# Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches

A comprehensive review of the literature of RL-based protocols

Master's Degree in Computer Science

**Alessio Bandiera** (1985878)


SAPIENZA
UNIVERSITÀ DI ROMA

# Table of Contents

Introduction

# Motivation

## Introduction

Modern networks have become far more complex, dynamic and diverse than early manually configured systems. This has made *human* management insufficient.

# Motivation

## Introduction

Modern networks have become far more complex, dynamic and diverse than early manually configured systems. This has made *human* management insufficient.

As a result, **Machine Learning (ML)** is used more and more often to handle tasks such as

- traffic prediction
- fault and configuration management
- congestion control

# Motivation

## Introduction

Modern networks have become far more complex, dynamic and diverse than early manually configured systems. This has made *human* management insufficient.

As a result, **Machine Learning (ML)** is used more and more often to handle tasks such as

- traffic prediction
- fault and configuration management
- congestion control

The goal is to **automatically** learn network conditions in order to improve the user experience, while optimizing network resources.

# The routing problem

## Introduction

In networks, **routing** is the problem of selecting paths for sending packets from source(s) to destination(s), while

- meeting **Quality of Service (QoS)** requirements
- optimizing network resources

# The routing problem

## Introduction

In networks, **routing** is the problem of selecting paths for sending packets from source(s) to destination(s), while

- meeting **Quality of Service (QoS)** requirements
- optimizing network resources

However, whenever multiple metrics are required, the routing problem becomes **NP-complete**.

# The routing problem

## Introduction

In networks, **routing** is the problem of selecting paths for sending packets from source(s) to destination(s), while

- meeting **Quality of Service (QoS)** requirements
- optimizing network resources

However, whenever multiple metrics are required, the routing problem becomes **NP-complete**.

This is the reason why ML is seen as a strategy to revolutionize current **routing** techniques.

# Reinforcement Learning

## Introduction

In particular, among all the various ML techniques known today, **Reinforcement Learning (RL)** stands out in terms of adoption for solving the routing problem.

# Reinforcement Learning

## Introduction

In particular, among all the various ML techniques known today, **Reinforcement Learning (RL)** stands out in terms of adoption for solving the routing problem.

As already discussed throughout the lectures of this course, **Reinforcement Learning (RL)** is an ML technique inspired by behavioral psychology that provides system modeling based on *agents* that interect with their *environment*.

# Reinforcement Learning

## Introduction

In particular, among all the various ML techniques known today, **Reinforcement Learning (RL)** stands out in terms of adoption for solving the routing problem.

As already discussed throughout the lectures of this course, **Reinforcement Learning (RL)** is an ML technique inspired by behavioral psychology that provides system modeling based on *agents* that interect with their *environment*.

RL-based algorithms are particularly useful in the context of routing because it can continuously **learn and adapt** to changing network conditions, selecting routes that optimize performance based on real-time experience rather than fixed rules.

# Q-learning
## Introduction

In 1989 Watkins et al. [Wat+89] proposed the most-widely adopted flavour of RL, called **Q-learning**.

Q-learning is a **model-free** approach that aims at estimating the action function $Q_{\pi^*}(s, a)$, where $\pi^*$ is the optimal policy

# Q-learning
## Introduction

In 1989 Watkins et al. [Wat+89] proposed the most-widely adopted flavour of RL, called **Q-learning**.

Q-learning is a **model-free** approach that aims at estimating the action function $Q_{\pi^*}(s, a)$, where $\pi^*$ is the optimal policy

Very importantly, his approximation of the action function is *independent of the policy* followed by the agent, making Q-learning applicable in a wide variety of contexts.

# Q-learning

## Introduction

In 1989 Watkins et al. [Wat+89] proposed the most-widely adopted flavour of RL, called **Q-learning**.

The action-value is 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 $\alpha$ is the **learning factor**, and $\gamma$ is the **discount rate**.

# Q-learning
## Introduction

In 1989 Watkins et al. [Wat+89] proposed the most-widely adopted flavour of RL, called **Q-learning**.

Moreover, this function can be rewritten in its more common **discrete time** $t$ form:

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

# Q-learning
## Introduction

In 1989 Watkins et al. [Wat+89] proposed the most-widely adopted flavour of RL, called **Q-learning**.

Most importantly, Watkins showed that Q-learning **converges** to the optimum action-values with probability 1, as long as all actions are repeatedly sampled in all states.

Indeed, this is the reason why Q-learning is the most popular and effective learning technique in the field.

# Q-routing
## Introduction

In 1993 Boyan and Littman [BL93] proposed a hop-by-hop routing algorithm based on Q-learning, called **Q-routing**.

# Q-routing
## Introduction

In 1993 Boyan and Littman [BL93] proposed a hop-by-hop routing algorithm based on Q-learning, called **Q-routing**.

After their seminal work, tens of works followed the original idea of using RL to optimize routing, while also considering the evolution of commuinication networks and users requirements.

# Q-routing

## Introduction

In 1993 Boyan and Littman [BL93] proposed a hop-by-hop routing algorithm based on Q-learning, called **Q-routing**.

After their seminal work, tens of works followed the original idea of using RL to optimize routing, while also considering the evolution of commuinication networks and users requirements.

In fact, most of the existing RL-based routing protocols today are extensions of their original work.

# The work in exam

## Introduction

The paper that will be discussed today was published in 2019 by Mammeri [Mam19].

Their work is a review of **60 papers**, which essentially covers the literature of RL-based routing algorithms completely.

# The work in exam

## Introduction

The paper that will be discussed today was published in 2019 by Mammeri [Mam19].

Their work is a review of **60 papers**, which essentially covers the literature of RL-based routing algorithms completely.

In particular, their work has 2 main objectives:

# The work in exam

## Introduction

The paper that will be discussed today was published in 2019 by Mammeri [Mam19].

Their work is a review of **60 papers**, which essentially covers the literature of RL-based routing algorithms completely.

In particular, their work has 2 main objectives:

1. provide a **comprehensive presentation** of the main characteristics of RL-based routing protocols

# The work in exam

## Introduction

The paper that will be discussed today was published in 2019 by Mammeri [Mam19].

Their work is a review of **60 papers**, which essentially covers the literature of RL-based routing algorithms completely.

In particular, their work has 2 main objectives:

1. provide a **comprehensive presentation** of the main characteristics of RL-based routing protocols
2. provide **classification criteria** to enable analysis and comparison of existing protocols

# Table of contents

Introduction

This presentation will first provide a general idea of **Q-routing**, which has been a very influential idea and most of the current literature is based on this RL-based algorithm.

# Table of contents

Introduction

This presentation will first provide a general idea of **Q-routing**, which has been a very influential idea and most of the current literature is based on this RL-based algorithm.

Subsequently, the most important segment of this review will be presented: the **classification criteria** that the authors defined in order to categorize the papers.

# Table of contents

Introduction

This presentation will first provide a general idea of **Q-routing**, which has been a very influential idea and most of the current literature is based on this RL-based algorithm.

Subsequently, the most important segment of this review will be presented: the **classification criteria** that the authors defined in order to categorize the papers.

To their knowledge, the authors state that their work is the first in the literature that proposes classification criteria to help comparing all available RL-based routing protocols.

# Table of Contents

Q-routing

# Introduction to Q-routing

## Q-routing

**Q-routing** is an RL-based approach to distributed routing in packet-switched networks.

# Introduction to Q-routing

## Q-routing

**Q-routing** is an RL-based approach to distributed routing in packet-switched networks.

Each node maintains **estimates** of the delivery time to every destination through each neighbor, and updates these values through *continuous interaction* with the network.

# Introduction to Q-routing

## Q-routing

**Q-routing** is an RL-based approach to distributed routing in packet-switched networks.

Each node maintains **estimates** of the delivery time to every destination through each neighbor, and updates these values through *continuous interaction* with the network.

By learning from real traffic rather than relying on static metrics, Q-routing adapts to

- congestion
- topology changes
- varying link qualities

allowing routing decisions to improve dynamically over time.

# The algorithm

## Q-routing

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 $Q_i(d,j^*) \leftarrow (1-\alpha) \cdot Q_i(d,j^*) + \alpha \cdot \left[W_i^q(P) + T_{ij^*} + \theta_{j^*}(d)\right]$
9: **end if**
10: **end while**
11: **end function**

- $i$ 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 $j$

# The algorithm

## Q-routing

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 $Q_i(d,j^*) \leftarrow (1-\alpha) \cdot Q_i(d,j^*) + \alpha \cdot \left[W_i^q(P) + T_{ij^*} + \theta_{j^*}(d)\right]$
9: **end if**
10: **end while**
11: **end function**

Upon sending packet $P$ to node $j^*$, node $i$ receives back from node $j^*$ the estimate

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

# The algorithm

## Q-routing

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 $Q_i(d,j^*) \leftarrow (1-\alpha) \cdot Q_i(d,j^*) + \alpha \cdot \left[W_i^q(P) + T_{ij^*} + \theta_{j^*}(d)\right]$
9: **end if**
10: **end while**
11: **end function**

Then, node *i* updates $Q_i(d,j^*)$ based on the *update formula* for Q-learning described earlier:

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

# Flaws of Q-learning

## Q-routing

Despite the wide adoption, Q-routing has some flaws. Some problems are direct consequences of Q-learning such as

- *slow convergence*
- *high parameter setting sensitivity*

# Flaws of Q-learning

## Q-routing

Despite the wide adoption, Q-routing has some flaws. Some problems are direct consequences of Q-learning such as

- *slow convergence*
- *high parameter setting sensitivity*

However, there are also problems arising from the algorithm itself, for instance the **Q-value freshness**: $\theta_j(d)$ is evaluated only upon packet transmission on a route, therefore if a route is not used for a long time its estimate becomes *outdated*.

# Table of Contents

Classification criteria

# The criteria groups

Classification criteria

To make sense of this growing body of research, the authors introduced a set of **classification criteria** that allow to systematically *categorize* and compare RL-based routing approaches of the literature.

# The criteria groups

Classification criteria

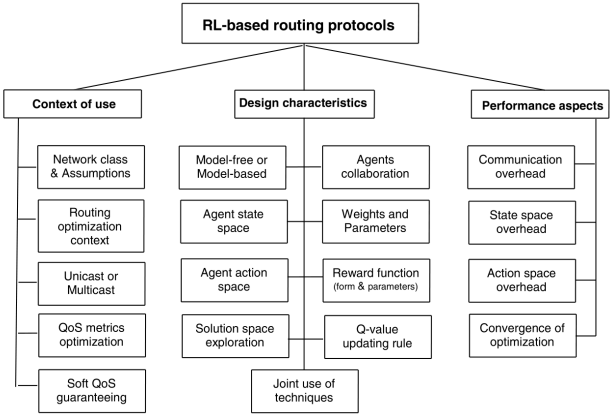
To make sense of this growing body of research, the authors introduced a set of **classification criteria** that allow to systematically *categorize* and compare RL-based routing approaches of the literature.

These criteria are divided into 3 groups:

1. **Context of use**: criteria based on the *target applications*
2. **Design characteristics**: criteria based on the *design* of the protocols
3. **Performance**: criteria based on qualitative evaluation on *overhead* and *metrics*

# The criteria groups

## Classification criteria



# Context of use

Classification criteria: Context of use

The first group of criteria focuses on the **context** in which an RL-based routing protocol is applied.

# Context of use

Classification criteria: Context of use

The first group of criteria focuses on the **context** in which an RL-based routing protocol is applied.

This includes the nature of the target application and the operational conditions under which the protocol must perform.

# Addressed network classes

## Classification criteria: Context of use

Not suprisingly, the first criteria of categorization is the **type of network** of the protocol, since there are various different types:

- WSNs
- DTNs
- FANETs
- MANETs
- etc.

# Addressed network classes

## Classification criteria: Context of use

Not suprisingly, the first criteria of categorization is the **type of network** of the protocol, since there are various different types:

- WSNs
- DTNs
- FANETs
- MANETs
- etc.

Moreover, some protocols rely on **specific assumptions** such as having

- prior knowledge of traffic distributions
- node localization services
- the possibility of transmission errors

# Routing optimization context

## Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

# Routing optimization context

## Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

From this observation, the authors outlined 6 different **routing optimization contexts**:

1. **Data-packet driven optimization context**:
   - the transmission of packets happens *hop-by-hop* from $s$ to $d$
   - upon receival of a packet, $d$ sends back a feedback on the transmission

   After some amount of forwarded packets, the routing process converges to the selection of optimal paths

# Routing optimization context

## Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

From this observation, the authors outlined 6 different **routing optimization contexts**:

2. **Route request diven optimization context**:
   - a source $s$ that has data to send to $d$ first sends a *Route Request (RR)* packet, which is disseminated in the network
   - each node can decide to partecipate or not: if a node agrees to partecipate, it selects the next node to forward the RR to, and this process continues until $d$ is reached
   - once a path is found, all packets $s \rightarrow d$ are routed through this path
   - at the end of each transmission a feedback is sent back to $s$

   Most protocols in this category are extensions to the **AODV** protocol [PBRD03].

# Routing optimization context

## Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

From this observation, the authors outlined 6 different **routing optimization contexts**:

3. **Context request driven optimization context**: it describes P2P systems
   - in this context a node $s$ is interested in some content $C$ possessed by $d$
   - then, it sends a request to $d$ to receive data packets from $d$
   - nodes on the path $s \rightarrow d$ can then decide to forward the requrest to locate the requested content
   - when data packets containing $C$ are forwarded the relay nodes receive feedback and adapt their paths accordingly

# Routing optimization context

Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

From this observation, the authors outlined 6 different **routing optimization contexts**:

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

# Routing optimization context

## Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

From this observation, the authors outlined 6 different **routing optimization contexts**:

5. **Cluster driven optimization context**:
   - a *cluster* is a partition of the nodes that has a *cluster-head*
   - data patckets are transmitted from $s$ to $d$ following a clustered hierarchy of the network
   - a cluster-head determines the number of members that can join in the cluster depending on the resources
   - after a transmission, cluster-heads receive feedback and adjust their cluster size accordingly

# Routing optimization context

## Classification criteria: Context of use

The ability of a protocol to perform path selection optimally depends on

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

From this observation, the authors outlined 6 different **routing optimization contexts**:

6. **Routing protocol driven optimization context**:
   - a *central node* has a set of routing protocols candidates (e.g. AODV, DSDV, DSR, OLSR, GPSR, etc.) that can be used for forwarding packets to *slave nodes*
   - in each $\Delta t$ the central node selects a routing protocol and calculates the routing tables, which are then sent to slave nodes for internal configuration
   - then, a feedback is collected by the central node about the performance of the protocol, which may make the central node change the current routing protocol for the next $\Delta t$

   After some $\Delta t$, the system converges to the most adequate routing protocol.

# Unicast or Multicast

Classification criteria: Context of use

Categorizing between **unicast** or **multicast** approaches is a natural choice, given the inherent *overhead* that multicast routing protocols introduce.

# Unicast or Multicast

Classification criteria: Context of use

Categorizing between **unicast** or **multicast** approaches is a natural choice, given the inherent *overhead* that multicast routing protocols introduce.

Indeed, RL should be applied in multicasting scenarios only when links are sufficiently stable and/or partial delivery is allowed, otherwise convergence may be outright *impossible*.

# QoS metrics for optimization

## Classification criteria: Context of use

The choice of the **metrics** is one of the most important aspects of a protocol. When multiple metrics are utilized, they are *weighted* based on the importance — which depends on the target application.

# QoS metrics for optimization

Classification criteria: Context of use

The choice of the **metrics** is one of the most important aspects of a protocol. When multiple metrics are utilized, they are *weighted* based on the importance — which depends on the target application.

QoS metrics that have been addressed as objectives for RL-based routing include:

- **delivery rate**: average time to deliver a packet
- **delivery ratio**: proportion of packets successfully delivered
- **hop count**: average number of hops from source to destination
- **loss ratio**: proportion of packets not delivered

# QoS metrics for optimization

Classification criteria: Context of use

The choice of the **metrics** is one of the most important aspects of a protocol. When multiple metrics are utilized, they are *weighted* based on the importance — which depends on the target application.

QoS metrics that have been addressed as objectives for RL-based routing include:

- **bandwidth**: average bandwidth provided to sources
- **throughput**: average amount of bytes delivered in the entire network per time unit
- **path stability**: change in path between source and destination over time
- **energy consumption**: average energy consumption of the network

# QoS guaranteeing

## Classification criteria: Context of use

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

# QoS guaranteeing

## Classification criteria: Context of use

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

For instance, QoS guarantees are essential in *multimedia applications*, such as video streams and streaming services.

# Design characteristics

Classification criteria: Design characteristics

The second group examines the **internal design choices** behind each protocol.

# Design characteristics

Classification criteria: Design characteristics

The second group examines the **internal design choices** behind each protocol.

This encompasses how states, actions, rewards, and learning models are defined, as well as the architectural decisions that shape how an agent interacts with the network.

# Learning model

## Classification criteria: Design characteristics

In RL there are two possible approaches, **model-free** and **model-based** learning.

The vast majority of RL-based routing algorithms are **model-free**, since constructing a model requires knowledge about the environment that can be difficult to collect.

# Learning model

## Classification criteria: Design characteristics

In RL there are two possible approaches, **model-free** and **model-based** learning.

However a few algorithms are actually **model-based**, in particular

- some of them use *offline*-collected information of the environment model
- some others calculate and improve the environment model in an *online* fashion

# Learning model

## Classification criteria: Design characteristics

In RL there are two possible approaches, **model-free** and **model-based** learning.

Model based approaches are known to converge quickly, and thus can offer an interesting opportunity when the **speed of convergence** is a crucial requirement.

# Agent states

Classification criteria: Design characteristics

In order to apply RL to any optimization problem, we need some definition of the **state space**, the set of all possible states that the agent can be in.

# Agent states

## Classification criteria: Design characteristics

In order to apply RL to any optimization problem, we need some definition of the **state space**, the set of all possible states that the agent can be in.

The definitions of state spaces in the reviewed literature include:

- set of *nodes*, the most popular in RL-based routing protocols
- set of *grids*, used in grid-organized networks
- set of *couples*, relating to the dynamics of the nodes — for instance in VANETs a *couple* is a vehicle speed class together with its context of move (urban, highway, etc.)
- set of *paths* and their characteristics

# Actions spaces

## Classification criteria: Design characteristics

Protocols can be also classified based on the **action space** they define. This table contains the possible *single-type actions* and the corresponding *action 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 |

# Actions spaces

## Classification criteria: Design characteristics

Protocols can be also classified based on the **action space** they define. This table contains the possible *single-type actions* and the corresponding *action spaces*:

| **Action selection** | **Action space** |
|---|---|
| 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 |

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

The *action selection* strategies in RL-based routing include:

- **Greedy strategy**: only the highest Q-value is used for selection — this strategy may take a very long time to converge

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

The *action selection* strategies in RL-based routing include:

- $\varepsilon$**-greedy strategy**: in addition to the greedy strategy, the learner uses a small amount of randomness (that depends on $\varepsilon$) to explore new solutions — the most used form of selection

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

The *action selection* strategies in RL-based routing include:

- **Proabability based strategy**: similar to $\varepsilon$-greedy, but the value of $\varepsilon$ is calculated from the history of learning

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

The *action selection* strategies in RL-based routing include:

- **Bayesian network decision strategy**: the action selection uses *Bayesian networks* to better explore the solution space

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

The *action selection* strategies in RL-based routing include:

- **Devaluation of solutions based strategy**: the Q-values are periodically decayed in order to enforce exploration of the solution space

# Solution space exploration

## Classification criteria: Design characteristics

In RL the **Exploration vs Exploitation dilemma** is a well-known problem. Indeed, the *speed of convergence* strictly depends on the approach utilized to balance between *exploring* and *exploiting* the solution space.

The *action selection* strategies in RL-based routing include:

- **New neighbors first strategy**: newly discovered nodes are favored in next hop selection — this approach is particularly useful in *mobile networks*

# Agents collaboration

## Classification criteria: Design characteristics

The original version of RL defines each agent as *independent*, and only able to interact with the environment.

However, when applying RL to routing it is more effective to allow **collaborating agents**, indeed almost all reviewed protocols are based on this idea.

Note that *collaboration* only concerns *non*-RL related exchanges, such as the exchange of **link-state information**.

# Agents collaboration

## Classification criteria: Design characteristics

The original version of RL defines each agent as *independent*, and only able to interact with the environment.

Indeed, collaboration is so prevalent among the protocols in the literature that it is possible to categorize them w.r.t. how the nodes cooperate:

- **Reactive collaboration**: nodes only provide feedback upon reception of packet
- **Proactive collaboration**: similar to the *reactive* approach, but nodes additionally broadcast their link-state information through *Hello packets* to their neighbors

# Hybridization with other optimization techniques

Classification criteria: Design characteristics

Most of RL-based routing algorithms involve *pure* RL approaches, however some algorithms combine RL with other **optimization techniques** to speed up convergence.

# Hybridization with other optimization techniques

Classification criteria: Design characteristics

Most of RL-based routing algorithms involve *pure* RL approaches, however some algorithms combine RL with other **optimization techniques** to speed up convergence.

Hybrid optimization approaches include:

- Gradient methods
- Game Theory approaches
- *Bayesian network* methods
- Least square policy iteration

# Hybridization with other optimization techniques

Classification criteria: Design characteristics

Most of RL-based routing algorithms involve *pure* RL approaches, however some algorithms combine RL with other **optimization techniques** to speed up convergence.

Hybrid optimization approaches include:

- Neural Networks
- Genetic algorithms
- Ants optimization

# Numbers of parameters to tune

Classification criteria: Design characteristics

A well-designed protocol should be **easily tunable**. However, in addition to $\alpha$ and $\gamma$ a multitude of protocols utilize many more tunable parameters in their algorithms.

# Numbers of parameters to tune

Classification criteria: Design characteristics

A well-designed protocol should be **easily tunable**. However, in addition to $\alpha$ and $\gamma$ a multitude of protocols utilize many more tunable parameters in their algorithms.

Additionally, weights must be assigned whenever there are **multiple metrics** to consider. This may add too much complexity in terms of usability for the correct choice of the parameters.

# Numbers of parameters to tune

## Classification criteria: Design characteristics

A well-designed protocol should be **easily tunable**. However, in addition to $\alpha$ and $\gamma$ a multitude of protocols utilize many more tunable parameters in their algorithms.

Additionally, weights must be assigned whenever there are **multiple metrics** to consider. This may add too much complexity in terms of usability for the correct choice of the parameters.

Therefore the authors categorized the routing protocols also based on the **number of tunable QoS metrics and parameters** each paper offers.

# Reward functions

## Classification criteria: Design characteristics

The authors outline that the **reward function** is the most distinctive feature of existing RL-based routing protocols.

# Reward functions

## Classification criteria: Design characteristics

The authors outline that the **reward function** is the most distinctive feature of existing RL-based routing protocols.
Reward functions may be categorized into 3 classes:

1. **Test-based reward functions**: the reward is assigned a constant value, depending the outcome of some *test*.

   The most common test is checking if the packet was actually delivered to destination, which yields a *binary outcome* for the reward.

# Reward functions

## Classification criteria: Design characteristics

The authors outline that the **reward function** is the most distinctive feature of existing RL-based routing protocols.
Reward functions may be categorized into 3 classes:

2. **Linear reward functions**: they have the following general form

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

- $C$ is a constant factor that depends on the test chosen by the protocol
- $H$ is the number of metrics of the protocol
- $\omega_k$ is the weight of the $k$-th metric
- $M_k$ is the value of the $k$-th metric

# Reward functions

## Classification criteria: Design characteristics

The authors outline that the **reward function** is the most distinctive feature of existing RL-based routing protocols.
Reward functions may be categorized into 3 classes:

3. **Nonlinear reward functions**: this type is less common among RL-protocols, and they are designed with different forms of combinations of metrics depending on the specific application

# Q-value updating rule forms

Classification criteria: Design characteristics

Over half of proposed RL-based routing algorithms are direct applications of Q-learning as originally proposed by Watkins.

# Q-value updating rule forms

Classification criteria: Design characteristics

Over half of proposed RL-based routing algorithms are direct applications of Q-learning as originally proposed by Watkins.

However the remaining half of the protocols use procedures that either

- use a *modified* Q-value updating rule, or
- do not rely on Q-learning at all

# Performance aspects

Classification criteria: Performance aspects

Lastly, the third group evaluates protocols through their **performance outcomes**.

# Performance aspects

Classification criteria: Performance aspects

Lastly, the third group evaluates protocols through their **performance outcomes**.

This includes qualitative assessments of overhead, responsiveness, and the metrics used to judge routing effectiveness.

# Communication overhead

Classification criteria: Performance aspects

**Communication overhead** is a crucial part of the design of a routing protocol, which depends on how the protocol defines the exchange of relvant information between nodes of the network.

# Communication overhead

Classification criteria: Performance aspects

**Communication overhead** is a crucial part of the design of a routing protocol, which depends on how the protocol defines the exchange of relvant information between nodes of the network.

Therefore, the overhead of the reviewed protocols have been categorized from a *qualitative* point of view into:

- **null overhead**: there is no exchange of information between agents

# Communication overhead

Classification criteria: Performance aspects

**Communication overhead** is a crucial part of the design of a routing protocol, which depends on how the protocol defines the exchange of relvant information between nodes of the network.

Therefore, the overhead of the reviewed protocols have been categorized from a *qualitative* point of view into:

- **low overhead**: the chosen next hop returns a feedback in an explicit ACK packet, or it includes its feedback when, in turn, it (re)forwards the packet

# Communication overhead

Classification criteria: Performance aspects

**Communication overhead** is a crucial part of the design of a routing protocol, which depends on how the protocol defines the exchange of relvant information between nodes of the network.

Therefore, the overhead of the reviewed protocols have been categorized from a *qualitative* point of view into:

- **medium overhead**: this is the case of protocols in which the feedback from the destination is propagated to all hops through an explicit ACK packet

# Communication overhead

Classification criteria: Performance aspects

**Communication overhead** is a crucial part of the design of a routing protocol, which depends on how the protocol defines the exchange of relvant information between nodes of the network.

Therefore, the overhead of the reviewed protocols have been categorized from a *qualitative* point of view into:

- **medium overhead**: this is the case of protocols in which the feedback from the destination is propagated to all hops through an explicits ACK packet

# Communication overhead

Classification criteria: Performance aspects

**Communication overhead** is a crucial part of the design of a routing protocol, which depends on how the protocol defines the exchange of relvant information between nodes of the network.

Therefore, the overhead of the reviewed protocols have been categorized from a *qualitative* point of view into:

- **high overhead**: these protocols require that nodes periodically exchange link-state information

# State space overhead

Classification criteria: Performance aspects

Even if this aspect is sometimes neglected, RL-based algorithms require *memory* to store the **states of the agents**, and the number of states may be very high.

# State space overhead

Classification criteria: Performance aspects

Even if this aspect is sometimes neglected, RL-based algorithms require *memory* to store the **states of the agents**, and the number of states may be very high.

Hence, the protocols can be *qualitatively* grouped based on the **state space overhead**:

- **very low overhead**: when state space is the possible states of a packet

# State space overhead

Classification criteria: Performance aspects

Even if this aspect is sometimes neglected, RL-based algorithms require *memory* to store the **states of the agents**, and the number of states may be very high.

Hence, the protocols can be *qualitatively* grouped based on the **state space overhead**:

- **low overhead**: when the state space is the node IDs

# State space overhead

Classification criteria: Performance aspects

Even if this aspect is sometimes neglected, RL-based algorithms require *memory* to store the **states of the agents**, and the number of states may be very high.

Hence, the protocols can be *qualitatively* grouped based on the **state space overhead**:

- **limited overhead**: when the state space depends on external factors — e.g. the number of transmission power levels, the maximum number of available channels, etc.

# State space overhead

Classification criteria: Performance aspects

Even if this aspect is sometimes neglected, RL-based algorithms require *memory* to store the **states of the agents**, and the number of states may be very high.

Hence, the protocols can be *qualitatively* grouped based on the **state space overhead**:

- **high overhead**: when the state space is a list of whole paths with their current characteristics

# Action space overhead

Classification criteria: Performance aspects

Additionally, RL-based algorithms also require *memory* to store all the **possible actions** that agents can perform.

# Action space overhead

Classification criteria: Performance aspects

Additionally, RL-based algorithms also require *memory* to store all the **possible actions** that agents can perform.

Therefore again, the protocols can be *qualitatively* grouped based on the **action space overhead**:

- **low overhead**: when the action space depends on external factors

# Action space overhead

Classification criteria: Performance aspects

Additionally, RL-based algorithms also require *memory* to store all the **possible actions** that agents can perform.

Therefore again, the protocols can be *qualitatively* grouped based on the **action space overhead**:

- **medium overhead**: when the action space depends on the number of nodes in the neighborhood

# Action space overhead

## Classification criteria: Performance aspects

Additionally, RL-based algorithms also require *memory* to store all the **possible actions** that agents can perform.

Therefore again, the protocols can be *qualitatively* grouped based on the **action space overhead**:

- **high overhead**: when the action space depends on either
  - the number of *dynamic paths*
  - the number of or *predefined paths*
  - the number of *grids* in the network

# Action space overhead

Classification criteria: Performance aspects

Additionally, RL-based algorithms also require *memory* to store all the **possible actions** that agents can perform.

Therefore again, the protocols can be *qualitatively* grouped based on the **action space overhead**:

- **very high overhead**: when the state space depends on combinations of channels subsets or paths


# Proof of convergence

## Classification criteria: Performance aspects

In the optimization field, the **convergence** to optimal solutions is an *expected* property. Nevertheless, many existing techniques to solve multicriteria optimization problems are not guaranteed to reach the optimal solution.

Regarding RL-based algorithms, from the original work of Watkins it is possible to derive proofs of convergence, however:

- not all RL-based approaches are based on the standard implementation of Q-learning
- many papers rely on **additional assumptions** that "guarantee" convergence, but in real world scenarios it is hard to establish the **satisfiability** of such assumptions

# Proof of convergence

## Classification criteria: Performance aspects

In the optimization field, the **convergence** to optimal solutions is an *expected* property. Nevertheless, many existing techniques to solve multicriteria optimization problems are not guaranteed to reach the optimal solution.

In general, proving convergence rigorously remains an **open issue** for most protocols that are not perfectly Q-learning compliant.

Rather, convergence is usually assessed from **simulations** or just *stated as reachable eventually* without providing any proof of such claim.

# Table of Contents

Conclusion and challenges

# Conclusion

## Conclusion and challenges

RL is an efficient alternative to design routing protocols that

- provide higher level of QoS
- optimize resource utilization

# Conclusion

## Conclusion and challenges

RL is an efficient alternative to design routing protocols that

- provide higher level of QoS
- optimize resource utilization

However, some **challenges** still remain and should be investigated further to provide evidence on applicability of RL-based protocols *at large scale*.

# Proof of optimality

## Conclusion and challenges

As already mentioned, almost all reviewed papers *did not* convincingly address **proof of convergence**.

# Proof of optimality

## Conclusion and challenges

As already mentioned, almost all reviewed papers *did not* convincingly address **proof of convergence**.

Since convergence is such an important requirement in the context of optimization, it should be treated as a core property of any RL-based routing approach.


# Proof of optimality

## Conclusion and challenges

As already mentioned, almost all reviewed papers *did not* convincingly address **proof of convergence**.

Since convergence is such an important requirement in the context of optimization, it should be treated as a core property of any RL-based routing approach.

Indeed, without it there is no guarantee that the learned policy will stabilize or consistently produce optimal/reliable routing decisions.

# Speed of convergence

## Conclusion and challenges

Whenever large networks are considered, *space exploration* may take a very long time before optimal paths are discovered. This may result in poor end-to-end performance of the network.

# Speed of convergence

## Conclusion and challenges

Whenever large networks are considered, *space exploration* may take a very long time before optimal paths are discovered. This may result in poor end-to-end performance of the network.

**Convergence rates** should be investigated to provide **bounds** of delay for let users know when the network can or cannot provide *acceptable QoS levels*.

# Link-state information dissemination

## Conclusion and challenges

In most protocols, **link-state information** is used to calculate metrics, hence the convergence of the routing algorithms strictly depends on the *freshness* of disseminated information.

# Link-state information dissemination

## Conclusion and challenges

In most protocols, **link-state information** is used to calculate metrics, hence the convergence of the routing algorithms strictly depends on the *freshness* of disseminated information.

Therefore, the frequence of *Hello packets* should be addressed in order to find a compromise between **protocol overhead** and **values of reward**.

# Hybridization

## Conclusion and challenges

A few routing protocols have sufficiently reduced the search space by **hybridizing** the optimal-path search with external optimization techniques.

# Hybridization

## Conclusion and challenges

A few routing protocols have sufficiently reduced the search space by **hybridizing** the optimal-path search with external optimization techniques.

However, the authors underline that this strategy is not explored enough. RL should be used *jointly* with **other techniques** to provide more (soft) guarantees on exploration of the solution space.

# Hybridization

## Conclusion and challenges

A few routing protocols have sufficiently reduced the search space by **hybridizing** the optimal-path search with external optimization techniques.

However, the authors underline that this strategy is not explored enough. RL should be used *jointly* with **other techniques** to provide more (soft) guarantees on exploration of the solution space.

In recent years **Deep Learning (DL)** has been proposed to enable RL to scale to complex problems.

# Predicting traffic demands

## Conclusion and challenges

Learning in current RL-based protocols is mainly based on **network-oriented metrics** like

- delays
- loss rate
- transmission success
- mobility of nodes

# Predicting traffic demands

## Conclusion and challenges

Learning in current RL-based protocols is mainly based on **network-oriented metrics** like

- delays
- loss rate
- transmission success
- mobility of nodes

However, **predicting traffic** from sources to destinations would result in more efficient selection of forwarders. Indeed, in *supervised learning scenarios* traffic predicion has resulted in more efficient route selection.

# Cooperative learning

## Conclusion and challenges

Lastly, almost all proposed protocols are **independent-agent-based**. In fact even if the agents “collaborate” by exchanging link-state information, they don’t *learn cooperatively*.

# Cooperative learning

## Conclusion and challenges

Lastly, almost all proposed protocols are **independent-agent-based**. In fact even if the agents “collaborate” by exchanging link-state information, they don’t *learn cooperatively*.

To face complexity of future networks, the authors suggest to enable **collaboration** between agents to help design more robust and efficient learning approaches.


# Thanks for your attention

# Unicast or Multicast

Classification criteria: Context of use

## Example: a Multicast protocol

An example of a multicast protocol is the **FROMS (*Feedback Routing for Optimizing Multiple Sinks*)**, introduced by Forster and Murphy [FM07].

# Unicast or Multicast

Classification criteria: Context of use

## Example: a Multicast protocol

An example of a multicast protocol is the **FROMS (*Feedback Routing for Optimizing Multiple Sinks*)**, introduced by Forster and Murphy [FM07].

This was the first RL-based protocol for multicast routing in WSNs, and it operates as follows:

- it constructs a tree similar to a *Steiner tree* with the selected paths
- routing to multiple destinations is defined as the **minimum cost multicast tree** starting at the source and reaching all interested destinations
- the *cost* of the *Steiner-like tree* is defined as the number of one-hop broadcasts to reach all sinks

# QoS metrics for optimization

Classification criteria: Context of use

**Example: a multi-metric protocol**

The first protocol in the literature that considered *multiple metrics* is the **AdaR (*Adaptive Routing*)**, proposed by Wang and Wang [WW06].

# QoS metrics for optimization

Classification criteria: Context of use

**Example: a multi-metric protocol**

The first protocol in the literature that considered *multiple metrics* is the **AdaR (*Adaptive Routing*)**, proposed by Wang and Wang [WW06].

In particular, they considered **four QoS metrics** for path selections:

1. number of hops
2. residual energy
3. link reliability
4. number of routes crossing in a node

# QoS guaranteeing

Classification criteria: Context of use

**Example: a delay-aware protocol**

An example of a protocol that provides *soft* delay guarantees to delay-sensitive applications was introduced by Lin and Schaar [LS10], called **RL-RPC (*RL-based Routing and Power Control*)**.

# QoS guaranteeing

Classification criteria: Context of use

**Example: a delay-aware protocol**

An example of a protocol that provides *soft* delay guarantees to delay-sensitive applications was introduced by Lin and Schaar [LS10], called **RL-RPC (*RL-based Routing and Power Control*)**.

In this protocol, RL is used to learn **channel conditions**. Moreover

- at each node the protocol selects the best route and the best power to forward packets
- a packet is dropped when the *deadline* — included in the packet — can no more be satisfied

# Learning model

Classification criteria: Design characteristics

**Example: a moded-based protocol**

For instance, the **QGrid (Q-learning-based Grid routing)** protocol introduced by Li, Li, Li, et al. [LLL+14] is a model-based protocol for VANETs, since it is composed of two phases:

- *offline learning* of Q-values
- *online use* of the Q-value table to forward packets

A vehicle uses its offline Q-table to pick the *next grid* and forwards the packet to a neighbor in that new grid; if none exists, it forwards to a closer neighbor, or otherwise stores the packet until a new neighbor appears.

# Learning model

Classification criteria: Design characteristics

**Example: a moded-based protocol**

For instance, the **QGrid (Q-learning-based Grid routing)** protocol introduced by Li, Li, Li, et al. [LLL+14] is a model-based protocol for VANETs, since it is composed of two phases:

- *offline learning* of Q-values
- *online use* of the Q-value table to forward packets

The assumption of this protocol is that it is possible to infer the optimal path to reach some destination $d$ located in a given grid from the **history of inter-grid movements**.

# Hybridization with other optimization techniques

Classification criteria: Design characteristics

**Example: a hybrid protocol**

An example of a protocol that combines RL with other optimization techniques is **AdaR** (the first multi-metric protocol) [WW06].

# Hybridization with other optimization techniques

Classification criteria: Design characteristics

**Example: a hybrid protocol**

An example of a protocol that combines RL with other optimization techniques is **AdaR** (the first multi-metric protocol) [WW06].

In particular, AdaR uses **Least Squares Policy Iteration (LSPI)**, which indeed enables *faster convergence* to optimal solution without suffering initial parameter setting.