CONFIDENCE-BASED Q-ROUTING: AN ON-LINE ADAPTIVE NETWORK 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 how confidence values can be used to improve the quality of exploration in Q-Routing for adaptive packet routing in communication networks. In Q-Routing 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 maintain their view of the network in terms of Q values which are updated as the routing takes place. In Confidence based Q-Routing (CQ-Routing), the improved implementation of Q-Routing with confidence values, each Q value is attached with a confidence (C value) which is a measure of how closely the corresponding Q value represents the current state of the network. While the learning rate in Q-Routing is a constant, the learning rate in CQ-Routing is computed as a function of confidence values of the old and estimated Q values for each update. If either the old Q value has a low confidence or the estimated Q value has a high confidence, the learning rate is high. The quality of exploration is improved in CQ-Routing as a result of this variable learning rate. Experiments over several network topologies have shown that at low and medium loads, CQ-Routing learns the adequate routing policies significantly faster than Q-Routing, and at high loads, it learns routing policies that are significantly better than those learned by Q-Routing in terms of average packet delivery time. CQ-Routing is able to sustain higher network loads than Q-Routing, non-adaptive shortest-path routing and adaptive Bellman-Ford Routing. Finally, CQ-Routing was found to adapt significantly faster than Q-Routing to changes in network topology.

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 is 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 $P(d)$ 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 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: (1) The goal is to optimize a global metric, the *average packet delivery time of all packets*, using only local information. (2) There is no "training signal" available for directly evaluating a routing policy until the packets have reached their

[1]THIS RESEARCH WAS SUPPORTED IN PART BY NATIONAL SCIENCE FOUNDATION UNDER GRANT #IRI-9504317. (THIS PAPER IS TO APPEAR IN THE *PROCEEDINGS OF THE ARTIFICIAL NEURAL NETWORKS IN ENGINEERING CONFERENCE*, 1998)

destination. (3) 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 network resources. (4) 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).

Q-Routing [Boyan and Littman (1994)] uses the Q-learning framework [Watkins and Dayan (1989)] for this task. Each node makes its routing decisions based on the local routing information, represented as 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 a node routes packets, its Q values gradually incorporate more global information. Such exploration was shown capable of adapting to load changes and to perform better than the non-adaptive shortest-path routing with high loads. When a Q value is not updated for a long time, it does not necessarily reflect the current state of the network and hence a routing decision based on such unreliable Q value will not be accurate. Through exploration, however, in Q-Routing, the Q values are updated but depending on the traffic pattern, and load levels, only a few Q values are current while most of the Q values in the network are unreliable. The update rule in Q-Routing does not take into account the reliability of the estimated or updated Q value. In other words, in Q-Routing all update rules use the same learning rate. A better alternative would be to update those Q values that are more reliable by higher amount than the Q values that are less reliable. That is, to have different learning rates.

This paper presents an improvement over Q-Routing, namely Confidence based Q-Routing (CQ-Routing), which addresses this issue by incorporating a confidence measure (C value) with each Q value. The C value denotes how closely the corresponding Q value represents the current state of the network. As the time since the last update of a Q value increases, its C value decreases exponentially. The novelty of CQ-Routing is that the C values are used to compute the learning rate for the Q value update rule, while the learning rate was fixed in Q-Routing. The C values are themselves updated to depict changes in the reliability of Q values. The quality of exploration in Q-Routing is thereby improved by adapting Q values with smaller C values more than those with higher C values. As a result of this improvement in quality of exploration, the speed of adaptation at low and medium loads increases significantly and the quality of the final policy learned is also superior to that in Q-Routing. CQ-Routing also adapts faster to change in network topology and can sustain higher load levels than Q-Routing.

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

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

where η is the "learning rate" and δ is the transmission delay over the link between nodes x and y (assumed same for all links). Note that the learning rate is constant for all Q value updates and does not take into account any information about how reliable (or accurate) the new estimates and old Q values are. The CQ-Routing algorithm discussed next incorporates this improvement into Q-Routing.

3 CONFIDENCE-BASED Q-ROUTING

The quality of the routing policy depends mainly on how closely the Q-values in the network represent its current state. Therefore, they must be continuously updated. Depending on the network load and traffic patterns, some of the Q-values may not be updated for a long time. Decisions based on such unreliable Q values is going to be inadequate.

3.1 Confidence Measure

In order to quantify the amount of reliability in the Q values, *confidence measures* are introduced in Q-Routing. Each Q value $\mathbf{Q}_x(y,d)$ in the network is associated with a measure of confidence $\mathbf{C}_x(y,d)$, which is a real number between 0 and 1. A value of 1 means that there is maximum confidence in the corresponding Q value. This Q value has recently been updated and is therefore likely to reflect the current state of the network. A value of 0, on the other hand, means that the corresponding Q value is completely random. It has never been updated and does not necessarily reflect anything about the current state of the network.

With this interpretation, the base case C values follow directly from the base case Q values that is, $\mathbf{Q}_x(y,y) = \delta$ and $\mathbf{C}_x(y,y) = 1$, which implies that there is full confidence in all Q values of the form $\mathbf{Q}_x(y,y)$ since all these Q values are held constant, the corresponding C values are also held constant. All the other Q values are initialized to low random values while the corresponding C values are initialized to 0. These Q values are being learned as routing takes place.

3.2 Using C values

In Q-Routing, the Q value update rule (equation 2) makes no use of any information regarding how reliable are the Q values, both the one that is updated and the estimated. As a result, the quality of exploration is not good. In order to incorporate this reliability information into the update rules, a measure of this reliability for each Q value is required. The confidence values in CQ-Routing provide this reliability measure. The amount of update in a Q value depends on the learning rate. Hence one way of incorporating the reliability information into the update rules is to compute the learning rate η as a function of the confidence values such that the less reliable the Q value being updated is, or the more reliable the estimated Q value is, the higher is the learning rate.

When node x sends a packet $P(d)$ to its neighbor y, it gets back not only the best estimate of node y for the remaining part of $P(d)$'s journey, namely $\mathbf{Q}_y(\hat{z},d)$, but also the confidence value associated with this Q value, namely, $\mathbf{C}_y(\hat{z},d)$. Now when node x updates its $\mathbf{Q}_x(y,d)$ value, it first computes the learning rate η which depends on both $\mathbf{C}_x(y,d)$ (=$\mathbf{C}_{old}$) and $\mathbf{C}_y(\hat{z},d)$ (=$\mathbf{C}_{est}$). The update rule (2) is now given by:

$$\Delta\mathbf{Q}_x(y,d) = \eta(\mathbf{C}_{old}, \mathbf{C}_{est})(\overbrace{\mathbf{Q}_y(\hat{z},d) + q_y + \delta}^{new\ estimate} - \mathbf{Q}_x(y,d)) \tag{3}$$

The *learning rate function,* η($\mathbf{C}_{old}$, $\mathbf{C}_{est}$) is chosen based on the following rule:

Rule 1: *Learning rate should be* **high** *if either (or both):*

- *Confidence in the* **old** Q value *is* **low** *or*
- *Confidence in the* **estimated** Q value *is* **high**.

One of the simplest and also most effective learning rate functions implementing rule 1 is given by:

$$\eta(\mathbf{C}_{old}, \mathbf{C}_{est}) = \max(\mathbf{C}_{est}, 1 - \mathbf{C}_{old}) \tag{4}$$

3.3 Updating the Confidence Values

The confidence values (except for the base cases) are updated over time, in order to reflect the reliability of the corresponding Q values. Depending on whether a Q value was updated in the last time step or not, different update rules for the corresponding C value apply:

Rule 2(a): *If their corresponding Q values are not updated in the last time step, every C value (except for the base case values) decay with some decay constant* $\lambda \in$ *(0,1).*

Rule 2(b): *If a Q value is updated in the last time step, then the corresponding C value is updated based on the C values corresponding to the Q values used in the Q value update.*

$$\mathbf{C}_{upd} = \begin{cases} \lambda\mathbf{C}_{old} & \text{Rule 2(a)} \\ \mathbf{C}_{old} + \eta(\mathbf{C}_{old}, \mathbf{C}_{est})(\mathbf{C}_{est} - \mathbf{C}_{old}) & \text{Rule 2(b)} \end{cases} \tag{5}$$

The C value update equation (5) uses the old C value of the Q value being updated and the C value of the Q estimate coming from neighboring node to first compute the learning rate. This learning rate is used in the Q value update rule and the same learning rate is used to update the C value. If this learning rate is high, the confidence in the updated Q value will be closer to that of the estimated Q value's confidence.

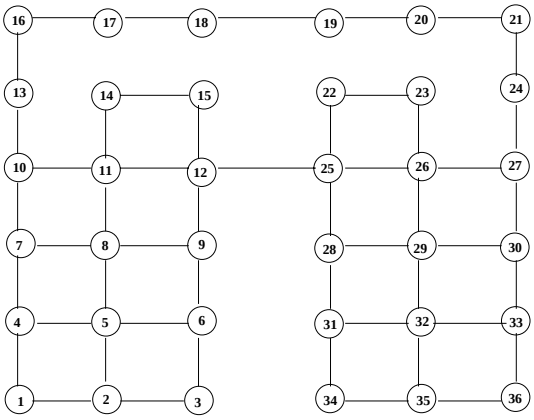

Figure 1. The 6x6 Irregular Grid.

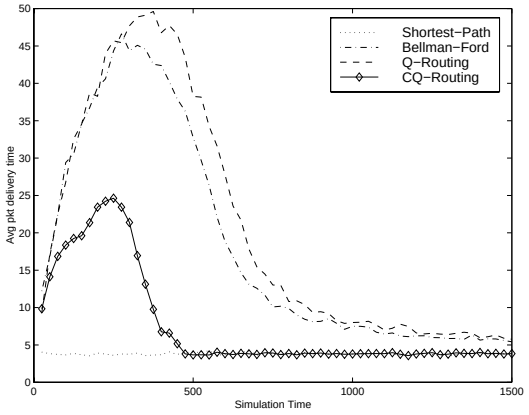

Figure 2. Learning at low network load

4 EXPERIMENTS

Experiments described in this paper use a simulated network represented by a collection of nodes and links between these nodes. 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 (at a steady state) measures the quality of the final routing policy.

The performance of CQ-Routing was tested against Q-Routing, non-adaptive shortest-path routing and Bellman-Ford routing, on a number of network topologies including 128 node 7-D hypercube, 116-node LATA telephone network, and an irregular 6×6 grid (shown in figure 1) [Boyan and Littman (1994)]. 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 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).

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 (1.25 packets per simulation step), this routing policy works fine and throughout the simulation, the average packet delivery time is close to optimum (figure 2). At medium load levels (2.0 packets per simulation step), the shortest-path strategy breaks down as nodes 12 and 25 become flooded with packets (figure 3) as the number of packets coming into these nodes was more than the number of packets they could handle. The average delivery time increases linearly with simulation time. For high load levels (2.75 packets per simulation time) the flooding of node 12 and 25 takes place at an even faster rate (figure 4).

The main result, however, is the comparison between Q-Routing and CQ-Routing. In both methods, the Q-tables at all nodes were initialized to low random values (except the base case values). The learning rate η in Q-Routing (equation 2) was set at 0.85 (best results obtained at this value). The value of the decay constant λ for which CQ-Routing performed best was 0.95. As the simulation progressed, the average packet delivery times at regular intervals of 20 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 loads, respectively, are averages over 50 test runs each with random starts. 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. Both Q-Routing and Bellman-

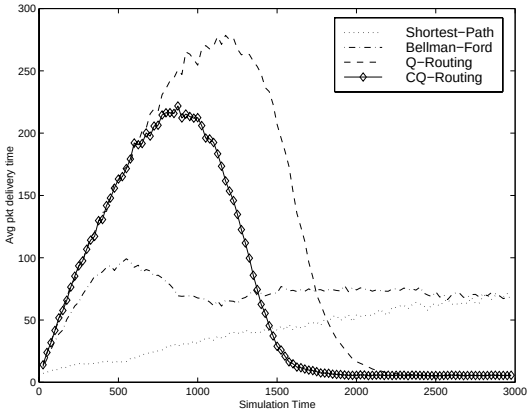

Figure 3. Learning at medium network load

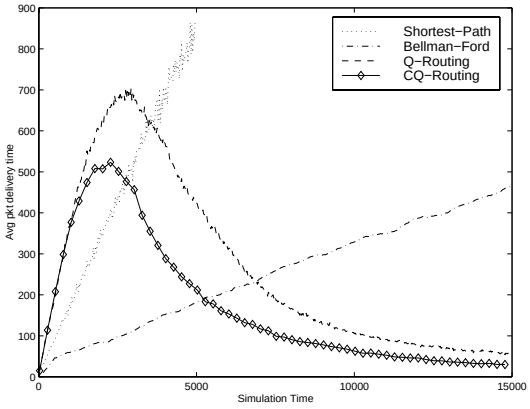

Figure 4. Learning at high network load

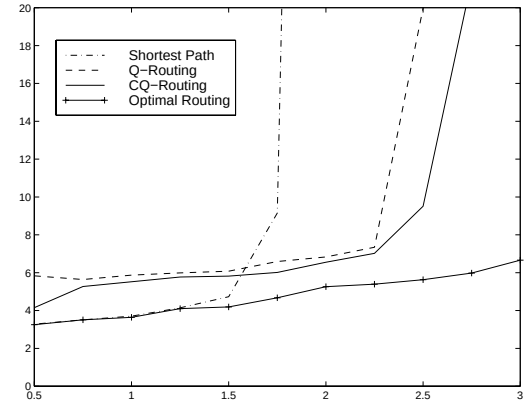

Figure 5. Avg. packet delivery times at different loads

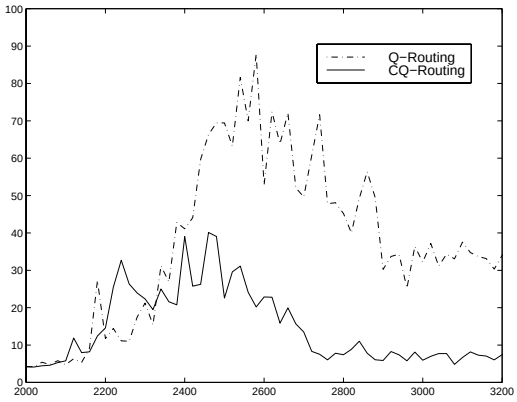

Figure 6. Adaptation to changes in network topology.

Ford Routing learned comparetively poor routing policy, and took significantly more time than CQ-Routing (figure 2). At medium load levels, again CQ-Routing learned an effective policy significantly faster than Q-Routing while Bellman-Ford routing settles learns an inferior policy (figure 3). At high load level too, CQ-Routing converged to a significantly superior routing policy than Q-Routing while Bellman-Ford routing failed showed congestion symptoms. (figure 4).

The results are summarized in figure 5, 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. CQ-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 2.25 packets per simulation time while CQ-Routing can sustain load levels up to 2.5 packets per simulation time of load.

Apart from being able to "learn" a routing policy from scratch, as shown in previous experiments, an adaptive routing algorithm should be able to "adapt" to changes in network topology *after* it has already learned an adequate routing policy for the current topology and load level. Results for adaptation to change in topology are given next. A link was added between nodes 3 and 34 in the 6×6 grid topology (figure 1) for these experiments. The routing algorithms were first allowed to learn an adequate routing policy for the new network at a load level of 2.0 pkts/step until they converged (2000 time steps). At time step 2000, the link between node 12 and 25 was removed. This link removal was simulated by reinitializing the Q tables of node 12 and 25 such that $Q_{12}(25,*)$ and $Q_{25}(12,*)$ were all set to *Infinite Cost* and the corresponding routing tables were also updated accordingly. This guaranteed that packets were not routed through the link as would be the case when the link goes down. The C values were not changed.

Figure 6 shows the average packet delivery time between time steps 2000 and 3200. As soon as the link between nodes 12 and 25 went down at time 2000, the average packet delivery time of both algorithms

started increasing. They try to adapt to the change in network topology and settle at the policy after 3000 time steps. CQ-Routing learns a better routing policy faster than Q-routing, performing almost at the same level as before the link went down. The main reason being that the alternative routes from nodes 12 and 25 which for destination in the other cluster were learned faster in CQ-Routing due to improved quality of exploration.

5 DISCUSSION AND FUTURE WORK

Exploration makes it possible for a routing algorithm to adapt. In this paper, the *quality of exploration* in Q-Routing, was improved by incorporating confidence measures in the Q value update rule. The variable learning rate thereof improved the quality of exploration in two ways (1) when the Q values were unreliable they were updated more and (2) when the new estimates had high confidence, they had a larger effect.

Exploration also adds overhead into a routing algorithm. It is important to analyze the tradeoff between the improvements and the overhead incurred. One obvious overhead is the additional confidence table, increasing the memory requirements of each node by a factor of 2. As the cost of memory is not high, this overhead is not significant. The main overhead is in the exploration of routing information. Exploration overhead in Q-Routing are discussed in detail in [Kumar and Miikkulainen (1997); Kumar (1998)]. In CQ-Routing an additional C value is appended to the estimate sent by node y back to node x. If we assume same number of bits for both the estimate and C value, then the additional overhead is less than 0.2% [Kumar (1998)], which is not significant.

An extension of Q-Routing, Dual Reinforcement Q-routing was proposed in [Kumar and Miikkulainen (1997); Kumar (1998)], which improved the *quantity of exploration* by additional backward exploration. Improvements in both quality and quantity of exploration are orthogonal to each other and combining CQ-Routing with the DRQ-routing should further improve the performance of CQ-routing. In [Choi and Yeung (1996)], 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 to predict the Q value at current time. The same idea could be extended to CQ-Routing without changing the rest of the algorithm, except that all the learning rates in PQ-Routing can be computed based on the confidence values instead of being constants. The speed of adaptation is again expected to improve because of improved quality of exploration. These combined algorithms constitute the main direction for future work.

Another direction is more realistic simulations. For example, 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 CQ-Routing to address the problem of finite buffer networks is an important future direction. This extension would make CQ a more realistic routing strategy that does not only route efficiently, but can also sustain higher loads in finite buffer networks to avoid congestion and adapt quickly to changing network loads and traffic patterns. Another future work could be the use of heterogeneous networks where the speed of various links and routers could be different.

6 CONCLUSION

In this paper the quality of exploration in Q-Routing is improved by incorporating confidence values, a measure of reliability of corresponding Q values. CQ-Routing, the implementation of this improvement, was shown to learn a better routing policy significantly faster than Q-Routing at low load and medium loads. At high loads, the routing policy learned by CQ-Routing performs more than twice as good as Q-Routing in terms of average packet delivery time. Moreover, CQ-Routing can sustain higher load levels than Q-Routing and shortest-path routing. CQ-Routing was also found to adapt to changes in network topology significantly faster than Q-Routing. The additional overhead of adding C values and maintaining them does not add significantly to the exploration overhead of Q-Routing.

REFERENCES

Bellman, R. E., 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.

Kumar, S., 1998, "Confidence based Dual Reinforcement Q-Routing: An On-line Adaptive Network Routing Algorithm," Master's thesis, Department of Computer Sciences, The University of Texas at Austin, Austin, TX-78712, USA Tech. Report AI98-267.

Kumar, S., Miikkulainen, R., 1997, "Dual Reinforcement Q-Routing: An On-line Adaptive Routing Algorithm," Proc. Proceedings of the Artificial Neural Networks in Engineering Conference.

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

Watkins, C. J. C. H., Dayan, P., 1989, "Q-Learning," *Machine Learning*, Vol. 8, pp. 279–292.