INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY
THIRUVANANTHAPURAM

Reinforcement Learning Case Study Report

Akhil P M (SC14M044)

Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach

A packet Routing Policy in general answers the question: To which adjacent node the current node should send its packets to get the final destination as early as possible. The performance measure of a policy is its total time to deliver a packet. We can easily observe all the essential characteristics in this problem to model it as a reinforcement learning task, especially in the case of dynamically changing networks.

- A routing policy should balances minimizing the number of "hops" a packet will take with the possibility of congestion along popular routes. The congestion rate varies dynamically in a network, thus algorithms with fixed behaviour will not fit for this problem, it should have good adaption capability.
- The learning should be continual and online.
- The algorithm should be robust enough to handle irregular and dynamically changing network topology.

A Reinforcement Learning based algorithm definitely meets all these requirments and hence it is a natural fit for this problem. Moreover, it uses only local information for building the routing table, thereby reducing the computational overhead and increasing the effectiveness in a network with high load[1].

1. Modelling the Routing problem as an RL based Learning Task

This RL based Learning Algorithm is called Q-routing because its underlying principles are same as that of Q-Learning. Consider a network topology shown in figure 1. Here each node in the network can be considered as a

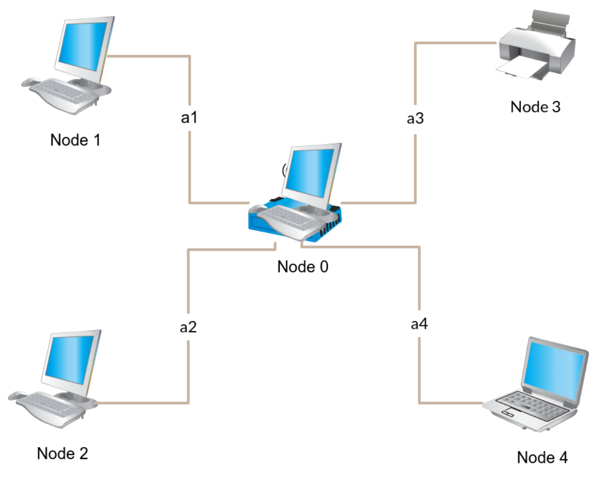


Figure 1: A simple network topology

state in the Reinforement Learning terminology. The actions are links to the adjacent nodes from the current node. In this case from Node 0 we can take 4 actions namely a1, a2, a3, a4. Unlike the classical Q-learning algorithm, here we don't have any reward function to give immediate reward for every action the agent takes. The reason is because

the performance of a policy can only be measured if the packets are reached at their intended destination, so there is no intermediate evaluation for the action we take[1].
Let $Q_x(d, y)$ be the time that a node x estimates it takes to deliver a packet P with destination d by routing it through the neighbouring node y(this includes the time that P spends in node x‘s queue for processing). When the packet reaches the node y, y immediately sends back its estimate of the Q-value to x(ie, the remaining time in the trip starting from node y) namely

$$t = \min_{z \in \text{neighbours of y}} Q_y(d, z) \tag{1}$$

If the packet spends q units of time in x’s queue and s units of time in transmission between nodes x and y, then node x can revise its Q value as follows[1]:

$$Q_x^{new}(d, y) = q + s + t$$

$$\Delta Q_x(d, y) = \alpha(\overbrace{q + s + t}^{\text{new estimate}} - \overbrace{Q_x(d, y)}^{\text{old estimate}}) \tag{2}$$

where α is the learning rate parameter. With this estimated $\Delta Q_x(d, y)$ value, node x can revise its $Q_x(d, y)$ as follows:

$$Q_x(d, y) = Q_x(d, y) + \Delta Q_x(d, y) \tag{3}$$

The objective of the algorithm is to minimize $Q_x(d, y)$.

2. Q-Routing Algorithm

The $Q_x(d, y)$ value is learned by the algorithm is stored in the form of a table. The final output of the algorithm is the $Q_x(d, y)$ table[2].

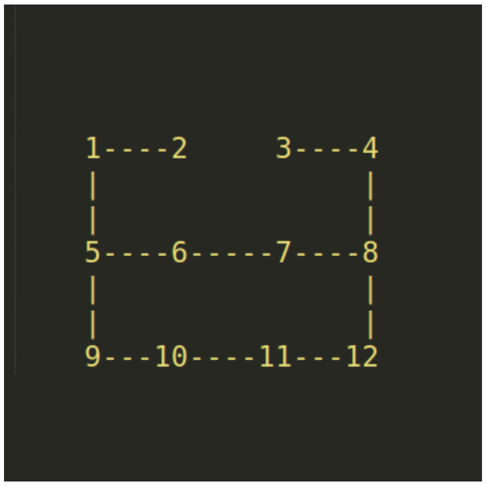
1. Generate a packet, insert it to a random node(say x), assign a random destination.
2. When a node x receives a packet, it sends that to y, where y is

   $$y = \arg \min_{z \in \text{neighbours of x}} \{Q_x(d, z) + s\}$$

   Here s is the transmission time between nodes x and z.
3. When the packet reaches y, y sends back its estimate of the $Q_y(d, z)$ value(say t) to x.

   $$t = \min_{z \in \text{neighbours of y}} Q_y(d, z)$$

4. $\Delta Q_x(d, y) = \alpha(q + s + t - Q_x(d, y))$
   where q is the time spend by packet P in x’s queue and α is the learning rate.
5. $Q_x(d, y) = Q_x(d, y) + \Delta Q_x(d, y)$
   terminate the process if $Q_x(d, y)$ converges. Otherwise goto next step.
6. if y is the destination d, goto step 1. Otherwise Assign x=y, goto step 2.



```
1----2     3----4
|               |
|               |
5----6-----7----8
|               |
|               |
9---10----11---12
```

Figure 2: A simple network with 12 nodes

3. Experimental Analysis

For the experimental study we considered a network structure as shown in figure 2. Simulation of the network is done as follows

- initial $Q_x(d, y)$ values are generated randomly for each nodes.
- the adjacent nodes of the destination node are set with minimum cost of delivery.
- the transmission delays are generated randomly from a uniform distribution.
- the process terminates when there is no change in next node chosen for all nodes in the network over ten successive iterations.

This simulation gives low traffic path for one specific destination node at a time, hence to determine all such paths for all nodes we have to run it many times with different destination nodes. The path learned by the algorithm for destination node 4 is

final route to destination 4
next node for [1] :5
next node for [2] :1
next node for [3] :4
next node for [5] :6
next node for [6] :7
next node for [7] :8
next node for [8] :4
next node for [9] :5
next node for [10] :11
next node for [11] :12
next node for [12] :8

final route to destination 12
next node for [1] :5
next node for [2] :1
next node for [3] :4

```
next node for [4] :8
next node for [5] :9
next node for [6] :7
next node for [7] :8
next node for [8] :12
next node for [9] :10
next node for [10] :11
next node for [11] :12
```

References

[1] Justin A. Boyan, Michael L. Littman, "Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach", *Advances in Neural Information Processing Systems* 6, 671-678, 1994.

[2] Suciu Andrei, "A Survey on Network Routing and Reinforcement Learning", 2000.