SAPIENZA
UNIVERSITÀ DI ROMA

"Sapienza" University of Rome
Faculty of Information Engineering,
Informatics and Statistics
Department of Computer Science

RL-based protocols review

Author
Alessio Bandiera

December 4, 2025

Contents

Information and Contacts

Personal notes and summaries collected as part of the *RL-based protocols review* course offered by the degree in Computer Science of the University of Rome "La Sapienza".

Further information and notes can be found at the following link:
https://github.com/aflaag-notes. Anyone can feel free to report inaccuracies, improvements or requests through the Issue system provided by GitHub itself or by contacting the author privately:

- Email: **alessio.bandiera02@gmail.com**
- LinkedIn: **Alessio Bandiera**

The notes are constantly being updated, so please check if the changes have already been made in the most recent version.

Suggested prerequisites:

Basic concepts of Reinforcement Learning.

Licence:

1 Introduction

TODO

todo

2 Q-Learning

TODO intro

2.1 Reinforcement Learning

An RL problem is modeled by a 4-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where

- $\mathcal{S}$ is the set of states
- $\mathcal{A}$ is the set of actions
- $\mathcal{P}$ is the matrix of state transition probabilities
- $\mathcal{R}$ is the reward function

In particular, the environment model is described by $\mathcal{P}$ and $\mathcal{R}$. When the matrix P is defined we call the problem **model-based**, however the vast majority of the approaches is actually **model-free** because they require less storage cost and they do not depend on the accuracy of *a priori* knowledge of the environment model.

The behaviour of the learning agent is defined by a **policy** π_t, where

$$\pi_t[a|s] := \Pr[a_t = a, s_t = s]$$

Assuming that time is discrete, at each timestep the agent selects an action among the set of actions $\mathcal{A}$, and the environment provides a **reward** denoted by R_t where

$$R_t := \mathcal{R}(s, a)$$

The objective of the agent is to take actions in order to maximize the **global discounted reward (GDR)**, denoted by G_t, that receives over the future. There are three standard approaches to define G_t:

- **Finite-horizon**: the agent should optimize the reward for the next h steps

$$G_t := \sum_{k=1}^{h} R_{t+k}$$

This model is appropriate when the agent lifetime is known

- **Infinite-horizon**: this model prevails in literature regarind RL-based systems in general, and it defines the GDR as follows

$$G_t := \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1}$$

where $0 \leq \gamma \leq 1$ is called **discount rate**. If $\gamma = 0$ the agent is called *myopic*, since it can only "see" the immediate reward, while as γ approaches 1 the awareness to future rewards increases.

- **Average-reward**: as the name suggests, this model specifies that

$$G_t := \lim_{h \to +\infty} \frac{1}{h} \sum_{k=0}^{h} R_{t+k+1}$$

In the field of RL applications it is usually assumed that the environment state has the **Markov property**, or some approximation of it. This is because under this assumption the expected value of the next reward is independent of past rewards. Indeed, the Markov property states that the next state depends only on the *present state* and the action to take

$$\Pr[R_{t+1} = r, s_{t+1} = s' \mid (s_0, a_0, r_0), \ldots, (s_t, a_t, r_t)] = \Pr[R_{t+1} = r, s_{t+1} = s' \mid s_t, a_t]$$

RL algorithm involve two types of **Q-value functions**:

- **state-value** function: it estimates how good it is for the agent to be in a state. The state-value of a state s under policy π is defined as the expected GDR in s

$$V_\pi(s) := \mathbb{E}[G_t \mid a_t = s]$$

- **action-value** function: it assesses how good it is to perform a given action in a given state. The state-value of taking action a in state s under policy π is the expected GDR in starting from s and taking action a

$$Q_\pi(s, a) = \mathbb{E}[G_t \mid s_t = s, a_t = a]$$

Solving an RL problem means finding a policy π^* that is able to achieve the maximum reward over the long term – a policy π is better than a policy π' if it yields better GDR over all the states, i.e.

$$\forall s \in \mathcal{S} \quad V_\pi(s) \geq V_{\pi'}(s)$$

Therefore, by definition it holds that

- $V_{\pi^*}(s) = \max_\pi V_\pi(s) \quad \forall s \in \mathcal{S}$
- $Q_{\pi^*}(s, a) = \max_\pi Q_\pi(s, a) \quad \forall s \in \mathcal{S}$

many approaches have been proposed to lean an optimal policy at reasonable cost, depending on whether the learning is model-free or model-based. The authors of this paper limited their presentation to the most used learning technique among all the RL-based protocols reviewed, namely *Q-learning* which will be discussed in the next section.

2.1.1 Q-learning

In 1989 Watkins et al. [Wat+89] introduced **Q-learning**, a model-free approach to estimate the action function $Q_{\pi^*}(s, a)$. Very importantly, his approximation of the action function is independent of the policy followed by the agent, which makes Q-learning applicable in wide variety of contexts. In Q-learning, the learning process consists in a sequence of stages called *epochs*: in epoch n, the agent is in state s_n, performs actions a_n, receives a reward R_n and it m oves to state s_{n+1}. The action-value is then updated through the following formula

$$Q_n(s_n, a_n) = (1 - \alpha) \cdot Q_{n-1}(s_n, a_n) + \alpha \cdot \left[R_n + \gamma \cdot \max_{a \in \mathcal{A}} Q_{n-1}(s_{n+1}, a) \right]$$

where α is called **learning factor**. The initial Q-values $Q_0(s, a)$ are assumed given. Moreover, this function can be rewritten in discrete time t as follows:

$$Q(s_t, a_t) = (1 - \alpha) \cdot Q(s_t, a_t) + \alpha \cdot \left[R_{t+1} + \gamma \cdot \max_{a \in \mathcal{A}} Q(s_{t+1}, a) \right]$$

Most importantly, Watkins showed that Q-learning converges to the optimum action-values with probability 1 as long as all actions are repeatedly samples in all states. Indeed, this is the reason why Q-learning is the most popular and most efefctive learning technique for learning from delayed reinforcement. However, the *speed of convergence* remains an open issue.

2.2 Q-routing

2.2.1 Application of RL to routing protocols

TODO intro

In RL-based design, the following aspects are addressed:

- identification of the most appropriate states and actions of the agent
- definition of the reward function depending on the metrics to optimize
- identification of environment model when available

Given a target field of application, different design models may be elaborated, which differ in how they address each of the previous aspects.

In the last 25 years many RL-based routing protocol have been proposed, but most of them share the same high-level structure.

In literature, nodes are confused with agents, and in almost all protocols the reward is (at least partially) calculated by a node upon selecting an entire route to use for all packtes to transmit, or just a next hop to transmit the current data packet. Thus, a *node* should be considered to consist of an agent and optional *modules*:

- **local reward** module: it calculates reward based on local view, which reflects the cost of communication as seen by the packet sender

- **remote reward** module: it receives feedback sent by the next hop or by the destinatino node — if local and remote modules are both employed they are combined to form the reward return to the agent
- **link-state information maintenance** module: it collects useful link state information through periodi or on-demand *Hello packets*

Therefore, the neighboring nodes of a node define the environment of the agent representing a node.

TODO

parlare dei tipi di reti?

2.2.2 Q-routing

Boyan and Littman [BL93] were the first to propose a hop-by-bop routing algorithm based on Q-learning, called Q-routing, and the most of exists RL-based routing protocols today are just extensions of this algorithm. The following is the algorithm that defines Q-routing in detail.

1: **function** QROUTING( )
2: Initialize Q_i matrix randomly
3: **while** termination condition holds **do**
4: **if** packet P is ready to be sent to d **then**
5: Determine node $j^* \leftarrow \arg\min_{j \in \mathcal{N}(i)} Q_i(d, j)$
6: Send packet to node j^*
7: Collect estimate $\theta_{j^*}(d)$ from node j^*
8: Update $Q_i(d, j^*) \leftarrow (1 - \alpha) \cdot Q_i(d, j^*) + \alpha \cdot [W_i^q(P) + T_{ij^*} + \theta_{j^*}(d)]$
9: **end if**
10: **end while**
11: **end function**

We will briefly explain the algorithm. First, let's present the notation:

- i is the node that is currently running the algorithm
- P is a packet that node i needs to forward to destination d
- $Q_i(d, j)$ is the *delivery delay* that i estimates it takes, for node j, to deliver the packet P at destination i
- $\mathcal{N}(j)$ is the set of j's neighbors
- $\theta_j(d)$ is j's estimate for the time remaining in the trip to destination d of packet P
- $W_i^q(P)$ is the time spent by packet P in node i's queue
- T_{ij} is the transmission time between nodes i and j

Each entry of the table Q_i is called **Q-value**, and when node i wants to send a packed, it selects the node j^* that minimizes the Q-value. Upon sendin packet P to node j^*, node i receives back from j^* the value

$$\theta_{j^*}(d) = \min_{k \in \mathcal{N}(j^*)} Q_{j^*}(d, k)$$

Then, node i has to update its estimate of $Q_i(d, j^*)$ based on $\theta_{j^*}(d)$, which can be performed utilizing the formula of the discrete time Q-function update outlined earlier, by setting

- $R_{t+1} = W_i^q(P) + T_{ij^*}$ since it represents the *link cost*
- $\gamma = 1$
- $\max_{a \in A} Q(s_{t+1}, a) = \min_{k \in \mathcal{N}(j^*)} Q_{j^*}(d, k)$ since the "action to take" corresponds to choosing an neighbor in this context, however we seek to *minimize* the Q-value since the delay is clearly a decreasing metric

Despite the wide adoption of this protocol over the years, Q-routing is still far from perfect and it has its flaws. Some of the problems are inherent problems of Q-learning in general, such as *slow convergence* rate and high *parameter setting sensitivity*, but there are problem that arise from the protocol itself. For instance, to avoid frequen oscillations of the Q-values — in case of sudden variations of traffic in the network — and limit the overhead of the protocol, Nowe, Steenhaut, Fakir, et al. [NSF+98] proposed an extension in which j^* sends an average $\overline{\theta}_{j^*}(d)$ to i only after a certain number of exchanged packets. Another known problem is called *Q-value freshness*: the estimate $\theta_j(d)$ is evaluated upon packet transmission on a route, therefore if a route is not selected during a long period of time, the agent does not have an accurate estimate of the current condition of such route.

3

Classification criteria

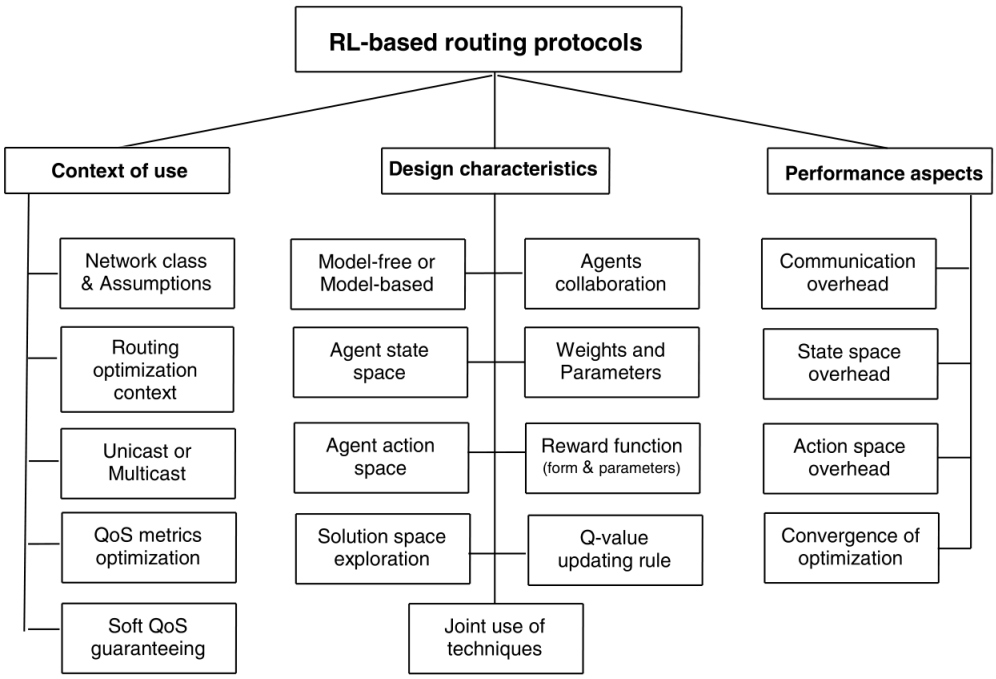

Figure 3.1: TODO

The authors underline that, to their knowledge, their work is the first in the literature that proposes classification criteria to help understanding and comparing all the available RL-based routing protocols. These criteria are divided into 3 groups:

- **context of use**: these are criteria that describe the targeted applications and their characteristics and requirements
- **design characteristics**: criteria in this group highlight how authors designed their protocols to make them efficient and different from the others

- **performance**: in this last category, criteria provide a qualitative evaluation of the overhead of protocols and the metrics used by the authors

We will cover each criteria in the

3.1 Context of use

3.1.1 Addressed network classes

RL-based routing protocols have progressively evolved across many *types of networks* — wired, wireless, mesh, cooperative, optical, MANETs, WSNs, vehicular, DTNs, FANETs, social networks, cognitive radio, SDN, NDN, and P2P. Indeed, the first kind of general categorization is the network class of each protocol. Beyond network type, some protocols also rely on **specific assumptions**, such as having prior knowledge of traffic distributions, node localization services, or the possibility of transmission errors.

3.1.2 Routing Optimization Context

From users' perspective, routing protocols should always be able to determine and select the optimal paths to convey data from sources to destinations. There are different ways to achieve such goal, that depend on

- roles assigned to data sources
- roles assigned to relaying nodes
- initial assumptions about routing

The authors outlined 6 different *routing optimization contexts*, which we will briefly explain one by one.

1. **Data-packet driven optimazion**: in this context the transmission of packets happens hop-by-bop from source s to destination d, and upon receival d sends back a feedback. After a given amount of forwarded packets, the routing process converges to the selectoin of optimal paths.
2. **Route requrest driven optimization**: a source s that has data to send to d, first sends a Route Request (RR) packet. The latter is then disseminated in the network, and each node that receives the RR packet can decide to partecipate or not — if it agrees to partecipate, it selects the next node to forward the RR packet to, and this process continues until d is reached. Once a path is found, all packets from s to d are routed through this path. Then, at the end of each transmission a feedback is sent back to the sender regarding the performance of current nodes. Most protocols in this category are extensions to the **AODV protocol** [PBRD03].
3. **Context request driven optimization**: this is a setting that describes peer-to-peer systems and named data networks, in which a node s that is interested in some content C sends its requrest to receive data packets from the nodes d possessing C. Nodes on the path from s to d can then decide to forward the request to locate

the requested content, and when data packets containing C are forwarded the relay nodes receive feedback and adapt their paths accordingly.

4. **Predefined routes driven optimization**: each source builds *offline* a list of paths of reachability for any target destination. Hence, when a source has packets to send it selects a path amonde the predefined ones. If a link break on the selected path is detected, the source switches to another predefined path. Periodically, a feedback is sent backward to the source, which will adapt its path selection among the predefined list.

5. **Cluster driven optimization**: *clustering* is the process of partitioning the set of nodes into groups, with one "cluster-head" per group. Data packets are transmitted from sources to destination following a clustered hierarchy of the network — i.e. nodes (members of a cluster) send their data packets to their cluster-head, which in turn send the packets to a higher level cluster-head, which in turn sends the packet to a higher level cluster-head or to the destination. Depending on the resources, a cluster-head determines the number of members that can join in the cluster. After the transmissionof data packets from members, cluster-heads receive feedback and adjust their cluster size.

6. **Routing protocol driven optimization**: it is a fairly recent approach, and in thic framework there is a central node that ha sa set of routing protocols candidates (e.g. AODV, DSDV, DSR, OLSR, GPSR, etc.) that can be used for forwarding packets by *slave nodes*. In each time interval, the central node selects a routing protocol and calculates the routing tables, then sends them to slave nodes to configure their forwarding tables. Afterwards, a feedback is collected by the central node regarding the performance of computed routing tables, and it may decide to change the current routing protocol for the next time interval depending on the observed performance. AFter some time intervals, the system converges to the most adequate routing protocol.

3.1.3 Unicast or Multicast

Unicast and Multicast routing strategies are vastly different in terms of optimization, so it comes natural to define this criteria in order to categorize the algorithm proposed in the literature. The difference between the two approaches lies in the overhead that Multicast trees requires, both in terms of times and communications, in order to reach optimal trees. Additionally, when some links are not sufficiently stable, the convergence to optimal trees is outright *impossible*. Indeed, RL shoule be applied for multicasting scenarios only when links are sufficiently stable and/or when partial delivery is allowed — for instance, it is greatly discouraged by the authors on wireless networks.

3.1.4 QoS metrics for optimization

In general, routing problems in networks are **multicriteria decision making (MCDM)** problems, which are notoriously difficult to solve because of the heterogeneity nature of the metrics utilized. In fact, the choice of the metrics is one of the most important aspects of a user, which depends on the specificities of their application. Consequently, MCDM

solving approaches are based on *weights* that express the relative importance of each metrics. **Quality of Service (QoS)** metrics that have been addressed as objectives for RL-based routing include:

- **delivery rate**: the average time to deliver a packet at destination
- **delivery ratio**: the proportion of packets successfully delivered at destination
- **hop count**: the average number of hops from source to destination
- **loss ratio**: the proportion of packets not delivered at destination
- **symbol error rate**: the proportion of *symbols* incorrectly transmitted
- **light-path blocking probability**: the percentage of the blocked light-paths of all requrests in optical networks — it is similar to the *loss ratio*
- **bandwidth**: the average bandwidth provided to sources
- **throughput**: the average amount of bytes delivered in the entire network per time unit
- **path stability**: it indicates how a path between source and destination changes over time
- **energy consumption**: the average energy consumption due to transmissions, receptions and processing
- **network lifetime**: the average time over which the network is still alive — this is essential in wireless sensor networks (WSNs)
- **transmission power**: the power for performing transmission — usually results in energy saving and interference reduction
- **PU-SU interference (ratio)**: it indicates how Primary users (PU) are prevented from transmissing by secondary users (SU)
- **hit delay**: the average delay to return requested data in peer-to-peer and named data networks
- **hit ratio**: the proportion of statisfied requrests in peer-to-peer and named data networks
- **gain** or **revenue**: the average revenue (in $ or any other currency) received by the agent when routing is seen from a business point of view, and routing should result in profit
- **overhead**: the average *cost* to deliver data packets at destinatino — the definition of *cost* may vary depending on the application

3.1.5 QoS guaranteeing

Lastly, there are a few routing protocols aimed at providing QoS guarantees, regarding delivery delay to meet requirements of some **delay-sensitive applications** – such as multimedia applications.

3.2 Design characteristics

3.2.1 Learning model

In RL there are two classes of learning strategies, namely **model-free** and **model-based**. Even if the vast majority of RL-based routing algorithms are model-free, since constructing a model requires knowledge about the enviroment that can be very hard to collect, it is worth mentioning that a few algorithms are actually model-based. Some of them use offline-collected information, regarding the environment model, while other caluclate and improve the environment model online. Model-based learning can offer an interesting opportunity when the the speed of convergence is a crucial requirement, as model-based approaches are known to have a faster convergence rate.

3.2.2 Agent states and Action spaces

In order to apply RL to any optimization problems we obviously need some definitions of **agent states** and **action spaces**. Let's discuss the former first. The following is a brief list of possible *agent state spaces* utilized in the reviewd literature:

non ho capito che ho scritto in questa lista

- *set of nodes*, which is the most popular in RL-based routing protocols
- *set of grids*, used in grid-organized networks
- *set of couples* relating to the dynamics of nodes, for instance in VANETSs a *couple* is a vehicle speed class and context of move (urban, highway...)
- *set of paths* and their characteristics
- *set of QoS levels required by flows*
- *set of transmission power levels*
- *set of available wavelengths*, in optical networks
- *set of packet states*

Next, we need to outline the possible *action spaces*. Broadly speaking, an action space is a set of single-type actions, or a set of actions of different types. The following is a table containing the possible *single-type actions* and corresponding action state spaces:

Action selection	**Action space**
Select node j as next hop and forward packet	Set of node IDs
Select a subset of neighbors S and broadcast packet	Set of partitions of node IDs
Select output link l and transmit packet	Set of links
Select grid g and send packet to one of the nodes in g	Set of grids
Select predefined path p and send packet along p	Set of predefined paths
Allocate m free channels	Set of channels
Select a transmission power pw	Set of transmission power levels
Select a protocol prt among a list of routing protocols and configure the network with prt	Set of standard protocols

3.2.3 Solution space exploration

In Reinforcement Learning the **Exploration vs Exploitation dilemma** is a well-known problem, and the speed of convergence is directly dependent on the approach utilized to balance between exploring and exploiting the solusion space. In RL-based routing protocols there are mainly 6 *selection techniques* to handle the solution space:

- **Greedy**: only the highest Q-value is used for selection — this strategy may take a very long time to converge
- **ε-greedy**: in addition to greedy selection, the learner uses a small amount of randomness to explore new solutions — this is the most used form of selection in the reviewed protocols
- **Probability based**: the same as ε-greedy, except that the value of ε is calculated from the history of learning
- **Bayesian network decision**: action selection uses Bayesian networks to better explore the solution space
- **Devaluation of solutions based**: with this strategy Q-values are either periodically decayed in order to enforce exploration
- **New neighbors first**: new discovered nodes are favored in next hop selection — this approach is particularly useful in mobile networks

3.2.4 Agents collaboration

In its basic form, RL defines each agent as *independent* and only able to interact with the environment. However, the authors point out that when applyin RL to routing it is much more effective to allow **collaborating agents**, indeed almost all proposed protocols are based on this idea. This technique involves not only rewards from the environment

but also exchanges of link-state information without actions undertaken from RL point of view. In particular, from an RL pointof view selecting the next hop *is* an RL action, but receiving periodic *Hello packets* is not. However, such information exchange is crucial to ensure reliability of the connections between hops. Indeed, collaboration between agents is so prevalent among the studied algorithms that it is possible to categorize them w.r.t. how the nodes of the network actually cooperate:

- **Reactive collaboration**: when the selected neighbors receives a data packet, either it directly returns its feedback in the ACK packet or it includes its link-state information in each data packet when it forwards it
- **Proactive collaboration**: in addition to sending their feedback upon reception of a packet, nodes periodically broadcast their link-state information in the form of *Hello packets* to their neighbors

3.2.5 Hybridation with other optimization techniques

Most of RL-based routing algorithms are "pure reinforcement learning", however some algorithms combine RL and other optimization techniques to speed up convergence. Some examples include:

- gradient methods
- game theory approaches
- fuzzy logic techniques
- Bayesian networks methods
- least square policy iteration
- neural networks
- genetic algorithms
- ants optimization

3.2.6 Number of parameters to tune

In general, a well-designed protocol should be easy to tune, i.e. it should provide easy-to-use tools and parameters in order to achieve high performance. As we already discussed, when RL is used we consider two basic parameters α and γ, the *learning factor* and the *discount rate* respectively. Additionally, when multiple metrics are of interest it is crucial to assign a weight to each metric or group of metric in order to establish importance. However, tuning such parameters can be *very* difficult. Therefore, the authors categorized the routing protocols also based on the number of tunable QoS metrics each paper offers.

3.2.7 Reward functions

The authors underline that the **reward function** is the most distinctive feature of existing RL-based routing protocols, in fact there is a wide variety of reward functions employed

based on the metrics that each protocol decided to include. Reward functions may be categorized into three classes:

- **Test-based reward functions**: they are the simplest form of reward, in which the reward is assigned a constant value depending on the outcome of some *test*. Not surprisingly, the most common test is checking if the packet is was actually delivered to destination — which would yield a *binary outcome* for the reward.
- **Linear reward functions**: they have the following general form

$$R = C + \sum_{k=1}^{H} \omega_k \cdot M_k$$

  where
  - C is a constant factor that depends on the test choosen by the protocol
  - H is the number of metrics of the protocol
  - ω_k is the weight of the k-th metric
  - M_k is the value of the k-th metric
- **Nonlinear reward functions**: this type is less common among the RL-protocols, and they are designed with different forms of combinations of metrics depending on the specific application

3.2.8 Q-value updating rule forms

While over half of proposed RL-based routing algorithms are direct applications of Q-learning as presented in the previous chapter, the remaining half of protocols proposed RL-based routing procedures that either used a modifice QL Q-value updating rule, or do not rely on Q-learning at all.

3.3 Performance aspects

3.3.1 Communication overhead

Communication overhead of protocols have been categorized from a *qualitative* point of view into the following groups:

- **null**: no exchange required between agents
- **low**: the selected next hop returns a feedback in an explicit ACK packet, or it includes its feedback when, in turn, it (re)forwards the packet — half of the reviewed protocols fall under this category
- **medium**: this is the case of protocols in which the feedback from the destination is propagated to all hopds through an explicit ACK packet

- **high**: these protocols require that nodes periodically exchange link-state information — clearly the amount of control packets needed depend on the period of *Hello packets*

3.3.2 State space overhead

RL-based algorithms require memory to store the **states of the agents**, and in some RL-based applications the number of states may be *very high*. Hence, from a *qualitative* perspective the protocols can be grouped based on the **state space overhead** as follows:

- **very low**: when the state space is states of a packet what?
- **low**: when the state space is the node IDs — most of the reviewed papers fall under this category
- **limited**: when the state space depends on external factors (such that the number of transmission power levels, maximum number of available channels etc.)
- **high**: when the state space is a list of paths with their current characteristics

3.3.3 Action space overhead

In addition to the space required to store states, RL-based algorithms also require memory to store **the possible actions** that agents can select. Again, from a *qualitative* point of view **action overhead** is categorized as follows:

- **low**: when the action space depends on external factors
- **medium** when the action space depends on the number of nodes in the neighborhood
- **high** if the action space depends on the number of dynamic or predefined paths, or on the number of grids in the network
- **very high**: if the state space depends on combinations of channel subsets or paths

3.3.4 Proof of convergence

In optimization field, the **convergence** to optimal solutions is an *expected* property, however many existing techniques to solve multicriteria optimization (MCO) problems are known to be sub-optimal. RL-based solution would be widely deployed if their convergence rate could be formally demonstrated, and indeed the proof of convergence can be derived from the work of Watkins et al. [Wat+89] but only as long as some assumptions are satisfied. In real world scenarios it is hard to establish the satisfiability of such assumptions, and proving convergence rigorously remains an issue to this day for most of the protocols that are not perfectly Q-learning compliant. Rather, convergence is usually assessed from simulations or just stating that convergence can be reached eventually without providing any proof of such claims.

3.3.5 Protocol performance (in simulations)

Given the large number of protocols and the wide variety of reward functions and metric weights, the authors found impractical to provide qualitative evaluation of the performance of each single protocol, so they limited their effort to present the data reported by the original authors *as-is*.

4

Conclusions and Challenges

RL is an efficient alternative to enforce online-awareness of routing protocols to their environment changes, and through RL-based routing protocols it is possible to provide higher level of QoS while still optimizing resource utilization. However, some challenges still remain and should be investigated to provide evidence on applicability of RL-based protocols *at large scale*. Some of the challenges pointed out by the authors include:

- **Proof of optimality**: as already mentioned, almost all reviewed papers did not convincingly address proof of convergence. Watkins proved convergence under specific conditions, however it is unclear how and when the latter apply to routing.
- **Speed of convergence**: whenever large networks are considered, space exploration may take a bery long time before optimal paths are discovered, resulting in poor end-to-end performance of the network. Convergence rates should be investigated further to provide bounds of delay for let users know when the network can or cannot provide acceptable QoS levels.
- **Link-state information dissemination**: in most protocols, link-state information is used to calculate metrics, consequently the convergence of the routing algorithms strictly depends on the freshness of disseminated link-state information. The frequency of *Hello packets* should be addressed in order to find a compromise between protocol overhead and values of reward.
- **Metrics weights and learning parameters**: clearly, learning parameters and weights have a significant impact on the quality of paths and on the speed of convergence. However, determining the optimal set of parameters can be very difficult, therefore the authors suggest the development of a methodology to address this problem, which would make the depolyment of RL-based routing protocols easier.
- **Hybridization**:
- **Hybridization**:
- **Hybridization**:

da finire so stufo

Bibliography

[BL93] Justin Boyan and Michael Littman. "Packet routing in dynamically changing networks: A reinforcement learning approach". In: *Advances in neural information processing systems* 6 (1993).

[NSF+98] Ann Nowe, Kris Steenhaut, Mohamed Fakir, et al. "Q-learning for adaptive load based routing". In: *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218).* Vol. 4. IEEE. 1998, pp. 3965–3970.

[PBRD03] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. *Ad hoc on-demand distance vector (AODV) routing.* Tech. rep. 2003.

[Wat+89] Christopher John Cornish Hellaby Watkins et al. "Learning from delayed rewards". In: (1989).