DUAL REINFORCEMENT Q-ROUTING: AN ON-LINE ADAPTIVE ROUTING ALGORITHM [1]

Shailesh Kumar
The University of Texas at Austin
Dept. of Elec. and Comp. Engg.
Austin, TX 78712
skumar@pine.ece.utexas.edu

Risto Miikkulainen
The University of Texas at Austin
Dept. of Computer Science
Austin, TX 78712
risto@cs.utexas.edu

ABSTRACT

This paper describes and evaluates the Dual Reinforcement Q-Routing algorithm (DRQ-Routing) for adaptive packet routing in communication networks. Each node in the network has a routing decision maker that adapts, on-line, to learn routing policies that can sustain high network loads and have low average packet delivery time. These decision makers learn based on the information they get back from their neighboring nodes as they send packets to them (forward exploration similar to Q-Routing) and the information appended to the packets they receive from their neighboring nodes (backward exploration unique to DRQ-Routing). Experiments over several network topologies have shown that at low loads, DRQ-Routing learns the optimal policy more than twice as fast as Q-Routing, and at high loads, it learns routing policies that are more than twice as good as Q-Routing in terms of average packet delivery time. Further, DRQ-Routing is able to sustain higher network loads than Q-Routing and non-adaptive shortest-path routing.

1 INTRODUCTION

In a communication network [Tanenbaum (1989)] information is transferred from one node to another as data packets. The process of sending a packet from its source node s to its destination node d is referred to as *packet routing* [Bellman (1958)]. Normally it takes multiple "hops" to transfer a packet from its source to destination node. On its way, the packet spends some time waiting in the queues of intermediate nodes while they are busy processing the packets that came earlier. Thus the delivery time of the packet, defined as the time it takes for the packet to reach its destination, depends mainly on the total time it has to spend in the queues of the intermediate nodes.

Normally, there are multiple routes that a packet could take, which means that the choice of the route is crucial to the delivery time of the packet for any (s,d) pair. If there was a global observer with current information about the queues of all nodes in the network, it would be possible to make *optimal* routing decisions: always send the packet through the route that has the shortest delivery time at the moment. In the real world, such complete, global information is not available, and the performance of the global observer is an upper bound on actual performance. Instead, the task of making routing decisions has to shared by all the nodes, each using only local information. Thus, a routing policy is a collection of local decisions at the individual nodes. When a node x receives a packet P destined for node d, it has to choose one of its neighboring nodes y such that the packet reaches its destination as quickly as possible.

The simplest such policy is the shortest-path algorithm, which always routes packets through the path with the minimum number of hops. This policy is not always good because some intermediate nodes, falling in a popular route, might have large queues. In such cases it would be better to send the packet through another route that may be longer in terms of hops but results in shorter delivery time. Hence as the traffic builds up at some popular routes, alternative routes must be chosen to keep the average packet delivery time low. This is the key motivation for

[1]THIS RESEARCH WAS SUPPORTED IN PART BY NSF UNDER GRANT IRI-9504317.

To appear in *Proceedings of Artificial Neural Networks in Engineering*, 1997

adaptive packet routing strategies that learn alternate routes through exploration as the current routing policy begins to lead to degraded performance.

Learning effective routing policies is a challenging task for several reasons:

- The goal is to optimize a global metric, the *average packet delivery time of all packets*, using only local information.
- There is no "training signal" available for directly evaluating a routing policy until the packets have reached their destination.
- When a packet reaches its destination, such a training signal could be generated, but to make it available to the nodes responsible for routing the packet, the training signal would have to travel to all these nodes, thereby consuming a lot of network resources.
- It is not known which particular decision in the sequence of routing decisions deserve credit for the performance, and how much (the credit assignment problem).

A way of efficiently exploring the network environment and continually updating the decision makers based on the local information is necessary in order to learn good routing policies.

Q-Routing [Boyan and Littman (1994); Littman and Boyan (1993)] uses the Q-learning framework [Watkins (1989)] in this task. Each node makes its routing decisions based on routing information at their neighboring nodes. The node stores a table of Q values that estimate the quality of the alternative routes. These values are updated each time the node sends a packet to one of its neighbors. This way, as the node routes packets, its Q values are gradually incorporate more global information. Such exploration has been shown capable of adapting to load changes and to perform better than the non-adaptive shortest-path routing with high loads.

This paper presents a new adaptive routing algorithm called Dual Reinforcement Q-Routing (DRQ-Routing) that combines Q-Routing with Dual Reinforcement Learning [Goetz et al. (1996)]. Dual reinforcement learning was first applied to the satellite communication problem where the two ends of the communication system co-adapt using the reinforcement signal for the other end as their own. Dual reinforcement learning adds backward exploration to the forward exploration of Q-routing, making DRQ-Routing twice as good as Q-Routing in terms of speed of adaptation (at low loads) and average packet delivery time (at high loads).

The Q-Routing algorithm is described in detail next, followed by the DRQ-Routing. The performance of the two algorithms are evaluated experimentally in section 4 and compared to the standard shortest-path algorithm. The amount of overhead generated by these algorithms is analyzed in section 5, and a number of directions for future research outlined.

2 Q-ROUTING

In Q-routing, the routing decision maker at each node x makes use of a table of values $\mathbf{Q}_x(y,d)$, where each value is an estimate, for a neighbor y and destination d, of how long it takes for a packet to be delivered to node d, if sent via neighbor y, excluding time spent in node x's queue. When the node has to make a routing decision it simply chooses the neighbor y for which $\mathbf{Q}_x(y,d)$ is minimum. Learning takes place by updating the Q values.

On sending P to y, x immediately gets back y's estimate for the time remaining in the trip, namely

$$\mathbf{Q}_y(\hat{z},d) = \min_{z \in N(y)} \mathbf{Q}_y(z,d), \tag{1}$$

where $N(n)$ denotes the set of neighbors of node n. If the packet spent q_x units of time in x's queue, then x can revise its estimate based on this feedback:

$$\Delta\mathbf{Q}_x(y,d) = \eta_f(\overbrace{\mathbf{Q}_y(\hat{z},d) + q_x}^{new\ estimate} - \overbrace{\mathbf{Q}_x(y,d)}^{old\ estimate}), \tag{2}$$

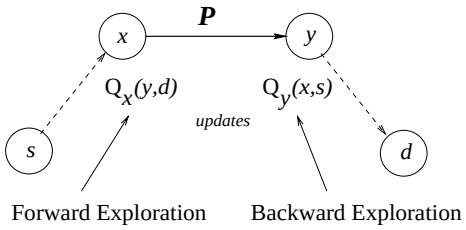

Figure 1. Forward and Backward Exploration. During forward exploration, the *sending* node x updates its $Q_x(y,d)$ value pertaining to the *remaining* path of packet P via node y. During backward exploration, the *receiving* node y updates its $Q_y(x,s)$ value pertaining to the *traversed* path of packet P via node x.

where η_f is the "learning rate".

In other words, the information about the remaining path is used to update the Q value of the sending node. Such exploration can be termed *forward exploration* (figure 2). In DRQ-Routing, the other possible direction of exploration, backward exploration, is added to this algorithm.

3 DUAL REINFORCEMENT Q-ROUTING

Dual reinforcement learning was first developed for adaptive signal predistorters in satellite communications [Goetz et al. (1996)]. Both ends of a satellite communication system have a predistorter that changes the signal, before it is transmitted, so that when the signal is received at the other end, the effects of distortion due to atmosphere have been cancelled out. Both predistorters start out equal and are adapted together while the system is performing. When the receiver gets the signal, it evaluates the performance of the transmitter's predistorter and uses the evaluation to adapt its own predistorter. Thus both predistorters learn on-line based on a training signal that is locally available.

The same idea is used to incorporate backward exploration into the Q-Routing algorithm, resulting in "Dual Reinforcement Q-Routing" (DRQ-Routing). When a node x sends a packet P to one of its neighbors, y, the packet can take along some Q value information of node x. When node y receives this packet, it can make use of this information for updating its own estimate pertaining to the neighbor x. Later when node y has to make a decision, it has the updated Q value for x. The only exploration overhead is a slight increase in the size of the packets.

Let s denote the source node of packet P, which is currently at node x. This packet carries to node y the estimated time it takes for a packet destined for node s from node x, that is $\mathbf{Q}_x(\hat{z},s)$ defined as:

$$\mathbf{Q}_x(\hat{z},s) = \min_{z \in N(x)} \mathbf{Q}_x(z,s). \tag{3}$$

With this information, node y can update its own estimate, $\mathbf{Q}_y(x,s)$, of sending a packet to node s via its neighbor x:

$$\Delta\mathbf{Q}_y(x,s) = \eta_b(\overbrace{\mathbf{Q}_x(\hat{z},s) + q_y}^{new\ estimate} - \overbrace{\mathbf{Q}_y(x,s)}^{old\ estimate}), \tag{4}$$

where η_b is the learning rate and q_y is the waiting time for this packet in node y.

In other words, the information about the path the packet has traversed so far is used to update the Q value of the receiving node. Such exploration can be termed *backward exploration* (figure 2). This way the packet is used to carry routing information from one node to the next as it moves from the source to destination.

In DRQ-Routing, both the forward exploration and backward exploration are used to update the Q values. Figure 2 illustrates these two updates as the packet P hops from node x to its neighbor y.

4 EXPERIMENTS

Experiments described in this paper use a simulated network represented by a collection of nodes and links between these nodes (figure 4). Packets destined for random nodes are periodically introduced into this network at random nodes. The number of packets introduced per unit simulation time step is called as the *network load*. Multiple packets at a node are stored in its *unbounded* FIFO queue. In one time step, each node removes the packet in front of its queue, examines the destination of this packet and uses its routing decision maker to send the packet to one of its neighboring nodes. When a node receives a packet, it either removes the packet from the network or appends it at the end of its queue, depending on whether or not this node is the destination node of the packet.

The *delivery time* of a packet is defined as the time between its introduction at the source node and its removal at the destination node. Delivery time is measured in terms of simulation time steps. Average packet delivery time, computed at regular intervals, is the average of packet delivery times of all the packets arriving at their destinations during the last interval. This measure is used to monitor the network performance *while* learning is taking place. Average packet delivery time *after* learning has settled measures the quality of the final routing policy.

The performance of DRQ-Routing was tested against Q-Routing and non-adaptive shortest-path routing, on a number of network topologies including 7-hypercube, 116-node LATA telephone network, and an irregular 6×6 grid. The results were similar in all cases; the discussion below focuses on the last one since it best illustrates adaptation. In the 6×6 irregular grid (due to [Boyan and Littman (1994); Littman and Boyan (1993)]), shown in figure 4, there are two possible ways of routing packets between the left cluster (nodes 1 through 10) and the right cluster (nodes 25 through 36): the route including nodes 12 and 25 (R_1) and the route including nodes 18 and 19 (R_2).

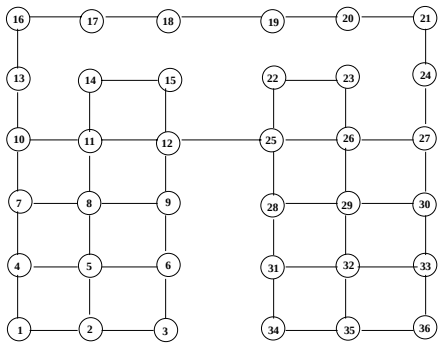

Figure 2. The 6x6 Irregular Grid. The left cluster comprises of nodes 1 through 10 and the right cluster comprises of nodes 25 through 36. The two alternative routes for traffic between these clusters are the route including the link between nodes 12 and 25 (route R_1) and the route involving the link between nodes 18 and 19 (route R_2). R_1 becomes a bottleneck with increasing loads, and the adaptive routing algorithm needs to learn to make use of R_2.

The shortest-path routing algorithm, which chooses the route with minimum hops, routes all the traffic between the left cluster and right cluster via route R_1. For low loads (0.5 to 1.5 new packets per simulation step), this routing policy works fine and throughout the simulation, the average packet delivery time is close to optimum (figure 3). At medium load levels (1.75 to 2.25 packets per simulation step), the shortest-path strategy breaks down as nodes 12 and 25 become flooded with packets. The average delivery time increases linearly with simulation time (figure 4). For high load levels (2.5 and more packets per simulation time) the flooding of node 12 and 25 takes place at an even faster rate.

The main result is the comparison between Q-Routing and DRQ-Routing. In both methods, the Q-tables at all nodes were initialized to low random values; the learning rate η_f was set at 0.7 after [Boyan and Littman (1994)], and η_b was set to 0.9. As the simulation progressed, the average packet delivery times at regular intervals of 50 simulation time steps were monitored to see the effect of learning. The results shown in figures 2, 3 and 4 for low, medium and high

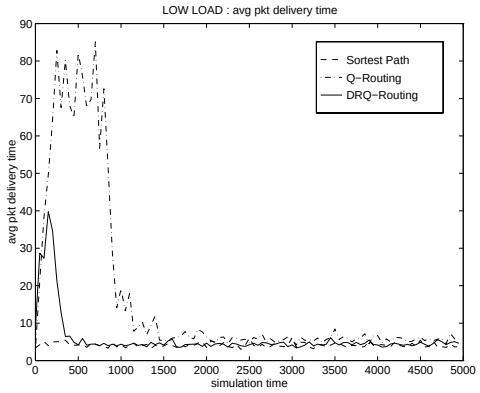

Figure 3. Learning at low network load (0.5 to 1.5 packets per simulation time.) Shortest-path routing performs best at low load. DRQ-Routing learns the optimal policy nearly three times as fast as Q-Routing. The initial hump in the curves show the learning phase after the Q-values have been initialized to random values.

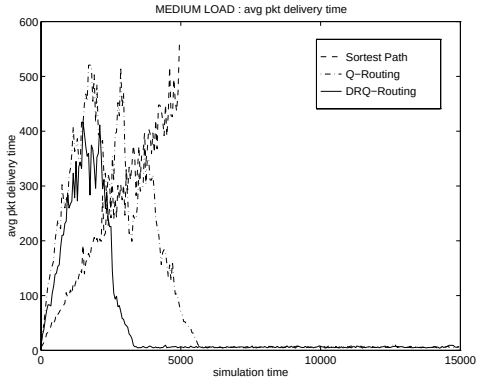

Figure 4. Learning at med ium network load (1.75 to 2.25 packets per simulation time.) DRQ-Routing learns the optim al routing policy twice as fast as the Q-Routing. The shortest-path routing suffers from severe congestion and the average packet delivery time increases linearly as the simulation progresses (the curve is not shown beyond 5000 simulation time steps.

loads, respectively, are typical single runs at these load levels. During the first few hundred time steps the average packet delivery times are small because the packets destined for distant nodes have not yet reached their destinations, and statistics are available only for packets destined for nearby nodes (with small delivery times). As distant packets start arriving, the average packet delivery time increases, while learning is still in progress. Eventually the learning converges, and each of the curves settles down indicating a stable routing policy.

At low network load levels, shortest path routing is the best policy. Q-Routing learnt a close-to-optimal routing policy, but it required almost three times the time it took DRQ-Routing to learn a slightly better policy (figure 3). At medium load levels, DRQ-Routing learnt an effective policy nearly twice as fast as Q-Routing (figure 4). At high load levels, DRQ-Routing converged to a routing policy which was twice as good, in terms of average packet delivery time, as the policy to which Q-Routing converged (figure 5).

The results are summarized in figure 6, which shows the average packet delivery times at different load levels after the learning has converged. The plots are averages over 10 simulations. The performance of the "optimal" routing policy with complete global information (section 1) is also shown for comparison. At low loads, shortest-path routing is the best routing policy. DRQ-Routing learns a slightly better routing policy at these load levels than Q-Routing. Both adaptive routing algorithms do well in the medium load levels too, but the shortest-path routing breaks down due to excessive congestion along R_1. In the high load region, Q-Routing breaks down at around 2.5 packets per simulation time while DRQ-Routing can sustain load levels up to 2.75 packets per simulation time of load.

5 DISCUSSION

Exploration makes it possible for a routing algorithm to adapt. In this paper, Q-Routing, which is based on forward exploration alone, was compared with DRQ-Routing, which uses both forward and backward exploration. Backward exploration makes adaptation more effective in two ways:

1. It leads to a two-fold increase in exploration: two Q values are updated with each hop of a packet in DRQ-Routing instead of one. Increased exploration in turn leads to increased

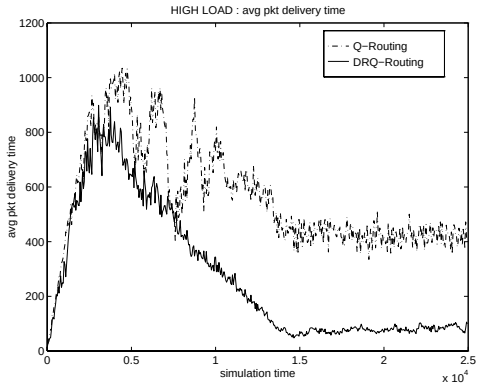

Figure 5. Learning at high network load (2.5 and more packets per simulation time.) DRQ-Routing settles down at a routing policy which is more than twice as good as that of Q-Routing. Shortest-path routing is off scale and not shown here.

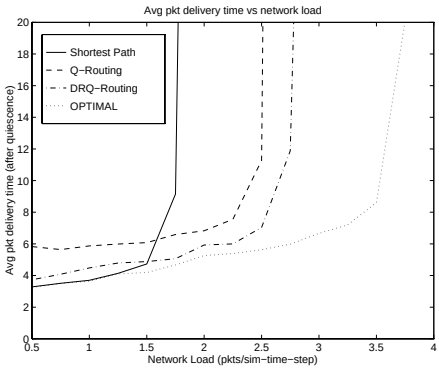

Figure 6. Average packet delivery times at different network loads. (averages of 10 test runs). Shortest path is the best routing policy for low load levels but breaks down as load levels are increased in the medium and high range. DRQ-Routing learns slightly better routing policies than Q-Routing at low and medium loads and sustains higher loads than Q-Routing. OPTIMAL shows the performance of an "optimal" router based on global information about the state of the network.

speed of learning, as can be seen in figures 3, 4, and 5 (cf. [Thrun (1992)]).

2. Backward exploration is more accurate than forward exploration. In backward exploration, information about the path already traversed is propagated, instead of *estimates* of the remaining path. At low load levels (such as shown in figure 3), DRQ-Routing is almost three times faster than Q-Routing because there is little exploration and learning depends more on accuracy. At high loads again, higher accuracy makes it possible to learn a better policy, as can be seen in figure 5.

However, exploration also adds overhead into a routing algorithm. It is important to analyze the tradeoff between the improvements and the overhead incurred. In forward exploration, when a node y receives a packet from node x, it sends back an estimate to node x. The estimate does not enter node x's queue, but instead node x waits for the estimate and processes it before the next packet in its queue. The total time of node y generating the estimate (t_g), the transmission time of this estimate over the link (t_l), and the processing time (t_p) when node x receives this estimate, constitute the *forward exploration overhead*. With fast node processors, the contribution of t_g and t_p is negligible. The main contribution comes from t_l. If $\mathbf{p}$ is the size of data packet and $\mathbf{e}$ is the size of estimate packet, then the overhead due to forward exploration is given by $\mathbf{e/p}$ (since the transmission time is proportional to the size of the packet). Because the estimate packet consists of just one Q value, this ratio is less than 0.1 %.

In backward exploration, an additional Q value is appended to the packet, increasing the packet size. The total time of node x computing and appending this Q value to the packet (t_a), the transmission time of this larger packet (t_l) and the time it takes for node y to extract this Q value and use it for updating its Q table (t_p) constitute the *backward exploration overhead*. Again with fast node processors, the contribution of t_a and t_p is negligible and the main contribution comes from t_l. The new packet is of size $(\mathbf{p} + \mathbf{e})$ where $\mathbf{p}$ is the size of the original packet and $\mathbf{e}$ is the size of Q value appended to it. The increase in packet size is again less 0.1 % and so is the overhead of using this larger packet.

Hence the total overhead due to forward and backward exploration is not significant, while the adaptability they establish improves the performance of the routing algorithm multi-fold.

Also, the improvement of DRQ-Routing over Q-Routing is by a factor of nearly two in terms of speed of adaptation at low loads and quality of routing policy at high loads, while the additional exploration overhead is just 0.1 %.

An adaptive routing algorithm should be able to learn alternative routing policies when a link goes down and should restore to the original policy when the link comes up again. In [Boyan and Littman (1994)], Q-Routing was found able to adapt when links go down. However, when the links come back up, it failed to restore to the original policy because it would not explore paths that were found defective. Similar behavior is expected for DRQ-Routing. In [Choi and Yeung (1996)] however, an extension of Q-Routing called Predictive Q-Routing, was proposed which keeps track of the last update time and the best Q values seen so far. Based on this information, PQ-Routing is able to explore paths that have been inactive for a long time, and thereby is able to restore the previous policy. The same idea could be extended to DRQ-Routing without changing the rest of the algorithm. The speed of restoration is again expected to be higher in DRQ-Routing than in the current PQ-Routing because of enhanced exploration in DRQ-Routing.

Unbounded FIFO queues were used in the current simulations for simplicity. In the real world, the queue buffers of the network routers are finite, leading to possible congestion in heavily loaded parts of the networks. Extension of the DRQ-Routing to address the problem of finite buffer networks is another important future direction. This extension would make DRQ a more realistic routing strategy that does not only route optimally, but can also sustain higher loads in finite buffer networks to avoid congestion and adapt quickly to changing network loads and traffic patterns.

6 CONCLUSION

In this paper a new adaptive network routing algorithm, DRQ-Routing was presented. It combines Q-Routing and Dual Reinforcement learning to get increased explorative capabilities. DRQ-Routing was shown to learn a better routing policy more than twice a fast as Q-Routing at low loads. At high loads, the routing policy learnt by DRQ-Routing performs more than twice as good as Q-Routing in terms of average packet delivery time. Moreover, DRQ-Routing can sustain higher load levels than Q-Routing and shortest-path routing. The additional overhead of adding this exploration being less than 0.1 %, DRQ-Routing is an efficient and practically viable adaptive network routing algorithm.

REFERENCES

Bellman, R., 1958, "On a Routing Problem," *Quarterly of Applied Mathematics*, Vol. 16, pp. 87–90.

Boyan, J. A., Littman, M. L., 1994, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach," In Cowan, J., Tesauro, G., Alspector, J., editors, *Advances in Neural Information Processing Systems 6*. MIT Press, Cambridge, MA.

Choi, S. P. M., Yeung, D.-Y., 1996, "Predictive Q-Routing: A Memory-based Reinforcement Learning Approach to Adaptive Traffic Control," In Touretzky, D. S., Mozer, M. C., Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pp. 945–951. MIT Press, Cambridge, MA.

Goetz, P., Kumar, S., Miikkulainen, R., 1996, "On-Line Adaptation of a Signal Predistorter through Dual Reinforcement Learning," Proc. Machine Learning: Proceedings of the 13th Annual Conference (Bari, Italy).

Littman, M., Boyan, J. A., 1993, "A Distributed Reinforcement Learning Scheme for Network Routing," Proc. Proceedings of the First Internation Workshop on Applications of Neural Networks to Telecommunications, Hillside, New Jersy, pp. 45–51. Erlbaum.

Tanenbaum, A., 1989, *Computer Networks* Prentice Hall, second edition.

Thrun, S. B., 1992, "The Role of Exploration in Learning Control," In White, D. A., Sofge, D. A., editors, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches.* Van Nostrand Reinhold, New York.

Watkins, C. J. C. H., 1989, "Learning from Delayed Rewards," Ph.D. thesis, University of Cambridge, England.