Reinforcement Learning-based Load Shared Sequential Routing

Fariba Heidari, Shie Mannor and Lorne G. Mason

Department of Electrical and Computer Engineering,
McGill University, Montreal, Quebec, Canada
fariba.heidari@mail.mcgill.ca, shie.mannor@mcgill.ca, lorne.mason@mcgill.ca

Abstract. We consider event dependent routing algorithms for on-line explicit source routing in MPLS networks. The proposed methods are based on load shared sequential routing in which load sharing factors are updated using learning algorithms. The learning algorithms we employ are either based on learning automata or on online learning algorithms that were originally devised for solving the adversarial multi-armed bandit problem. While simple to implement, the performance of the proposed learning algorithms in terms of blocking probability compares favorably with the performance of other event dependent routing methods proposed for MPLS routing such as the Success-to-the-top algorithm. We demonstrate the convergence of one of the learning algorithms to the user equilibrium within a set of discrete event simulations.

1 Introduction

In the early days of packet switching much attention was given to the routing problem. See [1] for an early survey. With the emergence of the Internet, destination-based IP routing was widely adopted for reasons of scalability and stability in spite of the fact that destination-based routing gives the user little control over how his/her traffic is routed. This in turn means that traffic may be routed over congested links (paths) while at the same time alternative less congested paths are available. The need for better control of traffic routing, also referred to as "traffic engineering", gave rise to the Multi Protocol Label Switching (MPLS) standard. MPLS is a connection oriented framework proposed by the IETF to enable traffic engineering, congestion management and QoS provisioning in traditional IP networks [2, 3, 4].

In the MPLS framework, constraint-based routing and label swapping replaces the hop-by-hop destination-based routing mechanism used in traditional IP networks. In MPLS the route selection can employ either hop by hop routing or explicit routing. In the explicit routing method, a single Label Switching Router (LSR) specifies all (or some of) the hops along the path. Explicit routing gives the designer the ability to control the traffic load distribution in the network. The purpose of this work is to introduce an adaptive method for explicit source routing in MPLS networks.

Most of the algorithms proposed for traffic engineering in MPLS networks are state dependent algorithms where traffic routing is based on the information about the current status of the network [5, 6, 7]. These methods impose an information flooding overhead on the network. Event dependent routing methods, on the other hand, update their knowledge about the status of the network from the observed events. In [8] and later in [9], it was shown that the MPLS protocol and signaling extensions for crank back permits the use of source based dynamic routing schemes in the Internet. Such routing schemes have been widely used in TDM networks and proposed for ATM networks. Results presented in the aforementioned references demonstrate the merits of event dependent routing schemes in MPLS networks in terms of performance and scalability. In particular, the presented blocking probability performance of the Success-To-the-Top (STT) algorithm makes it a viable routing method for a realistic network. Consequently, the STT algorithm was proposed by AT&T as an event dependent routing method in AT&Ts IP network.

In this paper, we present an event dependent routing scheme with the application to explicit source routing in MPLS networks. The proposed method is based on Load Shared Sequential Routing (LSSR) where load sharing factors are updated using learning techniques [10]. Learning automata and techniques that were developed for the multi-armed bandit problem are the learning algorithms used in this study. These algorithms are suitable choices in applications where the system has incomplete information about the environment and learns via trial and error. The algorithms are extremely simple to implement and computationally efficient. The application of learning automata in dynamic routing has been widely studied in different applications such as telephone routing [11], [12], wavelength routing in WDM networks [13] and MPLS routing [14]. An application of multi-armed bandit algorithms in adaptive routing was presented in [15].

The paper is organized as follows. We present an overview of user equilibrium in Section 2. The reinforcement learning algorithms used in this work are reviewed in Section 3. The the proposed routing algorithm is explained in Section 4. Simulation results for two simple network topologies are presented in Section 5. A summary and conclusions are given in Section 6.

2 User Equilibrium

The concept of Nash equilibrium is commonly used in non-cooperative strategic games. In a Nash equilibrium, there is no incentive in terms of cost decrease for an individual user to unilaterally change his/her strategy. Before defining a Nash equilibrium, let us define the cost structure of the model we consider in this paper.

Given a network that uses LSSR type routing we let $\lambda^{o,d}$ represent the total traffic from origin 'o' to destination 'd'. The load sharing model assumes that for every origin-destination pair $(o-d)$, there is a load sharing vector $\alpha^{o,d}$ such

that:

$$\sum_{\ell=1}^{|T(od)|} \alpha_\ell^{o,d} = 1, \quad \text{and} \quad \alpha_\ell^{o,d} \geq 0,\ \ell = 1, 2, \ldots, |T(od)|,$$

where $T(od)$ is the set of route trees from 'o' to 'd' and $\alpha_\ell^{o,d}$ is the load sharing factor of the ℓth route tree from 'o' to 'd'. We further assume that given the load sharing factors, there is a cost associated with every route tree. We let this cost be denoted by $L_\ell^{o,d}$ and note that it depends, generally in a non-linear way, on load sharing factors of other origin-destination pairs. This cost represents the blocking probability between 'o' and 'd' via the ℓth route tree. With some abuse of notation, we denote the total cost incurred from 'o' to 'd' when the load sharing factors are $(\alpha_1^{o,d}), (\alpha_2^{o,d}), \ldots, (\alpha_{|T(od)|}^{o,d})$ by:

$$L(\alpha^{o,d}) = \sum_{\ell=1}^{|T(od)|} \alpha_\ell^{o,d} L_\ell^{o,d},$$

and we note that $L(\alpha^{o,d})$ depends in general on all the sharing factors of all the pairs in the network.

The set of load sharing factors α is a Nash Equilibrium if and only if for each pair 'o' and 'd', given the load sharing factors of other pairs, there is no incentive in terms of decreasing the cost to change the load sharing vector $\alpha^{o,d}$ to $\overline{\alpha}^{o,d} \neq \alpha^{o,d}$. We note that the players in this game are the pairs of nodes and not the individual nodes, that is, every origin-destination pair is assumed to act selfishly with respect to other pairs, even if the other pairs include the same origin or destination. In other words, the set of load sharing factors α is a Nash Equilibrium solution if and only if, for every pair 'o' and 'd', given the load sharing factors of other pairs,

$$L(\overline{\alpha}^{o,d}) \geq L(\alpha^{o,d}) \quad \forall \overline{\alpha^{o,d}} \neq \alpha^{o,d}.$$

User equilibrium can also be explained in terms of Wardrop equilibrium [16]. The Wardrop equilibrium assumes the contribution of each user's traffic to the cost is negligible. It is therefore not surprising that as the number of users increases to infinity with constant total traffic, the Nash Equilibrium converges to the Wardrop equilibrium [17]. For the routing problem, the Wardrop Equilibrium is the solution to a non-linear minimization problem that is described in [18]. It is straightforward to show that at the Wardrop equilibrium the cost on all the route-trees on which there is traffic (i.e., $\alpha_k^{o,d} > 0$) is the same. The centralized solution for the load share optimization problem is studied in [18], where both user and system optimization perspectives are considered. We refer the reader to that paper and references therein.

3 Reinforcement Learning

The reinforcement learning framework [10] considers an agent that interacts with a dynamic unknown environment and attempts to learn an optimal, or at least

reasonable, policy via a sequence of trials. At each stage (t), based on his policy, the agent selects one of the possible actions $a_i \in \underline{A}$. The agent receives a reward $(x(t) \in \underline{X})$ which is a measure of the desirability of the selected action. The agent may use this signal to update his policy.

In the following subsections the policy updating scheme used in two standard simple reinforcement learning approaches is explained. We start with learning automata and then discuss algorithms for the multi-armed bandit problem.

3.1 Learning Automata

Learning automata is one of the earliest frameworks dealing with learning the optimal behavior via trial and error [19, 20]. Formally, a learning automaton is described as a quadruple $\{\underline{A}, \underline{P}, \underline{X}, T\}$ where $\underline{\mathrm{A}}$ is the set of actions with $K = |A|$, $\underline{\mathrm{P}}$ is the probability distribution over the set of actions, $\underline{X}$ is the response from the environment and $T : \underline{P} \times \underline{A} \times \underline{X} \rightarrow \underline{P}$ is the updating scheme.

Based on their Markovian behavior, updating schemes are categorized as either ergodic or absorbing algorithms. Absorbing schemes converge to specific states and are more suitable choices in stationary environments. Ergodic schemes converge in distribution independent of the initial states and are preferred in non-stationary environments. In this work, the set of possible values for the reward signal is equal to $\underline{X} = \{0, 1\}$. One of the most well-known updating schemes for this case is the following linear mapping:
Let $a(t) = a_i$:

$$x(t) = 1: \quad p_j(t+1) = \begin{cases} (1-G)p_j(t) & \forall j \neq i \\ p_j(t) + G(1-p_j(t)) & j = i \end{cases}$$

$$x(t) = 0: \quad p_j(t+1) = \begin{cases} \frac{B}{K-1} + (1-B)p_j(t) & \forall j \neq i \\ (1-B)p_i(t) & j = i \end{cases},$$

where G and B are parameters. The parameter G represents the gain in the case of a positive reward, while the parameter B represents the gain if the reward is 0.

Here, we use the Linear Reward Inaction method (LRI) as the first choice for updating load sharing factors. In the LRI method, the probability distribution is updated only when the selected action is rewarded. This can be derived from the above formula by choosing B=0. The LRI method is known to be $\epsilon-$optimal [19] if the reward stream is stationary. That is, if the reward process (for each action) is stationary and if one of the actions leads to a higher reward in expectation, by choosing G arbitrarily small, $P\{\lim_{t\rightarrow\infty} p_i(t) = 1\}$ can be as close to unity as desired (where i is the action leading to the highest reward). However, if G is not small enough, the algorithm converges to a wrong solution with non-zero probability.

Another variation for updating load sharing factors is the ergodic Linear Reward-ϵ Penalty (LRϵP) method. The LRϵP is derived from the above formula

by choosing $B << G$. LRϵP is sub-optimal in comparison with the LRI algorithm in stationary environments. However, it avoids being locked in a state and is a better choice for non-stationary environments. For the LRϵP algorithm, a large a value of G will result in a large variance in the steady state distribution.

3.2 Algorithms for the Multi-Armed Bandit Problem

The multi-armed bandit problem is the problem of learning the optimal action-selection policy through a sequence of trials and errors. The variant of the multi-armed bandit problem we consider is the so-called adversarial multi-armed bandit problem. The reward process is non-stochastic and is assumed to be generated by Nature or even by an adversary. The reward associated with the selected action at t^{th} trial may depend on the actions selected in the previous trials. This formulation is reminiscent of online learning problems and leads to performance guarantees even when stationarity assumptions cannot be made. This is especially useful when considering multi-agent learning where each agent follows a regret minimizing algorithm; see [21] and references therein.

Exponential weighting-type algorithms were proposed to solve the multi-armed bandit problem. We used an exponential weighing algorithm proposed to solve the problem in the non-stochastic case ([22]). In the real networks, the blocking probability of the links depend on their traffic load and the routing policies of different $(o-d)$ pairs. This makes the EXP3.P algorithm as a suitable candidate for updating load sharing factors. A pseudocode of the EXP3.P algorithm taken from [22] is as follows:

Algorithm 1: Algorithm EXP3.P

1. Parameters: $\alpha > 0$ and $\gamma \in (0, 1]$
2. Initialization: For i=1,...,K $\quad w_i(1) = \exp(\frac{\alpha\gamma}{3}\sqrt{\frac{T}{K}})$.
   At each stage $t = 1, ..., T$:
   (a) For $i = 1, ..., K$ set $p_i(t) = (1-\gamma)\frac{w_i(t)}{\sum_{j=1}^{K} w_j(T)} + \frac{\gamma}{K}$.
   (b) Choose $a(t) = a_i$ randomly according to the distribution $p_1(t), ..., p_K(t)$.
   (c) Receive reward $x_i(t) \in [0, 1]$.
   (d) For j=1,...,K set

$$\hat{x}_j(t) = \begin{cases} \frac{x_j(t)}{p_j(t)} & j = i \\ 0 & \text{otherwise} \end{cases}$$

$$w_j(t+1) = w_j(t) \exp\left(\frac{\gamma}{3K}(\hat{x}_j(t) + \frac{\alpha}{p_j(t)\sqrt{KT}})\right).$$

The parameters α and γ control the amount of exploration done by the algorithm initially (α) and persistently (γ). The choice of these parameters leads to a difference between the cumulative reward of the optimal action over the hindsight (consistently choosing the best action as if it was known) and the

cumulative reward obtained from the learning algorithm. By selecting the parameters appropriately, the per round value of this difference can be made to go to zero as the number of stages goes to infinity.

4 A Reinforcement Learning Approach to Load Shared Sequential Routing

Load Shared Sequential Routing (LSSR) randomly partitions the traffic load ($\lambda^{o,d}$) associated with an origin-destination pair ($o-d$) into n sub-streams using a set of load sharing factors ($\{\alpha_1^{o,d}, ..., \alpha_{|T(od)|}^{o,d}\}$). Each sub-stream is then offered to a route-tree which consists of one or more alternate paths. The number and the order of the alternate paths can be different from one route-tree to the other. The alternate paths of the selected route-tree are then tried sequentially. If there is not enough bandwidth available on at least one link of a path, an MPLS notification message is sent to the origin node and the origin node forwards the request to the next alternate path. Sending the notification messages can be done using the extensions of constraint-based routing using label distribution protocol ([23]) and the resource reservation protocol ([24]). This process is repeated until the requested bandwidth is allocated on one alternate path or all alternate paths have been tried unsuccessfully. If all paths have been tried unsuccessfully, the request is lost and rejected from the network. A pictorial representation of the LSSR model is provided in Fig. 1.

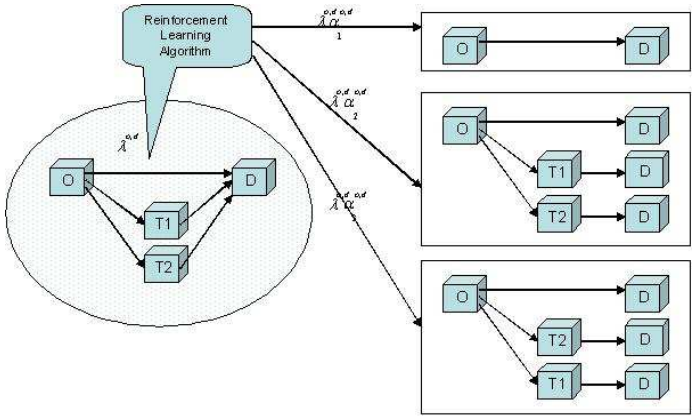

Fig. 1. LSSR Model: Different paths between O and D may be tried.

The LSSR model imposes no restriction on the load-sharing factors other than non-negativity and

$$\sum_k \alpha_k^{o,d} = 1 \ .$$

In Reinforcement Learning-based Load Shared Sequential Routing (abbreviated RL-based LSSR), load sharing factors $\alpha_k^{o,d}$ are updated using learning techniques. Here, the set of actions is equal to the set of available route-trees, response from the environment is the $\{0, 1\}$ rejected / accepted feedback from the network and the probability distribution over the set of actions is the set of load sharing factors $\{\alpha_k^{o,d}\}$.

5 Simulation Results

In this section, the performance of the RL-based LSSR algorithm is compared with that of another event dependent routing algorithm, Success-to-the-top (see the Introduction for references). STT is a decentralized on-line routing algorithm with a random updating scheme. In this algorithm, the bandwidth request between an $(o-d)$ pair is first sent through the primary path. In the case there is a direct link between $(o-d)$, the primary path would be the direct path. If the request is blocked on this path, it is sent through the last successful secondary path. If the bandwidth request is blocked on both the primary and the last successful secondary, another alternate path is selected at random and the request is forwarded through this path. The algorithm allows a maximum of N crank backs. If the request is accepted on one of the alternate paths, that path is labelled as the last successful path and will be used in routing the next bandwidth request between this $(o-d)$ pair.

As a measure of performance, we have used an estimation of the overall blocking probability. The blocking probability is estimated using exponential smoothing. The following calculation is done recursively whenever a bandwidth request is received:

$$R = SR + (1-S)(1-X) \ .$$

Here, R is bandwidth request blocking probability. X takes a value of '1' when the current bandwidth request is accepted and '0' otherwise. S is the smoothing constant. The larger the S, the slower R converges and the smoother the convergence curve appears. In our experiments S is set to 0.9999. The confidence interval used in this set of simulations is 90-th percentile.

In the first experiment a fully connected 4-node network is used. Between each $(o-d)$ pair, there are 5 route-trees. The first route-tree includes only the direct path; the second and third route-trees each include the direct path and one of the two-hop alternate paths. The last two route-trees include the direct path and both two-hop alternate paths. In the last two route-trees, the order of alternate paths is different. The capacity of each of the unidirectional links is equal to 50 trunks and traffic on each $(o-d)$ pair is 45 Erlangs. Sessions arrive according to a Poisson process and holding times are exponential distributed. All bandwidth requests are equal to 1 trunk. For the learning automaton model, the parameter 'G' is set to 0.001 and the parameter 'B' = .1G. Discrete event simulations are performed using OPNET Modeler 11.5.

A comparison of RL-based LSSR using different learning algorithms is illustrated in Fig. 2. The results of LRI-based LSSR simulation shows that load sharing factors converge to the user equilibrium solution of the problem as can be seen in Fig. 3. This is not the case for the LRϵP and EXP3.P algorithm and these algorithms do not necessarily converge to user equilibrium. In EXP3.P algorithm, the probability of selecting each action is lower bounded by a small value. Still, all the learning algorithms behave more or less the same in terms of performance.

In the next experiment, a more realistic non-symmetrical network with 9 nodes is used. The capacity and offered traffic matrices are listed in Appendix A. Here again, capacities are expressed in number of trunks (bandwidth units) and the offered traffic is in Erlangs. In order to have a reasonable comparison between STT and RL-based LSSR, the same set of alternate paths is used for RL-based LSSR and STT. Each route-tree has one direct link and three alternate paths and the maximum number of crank backs for STT is equal to three. The total blocking probability of RL-based LSSR and STT are compared with each other in Fig. 4. As can be seen in the figure, there is not much difference in performance of the different learning based algorithms. However, RL-based LSSR algorithms have better performance in terms of blocking probability in comparison with STT.

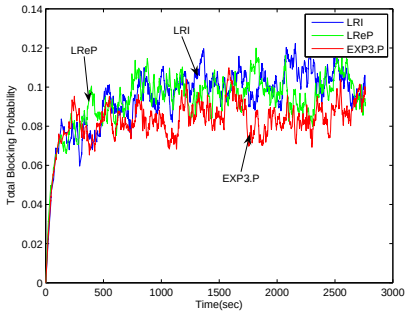

Fig. 2. Performance comparison in 4-Node network.

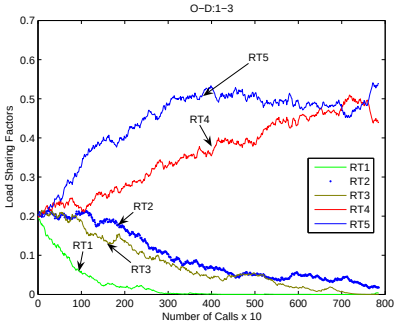

Fig. 3. Load Sharing Factors of 1-3 in 4-Node network using LRI learning.

In the last set of simulations the performance of the STT and RL-based LSSR algorithms are compared with each other in the case where there is a link failure in the system. As can be seen in the results (Fig. 5), STT has the highest blocking probability before and after the change has occurred. However, none of these algorithms react fast enough to be used in the link failure recovery and other restoration schemes such as the method presented in [25] can be used to improve the performance of the system.

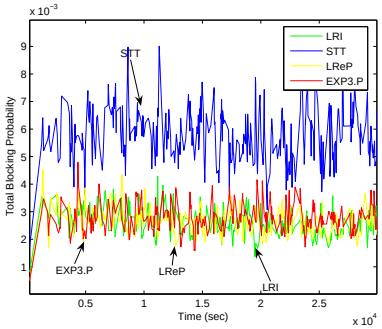

Fig. 4. Performance comparison in 9-Node network.

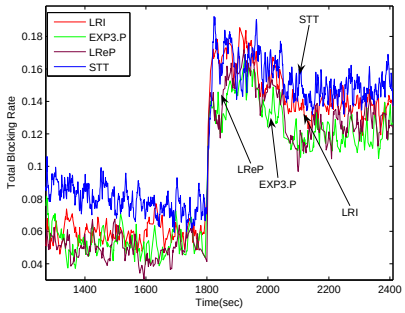

Fig. 5. Performance comparison when a link fails at $T = 1800$ seconds.

6 Discussion and Conclusion

In this paper, adaptive on-line routing algorithms for explicit source routing were presented. The proposed algorithmic framework (called RL-based LSSR) is based on load share sequential routing and uses reinforcement learning techniques to update the load sharing factors. We considered three learning algorithms: LRI, LRϵP and EXP3.P. The LRI algorithm is a suitable choice for stationary environments and by a suitable choice of learning parameter, the resulting load sharing factors converge to the user equilibrium solution. The LRϵP and EXP3.P algorithms are better choices for non-stationary environments. The EXP3.P algorithm has additional performance guarantees for the worst cases where the reward generation may be adversarial. In real networks, the blocking probability is a function of the traffic load and the routing policy of different $(o-d)$ pairs. This makes EXP3.P algorithm a suitable choice for routing in such environments.

The performance of the RL-based LSSR was compared with STT, another event dependent routing algorithm also proposed for routing in MPLS networks. The discrete event simulation results in some example networks show that RL-based LSSR compare favorably with STT in terms of network blocking probability. RL-based LSSR algorithms track the smooth changes in the traffic pattern. However, they are not fast enough for link failure recovery. One future area of research is addressing the problem of improving the response time to abrupt changes such as link failure.

The learning algorithms that were presented are rather simple. While this simplicity has the advantage of low computational cost and low informational needs, one may consider more complex algorithms that take the state of the network into account. Of course, since the complete state of the network is not known to each node, some reasoning in terms the uncertainty of the state estimate is needed. The advantage of such state dependent schemes may be

in their ability to synchronize between the different nodes as well as detecting abnormal traffic patterns.

In this paper, the analytical formulation of user equilibrium load sharing factors in LSSR model was also reviewed and it was shown that in the user equilibrium solution, for each $(o-d)$ pair, the traffic loss probability of all used route trees are equal and less than the traffic loss probability of other route trees. The simulation results presented in Section 5 confirm that by suitable choice of learning parameter, LRI-based LSSR converges to the user equilibrium solution.

Acknowledgements

This work was partially supported in part by the NSERC Strategic Project Grant STPGP 269449 03 and by the Canada Research Chairs Program. The authors thank Hanhui Zhang for supplying code and results which were useful in the research reported here.

References

[1] Mason, L.G.: Equilibrium Flows, Routing Patterns and Algorithms for Store-and-Forward Networks. Journal of Large Scale systems **8** (1985) 187–209
[2] Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol Label Switching Architecture. RFC 3031 (Jan. 2001)
[3] Awduche, D.O.: MPLS and Traffic Engineering in IP Networks. IEEE Communications Magazine **37**(12) (1999) 42–47
[4] LeFaucheur, F., Lai, W.: Requirements for Support of Differentiated Services-aware MPLS Traffic Engineering. RFC 3564 (July 2003)
[5] Kar, K., Kodialam, M., Lakshman, T.: Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. IEEE Journal on Selected Areas in Communications **18**(12) (December 2000) 2566–2579
[6] Suri, S., Waldvogel, M., Bauer, D., Warkhede, P.R.: Profile-Based Routing and Traffic Engineering. Computer Communications **26** (2003) 351–365
[7] Szeto, W., Boutaba, R., Iraqi, Y.: Dynamic Online Routing Algorithm for MPLS Traffic Engineering. Lecture Notes in Computer Science (2002) 936–946
[8] Ash, G.R.: Performance evaluation of QoS-routing methods for IP-based multi-service networks. Computer Communications **26**(8) (2003) 817–833
[9] Ash, G.R.: Traffic Engineering and QoS Optimization of Integrated Voice & Data Networks. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2006)
[10] Sutton, R.S., Barto, A.G.: Reinforcement Learning: an Introduction. MIT Press, Cambridge, MA (1998)
[11] Narendra, K.S., Wright, E.A., Mason, L.G.: Application of learning automata to telephone traffic routing and control. IEEE Trans. on Systems, Man and Cybernetics **7**(11) (1977) 785–792
[12] Brunet, G.: Optimisation de l'acheminement séquentiel non hiérarchique par automates intelligents. Master's thesis, INRS-Telecommunication (1991)
[13] Alyatama, A.: Dynamic routing and wavelength assignment using learning automata technique [all optical networks]. In: Proceedings of IEEE GLOBCOM. (2004) 1912–1917

[14] Zhang, H.: Simulation of learning Automata Load Shared Sequential Routing in MPLS network. Master's Project Report, Electrical and Computer Engineering, McGill University (2003)
[15] György, A., Ottucsák, G.: Adaptive routing using expert advice. The Computer Journal **49**(2) (2006) 180–189
[16] Wardrop, J.G.: Some theoretical aspects of road traffic research. In: Proceedings of the Institution of Civil Engineers, Part II. (1952) 325–378
[17] Haurie, A.B., Marcotte, P.: On the relationship between Nash-Cournot and Wardrop Equilibria. Networks **15** (1985) 295–308
[18] Brunet, G., Heidari, F., Mason, L.G.: Load Shared Sequential Routing in MPLS Networks: System and User Optimal Solutions. EuroFGI Net-COOP, to appear (2007)
[19] Narendra, K.S., Thathachar, M.A.L.: Learning Automata: an Introduction. Prentice Hall, Upper Saddle River, NJ, USA (1989)
[20] Lakshmivarahan, S.: Learning Algorithms: Theory and Applications. Springer, New York (1981)
[21] Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge University Press (2006)
[22] Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. SIAM Journal on Computing **32**(1) (2002) 48–77
[23] Jamoussi, B., Andersson, L., Callon, R., Dantu, R., Wu, L., Doolan, P., Worster, T., Feldman, N., Fredette, A., Girish, M., Gray, E., Heinanen, J., Kilty, T., Malis, A.: Constraint-Based LSP Setup using LDP. RFC 3212 (January 2002)
[24] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., Swallow, G.: RSVP-TE: Extension to RSVP for LSP tunnels. RFC 3209 (December 2001)
[25] Qin, Y., Mason, L.G., Jia, K.: Study on a joint multiple layer restoration scheme for IP over WDM networks. IEEE Network Magazine **17**(2) (2003) 43–48

A Appendix

Table 1. Traffic Matrix of 9-Node Network

0	9552	12976	0	0	3680	13856	8224	5104
8672	0	1376	5872	8176	0	3280	1824	0
12000	2848	0	19504	18000	1104	2224	4928	0
0	40480	17152	0	0	4848	5072	2704	2400
0	4576	12832	0	0	1728	6624	3776	2304
2976	0	1376	3200	5072	0	6352	4480	1232
13056	1696	2000	5504	5472	4848	0	5024	1056
5728	752	1952	2576	1776	4448	4304	0	3200
4304	704	1472	3152	976	672	3472	5024	0

Table 2. Capacity Matrix of 9-Node Network

0	9000	13000	0	0	6000	22000	9000	5000
7000	0	4000	6000	6000	0	2000	4000	3000
15000	3000	0	24000	15000	4000	3000	4000	8000
0	6000	24000	0	0	4000	7000	6000	4000
0	7000	20000	0	0	6000	7000	5000	5000
5000	0	3000	4000	5000	0	7000	7000	2000
10000	3000	4000	6000	9000	9000	0	6000	5000
5000	2000	4000	3000	6000	4000	4000	0	8000
8000	3000	5000	5000	5000	3000	3000	10000	0