Summarizing

Applications of Multi-Agent Deep Reinforcement Learning with Communication in Network Management: A Survey

This article is a survey about the application of MADRL with communication in network management. It proves that with the increasing complexity of network management, traditional centralized network management modes have gradually become unable to meet the demand. Traditional network management technologies usually rely on heuristic methods, which are hard to scale for automation and intelligent management in future networks. The article highlights the importance of deploying distributed and decentralized network engineering to improve network performance using Multi-Agent Deep Reinforcement Learning (MADRL) as a solution. It also emphasizes that in some partially observable distributed multi-agent systems, the mutual influence of adaptation strategies renders the system vulnerable to non-stationary problems—where each agent’s learning continuously affects the environment, making coordination and convergence challenging.

For the application of MADRL, the survey focuses on several network management domains. In traffic engineering, MADRL is used to optimize routing paths for data packets and balance the load among switches, routers, and links. The survey introduces key works in this field; for example, in [31] a MADRL routing method is proposed where each Autonomous System (AS) acts as an agent. The results show that a system with agent communication performs better than one where agents rely solely on their local observations to make decisions. Additionally, in transmit power control for wireless networks, network entities must control transmission power to reduce interference with other agents. Through communication, an agent can learn not only its own conditions but also the interference it causes or the power allocation of other agents, thereby finding a balance between improving transmission power and reducing interference. The article also touches on network security, presenting works where coordinated agent communication helps defend against threats like jamming and DDoS attacks.

After discussing these applications, the survey details the communication strategies that underpin MADRL in network management. The article categorizes the information exchanged among agents as "communication content," which refers to what information—such as an agent’s state, action, reward, or strategy—is encoded in the messages. While every agent internally uses both state and reward for learning, designers may choose to share only specific pieces of information externally to enhance coordination and reduce communication overhead. Furthermore, the survey defines "communication objects" as the targets of this information exchange. It categorizes them into three types: Neighbor Agents (agents share information only with those directly connected or in close proximity), All Agents (agents broadcast information to every other agent in the system), and Central Agent (agents communicate solely with a designated central controller that aggregates and redistributes decisions). Finally, it reviews "communication message processing" techniques—such as concatenation (directly combining agent messages), neural networks (learning effective communication patterns), and averaging (simplifying communication while potentially losing critical nuances)—which integrate the exchanged messages into each agent’s decision-making process. Together, these communication strategies enable MADRL systems to overcome the limitations of partial observability and non-stationarity, paving the way for more adaptive and intelligent network management solutions.

**Analysis of Independent Learning in Network Agents: A Packet Forwarding Use Case**

This article explores the use of Independent Q-Learning (IQL) for packet forwarding in Named Data Networking (NDN). The authors investigate whether independent reinforcement learning agents (where each agent learns without coordination with others) can efficiently make forwarding decisions in a distributed network. They propose an Independent Deep Q-Network Forwarding (IDQF) strategy and evaluate its performance against the basic Best Route (BR) strategy in NDN. The study highlights major challenges with independent learning in network environments, including non-stationarity, partial observability, and delayed rewards.

Problem Statement:  
 Traditional forwarding in large-scale networks often relies on centralized or cooperative approaches that are costly, complex, and prone to issues like single points of failure. The article addresses the challenge of deploying independent, autonomous agents for packet forwarding. However, such independent learning methods face significant difficulties due to non-stationarity (as agents adapt concurrently), partial observability (limited local information), and delayed rewards (latency in feedback), which collectively hinder convergence to optimal forwarding policies.

Proposed Solution and Methodology:

The article proposes an Independent Deep Q-Network Forwarding (IDQF) strategy for packet forwarding in Named Data Networking (NDN).In this approach, each router acts as an autonomous reinforcement learning agent that makes forwarding decisions based solely on its local observations.

State Representation:

Local Observations:  
 Each router gathers local network measurements such as:

Round-Trip Time (RTT)

Interest satisfaction ratio

Number of retransmitted packets

Time Window:  
 Observations are collected over a time window [t−δt,t] to average out fluctuations and provide a stable estimate of the network conditions.

Action Space:

The agent’s decision involves selecting the optimal next-hop interface for forwarding incoming Interest packets by evaluating the Q-value of each possible interface using a Deep Q-Network (DQN).

Reward Function:

The reward for a given time step t is defined as:

RW(t)=−(M1 k=t−δt∑t RTT(k)+C×R(t)) where:

* M is the number of data packets received in the interval [t−δt,t],
* RTT(k) is the round-trip time of a packet at time k,
* R(t) is the number of retransmissions during this time window,
* C is a constant penalty multiplier.

The DQN is trained using experience replay with parameters such as a decay rate of 0.001 and a learning rate of 1.0 over 50 episodes (each 60 seconds long). The experiments span various traffic loads by testing five different Interest packet request rates.

Environment Tested:  
 The experiments were conducted using ndnSIM in the NDN context. The network topology used is the Sprint topology, consisting of 11 routers and 18 links with varied capacities (1 Mbps, 2 Mbps, and 5 Mbps) and link delays. The testing scenario involves a single consumer requesting data from a single producer, with Interest packet rates varying from 100 to 300 packets per second, representing different levels of network load.

Results:

The IDQF strategy consistently underperforms compared to the basic Best Route (BR) approach, showing 10%-36% lower throughput and higher average end-to-end delays across all tested traffic rates.

Advantages:

Scalability: Unlike centralized RL, IDQF does not require coordination among agents, making it scalable for large networks.

Realistic Testing in NDN:

Identification of Key Challenges in Independent Learning: The article systematically outlines the limitations of IQL in networking, which helps guide future research.

Disadvantages:

Non-Stationarity Issues

Delayed Rewards Reduce Learning Efficiency: Since network conditions change dynamically, the delayed impact of forwarding decisions makes RL unstable.

Partial observations.

Multi Agent Reinforcement Learning Independent vs Cooperative Agents

The article investigates whether reinforcement‐learning agents that cooperate can outperform independent agents that learn without communication. It examines different ways in which agents can share information—such as instantaneous sensory data, complete episodic experiences, or learned decision policies—and studies the trade-offs associated with such cooperation.

**Problem Statement**

The central question the article addresses is: Given the same number of reinforcement learning agents, will those that cooperate learn more efficiently and perform better than independent agents that do not exchange information?

**Proposed Solution and Its Methodology Details**

The author proposes three main cooperative mechanisms:

* **Sharing Sensation:**  
  Agents share instantaneous sensory information such as their current perceptions (e.g., the relative distance of the closest prey), actions, or rewards.
* **Sharing Episodes:**  
  Agents exchange entire episodes, which are sequences of (sensation, action, reward) tuples gathered during trials. This enables an agent to learn from another agent’s experiences, effectively increasing its learning data.
* **Sharing Learned Policies:**  
  Agents exchange or average their learned decision policies, which allows them to complement each other’s exploration of the state space.

**Case 1: Sharing Sensation**

* **Scenario & Technical Details:**  
  In this case, each hunter (agent) primarily uses its own sensory input to determine the relative position of the nearest prey—represented by a coordinate pair (x, y) that indicates the distance along the x‐axis and y‐axis. Cooperation is introduced by allowing a separate “scout” agent (or another hunter acting as a scout) to share its sensory information with the hunter. For example, if the hunter cannot see any prey due to its limited field of view (e.g., visual depth of 2 gives a 5×5 area), it can use the scout’s perception to effectively “extend” its sensory range. Technically, this means the hunter’s state representation is augmented by the scout’s data, increasing the number of perceptual states—from a limited set (e.g., 5² + 1 = 26 states without a scout) to a much larger set (up to 442 states in a 10×10 grid if the scout covers the full grid). The extra sensory data helps the hunter take more informed decisions, resulting in fewer steps to capture the prey.

**Case 2: Sharing Episodes or Learned Policies**

* **Scenario & Technical Details:**  
  Here, rather than only sharing instantaneous sensory data, agents exchange longer sequences of experience (episodes) or even share their learned decision policies. An episode is a series of (sensation, action, reward) tuples gathered during a trial. When one hunter shares an episode with another, the receiving agent can "replay" that sequence in its own learning process, effectively doubling its learning experience. Alternatively, agents may periodically average or share their current policies (i.e., their Q-values or decision rules). Technically, this cooperation speeds up the convergence of learning because each agent benefits from the exploration done by its peers. However, this approach introduces communication overhead and requires that the agents’ state representations are compatible (which may be easier for homogeneous agents than for heterogeneous ones).

**Case 3: Joint Tasks Requiring Cooperation**

* **Scenario & Technical Details:**  
  In joint tasks, a single hunter is not enough to capture the prey; cooperation is mandatory. For example, two hunters might need to coordinate their actions to surround or herd a prey. In these scenarios, agents can share not only their instantaneous sensory information but also coordinate their movements (for example, by actively sharing their relative locations). This joint cooperation can be implemented in two modes:
  + **Passive Observation:** Agents simply observe each other and adjust their own policies accordingly.
  + **Active Mutual Scouting:** Agents actively share their sensory information (like their current relative positions to both the prey and each other), which enlarges each agent’s effective state space. Although this increases the complexity of the state space, it can lead to significantly better performance once the agents learn to coordinate.

**The Environment Tested**  
Experiments are conducted in a discrete grid world (for example, a 10×10 grid). In this environment:

* Hunters have limited visual fields defined by a “visual depth” parameter.
* Prey move randomly.
* Tasks vary from scenarios where a single hunter can capture the prey to joint tasks that require two or more hunters to coordinate their actions.

**The Results of the Article**

* Cooperative agents that share episodes or learned policies tend to learn faster and converge to effective strategies more quickly than independent agents.
* In joint tasks cooperative methods eventually lead to a significantly lower number of steps per trial compared to independent learning.
* However, the initial learning phase for cooperative agents may be slower due to the complexity of processing a larger state space and handling additional communication overhead.

**Relevance to my PFE: Multi-Agent Deep Reinforcement Learning for Forwarding Strategy in Named Data Networking (NDN)**

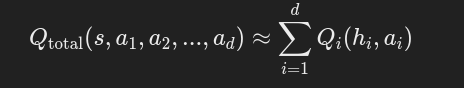
Based on the article, we can design a forwarding strategy in NDN. Routers act as agents that make forwarding decisions based on their local state. Similar to the hunter-prey scenario, routers could benefit from cooperative approaches. For example, they can share sensation to overcome the challenge of partial observation, share policy decisions so that less experienced agents can learn from experts thus speeding up training or share episodes (tuples of sensation, action, reward) to enhance learning. Although these approaches help agents make more optimal decisions, they do incur costs in terms of increased communication overhead and a larger state space.

Overview of the Article  
The article tackles the challenge of cooperative multi-agent reinforcement learning where all agents share a single team reward. Traditional approaches either use fully centralized learning which suffers from the exponential growth of the joint action space or independent learning which struggles with credit assignment and spurious rewards due to partial observability. The authors propose Value-Decomposition Networks (VDN) as a solution.

Problem Statement  
In cooperative settings, each agent only receives local observations but the reward signal is global. This makes it difficult for individual agents to understand how their actions contribute to overall performance. Independent learners tend to misattribute rewards, while fully centralized methods are computationally intractable. The article asks: How can we decompose the team’s joint Q-function so that each agent learns a useful local Q-function that, when summed, approximates the total expected reward?

To solve these problems, the authors propose Value-Decomposition Networks (VDN), which break down the total team Q-function into individual agent**-specific** Q-functions. Below, I provide the motivation, the formal mathematical formulation, and why VDN is a solution to the problem.

Value-Decomposition Networks (VDN) propose an approximation:



where:

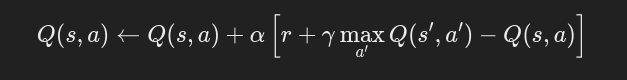
* Qtotal(s,a1,a2,...,ad)Q\_{\text{total}}(s, a\_1, a\_2, ..., a\_d)Qtotal (s,a1 ,a2 ,...,ad ) is the true joint Q-value.
* Qi(hi,ai)Q\_i(h\_i, a\_i)Qi (hi ,ai ) is the individual Q-value for agent iii, which depends only on agent iii’s local history hih\_ihi and action aia\_iai .

Why does this help?

* Each agent now only needs to learn its own local Q-function.
* The joint Q-function is automatically formed by summing up individual Q-values.
* This ensures that learning remains efficient and scalable.

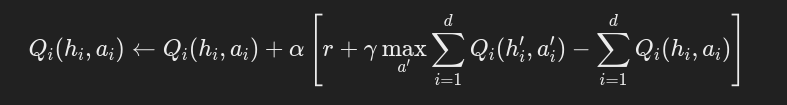
Since we do not have individual agent rewards, we train each agent’s Q-function using gradient backpropagation from the total team reward:

1. Q-learning update rule (in a single-agent case):



where α\alphaα is the learning rate.

1. For Value-Decomposition Networks, the update rule becomes:



* 1. The total team reward rrr is used to update all individual Q-functions.
  2. This update ensures that each agent learns a Q-function that contributes to the total team success.

The approach is evaluated on several two-dimensional grid-world domains inspired by Leibo et al. (2017), featuring:

* Partial Observability: Each agent only sees a small observation window (e.g., a 3×5×5 block).
* Tasks: The environments include tasks such as Switch (coordinating in narrow corridors), Fetch (synchronizing pickup and drop-off actions), and Checkers (managing asymmetric rewards between agents).

Results of the Article

Value-Decomposition Networks significantly outperform both fully centralized agents and independent learners.

The architecture successfully learns to decompose the joint Q-function into meaningful per-agent components, which helps in addressing spurious reward signals and improves overall coordination.