A Predictive Q-Learning Algorithm for Deflection Routing in Buffer-Less Networks

Soroush Haeri, Majid Arianezhad, and Ljiljana Trajković
Simon Fraser University
Vancouver, British Columbia, Canada
Email: {shaeri, arianezhad, ljilja}@sfu.ca

***Abstract*—In this paper, we introduce a predictive Q-learning deflection routing (PQDR) algorithm for buffer-less networks. Q-learning, one of the reinforcement learning (RL) algorithms, has been considered for routing in computer networks. The RL-based algorithms have not been widely deployed in computer networks where their inherent random nature is undesired. However, their randomness is sought-after in certain cases such as deflection routing, which may be employed to ameliorate packet loss caused by contention in buffer-less networks.**

We compare the proposed algorithm with two existing reinforcement learning-based deflection routing algorithms. Simulation results show that the proposed algorithm decreases the burst loss probability in the case of heavy traffic load while it requires fewer deflections. The PQDR algorithm is implemented using the ns-3 network simulator.

***Index Terms*—Computer networks, buffer-less networks, deflection routing, reinforcement learning, predictive Q-learning.**

I. Introduction

Reinforcement learning (RL) framework formalizes trial-and-error-based learning processes. Its four basic elements are: an agent or decision maker, the environment of the agent, actions that the agent performs, and feedback signals generated by the environment. The objective of an RL agent is to maximize/minimize the rewards/penalties that it receives from the environment. The agent observes environment variables known as the *system state* and performs a rewarding action according to its knowledge of the environment.

The environment evaluates the agent's action and generates a reinforcement signal. The agent employs this signal to enhance its ability to make decisions [1]. An RL agent requires: (a) policy that specifies the agent's action in each state; (b) reward function that maps the reinforcement signal received from the environment to a number; (c) value function that represents the total reward that an agent may expect during its interactions with the environment; and (d) model of the environment that enables the agent to predict future states and rewards [2].

RL-based algorithms were proposed in the early days of the Internet to generate routing policies [3]–[6]. Routing in communication networks is a process of selecting a path that connects two end-points for packet transmission. A common approach is to map the network topology to a weighted graph and set the weight of each edge according to metrics such as number of hops to destination, congestion, latency, link failure, or the business relationships between the edge nodes. The path with the minimum cost is then selected for end-to-end communications. Q-learning [7] is an RL algorithm that has been considered as a viable approach for generating routing policies. The Q-routing algorithm [3] requires that nodes make their routing decisions locally. Each node learns a local deterministic routing policy using the Q-learning algorithm. Generating the routing policies locally is computationally less intensive. However, the Q-routing algorithm does not generate an optimal routing policy in networks with low loads nor does it learn new optimal policies in cases when network load decreases. Predictive Q-routing [5] addresses these shortcomings by recording the best experiences learned, which may then be reused to predict traffic behavior. The distributed gradient ascent policy search [4] has also been proposed for generating optimal routing policies, where reinforcement signals are transmitted when a packet is successfully delivered to its destination.

RL algorithms have also been employed for deflection routing that may be used to resolve contention in buffer-less networks such as optical burst-switched (OBS) networks [8]. Two routing protocols may operate in such networks: an underlying routing protocol such as the Open Shortest Path First (OSPF) that primarily routes packets and a deflection routing algorithm that only deflects packets in case of a contention. Contention occurs when according to the routing table, multiple arriving traffic flows at a node need to be routed through a single outgoing link. In this case, only one flow is routed through the optimal outgoing link defined by the routing table. If no contention resolution scheme is employed, the remaining flows are discarded because the node possesses no buffers. In these cases, deflection routing helps temporarily misroute packets instead of buffering or discarding them.

Packet routing algorithms in large networks such as the Internet should also consider the business relationships between Internet service providers. Therefore, randomness is not a desired property for a routing algorithm to be deployed in such environment. Consequently, RL-based routing algorithms were not applied for routing in the Internet because of their inherent random structure. On the contrary, deflection routing may benefit from the random nature of RL algorithms. A deflection routing algorithm coexists in the network with an underlying routing protocol that usually generates a significant number of control signals. Therefore, it is desired that deflection routing

978-1-4799-0652-9/13/$31.00

protocols generate few control signals. RL algorithms enable a deflection routing protocol to generate viable deflection decisions by adding a degree of randomness to the decision making process. The deflection decision (action) is the index of an outgoing link that a node may use to deflect a traffic flow.

In this paper, we propose a predictive Q-learning deflection routing (PQDR) algorithm that combines the learning phase of the predictive Q-routing algorithm [5] and the signaling algorithm of the Reinforcement Learning Deflection Routing Scheme (RLDRS) [9]. We evaluate performance of the proposed algorithm using the ns-3 [10] network simulator and compare its performance with RLDRS and a newly proposed Node Degree Dependent (Q-NDD) algorithm [11].

This paper is organized as follows. In Section II, we provide a brief survey of work related to the application of reinforcement learning in deflection routing. We describe the proposed PQDR algorithm in Section III. Its performance is evaluated in Section IV. We conclude with Section V.

II. Related Work

The Q-learning path selection algorithm [12] calculates a priori set of candidate paths $P = \{p_1, \ldots, p_m\}$ for tuples (s_i, s_j), where $s_i, s_j \in S$ and $S = \{s_1, \ldots, s_n\}$ denotes the set of all edge nodes in the network. The i^{th} edge maintains a Q-table, which contains a quality value (Q-value) for every tuple (s_j, p_k), where $s_j \in S \backslash \{s_i\}$ and $p_k \in P$. The sets S and P are states and actions, respectively. The Q-value is updated after each decision is made and the score of the path is reduced or increased depending on the received rewards. This algorithm does not specify a signaling method or a procedure for handling the feedback signals.

RLDRS [9] employs the Q-learning algorithm for deflection routing. The advantage of RLDRS is its precise signaling and rewarding procedure. A drawback of the Q-learning path selection algorithm and RLDRS is that they are not scalable because their complexity depends on the size of the network. The Q-NDD algorithm [11] also employs Q-learning for deflection routing. However, it scales better in larger networks because its complexity depends on the node degree rather than the network size. In case of RLDRS, nodes receive feedback signals for every packet that they deflect while in case of Q-NDD, feedback signals are received only if the deflected packet is discarded by another node.

Decisions that repeatedly receive poor rewards have very low Q-values. Certain poor rewards that a decision receives might be due to a transient link failure or congestion and, hence, recovery and reselection of such decisions may improve the decision making. RLDRS and the Q-NDD algorithm do not provide a procedure to reselect those paths that have low Q-values as a consequence of transient network conditions. The proposed PQDR algorithm enables a node to recover and reselect such paths and to improve its decision making ability.

III. Predictive Q-learning Deflection Routing Algorithm

In this Section, we present details of the predictive Q-learning deflection routing algorithm (PQDR). PQDR determines an optimal output link to deflect traffic flows when contention occurs. The algorithm combines the predictive Q-routing (PQR) algorithm [5] and RLDRS [9] to optimally deflect contending flows. When deflecting a traffic flow, the PQDR algorithm stores in a Q-table the accumulated reward for every deflection decision. It also recovers and reselects decisions that are not well rewarded and have not been used over a period of time.

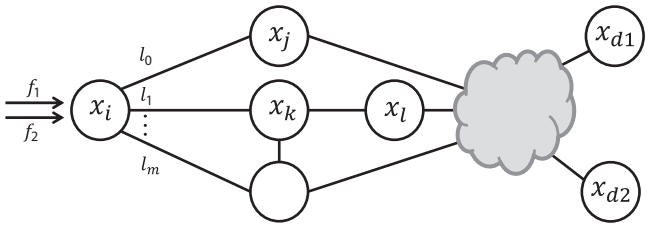


Fig. 1. A network with n buffer-less nodes.

An arbitrary buffer-less network is shown in Fig. 1. Let $\mathcal{N} = \{x_1, x_2, \ldots, x_n\}$ denote the set of all network nodes. Assume that each node possesses a shortest path routing table and a module that implements the PQDR algorithm to generate deflection decisions. Consider an arbitrary node x_i that is connected to its m neighbors through a set of outgoing links $\mathcal{L} = \{l_0, l_1, \ldots, l_m\}$. Node x_i routes the incoming traffic flows f_1 and f_2 to the destination nodes x_{d_1} and x_{d_2}, respectively. According to the shortest path routing table stored in x_i, both flows f_1 and f_2 should be forwarded to node x_j via the outgoing link l_0. In this case, node x_i forwards flow f_1 through l_0 to the destination x_{d1}. However, flow f_2 is deflected because node x_i is unable to buffer it. Hence, node x_i employs the PQDR algorithm to select an alternate outgoing link from the set $\mathcal{L} \backslash \{l_0\}$ to deflect flow f_2. It maintains five tables that are used by PQDR to generate deflection decisions. Four of these tables store statistics for every destination $x \in \mathcal{N} \backslash \{x_i\}$ and outgoing link $l \in \mathcal{L}$:

1. $Q_{x_i}(x, l)$ stores the accumulated rewards that x_i receives for deflecting packets to destinations x via outgoing links l.
2. $B_{x_i}(x, l)$ stores the minimum Q-values that x_i has calculated for deflecting packets to destinations x via outgoing links l.
3. $R_{x_i}(x, l)$ stores recovery rates for decisions to deflect packets to destinations x via outgoing links l.
4. $U_{x_i}(x, l)$ stores the time instant when x_i last updated the (x, l) entry of its Q-table after receiving a reward.

The size of each table is $m \times (n-1)$, where m and n are the number of elements in the sets $\mathcal{L}$ and $\mathcal{N}$, respectively. The fifth table $P_{x_i}(l)$ records the blocking probabilities of the outgoing links connected to the node x_i. A time window τ is defined for each node. Within each window, the node counts

successfully transmitted packets λ_{l_i} and the discarded packets ω_{l_i} on every outgoing link $l_i \in \mathcal{L}$. When a window expires, node x_i updates entries in its P_{x_i} table as:

$$P_{x_i}(l_i) = \begin{cases} \dfrac{\omega_{l_i}}{\lambda_{l_i} + \omega_{l_i}} & \lambda_{l_i} + \omega_{l_i} > 0 \\ 0 & \text{otherwise} \end{cases}. \quad (1)$$

The PQDR algorithm needs to know the destination node x_{d_2} of the flow f_2 in order to generate a deflection decision. For every outgoing link $l_i \in \mathcal{L}$, the algorithm first calculates a Δt value as:

$$\Delta t = t_c - U_{x_i}(x_{d_2}, l_i), \quad (2)$$

where t_c represents the current time and $U_{x_i}(x_{d_2}, l_i)$ is the last time instant when x_i had received a feedback signal as a result of selecting the outgoing link l_i for deflecting a traffic flow that is destined for node x_{d_2}. The algorithm then calculates $Q'_{x_i}(x_{d_2}, l_i)$ as:

$$Q'_{x_i}(x_{d_2}, l_i) = \max\Big(Q_{x_i}(x_{d_2}, l_i) + \Delta t \times R_{x_i}(x_{d_2}, l_i), B_{x_i}(x_{d_2}, l_i)\Big). \quad (3)$$

$Qx_i(x_{d_2}, l_i)$ is then used to generate the deflection decision (action) ζ:

$$\zeta \leftarrow \arg\min_{l_i \in \mathcal{L}} \{Q'_{x_i}(x_{d_2}, l_i)\}. \quad (4)$$

The deflection decision ζ is the index of the outgoing link of node x_i that may be used to deflect the flow f_2. Let us assume that $\zeta = l_1$ and, therefore, node x_i deflects the traffic flow f_2 via l_1 to its neighbor x_k. When the neighboring node x_k receives the deflected flow f_2, it either uses its routing table or the PQDR algorithm to forward the flow to its destination through one of its neighbors (x_l). Node x_k then calculates a feedback value ν and sends it back to node x_i that had initiated the deflection:

$$\nu = Q_{x_k}(x_{d_2}, l_{kl}) \times D(x_k, x_l, x_{d_2}), \quad (5)$$

where l_{kl} is the link that connects x_k and x_l, $Q_{x_k}(x_{d_2}, l_{kl})$ is the (x_{d_2}, l_{kl}) entry in x_k's Q-table, and $D(x_k, x_l, x_{d_2})$ is the number of hops from x_k to the destination x_{d_2} through the node x_l. Node x_i receives the feedback ν for its action ζ from its neighbor x_k and then calculates the reward r:

$$r = \frac{\nu(1 - P_{x_i}(\zeta))}{D(x_i, x_k, x_{d_2})}, \quad (6)$$

where $D(x_i, x_k, x_{d_2})$ is the number of hops from x_i to the destination x_{d_2} through x_k while $P_{x_i}(\zeta)$ is the entry in the x_i's link blocking probability table P_{x_i}, which corresponds to the outgoing link ζ (l_1). The reward r is then used by the x_i's PQDR module to update the (x_{d_2}, ζ) entries in the Q_{x_i}, B_{x_i}, and R_{x_i} tables. The PQDR algorithm first calculates the difference ϕ between the reward r and $Q_{x_i}(x_{d_2}, \zeta)$:

$$\phi = r - Q_{x_i}(x_{d_2}, \zeta). \quad (7)$$

The Q-table is then updated using ϕ as:

$$Q_{x_i}(x_{d_2}, \zeta) = Q_{x_i}(x_{d_2}, \zeta) + \alpha \times \phi, \quad (8)$$

where $0 < \alpha \leq 1$ is the learning rate. Table B_{x_i} keeps the minimum Q-values and, hence, its (x_{d_2}, ζ) entry is updated as:

$$B_{x_i}(x_{d_2}, \zeta) = \min(B_{x_i}(x_{d_2}, \zeta), Q_{x_i}(x_{d_2}, \zeta)). \quad (9)$$

Table R_{x_i} is updated as:

$$R_{x_i}(x_{d_2}, \zeta) = \begin{cases} R_{x_i}(x_{d_2}, \zeta) + \beta \dfrac{\phi}{t_c - U_{x_i}(x_{d_2}, \zeta)} & \phi < 0 \\ \gamma R_{x_i}(x_{d_2}, \zeta) & \text{otherwise} \end{cases}, \quad (10)$$

where t_c denotes the current time and $0 < \beta \leq 1$ and $0 < \gamma \leq 1$ are recovery learning and decay rates, respectively. Finally, the PQDR algorithm updates table U_{x_i} with current time t_c as:

$$U_{x_i}(x_{d_2}, \zeta) = t_c. \quad (11)$$

IV. Performance Evaluation

In order to evaluate and compare performance of the proposed PQDR algorithm, we have also implemented in ns-3 the existing RLDRS [9] and Q-NDD algorithm [11]. We compare the algorithms in terms of burst loss probability, number of deflections, average number of hops, and end-to-end delay by simulating the National Science Foundation (NSF) network topology. Algorithms are also compared in terms of memory requirements and Central Processing Unit (CPU) usage by simulating Waxman topologies [13] consisting of 500, 1,000, and 2,000 nodes.

A. Simulation of the NSF Network

For simulation scenarios, we employ the NSF network topology shown in Fig. 2. The topology was generated by

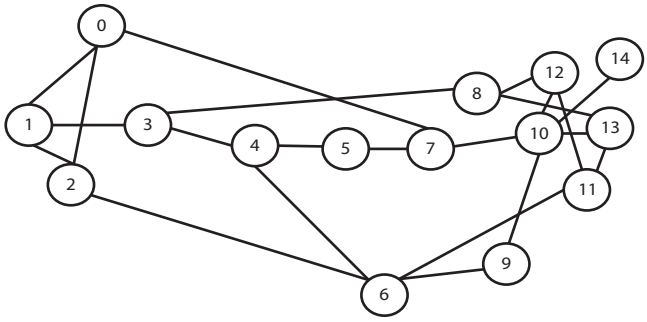


Fig. 2. Topology of the NSF network after the 1989 transition. Node 9 and node 14 were added in 1990.

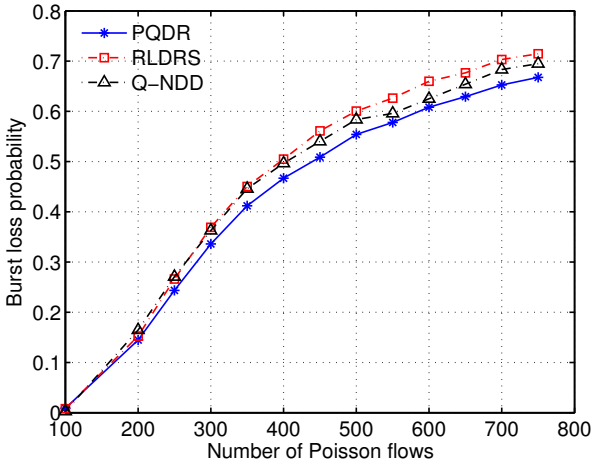
extracting the geodetic coordinates [14] of the NSF network nodes from the Google Earth [15]. They are then transformed to the Cartesian coordinates. We assume the buffer-less optical burst-switched architecture for data transmission where nodes are connected using bi-directional 1 Gbps fiber links with 8 and 64 wavelengths.

Multiple Poisson traffic flows with a data rate of 1 Gbps are transmitted randomly across the network. Each Poisson flow is 50 bursts long with each burst containing 12.5 kB of payload. The Poisson burst arrival process has been widely

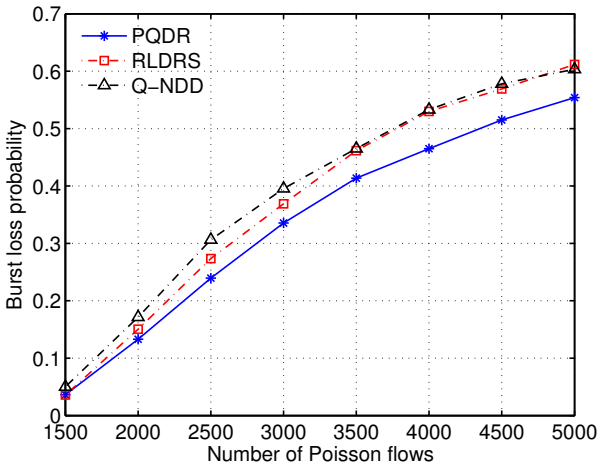
used for performance analysis of optical burst-switched networks because it is mathematically tractable [16], [17]. Each simulation scenario is repeated five times with various random assignments of nodes as sources and destinations. Simulation results are averaged over five runs.

In all simulations, we allow up to two deflections per burst. The burst header processing time is set to 0.1 ms. The duration of the sliding window that the nodes employ to calculate burst loss probability on each of their interfaces is set to $\tau = 50$ ms. The learning, learning recovery, and recovery decay rates are $\alpha = 0.1$, $\beta = 0.7$, and $\gamma = 0.9$, respectively [5].

The burst loss probability as a function of the number of Poisson flows for 8 and 64 wavelengths is shown in Fig. 3. In all scenarios, the PQDR algorithm performs better



(a)



(b)

Fig. 3. Burst loss probability as a function of the number of Poisson flows in the NSF network simulation scenario with (a) 8 wavelengths and (b) 64 wavelengths.

than RLDRS and Q-NDD in terms of burst loss probability. The results show that PQDR scales better as the number of wavelengths increases. For example, PQDR performs on average 7.0% better than RLDRS in case of 8 wavelengths and 800 traffic flows, as shown in Fig. 3(a). Simulation results shown in Fig. 3(b) illustrate that PQDR performs on average 10% better than RLDRS in case of 64 wavelengths and 5,000 traffic flows.

Although burst deflection reduces the burst loss probability, it introduces excess traffic load to the network. This behavior is undesired from the traffic engineering point of view. Therefore, the volume of the deflected traffic should also be considered as a performance measure for deflection routing algorithms. We use two metrics to compare the PQDR algorithm with RLDRS and the Q-NDD algorithm in terms of the volume of the deflected traffic: deflection ratio and average number of deflections. We define deflection ratio as the number of deflected bursts divided by the number of transmitted bursts:

$$\text{Deflection ratio} = \frac{\text{Number of deflected bursts}}{\text{Number of transmitted bursts}}. \quad (12)$$

Deflection ratio and average number of deflections as a function of the number of traffic flows are shown in Fig. 4 and Fig. 5, respectively. PQDR deflects on average 20% fewer bursts than RLDRS and Q-NDD in the 64-wavelength scenario, as shown in Fig. 5.

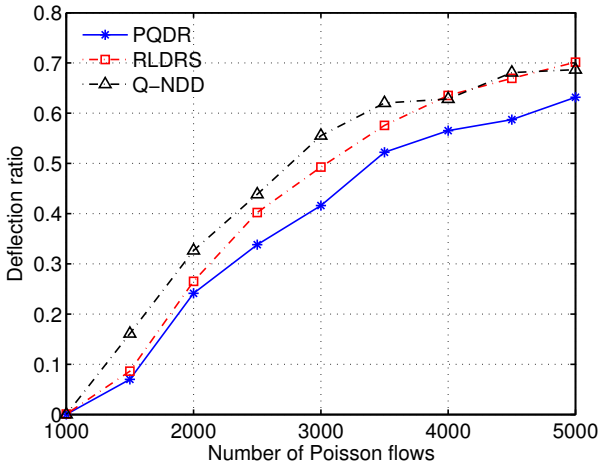


Fig. 4. Deflections ratio as a function of the number of traffic flows in the NSF network scenario with 64 wavelengths. PQDR deflects fewer packets.

Performance of the PQDR algorithm, RLDRS, and the Q-NDD algorithm in terms of average number of hops for the 64-wavelength scenario is shown in Fig. 6. When generating the reward signals that are used to update Q-values, the PQDR algorithm and RLDRS consider the number of hops to the destination. Hence, they perform better than Q-NDD in terms of the average number of hops traveled by bursts. PQDR does not rely only on the Q-values to generate deflection decisions, which may result in selection of longer paths. Therefore, bursts travel on average through more hops.

Performance of the PQDR algorithm, RLDRS, and the Q-NDD algorithm in terms of average end-to-end delay for the 64-wavelength scenario is shown in Fig. 7. Bursts experience smaller end-to-end delays in case of RLDRS because the scheme maintains only one table (Q-table) and, therefore, the

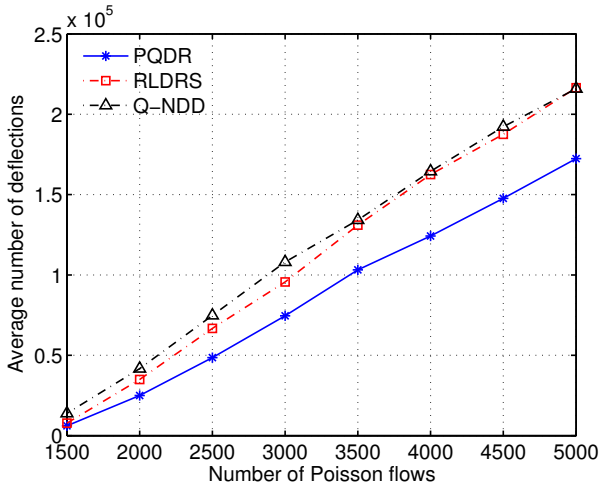


Fig. 5. Average number of deflections as a function of the number of traffic flows in the NSF network scenario with 64 wavelengths. PQDR deflects fewer packets.

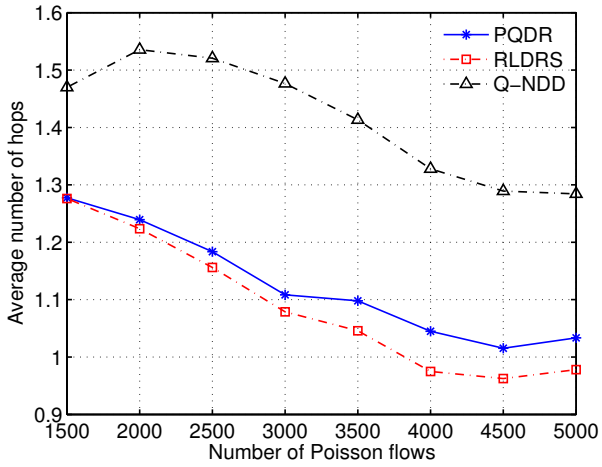


Fig. 6. Average number of hops traveled by bursts as a function of number of flows in the NSF network scenario with 64 wavelengths. In the case of RLDRS, the bursts travel the least number of hops.

link selection process and table updates are faster than in the case of PQDR.

B. Simulation of Waxman Network Topologies

We use the Boston University Representative Internet Topology Generator (BRITE) [18] to generate Waxman topologies [13] with 500, 1,000, and 2,000 nodes. In a Waxman graph, an edge that connects nodes u and v exists with a probability:

$$\Pr\left(\{u,v\}\right) = \eta \exp\left(\frac{-d(u,v)}{L\delta}\right), \tag{13}$$

where $d(u,v)$ is the distance between nodes u and v, L is the maximum distance between the two nodes, and η and δ are parameters in the range $(0,1]$. We use $\eta = 0.2$ and $\delta = 0.15$ [19] in simulation scenarios. The nodes are randomly placed and each node is connected to three other nodes using

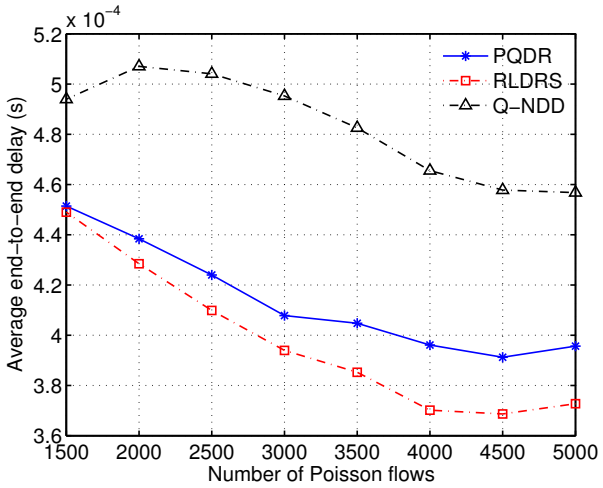


Fig. 7. Average end-to-end delay as a function of number of flows in the NSF network scenario with 64 wavelengths. In the case of RLDRS, the bursts experience the smallest end-to-end delay.

bi-directional single wavelength fiber links. A subset of nodes is randomly selected as sources and destinations of Poisson flows. Scenarios with 500, 1,000, and 2,000 nodes have 3,000, 6,000, and 12,000 Poisson traffic flows, respectively. Simulations were performed on a Dell Optiplex-790 with 16 GB memory and the Intel Core i7 2600 processor. The comparison of memory requirements and CPU usage of the PQDR algorithm, RLDRS, and the Q-NDD algorithm is shown in Table I.

The three algorithms initially have the same memory requirements. However, as the simulations proceed and tables are populated with new entries, the memory usage of PQDR grows faster compared to RLDRS and Q-NDD. For example, in the 1,000 node scenario, algorithms use 1,727 MB at the beginning of the simulations. The maximum memory requirements for PQDR, RLDRS, and Q-NDD are 1,817 MB, 1,769 MB, and 1,754 MB, respectively. This is to be expected because PQDR stores five tables while RLDRS and Q-NDD store two tables and one table, respectively. The memory usage of PQDR, RLDRS, and Q-NDD as a function of time is shown in Fig. 8. We used 70 sample points to generate the graphs. Although generating a deflection decision and table updates using PQDR involves additional steps compared to RLDRS, the results show that PQDR requires less CPU time because it performs smaller number of deflections. Q-NDD requires the least memory space and CPU time because its complexity depends on the node degree rather than network size.

V. Conclusion

In this paper, we introduced the PQDR algorithm that employs the predictive Q-routing to generate optimal deflection decisions. Deflection routing is employed in bufferless networks to reduce burst loss due to contention. The proposed algorithm recovers and reselects paths to generate viable deflection decisions.

TABLE I
COMPARISON OF MEMORY AND CPU USAGE OF PQDR, RLDRS, AND Q-NDD. THE ALGORITHMS INITIALLY USE EQUAL MEMORY. THE MEMORY USAGE OF PQDR GROWS FASTER COMPARED TO RLDRS AND Q-NDD.

Algorithm	Number of nodes	Number of links	Number of flows	Min. memory usage (MB)	Max. memory usage (MB)	Total CPU time used (mm:ss)	Total simulation time (s)
PQDR	500	1,500	3,000	566	609	4:00.65	2,857.5
	1,000	3,000	6,000	1,727	1,817	19:09.44	6,613.1
	2,000	6,000	12,000	6,342	6,526	107:33.22	17,770.8
RLDRS	500	1,500	3,000	566	588	4:04.57	2,919.5
	1,000	3,000	6,000	1,727	1,769	18:36.04	6,633.7
	2,000	6,000	12,000	6,341	6,424	110:56.76	18,069.7
Q-NDD	500	1,500	3,000	561	578	1:25.61	1,832.8
	1,000	3,000	6,000	1,727	1,754	16:44.46	4,872.8
	2,000	6,000	12,000	6,341	6,391	94:38.74	13,680.4

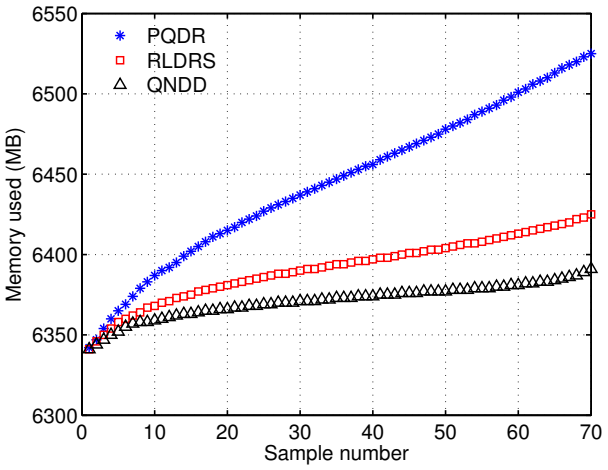


Fig. 8. Memory usage of PQDR, RLDRS, and Q-NDD in a network with 2,000 nodes. The graphs were generated by using 70 equally spaced sample points over each simulation run.

Simulation results indicate that the proposed PQDR algorithm reduces the burst loss probability up to 10% when compared to the existing reinforcement learning-based deflection routing scheme (RLDRS) and the Q-learning-based Node Degree Dependant (Q-NDD) deflection routing algorithm while requiring on average 20% fewer deflections. Performance of the PQDR algorithm is comparable to RLDRS and Q-NDD in terms of the number of hops traveled by bursts and end-to-end delay while having additional memory requirements.

REFERENCES

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *J. of Artificial Intell. Research*, vol. 4, no. 5, pp. 237–285, May 1996.

[2] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.

[3] J. A. Boyan and M. L. Littman, "Packet routing in dynamically changing networks: a reinforcement learning approach," in *Advances in Neural Inform. Process. Syst.*, vol. 6, pp. 671–678, 1994.

[4] L. Peshkin and V. Savova, "Reinforcement learning for adaptive routing," in *Proc. Int. Joint Conf. Neural Netw.*, Honolulu, HI, USA, May 2002, vol. 2, pp. 1825–1830.

[5] S. P. M. Choi and D. -Y. Yeung, "Predictive Q-routing: a memory-based reinforcement learning approach to adaptive traffic control," in *Advances in Neural Inform. Process. Syst.*, vol. 8, pp. 945–951, 1996.

[6] A. Nowe, K. Steenhaut, M. Fakir, and K. Verbeeck, "Q-learning for adaptive load based routing," in *Proc. IEEE Int. Conf. Syst., Man, and Cybern.*, San Diego, CA, USA, Oct. 1998, vol. 4, pp. 3965–3970.

[7] C. J. C. H. Watkins and P. Dayan, "Technical note, Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.

[8] C. Qiao and M. Yoo, "Optical burst switching (OBS)—a new paradigm for an optical Internet," *J. of High Speed Netw.*, vol. 8, no. 1, pp. 69–84, Mar. 1999.

[9] A. Belbekkouche, A. Hafid, and M. Gendreau, "Novel reinforcement learning-based approaches to reduce loss probability in buffer-less OBS networks," *Comput. Netw.*, vol. 53, no. 12, pp. 2091–2105, Aug. 2009.

[10] (2013, July) The ns-3 network simulator. [Online]. Available: http://www.nsnam.org/.

[11] S. Haeri, W. W-K. Thong, G. Chen, and Lj. Trajković, "A reinforcement learning-based algorithm for deflection routing in optical burst-switched networks," in *IEEE Int. Conf. Inf. Reuse and Integration*, San Francisco, USA, Aug. 2013, accepted for publication.

[12] Y. Kiran, T. Venkatesh, and C. Murthy, "A reinforcement learning framework for path selection and wavelength selection in optical burst switched networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 9, pp. 18–26, Dec. 2007.

[13] B. M. Waxman, "Routing of multipoint connections," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, Dec. 1988.

[14] E. J. Krakiwsky and D. E. Wells, *Coordinate Systems in Geodesy*. Fredericton, NB: Lecture Notes LN# 16, Department of Geodesy and Geomatics Engineering, Univ. of New Brunswick, 1971.

[15] (2013, July) The Google Earth. [Online]. Available: http://www.google.com/earth/index.html/.

[16] A. Zalesky, H. Vu, Z. Rosberg, E. W. M. Wong, and M. Zukerman, "Modelling and performance evaluation of optical burst switched networks with deflection routing and wavelength reservation," in *Proc. INFOCOM*, Hong Kong SAR, China, Mar. 2004, vol. 3, pp. 1864–1871.

[17] X. Yu, J. Li, X. Cao, Y. Chen, and C. Qiao, "Traffic statistics and performance evaluation in optical burst switched networks," *J. Lightw. Technol.*, vol. 22, no. 12, pp. 2722–2738, Dec. 2004.

[18] (2013, July) BRITE. [Online]. Available: http://www.cs.bu.edu/brite.

[19] E. W. Zegura, K. L. Calvert, and M. J. Donahoo "A quantitative comparison of graph-based models for Internet topology," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 770–783, Dec. 1997.