vol. 1, pp. 227-231, 1997.

5. Aranzulla P., Mellor J., "Implementing dynamic unicast VCC routing in ATM networks", Proceedings of NOC '97, Vol. 2, pp. 157-160.

6. Aranzulla P., Ritch M., "Improved learning algorithms for satisfying quality of service requirements: routing and nodal perspectives", 15th. IEE UK Teletraffic Symposium, 1998.

7. Nyong D., Aranzulla P., Cosmas J., Pitts J., "Resource based policies for design of interworking heterogeneous service networks", *Interoperable Communication Networks*, Vol. 1, No. 2-4, pp. 571-580, 1998.

8. Mellor J., Aranzulla P., "Using an S-model response environment with learning automata based routing schemes for IP networks", IFIP workshop on Performance Modelling and Evaluation of ATM Networks, 2000.

References

1. *B-ISDN ATM Layer Specification*, ITU-T Recommendation I.361.

2. *B-ISDN Asynchronous Transfer Mode Functional Characteristics*, ITU-T Recommendation I.150.

3. *B-ISDN General Network Aspects*, ITU-T Recommendation I.311.

4. *Internet Protocol*, rfc 791, 1981.

5. J. Walrand, *Communication Networks*, Aksen Associates, 1991.

6. *Internet Protocol, Version 6 (IPv6)*, rfc 1883, 1995.

7. *Congestion Management Overview*, Cisco product literature, http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/qcpart2/qcconman.htm

8. *Cisco –IOS Technologies – Quality – Weighted Fair Queueing (WFQ)*, Cisco product literature, http://www.cisco.com/warp/customer/732/Tech/wfq

9. *Cisco IOS Technologies – Quality – Random Early Detection (RED)*, Cisco product literature, http://www.cisco.com/warp/customer/732/Tech/red/

10. *Resource ReSerVation Protocol (RSVP) – version 1 Functional Specifications*, rfc 2205, 1997.

11. *Multiprotocol Label Switching Architecture*, Internet draft, 1998.

12. *An Architecture for Differentiated Services*, rfc 2475, 1998.

13. *A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)*, rfc 2430, 1998.

14. *Integrated Services in the Internet Architecture: An Overview*, rfc 1633, 1994.

15. Kelly F.P., "Network Routing", *Phil. Trans. R. Soc. Lond. A*, 1991, pp. 343-367.

16. Narendra K., Thathachar M.A.L., *Learning Automata – An Introduction*, Prentice-Hall, 1989.

17. Zgierski J.R., Oommen B.J., "SEATER: An Object-Oriented Simulation Environment Using Learning Automata for Telephone Traffic Routing", *IEEE Transactions on Systems, Man, and Cybernetics*, Feb. 1994, pp.349-356.

18. Chrystall M.S., *Adaptive Control of Communication Networks Using Learning Automata*, Ph.D. thesis, University of Aberdeen, 1982.

19. Reeve J.M., *Learning Algorithms for the Control of Routing in Integrated Service Communication Networks*, Ph.D. thesis, University of Durham, 1998.

20. Narendra K., Thathachar M.A.L., "Learning automata - a survey", *IEEE Transactions on Systems, Man, and Cybernetics*, July 1974, Vol. SMC-4, No. 4, pp. 323-334.

21. Lakshmivarahan S., Thathachar M.A.L., "Absolutely expedient learning algorithms for stochastic automata", *IEEE Transactions on Systems, Man, and Cybernetics*, May 1973, pp. 281-286.

22. Mars P., Chen J.R., Nambiar R., *Learning Algorithms - Theory and Applications in Signal Processing, Control and Communications*, CRC Press, 1996.

23. Economides A.A., "Multiple Response Learning Automata", *IEEE Transactions on Systems, Man, and Cybernetics*, Feb. 1996, pp. 153-156.

24. Thathachar M.A.L., "Learning automata Processing Ergodicity of the Mean: The Two-Action Case", *IEEE Transactions on Systems, Man, and Cybernetics*, Nov./Dec. 1983, pp. 1143-1148.

25. Thathachar M.A.L., Sastry P.S., "A new approach to the design of reinforcement schemes for learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, Feb. 1985, pp. 168-175.

26. Mukhopadhyay S., Thathachar M.A.L., "Associative learning of boolean functions", *IEEE Transactions on Systems, Man, and Cybernetics*, Sep./Oct. 1989, pp.1008-1015.

27. Lanctot J.K., Oommen B.J., "Discretized estimator learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, Nov./Dec. 1992, pp.1473-1483.

28. Oommen B.J., Lanctot J.K., "Discretised pursuit learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, July/Aug. 1990, pp. 931-938.

29. Oommen B.J., "Absorbing and ergodic discretized two-action learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, Mar./Apr. 1986, pp. 282-293.

30. Oommen B.J., Hansen E., "The asymptotic optimality of discretized linear reward-inaction learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, May/June 1984, pp.542-545.

31. Oommen B.J., Christensen J.P.R., "ε-optimal discretized linear reward-penalty learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, May/June 1988, pp.451-458.

32. Vasilakos A.V., Papadimitriou G.I., "A new approach to the design of reinforcement schemes for learning automata: Stochastic estimator learning algorithm", *Neurocomputing*, no. 7, 1995, pp.275-297.

33. Akselrod B., Langholz G., "A Simulation Study of Advanced Routing Methods in a Multipriority Telephone Network", *IEEE Transactions on Systems, Man, and Cybernetics*, Nov./Dec. 1985, pp. 730-736.

34. Vasilakos A.V., Paximadis C.T., "Fault-Tolerant Routing Algorithms Using Estimator Discretized Learning Automata for High-Speed Packet-Switched Networks", *IEEE Transactions on Reliability*, Dec. 1994, pp.582-593.

35. Girard A., *Routing and Dimensioning in Circuit-Switched Networks*, Addison-Wesley, 1990.

36. Narendra K.S., Wright E.A., Mason L.G., "Application of learning automata to telephone traffic routing and control", *IEEE Transactions on Systems, Man, and Cybernetics*, Nov. 1977, pp.785-792.

37. Economides A.A., "Learning Automata Routeing in Connection-Oriented Networks", *International Journal of Communication Systems*, Vol. 8 1995, pp. 225-237.

38. Narendra K.S., Thathachar M.A.L., "On the behavior of a learning automaton in a changing environment with application to telephone traffic routing", *IEEE Transactions on Systems,*

Man, and Cybernetics, May 1980, pp.262-269.

39. Nedzelnitsky O.V. Jr., Narendra K.S., "Nonstationary models of learning automata routing in data communication networks", *IEEE Transactions on Systems, Man, and Cybernetics*, Nov./Dec. 1987, pp. 1004-1015.

40. International Telecommunication Union, *ITU-T Recommendation I.371: Traffic control and congestion control in B-ISDN*, 1993.

41. ATM Forum, *ATM User-Network Interface Specification*, version 3.1, 1994.

42. Jamin S. et al, "A measurement-based admission control algorithm for integrated services packet networks", *Proceedings of the ACM Sigcomm*, September 1995, pp. 2-13.

43. Dziong Z., Choquette J., Liao K.-Q., Mason L., "Admission control and routing in ATM networks", *Computer Networks and ISDN Systems*, vol. 20, 1990, pp. 189-196.

44. Guerin R., Ahmadi H., Naghshineh M., "Equivalent capacity and its application to bandwidth allocation in high-speed networks", *IEEE Journal on Selected Areas in Communications*, vol. 9, 1991, pp. 968-981.

45. Gupta S., Ross K., El-Zarki M., "On routing in ATM networks", *IFIP Transactions C: Modelling and Performance Evaluation of ATM Technology*, Elsevier Science Publishers, 1993, pp. 229-239.

46. Jordan T.P., *Design and Analysis of Routing Algorithms for ATM Networks*, Ph.D. thesis, De Montfort University, 1995.

47. Maglaris B., et al, "Performance models of statistical multiplexing in packet video communications", *IEEE Transactions on Communications*, Vol. 36, No. 7, July 1988.

48. Naldi M., Pelusi P., "Efficiency and usability of equivalent bandwidth algorithms for ATM network dimensioning", *Proceedings of NOC '97*, Vol. 2, pp. 218-225.

49. O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modelling in ATM systems", University of Wurzburg, Institute of Computer Science, Report no. 101, Feb. 1995.

50. Moy J., "Link-state routing", *Routing in Communications Networks*, Prentice-Hall, 1995, pp. 135-157.

51. Ma Q., Steenkiste P., "Quality of service routing for traffic with performance guarantees", (version obtained by ftp from Carnegie Mellon University), 1997.

52. Ash G.R., "Dynamic network evolution, with examples from AT&T's evolving dynamic network", *IEEE Communications Magazine*, July 1995, pp. 26-39.

53. Guerin R., Orda A., Williams D., "QoS routing mechanisms and OSPF extensions", *IETF Internet Draft*, draft-guerin-qos-routing-ospf-00.txt, November 1996.

54. Chugo A., Iida I., "Dynamic path assignment for broadband networks based on neural computation", *IEICE Transactions on Communications*, vol. E75-B (pt. 7), 1992, pp. 634-641.

55. Rampal S., Reeves D.S., "Routing and admission control algorithms for multimedia networks", 1995.

56. Ma Q., Steenkiste P., Zhang H., "Routing high-bandwidth traffic in max-min fair share

networks", *ACM Sigcomm96*, August 1996, pp. 206-217.

57. Gibbens R.J., Kelly F.P., Key P.B., "Dynamic alternate routing", *Routing in Communications Networks*, Prentice-Hall, 1995, pp. 13-48.

58. Shaikh A., Rexford J., Shin K.G., "Dynamics of quality-of-service routing with inaccurate link-state information", (version obtained by ftp from University of Michigan), 1998.

59. Mitra D., Gibbens R.J., Huang B.D., "Analysis and optimal design of aggregated-least-busy-alternative routing on symmetric loss networks with trunk reservation", *Proceedings of the 13th. International Teletraffic Congress*, June 1991.

60. Breslau, *Adaptive Source Routing of Real-time Traffic in Integrated Services Networks*, Ph.D. Thesis, University of Southern California, 1996.

61. Moy J., *OSPF version 2*, RFC 2328, April 1998.

62. Baran P., "On distributed communication networks", *IEEE Transactions on Communications Systems*, CS-12(1):1-9, March 1964.

63. Rajaraman K., Sastry P.S., "Finite time analysis of the pursuit algorithm for learning automata", *IEEE Transactions on Systems, Man, and Cybernetics*, 1996, Vol. 26, pp. 590-598.

64. Aranzulla P., Reeve J., Mellor J., Mars P., "Improved stochastic learning automata for routing in ISDNs", Proceedings of NOC '97, Vol. 1, pp. 227-231, 1997.

65. Nyong O.D.O., *Performance Modelling and the Representation of Large Scale Distributed System Functions*, Ph.D. thesis, University of Durham, 1999.

66. Nyong D., Aranzulla P., Cosmas J., Pitts J., "Resource based policies for design of interworking heterogeneous service networks", *Interoperable Communication Networks*, Vol. 1, No. 2-4, pp. 571-580, 1998.

67. Luo Z., Cosmas J., Nyong D., Pitts J., *Learning and Prediction*, DTI HPIP project ARMAN, version 1., 1996.

68. *CCITT Recommendation Z.100: Specification and Description Language SDL*, 1988.

69. *Cisco 12000 Series Gigabit Switch Routers*, Cisco product literature, http://www.cisco.com/warp/public/733/12000/12000_ov.htm, 1998.

70. *Distributed Weighted Random Early Detection*, Cisco product literature, http://cio.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.htm, 1998.

71. *Designing Large Scale IP Networks*, Cisco product literature, http://www.cisco.com/univercd/cc/td/doc/cisintwk/idg4/index.htm .

72. *The Point-to-Point Protocol (PPP)*, rfc 1661, 1994.

73. *Routing Information Protocol*, rfc 1058, 1988.

74. *Transmission Control Protocol*, rfc 793, 1981.

75. *RTP: A Transport Protocol for Real-Time Applications*, rfc 1889, 1996.

76. *User Datagram Protocol*, rfc 768, 1980.

77. *Post Office Protocol – Version 3*, rfc 1939, 1996.

78. *Internet Message Access Protocol – Version 4rev1*, rfc 1730, 1996.

79. *Simple Mail Transfer Protocol*, rfc 821, 1982.

80. *Hypertext Transfer Protocol – HTTP/1.0*, rfc 1945, 1996.

81. *Cisco IOS™ Software: Quality of Service Solutions*, Cisco product literature, http://www.cisco.com/warp/partner/synchronicd/cc/cisco/mkt/ios/qosio_wp.htm

82. *Cisco IOS Technologies – Quality – CAR*, http://www.cisco.com/warp/customer/732/Tech/car/index.html

83. Matloff N., *Probability Modelling and Computer Simulation*, PWS-KENT Publishing Company, 1988.

84. "Chapter 9: Campus Network Example", *BONeS Designer: MAC Modules Library Reference*, BONeS Designer Documentation, 1994.

85. "Chapter 3: Example Systems", *BONeS Designer: ATM Library Reference*, BONeS Designer Documentation, 1994.

86. *Comparison of IP-over-SONET and IP-over-ATM Technologies*, http://www.trillium.com, Trillium Digital Systems, 1997.

87. Aranzulla P., "A comparative evaluation of OPNET and BONeS computer modelling packages", Internal report 1, 1995.

88. Aranzulla P., "Dynamic routing in ATM networks", Internal report 2, 1995.

89. Griffin D. ed., *Integrated Communications Management of Broadband Networks*, Crete University Press, 1996.

90. *PNNI Draft Specification*, ATM Forum, R9, 1995.

91. *OPNET Tool Operations Manual*, OPNET Modeller Documentation, Vol. 5, 1997.

92. Liengme B., *A Guide to Microsoft Excel for Scientists and Engineers*, Arnold, 1997.