Call Admission Control and Routing in Integrated Service Networks Using Reinforcement Learning

Peter Marbach
LIDS
MIT, Room 35 -307
Cambridge, MA, 02139
email: marbach@mit.edu

Oliver Mihatsch[1]
Siemens AG
Corporate Technology, ZT IK 4
D-81730 Munich, Germany
email: oliver.mihatsch@mchp.siemens.de

John N. Tsitsiklis
LIDS
MIT, Room 35-209
Cambridge, MA, 02139
email: jnt@mit.edu

Abstract

In integrated service communication networks, an important problem is to exercise call admission control and routing so as to optimally use the network resources. This problem is naturally formulated as a dynamic programming problem, which, however, is too complex to be solved exactly. We use methods of reinforcement learning (RL), together with a decomposition approach, to find call admission control and routing policies. We compare the performance of our policy with a commonly used heuristic policy.

1 Introduction

The call admission control and routing problem arises in the context where a telecommunication provider wants to sell its network resources to customers in order to maximize long term revenue. Customers are divided into different classes, called service types. Each service type is characterized by its bandwidth demand, its average call holding time and the immediate reward the network provider obtains, whenever a call of that service type is accepted. The control actions for maximizing the long term revenue are to accept or reject new calls (*Call Admission Control*) and, if a call is accepted, to route the call appropriately through the network (*Routing*). The problem is naturally formulated as an average reward dynamic programming problem, which, however, is too complex to be solved exactly. We use the methodology of reinforcement learning (RL) to approximate the differential reward of dynamic programming. Furthermore, we pursue a decomposition approach, where the network is viewed as consisting of link processes, each having its own differential reward. This has the advantage, that it allows a decentralized implementation of the training methods of RL and a decentralized implementation of the call admission control and routing policies. Our method learns call admission control and routing policies which outperform the commonly used heuristic "Open-Shortest-Path-First" (OSPF) policy.

In some earlier related work, we applied RL to the call admission problem for a single communication link in an integrated service environment. We found that in this case, RL methods performed as well, but no better than, well-designed heuristics. Compared with the single link problem, the addition of routing decisions makes the network problem more complex and good heuristics are not easy to derive.

2 Call Admission Control and Routing

We are given a telecommunication network consisting of a set of nodes $\mathcal{N} = \{1, \ldots, N\}$ and a set of links $\mathcal{L} = \{1, \ldots, L\}$, where link l has a a total capacity of $B(l)$ units of bandwidth. We support a set $\mathcal{M} = \{1, \ldots, M\}$ of different service types, where a service type m is characterized by its bandwidth demand $b(m)$, its average call holding time $1/\nu(m)$ (here we assume that the call holding times are exponentially distributed) and the immediate reward $c(m)$ we obtain, whenever we accept a call of that service type. A link can carry simultaneously any combination of calls, as long as the bandwidth used by these calls does not exceed the total bandwidth of the link (*Capacity Constraint*). When a new call of service type m requests a connection between a node i and a node j, we can either reject or accept that request (*Call Admission Control*). If we accept the call, we choose a route out of a list of predefined routes (*Routing*). The call then uses $b(m)$ units of bandwidth on each link along that route for the duration of the call. We can, therefore, only choose a route, which does not violate the capacity constraints of its links, if the call is accepted. Furthermore, if we accept the call, we obtain an immediate reward $c(m)$. The objective is to exercise call admission control and routing in such a way that the long term revenue obtained by accepting calls is maximized.

We can formulate the call admission control and routing problem using dynamic programming (e. g. Bertsekas, 1995). We will use the following notation: S denotes the

[1] Author to whom correspondence should be addressed.

state space, Ω is the set of all possible events, and $\mathcal{U}$ is the set of decision/control actions. The state $x_t \in \mathcal{S}$ at time t consists of a list for each route, indicating how many calls of each service type are currently using that route. Events $\omega \in \Omega$ which incur state transitions, are arrivals of new calls and call terminations. The decision/control $u_t \in \mathcal{U}$ applied at the time t of an arrival of a new call is to decide, whether to reject or accept the call, and, if the call is accepted, how to route it through the network. The objective is to determine a policy

$$\mu : \mathcal{S} \times \Omega \to \mathcal{U}$$

that assigns decisions to a state and event, so as to maximize the average reward

$$v(\mu) = \lim_{N \to \infty} \frac{1}{E\{t_N\}} E \left\{ \sum_{k=0}^{N-1} g(x_{t_k}, \omega_k, u_{t_k}) \mid \mu \right\}$$

associated with policy μ. $E\{\cdot\}$ is the expectation operator, t_k is the time when the kth event happens, and $g(x_{t_k}, \omega_k, u_{t_k})$ is the immediate reward associated with the state x_{t_k}, the event ω_k and the decision u_{t_k}.

A policy μ^* is optimal if

$$v(\mu^*) \geq v(\mu)$$

for every other possible policy μ. We denote the average reward associated with an optimal policy μ^* as v^*.

3 Reinforcement Learning Solution

RL methods solve optimal control (or dynamic programming) problems by learning good approximations to the optimal differential reward h^*, given by the solution to the Bellman optimality equation for average reward problems. For the call admission control and routing problem, the Bellman optimality equation takes the following form

$$\begin{aligned} & v^* E_\tau\{\tau \mid x\} + h^*(x) = \\ & \qquad E_\omega \left\{ \max_{u \in U(x)} \left[g(x, \omega, u) + h^*(x') \right] \right\}, \quad x \in \mathcal{S}, \\ & h^*(\hat{x}) = 0, \end{aligned}$$

where v^* is the optimal reward, $U(x)$ is the set of control actions available in the current state x, τ is the time when the first event ω occurs, x' is the successor state and $\hat{x}$ is the state representing the empty network. Note that x' is a deterministic function of the current state x, the control u and the event ω. In our setting, there is an unique solution to the Bellman optimality equation.

RL uses a compact representation $\tilde{h}(\cdot, \theta)$ and $\tilde{v}$ to learn and store an estimate of $h^*(\cdot)$ and v^*, respectively. Temporal-difference (TD(λ)) algorithms for average reward problems (Tsitsiklis & VanRoy, 1997) can be used to train $\tilde{h}(\cdot, \theta)$ and $\tilde{v}$. In the *optimistic* version of these algorithms, on each event ω_k, $\tilde{h}(\cdot, \theta_k)$ is both used to make decisions and to update the parameter vector θ_k. In the call admission control and routing problem, one has only to choose a control action when a new call requests a connection. In such a case, $\tilde{h}(\cdot, \theta)$ is used to choose a control action according to the formula

$$u = \arg \max_{u \in U(x)} \left[g(x, \omega, u) + \tilde{h}(x', \theta) \right] \qquad (1)$$

This can be expressed in words as follows.

Decision Making: When a new call requests a connection, use $\tilde{h}(\cdot, \theta)$ to evaluate, for each permissible route, the successor state x' we transit to, when we choose that route, and pick a route which maximizes that value. If the sum of the immediate reward and the value associated with this route is higher than the value of the current state, route the call over that route; otherwise reject the call.

Usually, RL uses a global feature extractor $f(x)$ to form an approximate compact representation of the state of the system, which forms the input to a function approximator $\tilde{h}(\cdot, \theta)$. Using the temporal-difference algorithm TD(0), the update at the kth event takes the following form

$$\begin{aligned} \theta_k &= \theta_{k-1} + \gamma_k d_k \nabla_\theta \tilde{h}(f(x_{t_{k-1}}), \theta_{k-1}), \\ \tilde{v}_k &= \tilde{v}_{k-1} + \eta_k (g(x_{t_k}, \omega_k, u_{t_k}) - (t_k - t_{k-1}) \tilde{v}_{k-1}), \end{aligned}$$

where

$$\begin{aligned} d_k = {} & g(x_{t_k}, \omega_k, u_{t_k}) - (t_k - t_{k-1}) \tilde{v}_{k-1} + \\ & \tilde{h}(f(x_{t_k}), \theta_{k-1}) - \tilde{h}(f(x_{t_{k-1}}), \theta_{k-1}), \end{aligned}$$

γ_k and η_k are small step size parameters, and u_{t_k} is the control action chosen according to the decision making rule (1) described above.

Here we pursue an approach where we view the network as being composed of link processes. Furthermore, we decompose immediate rewards $g(x_{t_k}, \omega_k, u_{t_k})$ associated with the kth event, into link rewards $g_l(x_{t_k}, \omega_k, u_{t_k})$ such that

$$g(x_{t_k}, \omega_k, u_{t_k}) = \sum_{l=1}^{L} g_l(x_{t_k}, \omega_k, u_{t_k}).$$

We then define, for each link l, $\tilde{h}_l(f_l(x)), \theta_l$, and $\tilde{v}_l$, which are interpreted as an estimate of the differential reward, and the average reward, respectively, associated with link l. Here, f_l defines a local feature, which forms the input to the differential reward associated with link l. We use the functions $\tilde{h}_l(f_l(x)), \theta_l$, and $\tilde{v}_l$, to obtain an approximation of $h^*(x)$, and v^*, respectively, by computing the sums

$$\sum_{l=1}^{L} \tilde{h}_l(f_l(x), \theta_l), \quad \text{and} \quad \sum_{l=1}^{L} \tilde{v}_l.$$

On each event, we update $\tilde{v}_l$ and the parameter vector θ_l of link l, only if the event is associated with the link. Events

Table 1: Service Types for the 4 Node Network.

Service type m	1	2	3
Bandwidth demand $b(m)$	1	3	5
Average holding time $1/\nu(m)$	10	10	2
Immediate reward $c(m)$	1	2	50

associated with a link l are arrivals of new calls which are potentially routed over link l and termination of calls which were routed over the link l. The update rule of the parameter vector θ_l is very similar to the TD(0) algorithm described above

$$\begin{aligned} \theta_{l,k} &= \theta_{l,k-1} + \gamma_{l,k} d_{l,k} \nabla_{\theta_l} \tilde{h}_l(f_l(x_{t_{l,k-1}}), \theta_{l,k-1}), \\ \tilde{v}_{l,k} &= \tilde{v}_{l,k-1} + \\ & \eta_{l,k}(g_l(x_{l,t_k}, \omega_{l,k}, u_{t_k}) - (t_{l,k} - t_{l,k-1})\tilde{v}_{l,k-1}), \end{aligned}$$

where

$$\begin{aligned} d_{l,k} &= g_l(x_{t_{l,k}}, \omega_{l,k}, u_{t_{l,k}}) - (t_{l,k} - t_{l,k-1})\tilde{v}_{l,k-1} + \\ & \tilde{h}_l(f_l(x_{t_{l,k}}), \theta_{l,k-1}) - \\ & \tilde{h}_l(f_l(x_{t_{l,k-1}}), \theta_{l,k-1}) \end{aligned} \qquad (2)$$

$\gamma_{l,k}$ and $\eta_{l,k}$ are small step size parameters and $t_{l,k}$ is the time when the kth event $\omega_{l,k}$ associated with link l occurs. Whenever a new call of a service of type m is routed over a route r which contains the link l, the immediate reward g_l associated with the link l is equal to $c(m)/\#r$, where $\#r$ is the number of links along the route r. For all other events, the immediate reward associated with link l is equal to 0.

The advantage of this decomposition approach is that it allows decentralized training and decentralized decision making. Furthermore, we observed that this decomposition approach leads to much shorter training times for obtaining an approximation for h^* than the approach without decomposition. All these features become very important if one considers applying methods of RL to large integrated service networks supporting a fair number of different service types.

Decomposing complex systems into subsystem, which are then solved independently, is a popular approach in systems-engineering. In general, when a system is decomposed into smaller subsystems a "modeling error" is introduced. This is due to the interaction/coupling between the subsystems which is suppressed by the decomposition. However, the hope is that the ability to find better solutions for the individual subsystems compensate this "modeling error" and that the resulting control policy compares favorable to the one obtained by solving the entire system.

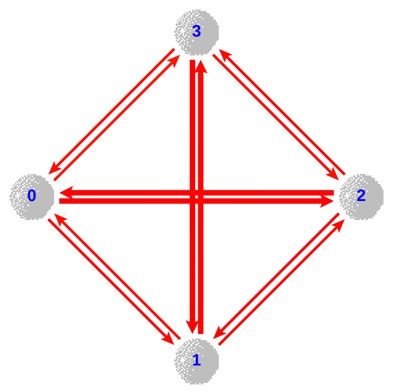

Figure 1: Telecommunication Network Consisting of 4 Nodes and 12 Unidirectional Links.

4 Experimental Results

In this section we present experimental results of two case studies involving a network with 4, and 16 nodes, respectively.

For both cases, we compare the policy obtained through RL with the commonly used heuristic OSPF (Open Shortest Path First). For every pair of source and destination nodes, OSPF orders the list of predefined routes. When a new call arrives, it is routed along the first route in the corresponding list, that does not violate the capacity constraint; if no such a route exists, the call is rejected.

For the RL approach, we use a quadratic approximator, which is linear with respect to the parameters θ_l, as a compact representation of $\tilde{h}_l$. Other approximation architectures were tried, but we found that the quadratic gave the best results with respect to both the speed of convergence and the final performance. As inputs to the compact representation $\tilde{h}_l$, we use a set of local features, which we chose to be the number of ongoing calls of each service type on link l. There are approximately $1.6 \cdot 10^{45}$ and $1.4 \cdot 10^{256}$ different feature configurations for the 4 node and 16 node network, respectively. Note that the total number of possible states is even higher.

We obtained the final control policies of both case studies after $5 \cdot 10^6$ iteration steps.

4.1 Network with 4 Nodes

In this section, we present experimental results obtained for the case of an integrated service network consisting of 4 nodes and 12 unidirectional links. There are two different classes of links with a total capacity of 60 and 120 units of bandwidth, respectively (indicated by thick and thin arrows in Figure 1). We assume a set $\mathcal{M} = \{1, 2, 3\}$ of three different service types. The corresponding bandwidth demands, average holding times and immediate rewards are given in

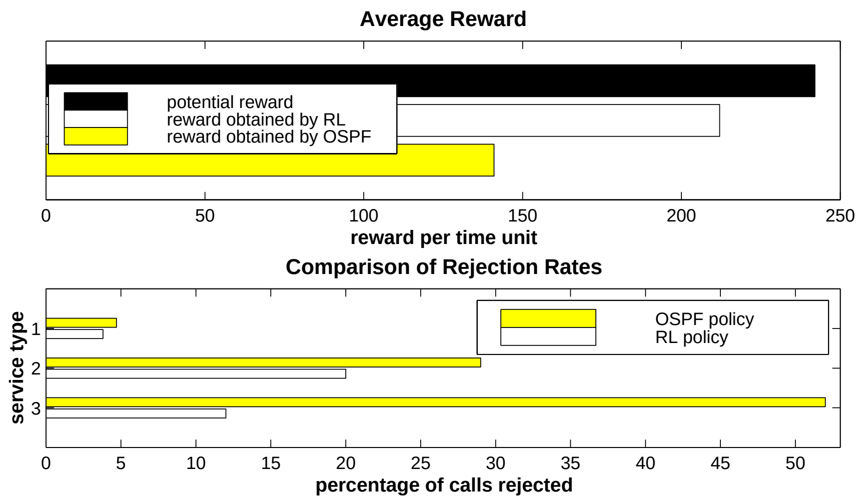

Figure 2: 4 Node Network: Comparison of the Average Rewards and Rejection Rates of the RL and OSPF Policies.

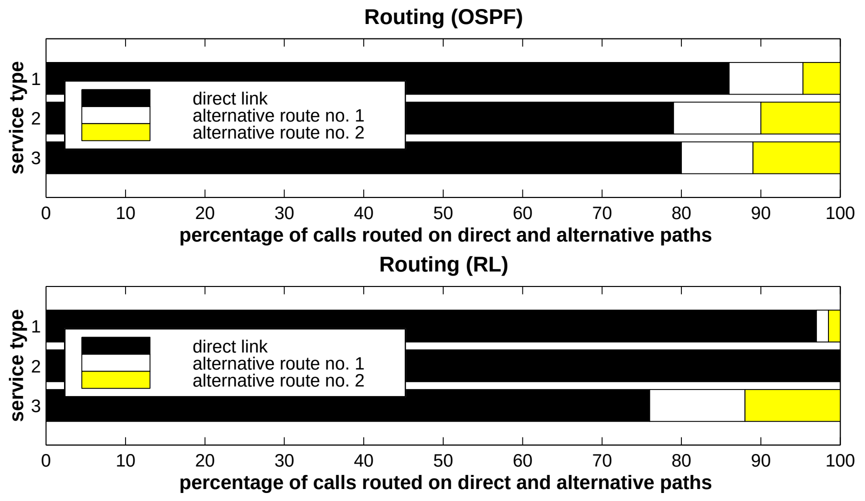

Figure 3: 4 Node Network: Comparison of the Routing Behaviour of the RL and OSPF Policies.

Table 1. Note that the calls of type 3 are much more valuable than the one of type 1 and 2. Call arrivals are modeled as independent Poisson processes, with a separate mean for each pair of source and destination nodes and each service type. Furthermore, for each source and destination node pair, the list of possible routes consists of three entries: the direct path and the two alternative 2-hop-routes.

This case study is characterized by a high traffic load and by calls of one service type having a much higher immediate reward than calls of the other types. We choose this setting to determine the potential of our optimization algorithm.

The results of the case study is given in Figure 2 (Performance) and Figure 3 (Routing Behaviour). We give here a summary of the results.

Performance Comparison: The policy obtained through RL gives an average reward of 212, which as about 50% higher than the one of 141 achieved by OSPF. Furthermore, the RL policy reduces the number of rejected calls for all service types. The most significant reduction is achieved for calls of service type 3, the service type, which has the highest immediate reward. Figure 2 also shows that the average reward of the RL policy is close to the potential average reward of 242, which is the average reward we would obtain if all calls were accepted. This leaves us to believe that the

Table 2: Service Types for the 16 Node Network.

Service type m	1	2	3	4
Bandwidth demand $b(m)$	1	2	3	4
Av. holding time $1/\nu(m)$	1.25	1.25	1.25	1.25
Immediate reward $c(m)$	0.25	1	6	15

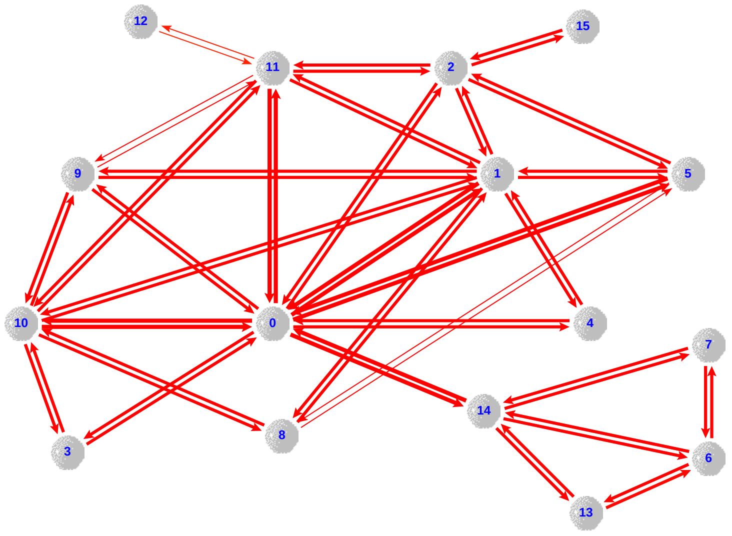

Figure 4: Telecommunication Network Consisting of 16 Nodes.

RL policy is close to optimal. Figure 3 compares the routing behavior of the RL control policy and OSPF. While OSPF routes about $15\% - 20\%$ of all calls along one of the alternative 2-hop-routes, the RL policy uses alternate routes for calls of type 3 (about 25%) and routes calls of the other two service types almost exclusively over the direct route. This indicates, that the RL policy uses a routing scheme, which avoids 2-hop-routes for calls of service type 1 and 2, and which allows us to use network resources more efficiently.

4.2 Network with 16 Nodes

In this section, we present experimental results obtained for a network consisting of 16 nodes and 62 unidirectional links (see Figure 4). The network topology is taken from Greenberg & Srikant (1997). The network consists of three different classes of links with a capacity of 60, 120 and 180 units of bandwidth, respectively. We assume four different service types. Table 2 summarizes the corresponding bandwidth demands, average holding times and immediate rewards. Again, call arrivals are modeled by Poisson processes. The arrival rates for each pair of source and destination nodes are published in the paper of Greenberg & Srikant (1997). Since the length of each link (in miles) is also mentioned in the above paper, we use this information to compute the list of possible routes. For each pair of source and destination nodes, it contains a maximum of six shortest path routes ordered by their path length.

The results of the case study are summarized by Figure 5 (Performance) and Figure 6 (Routing).

Performance Comparison: The OSPF policy almost exclusively routes all calls over the shortest path. This leads to an average reward of about 4254. The rate of rejected calls is positive for all service classes. The two most valuable service types 3 and 4 receive the highest rejection rate. In contrast, the RL policy comes up with a very different routing scheme that uses alternative paths for all types of services. Now, the rejection rates for calls of type 1, 3 and 4 vanish whereas that for service class 2 increases. The RL policy rejects these calls in a strategic way, i. e. RL is not forced to do so by the capacity constraint. Instead, it explicitly reserves bandwidth for the most valuable calls of type 3 and 4. The average reward of 4349 obtained through the RL policy is about 2.2% higher than the one achieved by OSPF. A more significant performance measure illustrating the power of the RL policy is the so-called lost average reward. It is defined as the difference between the potential average reward of 4438 and the actual one achieved by the current policy. RL reduces the lost average reward by about 52% compared with OSPF. This leaves us to believe that RL is close to optimal.

5 Conclusion

The call admission control and routing problem for integrated service networks is naturally formulated as an average reward dynamic programming problem, albeit one with a very large state space. Traditional dynamic programming methods are computationally infeasible for such large scale problems. We use reinforcement learning, based on $TD(0)$ for average reward problems (Tsitsiklis&Van Roy, 1997), combined with a decomposition approach, which views the network as consisting of link processes. This decomposition has the advantage that it allows decentralized decision making and decentralized training, which reduces significantly the time of the training phase. We presented a solutions for example networks with about 10^{45} and 10^{256} different

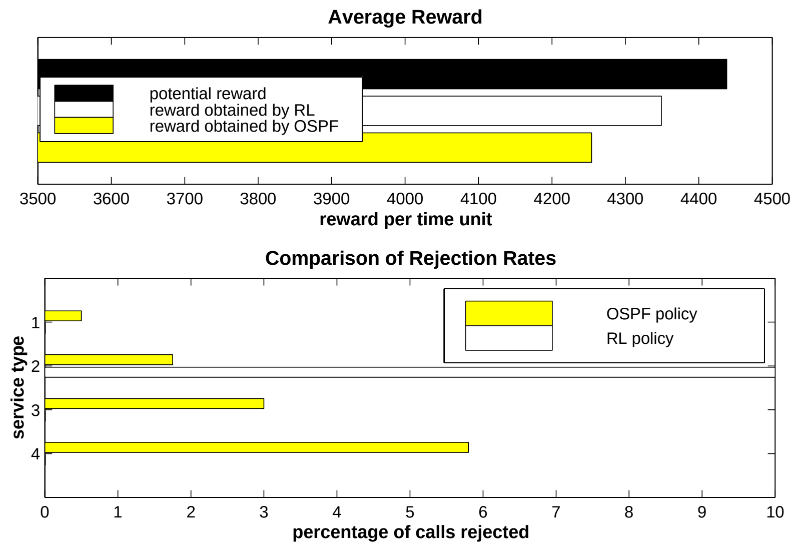

Figure 5: 16 Node Network: Comparison of the Average Rewards and Rejection Rates of the RL and OSPF Policies.

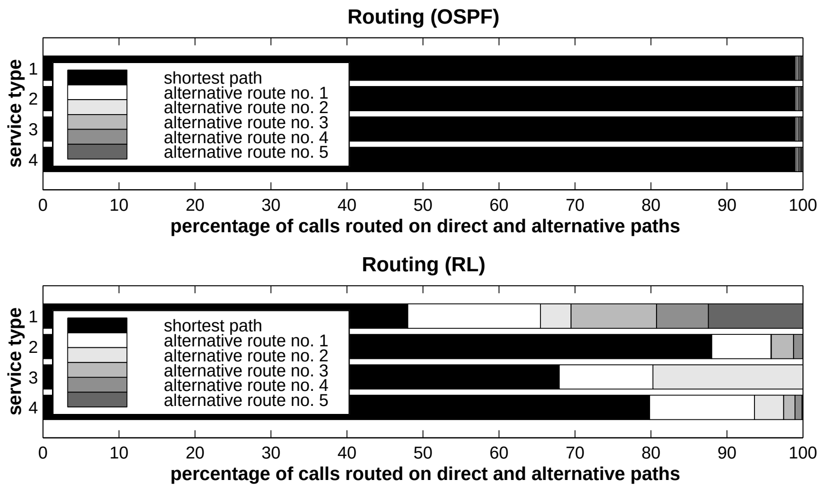

Figure 6: 16 Node Network: Comparison of the Routing Behaviour of the RL and OSPF Policies.

feature configurations. The case study involving a network with 16 nodes showed the capacity of RL to learn complex control policy which use strategic call rejections. Such policies are difficult to obtain through heuristics. More computational experiments are needed to determine if this method is useful for truly large scale networks.

References: Bertsekas, D. P. (1995) *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA.

Bertsekas, D. P., Tsitsiklis, J. N. (1996) *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Greenberg, A., Srikant, R. (1997) *Computational Techniques for Accurate Performance Evaluation in Multirate, Multihop Communication Networks*, IEEE/ACM Transactions on Networking.

Tsitsiklis, J. N., Van Roy, B. (1997) *Average Cost Temporal-Difference Learning*, Lab. for Info. and Decision Systems Report LIDS-P-2390, Massachusetts Institute of Technology, Cambridge, MA.