Comparing reinforcement learning techniques for embodied co-learning in a human-robot team

MSc Literature Study

by

Hugo W. Loopik

***Abstract* -**

This literature review focuses on embodied human-robot co-learning and how to achieve this using existing reinforcement learning techniques. To explore this the following two research questions are composed:

R1 What criteria are relevant for reinforcement learning in order to achieve co-learning in a human-robot team?

R2 What reinforcement learning technique is most suitable for human-robot co-learning according to the found criteria?

Relevant literature was studied. First we answered the R1 by defining three general criteria that ensure a reinforcement learning technique is suitable for co-learning: *C1 - Real-time Embodied learning*, *C2 - Reward based on performance* and *C3 - Adaptability*.

Mutual dependence, a shared group goal, observablility and explainability are the foundations of an interdependent relationship, from which the cascading effect of opportunistic dependencies arises. This contributes to the level of interdependence, and therewith to the collective-effectiveness, allowing the team to learn and operate as one integrated unit, which is crucial for co-learning. This leads to three criteria that are ensure for the emergence of an interdependent relationship: *C4 - Shared group rewards*, *C5 - Observability* and *C6 - Explainability*.

Four promising reinforcement learning techniques where selected to be explored using these criteria. To answer R2 an in-depth discussion was held to compare the four techniques according to the defined criteria.

MAXQ value decomposition is the most suitable reinforcement learning technique for human-robot co-learning according to the six criteria.

Student number:	4478231
Supervisiors:	Prof. dr. ir. D.A. (David) Abbink
	Dr. L. (Luka) Peternel
	Ir. E.M. (Emma) van Zoelen

TU Delft

Contents

1 Introduction

From smart vacuum robots and self-driving vehicles to industrial robotic arms, interaction between robots and humans is part of everyday life in modern society. With robots becoming more autonomous [1] and collaborative robots becoming more common as well [2], it is not surprising that in the last decades robots are taking evermore responsibility within that collaboration [3, 4]. Robots and humans operating as one collaborative team, can only be achieved if robots have increasing autonomy, because team members need to adapt to each other [5] and be able to share responsibilities [6]. Furthermore, team performance is highly influenced by the collective effectiveness of the team and the level of interdependence [7]. Interdependence between team members is present when there is a symbiotic relationship, and the team members depend on each other to increase task performance and efficiency [8]. In order to create good performing human-robot teams, robots should not be designed to learn autonomously, but be designed to learn as a team member that is part of an interdependent relationship. This learning as team members is called co-learning.

A handful of research has been conducted investigating how to design for achieving co-learning [8–11]. However, these researches have focused mostly on the high level theoretical requirements for co-learning and on virtual implementations only [10].

On the other hand there is research by Shafti [12] that shows that physical implementation of co-learning is possible on embodied robots. It achieves this by using the reinforcement learning technique called Soft Actor Critic (SAC) [13]. Shafti however, does not focus on the requirements needed to select machine learning techniques for embodied co-learning. It just uses SAC without a analysis of possible alternatives.

So there is literature that describes research into the theoretical side of how co-learning can be achieved, and there are some practical implementations of co-learning that show it is possible to achieve human-robot co-learning. However there is a research gap between these theoretical requirements and the actual practical implementations. For instance, there is no literature written to date, that documents research that investigates what learning algorithms are best to use for a robot that is designed for co-learning applications. In this review we aim to fill this research gap, in order to contribute to the development of practical implementations of co-learning in human robot collaborative teams. So, the goal of this research is to explore how humans and robots can co-learn as one collaborative team, using existing machine learning techniques found in the current literature.

1.1. Scope

This literature review focuses on finding existing machine learning techniques that could be used for embodied co-learning. To further define the scope of this research a few decisions have been made:

Firstly, to fit the research gap described above, the scope will be limited only to embodied robots. The word embodiment is used in different ways throughout the social robotics community [3, 14]. In this review the word embodiment is used with the same meaning as done by Kober [15] and Akalin [16], referring specifically to the robot having a physical body. This is in contrast to how for instance Fong [3] uses it, where a robot can be embodied virtually as well. Physical embodiment plays a big role in social-robotics and HRI [16]. As shown by Jung [14], humans will evaluate robots, as well as the interaction with them, more positively when the robot is physically embodied than when it is not. Accordingly, this research is focused as well on a physical robot.

Secondly, because the development of co-learning is still in its infancy, the human-robot teams considered in this review are limited to the most basic team, i.e. consisting of one human and one robot.

Lastly, we focus on reinforcement learning (RL) techniques only. RL is a form of machine learning that learns by trail and error. The algorithm learns what actions to take in what scenarios by getting rewarded for desired behaviour [17]. There are two reasons why we only focus on RL. The first reason is that the human and a robot have to learn a task together as a team, and therefore RL is the most logical choice form the available machine learning techniques, as it is the closest to the way humans learn new tasks. As RL learns by trial and error, it is a form of machine learning that is very similar to how humans learn [18, 19]. This makes it suitable for this application.

1.2. Research Questions

Before RL-techniques can be selected and compared based on their ability to co-learn, we first need to define which aspects of an RL-technique are important for embodied co-learning. Then using these aspects, existing reinforcement learning techniques can be selected and compared. So, in order to research what existing RL-techniques can be used to achieve embodied co-learning in a human-robot team, two research questions are formulated:

R1 What criteria are relevant for reinforcement learning in order to achieve co-learning in a human-robot team?

R2 What reinforcement learning technique is most suitable for human-robot co-learning according to the found criteria?

1.3. Research plan

To answer these research questions, this review is split in three parts which is documented in three chapters. In the first chapter, chapter 2, the main focus is to answer research question **R1**. The aspects of RL-techniques that are important in order to achieve co-learning are explored, and criteria on which RL-techniques could be selected and compared are constructed from these aspects. This is done in two parts, first, in section 2.2, general criteria based on the scope of this review, and the possibilities of RL in itself are discussed. Then, in section 2.3, criteria are constructed based on the necessities of building an interdependent relationship in a human-robot team in order to achieve collective effectiveness. Lastly, in section 2.4, the found criteria are listed and recapitulated.

To find an answer to **R2**, four different existing RL-techniques that seem promising for achieving co-learning are explained along with their relevant qualities. This is documented in chapter 3.

Next, in the discussion of this review (chapter 4), these four techniques are compared using the criteria from chapter 2. This is done by giving each RL-technique a score for each criterion, using a five point rating scale. An overview of these results is shown in Table 4.2. Explanations of these ratings are given per criterion in section 4.1 to section 4.6. The results are subsequently discussed in section 4.7 together with a retrospective review on the found criteria.

Lastly, a conclusion is drawn in the form of answers to the two research questions. This can be found in chapter 5 along with other findings and plans for future work.

1.4. Methodology

To answer the research questions above, relevant literature was scouted, read, absorbed and digested. Before searches where performed to find literature, a list of 11 papers provided by the cognitive robotics department of the TU Delft was used to get an overview of the subject. Using cross-references from these papers and general searches on google scholar, this list was expanded to 24 relevant papers that where studied to form the general scope and direction of this research. With this learned background information, more in-depth searches where conducted using relevant variations and combinations of the following key-words:

”*human-robot*”, ”*team*”, ”*co-learning*”, ”*collaborative*”, ”*reinforcement learning*”, ”*multi-agent*”, ”*embodiment*”, ”*real-time*”, ”*team responsibility*”, ”*interdependence*”, ”*team dependence*”, ”*collaboration*”, ”*interaction*”, ”*co-active*” and ”*joint activity*”.

During these searches papers where first selected on title after which the abstracts where read to evaluate the relevance of each paper. This selective in-depth search resulted in 59 relevant papers. One of the found publications was a recent survey on RL-approaches in social robotics by Akalin [16]. Using this survey 10 more papers of relevant RL-techniques where added to the list of literature, resulting in a total list of 69 papers. These where all studied in more detail to find answers to our two research questions **R1** and **R2**. From the found literature, 43 papers where actually used to answer the research questions. The eight most useful papers are the following citations: [7, 8, 10, 16, 20–23].

2 Criteria for RL-techniques to be used in embodied human-robot co-learning

Different types of RL-techniques suit different learning problems. Although, the aim of this research is not to find the RL-technique for one specific problem, we are interested in what aspects of RL-techniques make them suitable for what type of the learning problems. For instance a robot learning how to navigate a virtual environment might require a different RL-technique than a robotic arm that is physically interacting with a human. The aim of this study is to compare suitable RL-techniques for an embodied robot that co-learns with a human in a team.

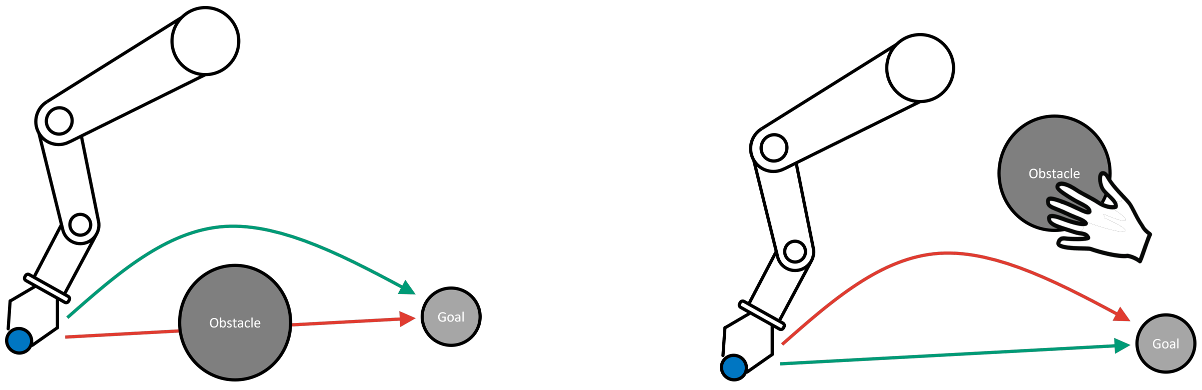
In this chapter we focus on how to compare RL-techniques. In order to select and compare different RL-techniques, criteria are composed. This is done in two parts; In section 2.2 criteria based on key aspects for co-learning an embodied task are composed. Then, section 2.3 focuses on which aspects are important to create an interdependent relationship between human and robot. In that section it is also explained in more detail what an interdependent relationship is, how this is created, and why this is important in order to allow for co-learning. In order to find the relevant criteria, the aspects for interdependence will first be elaborated, after which the criteria will follow.

In order to compose criteria based on a certain concept or aspect, sometimes these will be explained first on a higher lever of abstraction, before projecting it onto a human-robot team. Interdependence, for example, will first be viewed from an agent-agent perspective, where it is not relevant whether the agent is human or robot, before it is projected onto a criterion. Finally, in section 2.4 a list of the criteria and how they can be met is presented. However, first in section 2.1 an example of a robot and a human co-learning an embodied task will be given in order to help provide context to some of the concepts explained later in this chapter.

2.1. Running example

To help make some of the concepts explained in this chapter more clear, we will introduce a running example of a robot and a human co-learning an embodied task: a robotic arm and human have the task of moving a blue ball to a goal position as efficiently as possible (see Figure 2.1). The robot knows where its end-effector, and thus, where the blue ball is. It also knows where the goal position is. Furthermore, the robot can detect obstacles and has the ability to learn to move around them. The human has the ability to help the robot, but it has to learn how. The human can for instance guide the arm to learn a more efficient trajectory, or it can change the environment to make the robot do the task more efficient.

The scenario displayed in the figure is that the robot tries to move the ball in a straight line to the goal position at first, realising quickly that this is not possible. Then the robot learns that it should move around it, as shown in Figure 2.1a. Simultaneously, the human learns that it can move the obstacle out of the way, as can be seen in Figure 2.1b. Now the robot can learn to take the shorter, straight path to the goal position. This a is an example of co-learning because both agents learn to execute one team task, while collaborating as team members. Note that this example is deliberately very simple as it is meant to make specific concepts more understandable and is not meant as a well designed collaborative task for a human and a robot to co-learn.



(a) Policy A (b) Policy B

Figure 2.1: A robotic arm and a human are collaborating to move the blue ball (left) to a goal position (right) as efficiently as possible. In both sub-figures, the green trajectory shows the preferable path. In subfigure b the human team member has moved the obstacle out of the way.

2.2. General aspects of embodied co-learning in a human-robot team

In this section we discuss some key criteria for RL-techniques suitable for an embodied robot collaborating with a human in a team. To do so, subsection 2.2.1 will explain the concepts of embodiment and real-time RL together with the challenges that they bring. Then the possibilities for creating a human-robot team will be discussed in subsection 2.2.2. Lastly in subsection 2.2.3 we go deeper into why an RL-agent must stay adaptable in order to make co-learning possible. Throughout this section the first three criteria, ”*Real-time embodied learning*”, ”*Reward based on performance*” and ”*Adaptability*”, will be composed.

2.2.1. The conflict between physical embodiment and real-time learning

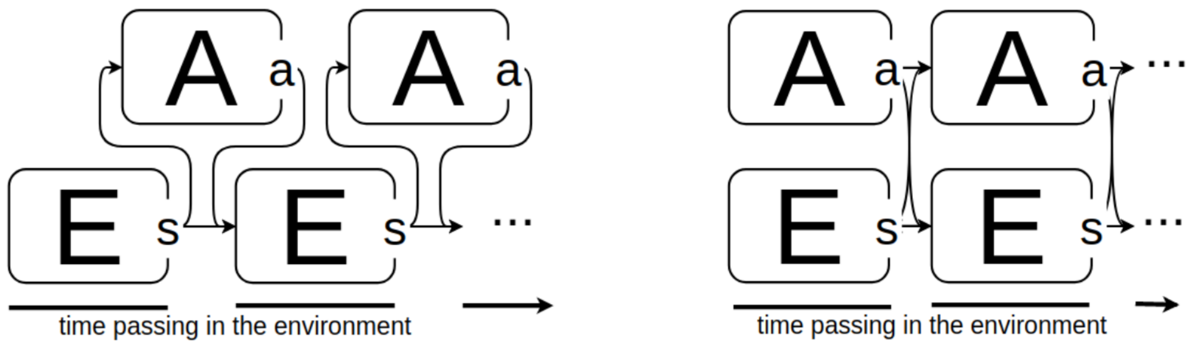
Co-learning can only exist when the team is able to operate as a team, instead of just being a set of two individuals. To achieve this, both agents have to decide and act in a similar time window. In the running example for instance, the human might move the obstacle within seconds. When the robot however takes minutes to even try the straight line, there is not much co-learning going on, as the robot reaches location of the obstacle after it has been removed. So, taking this example the robot learns to do the task as if the obstacle, nor the human was ever there. The same is true about the time window in which the agents learns. Only when both agents can interact and change their behaviour to match the strategy of the other agent at a similar time window, they have the potential to become an interactive team. Hence, for a robot to co-learn in a human-robot team the robot should operate and learn in the similar time window as a human would. This is called learning and operating in real-time.

To achieve real-time learning, computational times should be limited, which can be challenging. One way to overcome these challenges, real-time learning could be achieved by keeping the learning problem simple, so learning can occur without much computing time [10, 24]. However, a learning task that is physically embodied can make a learning problem more complex. This can make real-time learning within a physical embodied environment difficult.

In this section, this conflict will be explored in more depth. This is done by highlighting the challenges of embodiment first, followed by the challenges of real-time learning, along with some examples of how to counteract these challenges. This section concludes with the a resulting criterion, *Embodied Real-time learning*, to which suitable RL-techniques for co-learning should adhere.

An RL-agent learns by selecting better actions based on its current state and its environment. For this learning to take place the agent has to be aware of the state of the environment in order to know which better action to take. Real-world environments, however, come with a lot of uncertainties. Not only can measurements be noisy or imprecise, but it is often the case that some desired state variable cannot be measured directly. Indirect variables should then be measured to describe that part of the state. These indirect measured variables could either be manipulated to represent the desired part of the state, or could be used directly as an alternative state variable, meaning the algorithm should learn the causality between the measured values and the state. In both cases the learning problem gets bigger due to the embodiment of the task. This causality between measured values and the state of the robot is a model.

Reinforcement learning techniques used in physically embodied robots can be categorized into model-based and model-free techniques [16]. Model-based techniques, use a model of the real-world system dynamics. These models can be hard to obtain, and they have to make assumptions and simplifications. Some RL-techniques are designed specifically to learn such models. Another used strategy is to pre-train the system and learn the model in a simulation before implementing it in the physical world [16]. Furthermore, imperfections in the model, can accumulate over time, affecting the agents behavior [15]. Summarising, although model-based RL can be effective, it adds to the complexity of the technique. Therefore, in some cases, model-free RL can be more fitting than model-based RL because of the challenges, uncertainties and complexities that arise when creating a sufficient accurate model [16]. For the same reason, most applications on physically embodied robots to date are not model-based [16].


A
a
A
a
E
s
E
s
...
time passing in the environment
A
a
A
a
...
E
s
E
s
...
time passing in the environment

(a) turn-based (b) real-time

Figure 2.2: A visual representation of turn-based and real-time RL. E is the environment represented by state s, and a represents the action chosen from action-space A. Note that the action in turn-based is based on the current state and in real time it is based on the previous state. [23]

As explained in the beginning of this section, it is important for an RL-technique to be able to learn and operate at the same time window as a human for it to be able to co-learn with a human team member. This means that the RL-technique should have the ability to continuously learn, while operating in real-time. Real-time RL can be achieved in different ways. Ramstedt [23] distinguishes real-time RL from turn-based RL. In standard turn-based RL the action a_i is chosen based on the corresponding state s_i. The resulting state-action pair (s_i, a_i) determines the next state s_{i+1}. This means that the next action can only be selected when the turn is completed, and the next state s_{i+1} is reached. This is visualized in Figure 2.2a. When this action selection, however, takes any amount of computing time the process would not be real-time anymore. So, when computing times for sufficient performance are not negligible, operating in real-time while learning can be challenging. Real-time RL is often achieved in simulations, where the system is paused during computing times [23]. This is not possible in combination with embodiment.

There are, however, other strategies to achieve real-time RL without having to reduce the computing time; Hester [25] developed an algorithm that can take actions continually in real-time while learning at the same time. The learning, or updating of the policy, happens in the background parallel with the actions. This way, the robot will operate in real-time, while updating its policy. The background learning, however, here still happens in episodes. Ramstedt [23] achieves real-time learning, by developing RL-techniques that selects the actions before the next state can be observed. Their technique uses the previous state-action pair to predict the current state, on which the learning is based. This is visualized in Figure 2.2b, and further explained in section 3.4.

Alternatively, near real-time RL can be achieved with turn-based techniques, when the computing times for updating the policy and selecting the action are fast enough [10, 24]. This is only possible when the learning problem is not too complex. This is an acceptable solution for the scope of this review, as the aim of this review is to find an RL-technique with the ability of co-learning. For now the ability of learning and operating in real-time is more desirable than the ability of learning complex tasks.

In short, for co-learning to occur, the robot should be able to learn while operating in real-time, and the learning itself must be possible in a similar time window as the learning of the human agent. If the human would, for instance, be able to learn its task within a few minutes, while the robot needs an hour, co-adaption and co-learning are not feasible. In other words, the embodied RL-agent should be able to learn efficiently. This can be conflicting with the effects of embodiment described above, especially when for instance a model has to be learned first. This allows the definition of the first criterion for comparing RL-techniques: ***C1** - Real-time embodied learning*. This criterion is met if the RL-technique is able to learn an embodied task in real-time. As described above, there are multiple ways of achieving this. Some techniques are specifically designed to achieve this, while others have to rely on keeping the learning problem simple.

2.2.2. Human-Robot team

As stated in chapter 1 robots should be designed to learn as a team member for co-learning to occur. In other words, when choosing an RL-technique for a robot that has to form a team with a human, an RL-technique should be chosen that makes this is possible. Fortunately there are lots of RL-techniques that are designed to interact collaboratively with humans [16]. In order to compose a criterion based on the human and the robot being able to form a team, the different possible RL-techniques that allow interacting with humans will be explored in this section.

Akalin [16] organizes RL-techniques in Social Robotics into three categories: *interactive RL*, *intrinsically motivated techniques* and *task performance driven techniques*. This can be seen in Figure 2.3. All of these techniques include a form of human robot interaction, however the role of the human within the learning algorithm is different in each of these three. The category *intrinsically motivated techniques* will not be discussed here, because it solely includes techniques designed for the well-being of robots which is outside the scope of this review. *Interactive RL* and *task performance driven techniques*, however, make an interesting distinction when focusing on a human-robot team.

The learning algorithms of techniques within the category *Interactive RL* are based on feedback from interaction with the human. This feedback can be explicit, where the human directly gives the robot a reward, or implicit, in which case the algorithm changes its behavior due to the action of the human without this action being a reward. An example of *interactive RL* using explicit feedback is TAMER [26]. In this technique the human can give the RL-agent a positive or negative reward in order for the agent to learn.

In the example from Figure 2.1, the human should give the robot a positive reward for choosing a green path, and a negative reward for choosing a red path. This way the robot learns which path is preferred by the human. There are also *interactive RL*-techniques that use implicit feedback. COACH [27] is an example of this. In this technique, the human has the ability to give the robot constructive feedback by giving corrections when needed. Using the same example, the human can for instance move the robot hand along the preferred path when it takes a path that is not optimal according to the human. Such *interactive RL*-techniques, are based on an actor-critic relationship where the human critic is treated as an expert, providing feedback, and the robot is the actor, learning from or adapting to the human. However, for co-learning, the goal is to learn together, and adapt to each other, where the human learns as well and can therefore not always be treated as an expert. Moreover, in order to create a good team, team members should be equals without such a hierarchy [6, 7].

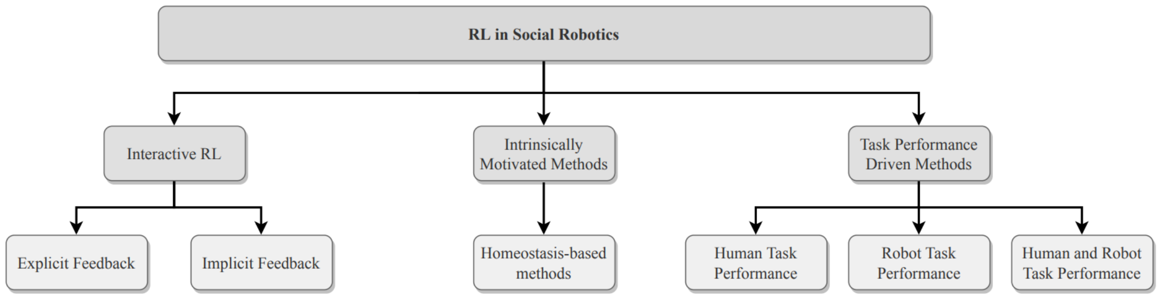

RL in Social Robotics
Interactive RL
Intrinsically Motivated Methods
Task Performance Driven Methods
Explicit Feedback
Implicit Feedback
Homeostasis-based methods
Human Task Performance
Robot Task Performance
Human and Robot Task Performance

Figure 2.3: The categorisation of RL-techniques that are used in social robotics by Akalin [16].

The remaining category is *Task performance driven techniques*. Here, the robot is rewarded based on the performance of the task. As depicted in Figure 2.3 Akalin distinguishes three subcategories for which two are relevant for the robot: reward based on its own performance, or reward based on the collaborative performance with the human. Especially this last sub-category, *Human and Robot task performance*, has the potential for co-learning, as the human and robot would have a collaborative group goal [7]. In other words, when the robot and the human are both reward on the same performance, they intrinsically are motivated to collaborate and work as a team. Benefits of a collaborative group goal and why this contributes to the ability to co-learn will be discussed in more detail in section 2.3. Furthermore, rewarding the agents based on overall performance would be most feasible for co-learning since the optimal policy of the robot is dependent on the behavior of both team members [10]. This forms the second criterion on which RL-techniques can be compared: ”***C2*** - *Reward based on performance*”. This criterion assures that the human and the robot will be team members. Ideally, the reward is based on the collaborative performance of the team. This will be captured in the criterion ***C4*** - *Shared group rewards*, which is further discussed in subsection 2.3.3.

2.2.3. Adaptability

In order to learn as a team it is crucial that the robot stays adaptable, as in co-learning both members of the team are learning and thus changing their behavior. When one team member changes its behaviour it can influence the optimal policy for the other agent. Take for instance the robotic arm from Figure 2.1. It is moving an blue ball from one place to another. At first it tries to move in a straight line but it encounters an obstacle in the way. Next it learns to maneuver around the obstacle, as in Figure 2.1a. After a while, the human team member realizes it can move the obstacle out of the way. At this point the robot should change its policy back to the straight line, even though this policy was not optimal when explored earlier, as in Figure 2.1b. So, because the human has learned, and adapted its behavior, the behavior of the robot should change as well.

The nature of most RL-techniques is to converge to some kind of optimal behavior, and to exploit that when it is found. To do so, the algorithm must explore the state-action-space first. When this space is sufficiently explored the algorithm can start to exploit the learned policy. Determining the balance between exploration and exploitation can be done in different ways. The most basic way to do so is using ϵ-decay, in which ϵ represents the probability that the agent will explore. At the beginning of the learning process ϵ has a value of one meaning it will always explore. During learning ϵ will decay towards zero, after which the policy will only exploit the learned policy.

There are also other, more efficient, approaches than ϵ-decay such as the Boltzmann strategy [28], or heuristic approaches based on an optimistic assumption [29]. Kapetanakis [30] showed that these approaches are suitable for an environment with two agents and that, given the right circumstances, the behavior of both agents could converge to a global optimum, even when this globally optimal policy seems to be worse than other strategies at first. This is done by rewarding the agents the same reward dependent on both their actions. When one agent changes their policy it influences the reward of the other agent as well. That way, they could influence each other's policy. So having *Shared group rewards* can also help with creating adaptability. This criterion (***C4*** *- Shared group rewards*) will be further discussed in section 2.3. Kapentenakis [30] however also found, that this strategy alone is not enough when the environment is not fully observable. The same is true when the actions of the other agent are only partially observable. The reason being that all these strategies use the same principle of converging to state of exploiting a seemingly optimal policy. This means that the policy does not necessarily changes when another policy becomes more optimal due to learning of another team member.

Looking at the robotic arm example: when the robot has learned that it is more optimal to move around the obstacle, it would later not notice that the human is has moved the obstacle out of the way, and therefore it would keep exploiting the learned policy. This would be a problem for co-learning since it might not always be observable that the other agent has learned something new. In other words, ***C3*** *- adaptability* is a criterion for the RL-algorithm. This criterion must ensure that the RL stays adaptable to changes, even if they can not be observed within the current policy. One way of doing so is to keep exploring even if the policy seems to be optimal. Exploring enough to stay adaptable can be a challenge without the behaviour acting too random, since exploration is done by trying random actions. So to meet this criterion, the RL-technique should have some smart or efficient way to re-explore already explored parts of the environment for changes.

2.3. Interdependence

The most important aspect for team members to collaborate [6–8] and more specifically to co-learn is interdependence [9–11]. So in this section we will dive deeper into how an interdependent relationship between a two agents can emerge, and how an RL-technique could be capable of facilitating that. As mentioned in chapter 1, interdependence is a symbiotic relationship where the team members depend on each other to increase task performance and efficiency. The concept of interdependence is often used in studies of team performance [6, 7], team task design [8, 9] and team learning [10, 11]. It is established by concepts like responsibility and regular dependence.

In subsection 2.3.1 we first describe in why interdependence is one of the most important aspects to create opportunities for co-learning. After which we focus on what interdependence is in more detail, and how an interdependent relationship can arise. This is described in subsection 2.3.2, subsection 2.3.3 and subsection 2.3.4. Then, in subsection 2.3.5, we derive three criteria for an interdependent relationship between a human and an RL-agent: ***C4*** *- Shared group rewards*, ***C5*** *- Observability* and ***C6*** *- Explainability*.

2.3.1. Interdependence: a crucial aspect for co-learning

The level of interdependence within a team determines the balance between the experience of self-effectiveness of individual team members and collective-effectiveness of the team [7]. Co-learning is learning as a team rather than collaborating agents learning individually. So, to be able to achieve this, the team must first operate as such. High level of interdependence results in the team operating as one integrated system where individual contributions are indistinguishable [7]. Tal [7] even concludes that interdependence is an essential aspect for the emergence of collective-effectiveness within a team.

2.3.2. From dependence to interdependence

Interdependence starts with dependence, which originates from the capacity of individual team members [8]. This capacity represent the needed skills or available set of actions that are required to achieve the team goal. When one agent alone does not have the different skills, knowledge or resources required to execute a task, a team needs to be formed to complete the task. If both agents do not possess all need capacities alone, but together they do, there is a mutual dependence to execute the task.

However, while mutual dependence can be the first step to create an interdependent relationship, interdependence is more than just mutual dependence [8]. The dependencies described above are considered hard dependencies, since they are required to complete the task. Hard dependencies can be distinguished from soft dependencies [8]. Soft dependencies are also called opportunistic dependencies because they are not strictly needed to achieve the group goal, but they arise from opportunities to perform better as a team. In other words; the team chooses a strategy, where the individual team members might get dependent on each other's actions to complete a part of the task, that they could otherwise have completed independently, if a different strategy was used.

For instance, looking back at the example from section 2.1, by choosing the strategy where the human moves the obstacle out of the way and the robot can move in a straight line, the task gets completed more efficiently. At the same time, a soft dependency is created here because the robot gets dependent on the human actually moving the obstacle, even thought the robot could have completed the task independently. Creating this soft, opportunistic dependency between the team members was beneficial for the completion of the task.

Two other examples are given in Figure 2.4. The dependency of the two trains, is them being physically linked. If this link was not there, the carriages could not be tow each other, making the dependency hard, and required. The two girls walking on the tracks on the right however, depend on each other by holding hands. They use each other to keep their balance. When one of them suddenly lets go, they would fall, making it a dependency. This dependency is soft however, because holding hands it is not strictly necessary to keep their balance, but it does make it easier. Soft dependencies consolidate an interdependent relationship. Moreover, the emergence of soft dependencies, is recursively subject to an interdependent relationship. Johnson [8] calls this the cascading effect. So when interdependence can be established, and there are opportunities for soft dependencies, this cascading effect retains the interdependent relationship. This cascading effect often starts with mutual dependence [8], but that alone would not be enough to establish required the interdependent relationship.

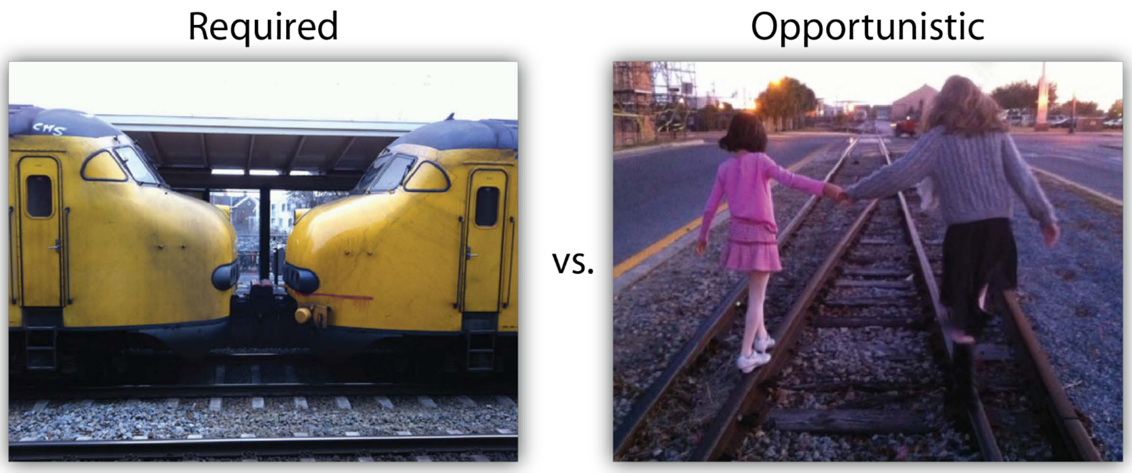


Figure 2.4: Two examples of different types of dependencies [8]. The hard dependency, shown on the left, is visualised by to carriages being linked. Without this link the train would not be able to transport everybody on board. The soft dependency, shown on the right, is visualised by to girls leaning on each other by holding hands. This dependency is not strictly needed to stay in balance, but is does make is easier.

2.3.3. Shared group goal

For a team to have the same shared goal, is important for interdependence to arise. According to Tal [7], there are two main parts that determine the level of interdependence: means and outcomes. The means, in this context, consists of the actions resulting from both types of dependencies that contribute to the interdependence as described in subsection 2.3.2. The outcomes on the other hand are effected by how complementary the group goals and rewards are. In other words, both team members must have interest in the same outcome [8]. This is not only true for the final outcome of the task, but for the intermediate rewards as well. If the robot in the example does not care what path the blue ball takes to the goal position or how long that takes, it would not benefit from having a soft dependency on the human when it can complete its goal independently. So, to function as one interdependent unit, the outcome should be beneficial for all team members, meaning that the team should have a shared group goal and rewards.

2.3.4. Observability, Predictability, Explainability and Directability

Besides dependencies and a shared team goal, there are other requirements for interdependence: *Observability*, *Predictability*, *Explainability* and *Directability* (OPED) [8, 11]. These requirements concern the interaction between the agents. According to Jonhson [8] the three requirements for the collaborative interaction that allow for interdependence are *Observability*, *Predictability*, and *Directability* (OPD). This is extended with *Explainability* to OPED, by van den Bosh [11]. In order to understand the role of these requirements in interdependence, they will be defined first:

- **Observability** indicates to what extent that the agents can observe each other's actions and states.
- **Predictability** stands for the ability to predict each other's actions.
- **Directability** denotes whether the agents are able to influence the actions of the other agent.
- **Explainability** is the ability of the agents to understand each others behaviour.

The contribution of OPED to interdependence can be understood by starting with the Sense-Plan-Act model of a single agent. The first step for the action selection process of an agent is to observe (sense) the environment. Based on this input, it can then determine (plan) the best action using for instance a policy, before it can take action itself (act). This action then influences the environment, closing the loop. In a team, however, there is more to sense, plan or act upon than just the environment. Take for instance *Predictability*; when one agent can predict the action of the other agent, it can base its decision on that. The other way around, when an agent is depending on certain actions of the other agent, it can influence the other agent's actions, using *Directability*. The first case is essentially the start of a soft dependency, because the agent is now depending on the action of the other agent. The second case is a way to deal with dependencies. In other words, *Predictability* and *Directability* are essential to create, or deal with dependencies, and therefore they are required in order to achieve interdependence.

For instance, when the robot arm from the running example explained in section 2.1 wants the human to move the obstacle out of the way, it can use *Directability* by moving towards it, influencing the human to move the obstacle. Alternatively, when it predicts that the human is going to move the obstacle, the robot can already start to plan its trajectory, making use of its *Predictability*. In both cases, the robot creates the dependency: that the human will actually move the obstacle out of the way. Evidently, Observability, or the ability to observe each other, is the foundation of both examples. Because when the robot, for instance, directs the human to move the obstacle the human must be able to observe that for it to have an effect. The addition of Explainability [11], creates the possibility of learning how to predict or influence the actions of the other agent. Without this learning ability it would not be possible to predict the actions of the other agent at all. When both agents have the ability to diagnose the reason or cause behind the actions of their team member they can actually start to successfully use their *Predictability* and *Directability*, creating soft dependencies and thus consolidating the interdependent relationship.

For a high level of interdependence, each team member should have direct interactions with other team members in order to make contributions to the team output. The outcome of such high interdependent tasks can no longer be measured by the sum of individual performances [7]. This only becomes possible with the presence of OPED. Because, with the use of OPED, the steps in this Sence-Plan-Act sequence are carried out by different agents. For instance, one agent sensing something in the environment might cause it to influence the other agent to act upon it. This demonstrates OPED as a tool for interdependence.

2.3.5. Criteria for interdependence in a human-robot team

Above, the requirements and cascading sequences leading towards interdependence are explained, as well as why interdependence is important for co-learning. This can be summarized in the following statement:

”Mutual dependence, shared group goal and OPED are the foundation of an interdependent relationship, from which the cascading effect of opportunistic dependencies arises. This contributes to the level of interdependence, and therewith to the collective-effectiveness, allowing the team to learn and operate as one integrated unit, which is crucial for co-learning.”

These requirements and sequences, can be captured in three distinct criteria, when being projected onto a human-robot team where the robot learns using an RL-algorithm: ***C4*** *- Shared group rewards*, ***C5*** *- Observability* and ***C6*** *- Explainability*. In this first criterion, ***C4*** *- Shared group rewards*, the requirements explained in subsection 2.3.3 are captured. Ensuring that both agents are rewarded similar on the same results, which is the first baseline for interdependence.

The other two criteria based on creating an interdependent relationship between human and robot, capture the requirements of OPED as described in subsection 2.3.4. As explained, the ability to observe the state and actions of the human is the foundation for *Predictability* and *Directability* for the robot. Without it the RL-agent has no idea what the human team member is doing and, therefore, could not base any actions on it. This makes ***C5*** *- Observability* a criterion on its own. The last criterion composed from interdependence is ***C6*** *- Explainability*, as it is the other foundation for *Directability* and *Predictability*. Because when an agent can not explain the actions and decision of its team member, it has no possibility of predicting or directing them. So the criteria ***C5*** *- Observability* and ***C6*** *- Explainability* together, form the foundation of OPED.

2.4. Criteria for RL-techniques to achieve embodied co-learning in a human-robot team

In this chapter six criteria for comparing RL-techniques for embodied co-learning applications where composed. In section 2.2 the first three criteria based on general requirements for embodied co-learning in a human-robot team where conducted: ***C1*** *- Embodied real-time learning*, ***C2*** *- Reward based on performance* and ***C3*** *- Adaptability*. Then in section 2.3 the three criteria that form a base line for interdependence where derived: ***C4*** *- Shared group reward*, ***C5*** *- Observability* and ***C6*** *- Explainability*. This section gives a summary of the six criteria and how they can be met within an RL-technique.

C1 ***Embodied real-time learning*** is met if the RL-technique is able to learn an embodied task in real-time, so that the human and the robot operate and learn in a similar time window. This can be extra challenging in an embodied environment.

C2 ***Reward based on performance*** is met if the RL-technique is reward on the performance rather than from either explicit or implicit feedback from the human team member. This is important to make sure that team members can be equals without a permanent hierarchical actor-critic relation.

C3 ***Adaptability*** is met if the policy of the RL-technique stays adaptable to change even when the current policy seems to be optimal. This is nescessary because the human team member might change its behaviour as it is learning as well.

C4 ***Shared group rewards*** is met when both the human and the RL-agent have the same goal, and therefore are rewarded similarly. This ensures a mutual motive for opportunistic dependencies to arise.

C5 ***Observability*** is met when the robot has the ability to observe the state and actions of the human. This is the first foundation for *Predictability* and *Directability*, since interaction and interdependence, can only be possible with *Observability*.

C6 ***Explainability*** is met when the RL-technique is able to understand the policy of the human agent. This is the second foundation for *Predictability* and *Directability*.

3 Reinforcement learning techniques for embodied human-robot co-learning

To critically compare different RL-techniques they must first be explored in more detail. In this chapter four RL-techniques are explored. They are selected as they are promising techniques for the co-learning of an embodied robot. These techniques are either specifically designed to meet some criteria mentioned in section 2.4 or they have been used for co-learning applications already. They are discussed in the subsections below. First each technique is introduced. This is followed by a detailed explanation of how the technique learns, so it can be evaluated whether it meets the criteria in the next chapter. These four techniques together cover a broad range of RL-technique and offer an overview of what is possible today.

3.1. Q-learning

The first technique that is to be explored is Q-learning. Although it has many times been shown that this basic technique can be outperformed by more complex techniques [30–32], it is still frequently used successfully in the domain of social robotics [33–35] and for co-learning [10]. This is because it is a robust technique that can easily be adapted to learn tasks in any domain. Furthermore Q-learning is the foundation of most other RL-techniques. To get a good overview of what RL-technique is, Q-learning will be explored first along with some fundamental principles such like exploration, exploitation, reward-functions and Q-values.

3.1.1. Q-values

Q-learning is a form of dynamic programming [20], and is one of the oldest and most general techniques of reinforcement learning. Q-learning is based on the concept of having a value, Q, that describes the quality of each state-action pair (s, a). All the Q-values $Q(s, a)$ are stored inside of a Q-table, or Q-function. When all the Q-values are known, a policy can determine which action to take. This is done by choosing the action with the highest Q-value for the given state. These Q-values, however, first have to be learned by the RL-algorithm. To learn these Q-values mathematically each state is assigned a reward $R(s)$. This reward-function $R(s)$ is to be designed based on the task to be learned. Since it is the state-action pair that determines the next state, this reward-function can be used to determine the quality of taking a certain action within a certain state. To fully describe the quality of a state-action pair $Q(s, a)$, however, not only the reward of the next state $R(s')$ should be taken into account but also the reward of the states that can possibly be reached in the future.

Take for instance an example of a maze game as shown in Figure 3.1. The goal is to move out of the maze. The state is the position of the agent. The reward function is defined as minus the euclidean distance to the exit of the maze. Note that the best possible reward is zero, when the agent is at the end. All other rewards are negative and thus worse. When the agent moves into a wall it is immediately placed back at the start, which is the state with the worst reward as it is furthest from the end. In the example in the Figure 3.1 the quality of moving down when the agent is at the red dot (position A) should be lower then the quality of moving to the right at the location of the green dot (position B), even though the reached state after the state-action pair A is closer to the end. Hence, the expected future rewards should be taken into account when calculating the quality of each state action pair.

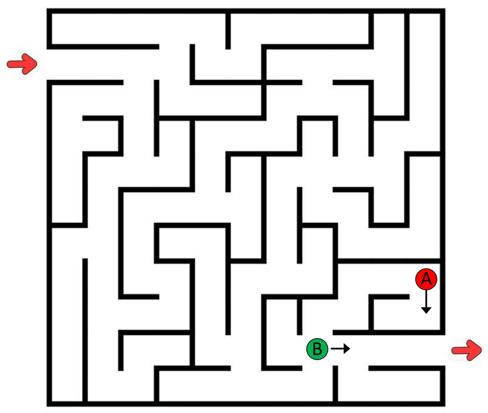


Figure 3.1: A maze game where an RL-agent starts at the red arrow in the top left, and has to learn how to navigate trough this maze. Two possible state-action pairs are shown as examples. The colored dot shows the location (state) of an agent, and the arrow shows the move it is going to make (action)

Learning these Q-values is done by iterative updating them using the Bellman equation (Equation 3.1). The expected future reward is based on the Q-values of the future state-action pairs, where future actions, and consequently the future states, are chosen by the policy based on the current Q-function. In other words, every time an action is taken the Q-value for that state-action pair $Q(s, a)$ is updated to become the sum of the reward received of entering the next state $R(s')$ and highest currently known Q-value for this next possible state-action pair $\gamma \max_{a'} Q_(s', a')$. This maximum Q-value in the next state, is called the expected cumulative future reward because it is recursively equal to the sum of all future rewards. The expected future reward is discounted using a factor γ. This γ must be smaller than one, so that each expected cumulative future reward will contribute less to the total of this Q-value when it is further in the future. This way it is assured that all the Q-values have a finite limit [20]. When iteratively updating the Q-values, the Q-values will eventually to converge to the optimal Q-function, $Q_*(s, a)$, as shown in Equation 3.1. Once this is achieved an optimal policy is learned.

$$Q_*(s, a) = E[R(s') + \gamma \max_{a'} Q_*(s', a')] \tag{3.1}$$

3.1.2. Exploration vs. Exploitation

Since the Q-function is based on the reward and on future values of the Q-function itself, all rewards for all states should be known to find the optimal Q-function. These rewards, however, are not always known beforehand, and often even part of the to-be-learned task. In the maze example from Figure 3.1 the reward for moving into the wall gets the worst reward possible as the agent is placed back at the start. Still the agent will learn for which state-action pairs this state is reached and thus when the corresponding reward is earned. Learning which state-action pairs result in moving into a wall is done through exploration. To explore the agent takes random actions not based on any learned policy. On the other hand, there is exploitation, where the agent follows the policy. If there only was exploration, the agent would not learn, and always take random actions. If there only was exploitation, the agent would also not learn, because it would keep following the same policy. A balance between the two, determines the essence of the learning algorithm. The more the algorithm exploits, the faster it wants to earn high rewards. This is called a greedy policy.

Traditionally, the greediness of the Q-learning algorithm is determined by a parameter ϵ, which portrays the chance that the agent explores. By starting with a high ϵ, the algorithm explores fast at first, after which a lower ϵ lets the algorithm explore more around the higher Q-values. When the optimal policy is found, ϵ could decay all the way to zero, so the algorithm would only exploit its learned policy. In an environment where the future reward is not predictable, because the actions of another agent influence the future state as well, this balance becomes crucial for convergence to joint activity [30].

Take for instance the maze game again (Figure 3.1), but now with two agents playing at the same time. When the two agents crash into each other, they both go back to the start. In that case they would have learned that the just executed state-action pair would have lead to a state with a low reward. When in this scenario the algorithm is to greedy, it would not learn that this situation was only incidental, and it will be avoided it in the future. Moreover, in the context of co-learning, this balance becomes extra important and can not converge to only exploitation to ensure adaptability, as explained in subsection 2.2.3.

Alternatively to ϵ-decay, there are other strategies to determine the greediness of the algorithm, such as the Boltzmann strategy [28], or the frequency maximum Q-value (FMQ) heuristic [30]. These strategies outperform traditional ϵ-decay on multiple aspects as is shown by Kapetanakis [30]. However, all these alternatives are also designed to converge to a greedy policy after a while, which is not beneficial for the adaptability in later stages of the learning process. This makes them less suitable for co-learning, as described in subsection 2.2.3.

3.2. MAXQ value decomposition

The second RL-technique explored in this chapter is MAXQ value decomposition [21], a Hierarchical RL-technique often used in social robotics [36, 37], and multi-agent environments [38]. MAXQ has two major advantages in comparison to standard Q-learning, that both make MAXQ more suitable for an embodied co-learning robot. The first advantage is that it makes use of value decomposition. Value decomposition replaces the reward-function in the main Q-function with the Q-function of a sub-learning problem. Mathematically this creates a hierarchical structure of the learning problem. Due to the value decomposition complex problems can become simpler and more structured, resulting in faster learning [31] and faster computations. The second advantage is that exploration can be done in bigger chunks due to the hierarchical structure of the action decision process. This also improves the learning rate at the beginning of the learning process and increases the adaptability in later stages as well [39]. How MAXQ value decomposition works will be explained later in subsection 3.2.2 in more detail.

3.2.1. An overview of the algorithm

Hierarchical reinforcement learning breaks the main or root task up into multiple subtasks that will be seen as actions of the root task. The action-space of such subtasks can then be made up from a combination of subtasks and primitive actions. Primitive actions are the actions which are actually executed by the agent. Dietterich [39] explains this concept using a taxi problem: Consider a taxi cab operating in a 5x5 grid. The primitive actions are that it can move one cell in each cardinal direction, pick a passenger up, or put a passenger down. The goal of the root task is to pick a passenger from a random starting cell and put it down at a random destination cell using as few actions as possible. This root task can be broken up into the two subtasks *Get* and *Put* as can be seen in Figure 3.2. Where the goal subtask *Get* is to drive to the passenger and pick it up, and the subtask *Put* is to drive to the destination of the passenger and put it down. *Get* can then for instance be broken up into the subtask *Navigate* and the primitive action *Pickup*. The subtask *Put* can then use the same subtask *Navigate* and the primitive action *Putdown*.

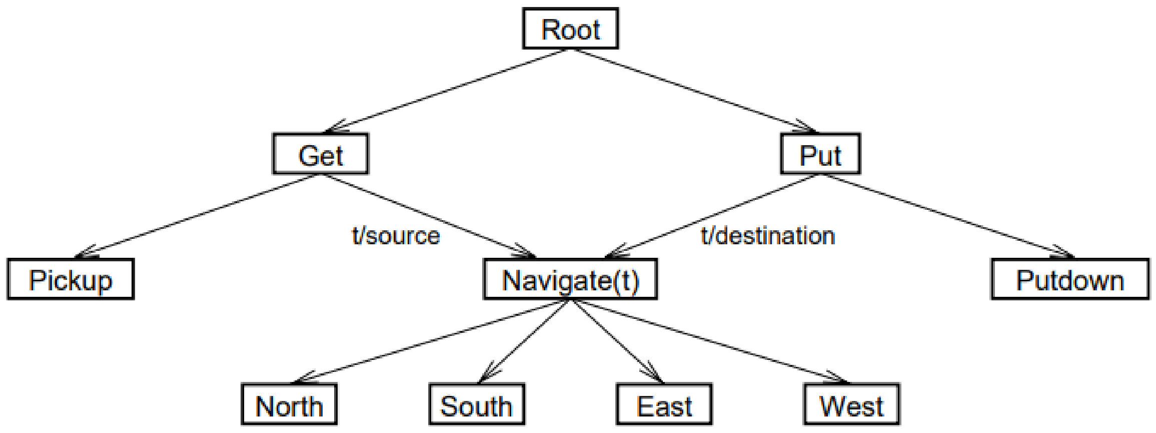


Figure 3.2: This task graph for the taxi problem shows the hierarchical structure of the root task and its subtasks [39]. The arrows point to the subtasks, and primitive actions that a subtask can execute. Primitive actions have no arrows originating from them.

A hierarchical structure of the learning problem can have multiple benefits: it might not only improve learning in early stages, but it could also make the algorithm more adaptable to changes later on. This is because exploration can happen in bigger steps [21]. Furthermore, the hierarchy makes it possible to keep parts of the learned policy unchanged, while other subtasks evolve. This improves the adaptability of the algorithm because often only the policy of a subtask needs to be adapted. Lastly, since each subtask has the ability to consider only relevant state variables for that task, the amount of state-action combinations that will be considered in general, can be reduced.

In the given example, there are 2 state variables: the location of the cab, and the location of the passenger. The first variable has 25 possibilities, i.e. the cab can be at one of 25 cells in the grid. The second state variable has 26 possibilities, one for each cell in the grid, and one extra value for when the passenger is in the cab. Without the hierarchical structure the task would have 3900 possible state-action combinations: 650 states and 6 primitive actions. Using the new structure, the root task would only have to consider 2 actions and 26 states, as the location of the cab does not matter for the action selection but only whether the passenger is in the cab or not. For the *Navigate* task, the location of the passenger does not matter, only the location of the cab. So this subtask would have a state-action-space with only 100 possibilities (4 actions and 25 states). In conclusion, the hierarchy makes it possible to ignore irrelevant parts of the state-action-space for a specific subtask, simplifying the problem into multiple smaller learning problems and making the learning process more efficient.

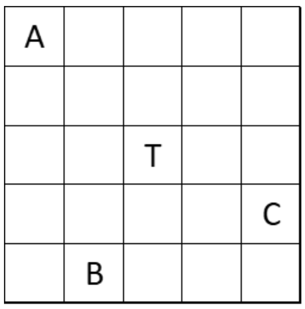


Figure 3.3: The domain of the taxi cab corresponding to Figure 3.2. The T allocates the position of the taxi cab and the A, B and C allocate positions where passengers can be picked up or put down.

The essence of the MAXQ value decomposition algorithm is to replace the reward function in the Q-function with a decomposed value function. This is different from decomposing the learning problem into multiple individually learning Q-learning problems because value decomposition allows each task to take the results of its subtask into account, while at the same time learning happens only for its own action selection. How this works is explained in more detail in subsection 3.2.2. The effect of the decomposed the value function, in contrast to decomposing learning problem, can be illustrated if we make the taxi example a bit more complex. Suppose a second passenger is added to the problem. The cab however only has place for one passenger at a time. The first passenger P_1 is located at point A and must go to point B, and the second passenger, P_2 starts at point C and wants to go to point A, see Figure 3.3. The taxi starts at point T. In this case it would be more efficient to start with transporting P_2 because the pickup location of P_1 is the same as the destination of P_2. If the root-task would not take the results of the *Navigate* subtask into account it would not be able to distinguish this difference. Using value decomposition it should be able to learn that the expected cumulative reward would be better if it would start with picking up P_2 first.

3.2.2. The algorithm in more detail

Dietterich [21] developed the hierarchical reinforcement learning technique called MAXQ value decomposition. MAXQ is based on the same principles as basic Q-learning, where the policy is based on the quality of state-action pairs. Like Q-learning, MAXQ takes expected future rewards into account when these quality values are calculated, using a similar bellman equation. However, since the used state-variables and actions are different in each subtask, this function is not as straight forward. This is where value decomposition comes into play. The quality $Q_i^\pi(s,a)$ of state-action pair (s,a) in subtask i, following policy π, is calculated as shown in Equation 3.2. This is similar to the Q-function of standard Q-learning (Equation 3.1). The reward $R(s')$ is however replaced by a value function $V_a^\pi(s)$ that represents the value for taking action a in state s under policy π. This value function $V_a^\pi(s)$ is equal to the reward $R(s')$ when a is a primitive action. However when action a is a subtask j, it is recursively equal to $Q_j^\pi(s,\pi(s))$, where $\pi(s)$ represents the action that would be selected in state s under policy π. The second term of Equation 3.2 is the completion function, $C_i^\pi(s,a)$, or the expected cumulative reward for completing subtask i following policy π after executing action a in state s.

To illustrate this refering to the taxi example, the Q-function for the subtask *Navigate* will be the same as in normal Q-learning, because all the actions of this function are primitive actions, and $V_a^\pi(s)$ will always be equal to $R('s)$ and the completion function is equal to the expected cumulative reward of this subtask. The Q-function of the root-task on the other hand is decomposed into two different Q-functions as both actions are actually subtasks. In practice the Q-function can be learned in the same way as in standard Q-learning, by iteratively exploring the environment and updating the entire Q-function while doing so.

$$Q_i^\pi(s,a) = V_a^\pi(s) + C_i^\pi(s,a) \tag{3.2}$$

So, in short, MAXQ breaks the value function down into chunks of rewards, which an efficient way of doing the bookkeeping of Q-values and will shorten the computation time. This also allows for efficient exploration, improving the adaptability. Both these advantages are great for co-learning embodied task as explained in subsection 2.2.1 and subsection 2.2.3.

3.3. Soft Actor Critic

Soft Actor Critic (SAC) is a new popular deep RL-technique that is proven to be able to efficiently learn global optimal policies for embodied robots [13, 22, 40], that are robust to adaptions in the environment. This is because, unlike the techniques mentioned above, SAC contains two agents, an actor that is responsible for action selection, and a critic that provides feedback on the actors actions. SAC is relevant for this research for its efficiency and its adaptability. This technique is one of the few RL-techniques that has already achieved embodied co-learning in a real-world task [12].

SAC is a different type of RL-technique compared to the two techniques mentioned above, as it is based on an actor-critic approach that makes use of neural networks. This makes it even more interesting as it might be able to cover a broad range of possibilities of RL. This section further explains how SAC works and what the advantages and disadvantages for co-learning are.

SAC uses an actor-critic approach, which splits the algorithm into two parts: the actor, and the critic. The actor is responsible for the action selection, and the critic teaches the actor what action is best given the state. Two other examples of actor-critic based RL-techniques that are famous in HRC are TAMER [26] and COACH [27]. In both these two techniques however, the role of the critic is executed by a human, as explained in subsection 2.2.2. In SAC both roles are covered within the robot-agent. This is done using two neural networks, one for the actor and one for the critic, as shown in Figure 3.4. Even though the name actor-critic approach suggest otherwise, the technique lies in the category of *Task Performance Driven Methods* as described in subsection 2.2.2 and shown in Figure 2.3. As described in chapter 2 this category is beneficial for co-learning, as the human does not permanently have to take an expert or teacher role.

3.3.1. Neural networks

SAC uses two neural networks (NN), one to select its actions, and a second NN that takes the rol of the critic. This second one eliminates the teaching role of the human, in a traditional actor-critic RL-technique such as COACH [27] and TAMER [26] (mentioned in subsection 2.2.2). In order to understand how SAC works, first the working of neural networks will be explained. NN's are based on the human brain, where neurons are connected to nodes to form a network that is able to process information [17]. In a NN these nodes are structured in layers. A NN contains a layer of input nodes, a layer of output nodes, and at least one hidden layer in between. The NN's shown in Figure 3.4 for example has two hidden layers. Each neuron, represented by a line in the figure, has a specific weight that determines how much the two nodes it connects are related. These weights are used to calculate the next layer. This process can be done efficiently using matrix multiplication. When the input layer is given, the NN can calculate the next layers and eventually the output layer. Since the hidden layers are hidden a NN acts as a black box that provides an output for every input.

A NN learns from examples. When a NN is presented an input it predicts the outcome by forward calculations through the layers. This calculated outcome is then compared to the outcome as given by the examples, and based on this the weights are updated in order to match the outcome better. This is done by using a process called back propagation. When a NN has been provided enough examples, it will eventually be able to predict the outcome given a new input [17].

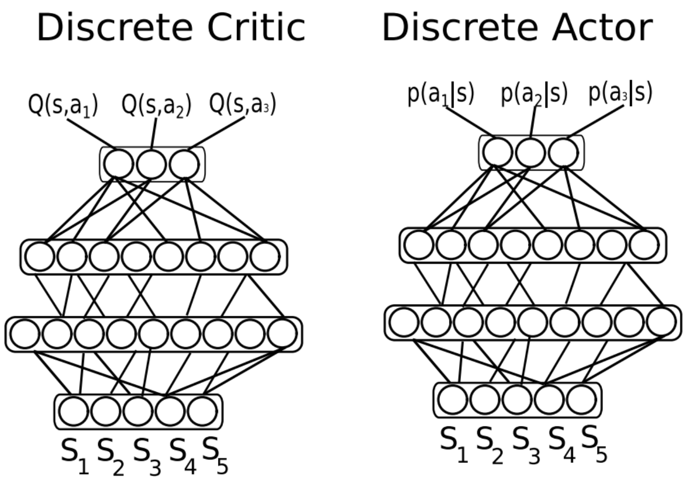


Figure 3.4: An overview of the two neural networks used in SAC [22, 41]. The left figure shows the actor that take the state as input (at the bottom) and returns the Q-values as output for the given state, and all the possible actions (shown at the top). The right figure shows the actor that also takes the state as input, but it outputs a probability for each action to be executed, given the state. Both neural networks show two hidden layers.

3.3.2. How SAC works

In the case of SAC, both the actor and critic NN's are initialized randomly. Meaning that the agent will act randomly at first and does not know the value of any state-action pair. The critic is trained first. The input is the state and the computed output is the quality, or Q-value, of each action given that state $Q(s, a)$ as can be seen in Figure 3.4. After each action the actual outcome is used to train the critic. The critic is used to train the actor. The actor is very similar to the critic; It takes the same input, the state, but it computes probabilities for each action to be taken (see Figure 3.4). These probabilities are based on the Q-values learned by the critic and, in addition, to something called entropy. Entropy is a term for the randomness of the actions, and it determines the trade-off between exploration and exploitation. For the critic it is important that future actions are predictable in order to compute meaningful Q-values. In contrast, the actor tries to maximize this random behavior as much as possible. So entropy maximization means that the actor tries to act as random as possible, while it stays possible for the critic to predict future actions somewhat reliable. How much this entropy weights is determined by the a variable called the temperature. In early versions of SAC [13, 40], this temperature, was a hyper-parameter that had to be tuned according to the task. The newest version [22] is able to learn the best value of this parameter by itself, making it easy to implement in a broad scope of tasks, including tasks that change. So, in short, first the critic learns the Q-values of each state-action pair by exploration. Then, the actor is trained to output a probability for each action to be taken, which is based on the output of the critic, but still as random as possible.

3.3.3. Advantages and Drawbacks

The biggest advantage of SAC is that is has the best of two worlds [40]; Within deep RL, there are two families of techniques. The first being stochastic techniques, such as TRPO [42] and PPO [43], that are able to stably learn the global optimum and stay adaptable to changes. Because of their stochastic nature, however, they can not reuse samples and therefore learn slower. The second family consists of deterministic techniques such as DDPG [44] and TD3 [45]. Deterministic techniques are able to reuse samples, which makes them learn efficiently, but often they get stuck in local optima. SAC has the best of both these two worlds. This is because it makes use of two NN's. The deterministic critic is able to reuse samples, and therefore learn efficient, and the stochastic actor maximizes the entropy in order to stay adaptable, and find the global optimum.

Furthermore, the newest version of SAC can be implemented for a broad scope of tasks and is adaptable to changes during the task [22]. Due to its computation efficiency, it can even be applied in embodied tasks. To show its applicability, Haarnoja performed additional research [22] demonstrating that SAC is suitable in different types of embodied tasks including learning a quadrupedal robot how to walk, and learning manipulations to a robotic arm.

There is however one big drawback to the use of neural networks, and therefore with all deep RL-techniques, including SAC. This drawback is that a NN acts as a black box, meaning no guaranties can be given on predictable behavior. When a robot is collaborating with a human, it is very important to have safety guarantees. So when SAC or any other deep RL is used for co-learning an embodied task together with a human, extra safety constraints might need to be added, to make sure the robot is safe.

3.4. Real-Time Actor-Critic

The last RL-technique covered in this review is Real-Time Actor-Critic (RTAC) [23]. RTAC is selected as a promising RL-technique, because it is an upgraded version of SAC that is specifically designed to learn in real-time. As described above, SAC was already able to learn physically embodied tasks as shown by Haarnoja [22] and Ramstedt [23]. With the added functionalities to be able to learn in real-time it will score specifically high on criterion ***C1*** - *Embodied real-time learning* from the list in section 2.4.

As described earlier in subsection 2.2.1, Ramstedt makes a distinction between real-time and turn-based RL. In turn-based RL the agent updates the algorithm and selects the next action after a state is reached, because the reached state is used to calculate the reward and to select the next action. In real-time RL, on the other hand, updating the value function and selecting the next action is done during the execution of the current action, and it is based on a prediction of the next state. This way the computing time needed to do so, does not effect the rate at which the algorithm learns or operates. Since real-time RL is based on this prediction, it must be assumed that the next state can be predicted based on the current state-action pair. This should not be a problem as reinforcement learning in general is based on that assumption already: The cumulative future reward in standard Q-learning for example, as explained in section 3.1, consist of the Q-value of the reached state and best corresponding action. This Q-value is calculated from the reward of the state reached after that action and the future rewards after that. In other words, it assumes that a state-action pair gets rewarded the same every time, and since the reward-function is based on the reached stated, Q-learning also uses the assumption that the reached state predictable is based on this state-action pair as well.

To predict the next state, the neural network that acts as the critic in SAC is updated, so that it does not only have the current state as input, but the current action as well. This way it will not compute the Q-values of the current possible state-action pairs, but the Q-values of the next state, and the possible actions after that. This way the actor can also be trained to compute the next action, based on the current state and the current action. Ramstedt [23] shows that this change allow the algorithm not only the ability to learn in real-time, but it makes the algorithm outperform SAC in learning an embodied task.

When this is put in a co-learning perspective however, RTAC might be less successful than SAC. Because, as mentioned in section 3.1, the next state might not be as predictable in a multi-agent environment. To understand this, reflect back to the maze game (Figure 3.1) with two agents playing at the same time. The next state is most of the time predictable, but sometimes, when the two agents crash into each other, the next state is different then originally predicted. This does not only cause the next action to be selected wrongly, but also the learned policy to be updated based on assumption that the crash is always happening. Moreover, where in this example scenario interactions between two agents might be rare, during co-learning in an interdependent team on the other hand, interactions are more the norm, rather then the exception. So, an RL-technique that relies heavily on the next state being predictable, might be less optimal, when in this state might in practice not always be as predictable.

4 Discussion

In this chapter the RL-techniques discussed in chapter 3 will be compared by applying the criteria defined in chapter 2. Per criterion a section is dedicated to compare the different techniques and to discuss to what extent they meet this criterion (section 4.1 to section 4.6). In each of these sections we first recapitulate what the criterion entails and how it can be met, followed by a discussion of each technique in relation to the criterion. We conclude this chapter with a summary in section 4.7. Here we also give a retrospective on the found criteria from chapter 2, now that the comparison is made.

To qualitatively compare the different RL-techniques against the criteria a five-point rating scale is used. This scale increases from the lowest rating -- via -, +/- and + to the highest rating ++. This score represents to what extent an RL-technique has the possibility to meet a certain criterion, relative to what would be ideal, given the possibilities of state-of-the-art technology. So, a ++ means that the RL-technique would perfectly meet a criterion, given what would currently realistically be is possible. A --, on the other hand would represent the worst case scenario; that the criterion can not be met. However, since all four discussed techniques are selected to be promising RL-techniques for co-learning with the criteria in mind, this worst case scenario does not apply to any of these RL-techniques.

An overview of the meaning of each score is given in Table 4.1. Table 4.2 gives an overview of the results from the comparison.

Table 4.1: The five-point scale used for rating each RL-technique with respect to a criterion, along with an explanation of each score.

--	Impossible to meet the criterion
-	Criterion can be met partially but only in specific cases
+/-	Criterion can be met partially in most cases and completely in some specific cases
+	Criterion can be met completely, in almost all cases
++	Criterion will be met naturally, as the RL-technique is designed specifically to do this.

Table 4.2: An overview of the ratings for each RL-technique from chapter 3 on the criteria for from chapter 2

	Q-learning	MAXQ	SAC	RTAC
C1 - Real-time embodied learning	+/-	+	+/-	++
C2 - Reward based on performance	++	++	++	++
C3 - Adaptability	+/-	++	++	+
C4 - Shared group reward	+	+	+/-	+/-
C5 - Observability	-	+/-	+	-
C6 - Explainability	+/-	+	+/-	-

4.1. C1 - Real-time embodied learning

The first criterion, ***C1*** *- Real-time embodied learning*, is met when the RL-technique is able to learn and operate an embodied task in real-time. The possibilities of achieving this is mostly impacted by the computational time, as described in subsection 2.2.1. This is connected to the simplicity of the technique, since a simpler algorithm takes less computational time. The trade-off here, however, is that less complex techniques are only effective for less complex learning problems. Take for instance Q-learning; as it is one of the simplest RL-techniques out there, it is able to learn and compute fast enough to meet this criterion. However, this technique is not suitable for more complex learning problems. So if the learning problem is suitable for Q-learning, it is able to learn in real-time.

The same can be said about MAXQ value decomposition. This technique fits best in combination with a task that has a hierarchical structure. It breaks a complex learning-problem up into smaller and simpler problems that can be learned without much computations or iterations. So, although MAXQ is most suitable for learning problems that have a hierarchical structure, it will meet this criteria when the technique fits the problem. This means that for both techniques it can be said that if the technique is suitable to learn the problem, it will also have the ability to learn it in real-time.

The other side of this criterion is that the technique should be able to learn embodied tasks. As explained in subsection 2.2.1, embodied tasks are generally more complex then virtual tasks. So for Q-learning to meet this part of this criterion, the task must be very simple in order for the learning problem to stay simple enough. In other words, even though Q-learning has the ability to learn in real-time, it will only be able to learn embodied problems in real-time when the task is very simple, leaving it with an average score. MAXQ on the other hand has the ability to learn complex problems, as long as the problem can be broken down into multiple sub-problems. In other words it will meet the criterion in all cases where the technique fits the problem, for which it receives a + as the score for this criterion.

The other two techniques discussed in this review are based on neural networks. The big advantage of a neural network is that it has the ability to learn complex tasks. This makes these techniques well suited for embodied problems. Computational times in neural networks, however, can be high making the learning pace in real-time very challenging. This is the main reason SAC only has a limited ability to learn embodied tasks in real-time. It is still granted an average score for it can be able to compute fast enough when the learning problem is simple, similar to Q-learning.

RTAC, on the other hand, is specifically designed to be able to learn complex tasks in real-time. It uses a prediction of the next state for action selection, as explained in section 3.4. This way it has the same abilities to learn a complex embodied problem as SAC, while also being able to learn and operate in real-time. This makes RTAC score the maximum rating on this criterion.

4.2. C2 - Reward based on performance

The criterion ***C3*** - *Reward based on performance* is met, when the RL-technique is rewarded purely on the performance of the task, contrary to being rewarded on any form of feedback from the human team member. Since all four discussed techniques are *task performance driven RL*, and not *interactive RL* (see Figure 2.3), they all meet this criterion due to the design of the technique. All four discussed techniques receive the maximum score on this criterion.

4.3. C3 - Adaptability

C3 - *Adaptability* is the last general criterion that the selected RL-techniques will be compared on in this section. To meet this criterion, the RL-technique needs to stay adaptable to changes, even when a preliminary optimal policy has already been found. The level of adaptability is partially determined by the balance between exploration and exploitation. In other words, in order to stay adaptable the agent must keep exploring even when the policy seems to be optimal.

Since exploration in Q-learning is random and not very efficient, it can be challenging to find the right balance. MAXQ on the other hand, has the ability to explore much more efficiently, due to its hierarchical structure. This means it can explore big chunks of the action-space at once without much random behavior. Therefore, it can still adapt specific parts of the policy while keeping most of it unchanged, even when the balance is converged mostly towards exploitation. This makes adaptability one of the main strengths of MAXQ granting it the maximum rating.

SAC is earned the maximum rating on this criterion as well for its use of entropy maximization. The technique is developed specifically to keep exploring without acting unstable or unpredictable. SAC uses entropy maximization to keep explore as much as possible. This ensures that the policy stays adaptable when converging to local optima, as explained in section 3.3. This also means that when the optimum changes overtime, it will be detected allowing for a change of policy.

This characteristic is similar for RTAC, since it uses the same principles as SAC. The difference however, is that RTAC bases its actions on predictions of future states. Since this predication is not necessarily true, it makes the behavior less stable. In other words, the entropy is naturally higher due to its real-time action selection. This leaves less room for exploration to happen without the entropy reaching its maximum. So even though RTAC is based on a concept that is designed to be adaptable, it has traded some of its adaptability for the ability of real-time learning.

4.4. C4 - Shared group rewards

To meet the criterion ***C4*** - *Shared group rewards*, the reward for the human and the robot must be the same. In practice this means that the reward-function should be implemented in such a way that the robot gets rewarded for the same things as the human. The main importance of having shared group rewards is that the robot and the human get rewarded as a team. An RL-agent gets rewarded based on a reward-function. This reward-function, is based on the global goal of the task and determines the intermediate rewards that the agent receives. Both the global goal, as the intermediate reward need to be similar to for the human and the robot. A human does not have a reward-function, so in order to meet this criterion the reward-function implemented on the RL-algorithm should be based on the success experience of the human. Capturing this success experience in a reward function can be a difficult process, since the reward function is a global function that only determines which states are preferable and which are not, while the success experience of the human can be influenced by many more factors then just the current state.

Comparing different techniques on this criterion might feel contradictory at first, since the implementation of the reward-function is in principle independent of the RL-technique. However, how the reward function is handled within the technique is of importance as to how well this criterion is met. Take for instance the actor-critic approach used in SAC and RTAC; The reward function is used to train the critic, while the actor is responsible for the policy. The actor is then recursively trained based on this critic. So the critic is the bridge between the implemented reward function and how the actor is actually being rewarded.

The problem with this approach arises when the critic is a complex neural network that acts as a black box. Due to this the intermediate rewards, that are actually received by the robot, can differ from what the human experience as success in the first place. In other words, the success experience of the human has to be captured first in a generalised reward function, which is then used to train a neural network: the critic. Next the actor is trained based on this critic. Capturing this success experience in a reward function can be quite a difficult process, regardless of how it is implemented within the technique. So, having these extra steps between the implementation of the reward function and how it is actually received by the human, can make these two techniques less suitable regarding this criterion. For this reason SAC and RTAC can only meet this criterion partially in most cases, while Q-learning and MAXQ score are only limited by the implementation of the reward function itself.

4.5. C5 - Observability

C5 - *Observability* is the ability to observe the state and actions of the other agent, so that they can then be taken into account in the policy of the RL-agent. When an RL-technique meets the criterion ***C5*** - *Observability* it should be able to take the actions and state of the human team member into account within the policy. As explained in subsection 2.3.5 observing the other agent is the foundation of *Predictability* and *Directability*.

In order for an RL-algorithm to observe something it should be part of the state variables. This is because anything that is not part of the state variables cannot be taken into account in the decision making process and will therefore not be observed. In other words, *Observability* goes hand in hand with an increased state space. An RL-technique needs to learn the result and value of each relevant state-action pair possible in order to learn the optimal policy. So, an increased state space or having more state variables rapidly increases the learning problem.

In the case of Q-learning the scaling of the learning problem could get problematic fast, since each and every state-action pair needs to be explored before the optimal policy can be found. For this reason Q-learning is not suitable for having a big state-space. So in the case of Q-learning meeting this criterion would only be possible if the learning problem is very small, for instance by having very few possible actions or states.

MAXQ has a more efficient way of handling state variables, since each subtask only takes relevant state variables into account, making it much more suitable for co-learning from an *Observability* perspective. Which state variables are important during which subtasks, however, is part of the implementation of the technique and is not something that can be learned by the technique itself. So although MAXQ has better *Observability* than Q-learning it is still very dependent on the implementation of the technique.

Neural networks, on the other hand, are proficient in handling big amounts of state variables. For that reason SAC naturally meets this criterion in most cases. In the case of RTAC however it is not that straightforward. Although the algorithm itself is based on neural networks that have no problem handling big state spaces, there is another complication withing RTAC that influences the possibility of have *Observability*. RTAC does not use the current state but a prediction of the next state for the action selection process. This prediction is made based on the current state-action pair, assuming there is a causality. When the state and actions of the human are part of the state space of the RL-agent however, this causality does not always exist, for the human can determine its own actions independently. In other words, RTAC dependents on a prediction of the next state, but since part of this state would be fully dependent on the actions of the human, this prediction can not reliably be made. For this reason it is not practical to make the state and action of the human team member part of the state-space. On the other hand, although the actions of the human can be unpredictable, they are not completely random. So in some specific cases it might be possible to predict the actions of the human reliably, and hence making it possible for RTAC to meet this criterion. Moreover, if predicting human action would be possible, this would contribute to meeting the last criterion, ***C6*** - *Explainablilty* as well, as will discussed in the next section.

4.6. C6 - Explainability

C6 - *Explainability* is the key to *Predictability* and *Directability*, as explained in subsection 2.3.4. This criterion is about being able to understand the policy of the other agent, so that it can base its actions on those of its team member. From the perspective of an RL-algorithm there are two sides to this; the ability of understanding the policy of the human agent and how understandable the policy of the robot is for its human team member. The first part, understanding the policy of its human team member, starts with meeting the criterion ***C5*** - *Observability*. This part of *Explainability* is about being able to do something useful with state variables that describe the human team member. In other words, the RL-technique should able to learn how to use the observation of the human agent for its action selection in order to understand the policy of the human. So, in order to meet the first part of the criterion *explainability*, the RL-technique should at least be able to meet the criterion ***C5*** - *Observability*.

In the specific case of RTAC there is an additional way to understand the policy of the human team member. RTAC is the only discussed technique that bases its actions on a prediction the future state. In order to predict the next state and action of the human the predicting algorithm must have the ability of understanding the human policy first. Whether the algorithm can however reliably predict this, is highly dependent on the implementation this feature. In other words, RTAC has the perfect feature in order to meet this first part of the criterion *explainablity*, but this is only possible in specific implementations.

The other side of explainability; how understandable the policy of the RL-technique is to the human team member is also of importance. The two techniques based on neural networks, score the worst here, since their policy consists of two trained neural networks which can be very hard to understand for a human. There is also no clear way of explaining the policy to the human, since a neural network is in essence a black box with only in- and outputs.

The other two techniques offer more opportunities to meet this part of the criterion. Q-learning for instance is a simple technique with the most basic policy. In most cases the policy might even be simple enough to be eventually understood by the human just by collaborating on the task together. Otherwise, since the policy is just based on the principle where the actions with the highest Q-value is chosen, the interaction patterns should be explainable to a human via some form of interface.

MAXQ value decomposition is also easily understandable for a human, due to the fact that each subtask is designed to be simple, they can be understood easily by a human. Furthermore, the hierarchy of subtasks already describes interaction patterns that can be shown to the human in order to help them understand the policy of the RL-agent.

To rate each RL-technique on this criterion, both parts of explainability have to be taken into account. Q-learning is almost ideal for being understood by the human, however it offers almost no possibilities of understanding the human, resulting in only an average score. MAXQ on the other hand scores the best on this criterion, since it is able to both take the policy of the human into account as described in section 4.5 and also offers opportunities to be understood. SAC is the best technique at understanding the human policy because of its ability to understand complex systems with big state-spaces. However, this complexity also makes it harder for the human to learn the policy of the RL-agent, resulting in an average score. RTAC receives the lowest score on this criterion. It might be the prefect technique for understanding the human policy due to its predictions, but only in specific implementations. Furthermore, its own policy can be hard to learn for a human due to the complexity of the neural networks used.

4.7. Retrospective

Before a complete conclusion can be drawn, a retrospective will be given on the results of the above comparison and the used criteria. This will be done in two parts. First in subsection 4.7.1 the results from Table 4.2 will be critically reviewed, and then in subsection 4.7.2 we will critically review the used criteria.

4.7.1. Retrospective on the results

In Table 4.2 it can be seen that MAXQ is the best RL-technique as it scores highest on average. This however, is not all that can be learned from this. As described in most of the sections above, some of the RL-techniques might in some specific implementations or scenarios be more suitable that the table suggests. In an implementation where the actions of the human, and thus the next state is very predictable, for instance, RTAC might be a very suitable RL-technique. This is because it would then score high on criteria ***C5*** - *Observability* and ***C6*** - *Explainability*, as described in the corresponding sections. Or, to give another counter example; MAXQ is not suitable for embodied tasks that are not easily structured hierarchically as it advantages on the criteria ***C3*** - *Adaptability* and ***C6*** - *Explainability* are depending mostly on this hierarchical structure. So for such a task SAC might be more suitable.

Furthermore, the importance of each criterion is dependent on the implementation as well. When the to-be-learned-task would for instance be turn-based in itself, such as a game where the team-members would act in succession, it would be less important whether the criterion ***C1*** - *Real-time* *embodied learning* would be met at all. Because this criterion is based on the requirement that the human and robot have to interact in real-time.

Moreover, all the discussed RL-techniques could be adapted to fit a specific implementation better. The state-space could for instance be based on what suits the RL-technique best, and in Q-learning and MAXQ the balance between exploration and exploitation can be chosen specifically to the task as well. In SAC and RTAC there are a lot of hyper parameters that can be set to fit the implementation, such as the amount of hidden layers or the weight of the entropy term that maximizes the exploration. In other words, what RL-technique is most suitable for embodied human-robot co-learning is very depended on the specific task that is to be learned and how well the RL-technique fits it.

4.7.2. Retrospective on the criteria

The criteria itself can be reflected on as well. As shown, they can be used to select and compare RL-techniques on there applicability on co-learning. However, these criteria solely are not enough to select the best RL-technique for a specific learning problem. The implementations itself needs to be taken in to account as well to determine which RL-technique is most suitable. So, when the specific implementation is known, these criteria become most useful.

Furthermore, the first three criteria (***C1*** to ***C3***) form a good foundation to select and discard promising RL-techniques for a specific implementation, as they are about general aspects of RL-techniques themselves. Especially ***C2*** can be used for this, as it categorizes all RL-techniques into suitable and unsuitable techniques for co-learning. The three criteria that are important to ensure an interdependent relationship (***C4*** to ***C6***), on the other hand, can not only be used to compare how suitable RL-techniques are for a specific co-learning implementation, but they should also be taken into account when implementing a RL-technique. Because, these criteria are mainly not about the RL-technique itself, but more about the possibilities that they offer when implemented. Take for instance ***C4*** - *Observability*, this criterion can be met if the robot is able to observe the human team member. To do so the actions and state of the human team member should be part of the state space of the RL-agent. So, this criterion can be used to compare RL-techniques, as some are able to handle a bigger state space than others. This criterion should however also be used to actually ensure that the robot uses the observations of the human as part of its state-space in the implementation.

So, in short, the criteria composed in chapter 2 offer a good foundation to select the most suitable RL-technique, but they are even more useful when the implementation is known as well. Moreover, they should not only be used to select the best RL-technique, but they should be taken into account during implementation as well.

5 Conclusion and Future work

5.1. Conclusion

Two research questions where composed to explore how humans and robots can co-learn an embodied tasks as one collaborative team, using existing reinforcement learning techniques found in the current literature:

R1 What criteria are relevant for reinforcement learning in order to achieve co-learning in a human-robot team?

R2 What reinforcement learning technique is most suitable for human-robot co-learning according to the found criteria?

Six criteria where composed to answer the first research question **R1**. Three general criteria and three criteria to ensure an interdependent relationship. The three general criteria that ensure a reinforcement learning technique is suitable for co-learning and are ***C1*** *- Real-time Embodied learning*, ***C2*** *- Reward based on performance* and ***C3*** *- Adaptability*. The three criteria that ensure for the emergence of an interdependent relationship are ***C4*** *- Shared group rewards*, ***C5*** *- Observability* and ***C6*** *- Explainability*.

The answer to the second research question **R2** is that MAXQ value decomposition is the most suitable reinforcement learning technique for human-robot co-learning according to these criteria. What technique is most suitable for a specific implementation, however, is depended on the nature of the problem as well. The defined criteria are even more useful when the implementation is known. The criteria should also be taken into account during implementation.

5.2. Future work

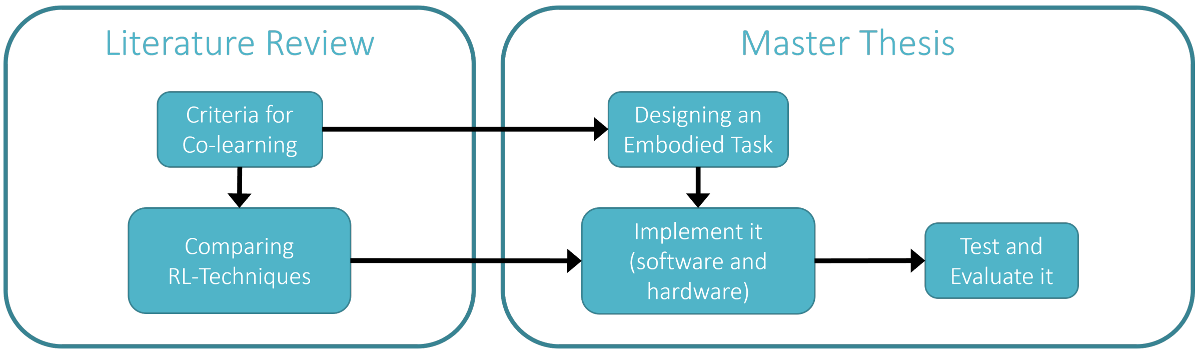


Figure 5.1: How this literature review and the master thesis are connected

Now that the six criteria have been defined and the connection between RL-techniques and embodied co-learning in a human-robot team has been studied, this knowledge can be used to design a physically embodied task for co-learning. Then one of the four promising RL-techniques that have been identified, can be implemented and evaluated quantitatively in a practical implementation. This is what we aim to do in the future, in the form of a master thesis project.

Bibliography

1. Azevedo, C. R. B., Raizer, K. & Souza, R. *A vision for human-machine mutual understanding, trust establishment, and collaboration* in *2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)* (2017), 1–3.
2. Ajoudani, A. *et al.* Progress and prospects of the human–robot collaboration. *Autonomous Robots* **42,** 957–975 (2018).
3. Fong, T., Thorpe, C. & Baur, C. *Collaborative control: A robot-centric model for vehicle teleoperation* (Carnegie Mellon University, The Robotics Institute Pittsburgh, 2001).
4. Briscoe, N. Mercedes set to take legal responsibility for 'autonomous' car crashes. *The Irish Times* (Apr. 2022).
5. Burke, C. S., Stagl, K. C., Salas, E., Pierce, L. & Kendall, D. Understanding team adaptation: a conceptual analysis and model. *Journal of Applied Psychology* **91,** 1189 (2006).
6. Doorewaard, H., Van Hootegem, G. & Huys, R. Team responsibility structure and team performance. *Personnel review* **31,** 356–370. ISSN: 0048-3486. `https://doi.org/10.1108/00483480210422750` (Jone 2002).
7. Katz-Navon, T. Y. & Erez, M. When Collective- and Self-Efficacy Affect Team Performance: The Role of Task Interdependence. *Small Group Research* **36,** 437–465. eprint: `https://doi.org/10.1177/1046496405275233`. `https://doi.org/10.1177/1046496405275233` (2005).
8. Johnson, M. *et al.* Coactive Design: Designing Support for Interdependence in Joint Activity. **3,** 43–69. `https://doi.org/10.5898/JHRI.3.1.Johnson` (Feb. 2014).
9. van Zoelen, E. M., van den Bosch, K., Rauterberg, M., Barakova, E. & Neerincx, M. Identifying Interaction Patterns of Tangible Co-Adaptations in Human-Robot Team Behaviors. *Frontiers in Psychology* **12.** ISSN: 1664-1078. `https://www.frontiersin.org/article/10.3389/fpsyg.2021.645545` (2021).
10. van Zoelen, E. M., van den Bosch, K. & Neerincx, M. Becoming Team Members: Identifying Interaction Patterns of Mutual Adaptation for Human-Robot Co-Learning. *Frontiers in Robotics and AI* **8.** ISSN: 2296-9144. `https://www.frontiersin.org/article/10.3389/frobt.2021.692811` (2021).
11. van den Bosch, K., Schoonderwoerd, T., Blankendaal, R. & Neerincx, M. *Six Challenges for Human-AI Co-learning* English. in *Adaptive Instructional Systems - 1st International Conference, AIS 2019, Held as Part of the 21st HCI International Conference, HCII 2019, Proceedings* (eds Sottilare, R. & Schwarz, J.) 1st International Conference on Adaptive Instructional Systems, AIS 2019, held as part of the 21st International Conference on Human-Computer Interaction, HCI International 2019 ; Conference date: 26-07-2019 Through 31-07-2019 (Springer, 2019), 572–589. ISBN: 9783030223403.
12. Shafti, A., Tjomsland, J., Dudley, W. & Faisal, A. A. *Real-world human-robot collaborative reinforcement learning* in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), 11161–11166.
13. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor* in *International conference on machine learning* (2018), 1861–1870.
14. Jung, Y. & Lee, K. M. Effects of physical embodiment on social presence of social robots. *Proceedings of PRESENCE,* 80–87 (2004).
15. Kober, J., Bagnell, J. A. & Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* **32,** 1238–1274. eprint: `https://doi.org/10.1177/0278364913495721`. `https://doi.org/10.1177/0278364913495721` (2013).

16. Akalin, N. & Loutfi, A. Reinforcement learning approaches in social robotics. *Sensors* **21,** 1292 (2021).

17. Dash, S. S., Nayak, S. K. & Mishra, D. in *Intelligent and Cloud Computing* 495–507 (Springer, 2021).

18. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction* (MIT press, 2018).

19. Neftci, E. O. & Averbeck, B. B. Reinforcement learning in artificial and biological systems. *Nature Machine Intelligence* **1,** 133–143 (2019).

20. Watkins, C. J. C. H. & Dayan, P. Q-learning. *Machine Learning* **8,** 279–292. ISSN: 1573-0565. `https://doi.org/10.1007/BF00992698` (May 1992).

21. Dietterich, T. G. *et al. The MAXQ Method for Hierarchical Reinforcement Learning.* in *ICML* **98** (1998), 118–126.

22. Haarnoja, T. *et al.* Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

23. Ramstedt, S. & Pal, C. J. Real-Time Reinforcement Learning. *CoRR* **abs/1911.04448.** arXiv: `1911.04448`. `http://arxiv.org/abs/1911.04448` (2019).

24. Weber, K., Ritschel, H., Lingenfelser, F. & André, E. Real-time adaptation of a robotic joke teller based on human social signals (2018).

25. Hester, T. & Stone, P. TEXPLORE: real-time sample-efficient reinforcement learning for robots. *Machine Learning* **90,** 385–429. ISSN: 1573-0565. `https://doi.org/10.1007/s10994-012-5322-7` (Mar. 2013).

26. Knox, W. B. & Stone, P. *Interactively shaping agents via human reinforcement: The TAMER framework* in *Proceedings of the fifth international conference on Knowledge capture* (2009), 9–16.

27. Celemin, C. & Ruiz-del-Solar, J. *COACH: Learning continuous actions from COrrective Advice Communicated by Humans* in *2015 International Conference on Advanced Robotics (ICAR)* (2015), 581–586.

28. Kaelbling, L. P., Littman, M. L. & Moore, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research* **4,** 237–285 (1996).

29. Lauer, M. & Riedmiller, M. *An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems* in *In Proceedings of the Seventeenth International Conference on Machine Learning* (Morgan Kaufmann, 2000), 535–542.

30. Kapetanakis, S. & Kudenko, D. *Improving on the reinforcement learning of coordination in co-operative multi-agent systems* in *Second AISB Symposium on Adaptive Agents and Multi-Agent Systems* (2002).

31. Chan, J. & Nejat, G. Social Intelligence for a Robot Engaging People in Cognitive Training Activities. *International Journal of Advanced Robotic Systems* **9,** 1 (Oct. 2012).

32. Stolle, M. & Precup, D. *Learning options in reinforcement learning* in *International Symposium on abstraction, reformulation, and approximation* (2002), 212–223.

33. Moro, C., Nejat, G. & Mihailidis, A. Learning and Personalizing Socially Assistive Robot Behaviors to Aid with Activities of Daily Living. *J. Hum.-Robot Interact.* **7.** `https://doi.org/10.1145/3277903` (Oct. 2018).

34. Papaioannou, I., Dondrup, C., Novikova, J. & Lemon, O. *Hybrid Chat and Task Dialogue for More Engaging HRI Using Reinforcement Learning** in (Sept. 2017).

35. Hemminahaus, J. & Kopp, S. *Towards Adaptive Social Behavior Generation for Assistive Robots Using Reinforcement Learning* in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI* (2017), 332–340.

36. Chan, J. & Nejat, G. *A learning-based control architecture for an assistive robot providing social engagement during cognitively stimulating activities* in *2011 IEEE International Conference on Robotics and Automation* (2011), 3928–3933.

37. Chan, J. & Nejat, G. *Minimizing task-induced stress in cognitively stimulating activities using an intelligent socially assistive robot* in *2011 RO-MAN* (2011), 296–301.

38. Makar, R., Mahadevan, S. & Ghavamzadeh, M. *Hierarchical Multi-Agent Reinforcement Learning* in *Proceedings of the Fifth International Conference on Autonomous Agents* (Association for Computing Machinery, Montreal, Quebec, Canada, 2001), 246–253. ISBN: 158113326X. `https://doi.org/10.1145/375735.376302`.

39. Dietterich, T. G. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of artificial intelligence research* **13,** 227–303 (2000).

40. Haarnoja, T., Tang, H., Abbeel, P. & Levine, S. *Reinforcement learning with deep energy-based policies* in *International Conference on Machine Learning* (2017), 1352–1361.

41. Sigaud, O. & Pierrot, T. `https://rl-vs.github.io/rlvs2021/class-material/pg/12_sac.pdf`.

42. Schulman, J., Levine, S., Abbeel, P., Jordan, M. & Moritz, P. *Trust Region Policy Optimization* in *Proceedings of the 32nd International Conference on Machine Learning* (eds Bach, F. & Blei, D.) **37** (PMLR, Lille, France, July 2015), 1889–1897. `https://proceedings.mlr.press/v37/schulman15.html`.

43. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

44. Lillicrap, T. P. *et al.* Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

45. Fujimoto, S., van Hoof, H. & Meger, D. *Addressing Function Approximation Error in Actor-Critic Methods* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, Oct. 2018), 1587–1596. `https://proceedings.mlr.press/v80/fujimoto18a.html`.