Artificial Intelligence 283 (2020) 103258

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Qualitative case-based reasoning and learning

Thiago Pedro Donadon Homem [a,b,*], Paulo Eduardo Santos [b,e], Anna Helena Reali Costa [c], Reinaldo Augusto da Costa Bianchi [b], Ramon Lopez de Mantaras [d]

[a] IFSP - Federal Institute of São Paulo, São Paulo, SP, Brazil
[b] FEI - University Center FEI, São Bernardo do Campo, SP, Brazil
[c] USP - University of São Paulo, São Paulo, SP, Brazil
[d] CSIC - Spanish National Research Council, Barcelona, Spain
[e] School of Science and Technology, Flinders University, Adelaide, Australia

ARTICLE INFO

Article history:
Received 29 June 2018
Received in revised form 17 June 2019
Accepted 23 February 2020
Available online 20 March 2020

Keywords:
Case-based reasoning
Qualitative spatial reasoning
Reinforcement learning
Robot soccer

ABSTRACT

The development of autonomous agents that perform tasks with the same dexterity as performed by humans is one of the challenges of artificial intelligence and robotics. This motivates the research on intelligent agents, since the agent must choose the best action in a dynamic environment in order to maximise the final score. In this context, the present paper introduces a novel algorithm for Qualitative Case-Based Reasoning and Learning (QCBRL), which is a case-based reasoning system that uses qualitative spatial representations to retrieve and reuse cases by means of relations between objects in the environment. Combined with reinforcement learning, QCBRL allows the agent to learn new qualitative cases at runtime, without assuming a pre-processing step. In order to avoid cases that do not lead to the maximum performance, QCBRL executes case-base maintenance, excluding these cases and obtaining new (more suitable) ones. Experimental evaluation of QCBRL was conducted in a simulated robot-soccer environment, in a real humanoid-robot environment and on simple tasks in two distinct gridworld domains. Results show that QCBRL outperforms traditional RL methods. As a result of running QCBRL in autonomous soccer matches, the robots performed a higher average number of goals than those obtained when using pure numerical models. In the gridworlds considered, the agent was able to learn optimal and safety policies.

1. Introduction

Much research in Artificial Intelligence (AI) is aimed at creating intelligent agents with human-level problem solving capabilities; take for instance the recent successes of AI agents playing against humans in a Jeopardy scenario [17,7], or on playing Go [59]. An essential part of the human problem solving ability is the identification of previous experiences that could be reused to solve new issues. The main contribution of the present paper is the definition of a novel Case-Based Reasoning (CBR) method, called Qualitative Case-Based Reasoning and Learning (QCBRL), that represents cases by means of a Qualitative Spatial Reasoning (QSR) formalism that also serves as the basis for case retrieval and reuse methods. New

* Corresponding author.
E-mail address: thiagohomem@ifsp.edu.br (T.P.D. Homem).

https://doi.org/10.1016/j.artint.2020.103258
0004-3702/

cases are learned by partially running a Reinforcement Learning (RL) algorithm, that is a sub-task of QCBRL. The proposal introduced in this paper is evaluated on robot-soccer domains that, not only are dynamic environments in which each robot must excel its action choices, but it is also a domain where state-of-the-art artificial agents are far from achieving a human-level performance.

The central idea of QCBRL is to model domains using a qualitative spatial representation of directions called $\mathcal{EOPRA}$ [42]. QCBRL creates a compact world model representation and solves the exponential space complexity problem described in Albrecht and Stone [3]. Case retrieval and reuse are executed assuming Conceptual Neighbourhood Diagrams (CND) [23] and a qualitative similarity function, that computes the similarity between a new problem and an element of the case base, retrieving the most similar case to a given situation and reusing its solution to solve a new problem [28]. New cases are learned by executing a partial RL method; that is, when no similar case is retrieved, QCBRL iterates the RL method until the occurrence of the first successful episode. The set of states and actions of this successful episode is, then, stored as a new case [30]. The case-based maintenance is inspired by the work of Yan et al. [71]. It is performed by considering a *trust value* for each case that is incremented (or decremented) when the retrieved case solves (or not) the problem. Since the partial RL algorithm may return non-optimal actions, the cases with *trust values* that meet the removal conditions are deleted. With these procedures, the method described in this paper executes the complete CBR cycle of Retrieval, Reuse, Revision and Retention as defined in Aamodt and Plaza [1].

QCBRL can model any domain where objects' positions can be represented as qualitative relations with respect to an elevated observer (where the observer may need to interact with the environment to solve a given problem). The present work was evaluated in a simulated robot-soccer environment, running the RoboCup Soccer 2D Simulator, in a real humanoid-robot environment and on simple tasks in two distinct gridworld domains, as described in Section 5.

In the robot soccer simulated environment, empirical tests were performed on the Half Field Offense (HFO) task, a soccer game played over one half of the soccer field, where a team of offensive robot players attempts to score a goal on a team of defensive players [31]. In the real robot domains, the tests were performed with humanoid robots in an environment similar to HFO, where an attacker robot attempts to score a goal on a team of defensive robots. In the simulations, results were compared to the traditional RL method of Hausknecht et al. [25], to our previous work that implements a qualitative RL method [29] and to a qualitative CBR system that does not execute case-based maintenance [30]. Aiming to show that QCBRL opens new directions in different research fields, and that it can be applied to other domains, this paper describes tests in two further scenarios: first, a simple gridworld was considered whereby scalability of our proposal is investigated considering grids of increasing sizes; and, second, we present a solution to the lava safety gridworld defined in [37] (as presented in Section 5.2).

The next section presents the background knowledge upon which this research was built.

2. Research background

This section presents the three main methods used in this work, Case-Based Reasoning, Reinforcement Learning and Qualitative Spatial Reasoning.

2.1. Case-based reasoning

Case-Based Reasoning (CBR) is an AI paradigm that uses the knowledge obtained in past situations, referred as *cases*, to solve new problems. Aamodt and Plaza [1] presented a review of CBR, describing the CBR cycle and the core CBR problem-solving processes: Retrieval, Reuse, Revision and Retention. Given a new problem, the retrieval process searches in the case base for the most similar cases and, through a similarity evaluation, selects the case with the greatest potential to be reused. The reuse process tries to apply the selected case as a solution to the problem at hand, making some adaptations in the case description if necessary. The revision process evaluates the proposed solution and, if the retrieved case has successfully solved the problem, the retention process stores the problem and the proposed solution as a new case [69].

According to Richter and Weber [51], cases can have a complex description since they represent the knowledge, that is, the experience in solving a problem. Ros et al. [54] defines a case C as a triple:

$$C = (P, A, K), \tag{1}$$

where P is the problem description (or part of the problem situation), A is the solution description (or part of the solution) that describes how the problem can be solved and K is the case scope that defines elliptic regions around the objects' positions.

The problem description (P) represents the problem situation according to the domain and the solution description (A) is the sequence of actions an agent can perform to solve P.

As CBR aims to solve a problem considering the context, or how it is represented, different similarity measurement definitions can be used to retrieve the most appropriate case. Depending on the characteristics of the cases, the use of an inadequate similarity measurement method can result in an inadequate case retrieval and, consequently, in an inadequate solution. Similarity strategies can combine different traditional methods, as those proposed in Burkhard [11], Khajotia et al. [33], Pal and Shiu [44], Kendall-Morwick and Leake [32] and Zeyen et al. [73].

In this work the entire CBR cycle of Retrieval, Reuse, Revision and Retention is implemented using a combination of Reinforcement Learning (RL) and Qualitative Spatial Reasoning (QSR) methods. RL is described in the next section.

2.2. Reinforcement learning

Reinforcement Learning (RL) is a traditional machine learning method whereby an agent learns through interactions with the environment, without assuming any prior knowledge [64]. RL relies on the agent's actions and the feedback from the environment obtained after the actions' execution; this feedback is called reward or reinforcement. The goal of RL is to maximise the reinforcement during the agent interactions with the environment [56]. Reinforcement values observed at each state transition could be positive (e.g., in the soccer domain it could represent a scored goal) or negative (i.e., a punishment, when the ball is defended or is out of bounds in the soccer scenario).

An RL problem can be formalised as a Markov Decision Process (MDP) composed of a 4-tuple (S, A, T, R), where S is a finite set of states, A is a finite set of possible actions, $T : S \times A \times S \rightarrow [0, 1]$ is a transition function. This function specifies the probability $T(s, a, s')$ with which an agent moves to the next state s' when an action a is executed in a state s, and $R : S \rightarrow \mathbb{R}$ is the reinforcement function, a numerical value $r = R(s)$ obtained when the state s is reached.

In an RL problem, the agent's goal is to maximise a cumulative reward, i.e., it aims to learn an optimal policy $\pi^* : S \rightarrow A$ that maps the best action for each state. A common way of obtaining π^* is by Temporal Difference (TD) methods, where the agent iteratively learns the action-value function $Q : S \times A \rightarrow \mathbb{R}$. According to Watkins and Dayan [68], starting from the current state/action pair, the function maps all the combinations of state and action to an estimated long-term reward.

Among the RL methods that iteratively approximate the Q function ($Q(s, a)$), the two most used are Q-learning [67,68] and State-Action-Reward-State-Action (SARSA) [55,60]. They differ from each other on how the Q-value function is updated for each state-action pair: Q-Learning updates the maximum Q-value of the next state, while SARSA updates Q-values with respect to the Q-value of the next action to be taken. Therefore, Q-learning is considered an *off-policy* method because target and behaviour policies are different, i.e., the policies used to generate the current action and the next action are different, while SARSA is considered an *on-policy* method because target and behaviour policies are the same, i.e., the policies used to generate the current and the next actions are the same [64].

In this work, SARSA was chosen as the key RL method, since *off-policy* methods can be unstable with some function approximations [63]. According to Sutton and Barto [64] the distribution of updates in *off-policy* methods with function approximation is still a challenge because the state-action pairs are updated according to a different distribution and this makes the estimated values diverge to infinity. In this case, however, *on-policy* methods are well suited since the distribution of updates of the state-action pairs are the same [46]. SARSA aims to maximize the following Q-function:

$$Q(s, a) = (1 - \alpha) Q(s, a) + \alpha [r + \gamma Q(s', a')], \tag{2}$$

where s is the current state, a is the current action, s' is the next state, a' is the next action, $r \in R$ is the reinforcement obtained when applying a in s, $\alpha \in (0, 1)$ is the learning rate and $\gamma \in [0, 1]$ is the discount factor.

SARSA algorithm can also be augmented with eligibility traces and TD(λ) methods. In this case, it is named SARSA(λ) [64] and the method may learn more efficiently. SARSA(λ) is similar to the SARSA algorithm, except for the eligibility traces, that keep a record and are updated for every visited state-action pair. Algorithm 1 presents the SARSA(λ) method that is used in the *Problem Solver* process in this work.

As the Problem Solver is the key to achieve a feasible solution, various RL methods can be used in the interest of improving the learning performance. For instance, a Multi-step Q(σ) [64] could be used, that executes a mixture of full-sampling (SARSA, $\sigma = 1$) and pure-expectation (Tree-backup(σ), $\sigma = 0$), presenting a better performance than SARSA on extreme cases ($\sigma = 0$ or 1) [4]. Yang et al. [72] present a novel algorithm, named $Q^{\pi}(\sigma, \lambda)$, that unifies SARSA(λ) and Q(λ) by defining a mixed-sampling operator. This operator facilitates the application of methods that vary from pure-expectation to full-sampling.

In this work, the domain is modelled as an MDP where the states are described by the positions, orientations and relative locations of robots that are represented by means of a qualitative spatial reasoning formalism.

Algorithm 1 SARSA(λ) algorithm [64].

```
function SARSA(λ)
    Initialize Q(s, a) arbitrarily and e(s, a), for all s, a
    repeat (for each episode)
        Initialize s, a
        repeat (for each step of episode)
            Take action a, observe r,s'
            Choose a' from s' using policy derived from Q (ε-greedy)
            δ ← r + γ Q(s', a') − Q(s, a)
            e(s, a) ← e(s, a) + 1
            for all s, a do:
                Q(s, a) ← Q(s, a) + αδe(s, a)
                e(s, a) ← γλe(s, a)
            end for
            s ← s'; a ← a'
        until s is terminal
    until some stopping criterion is reached
end function
```

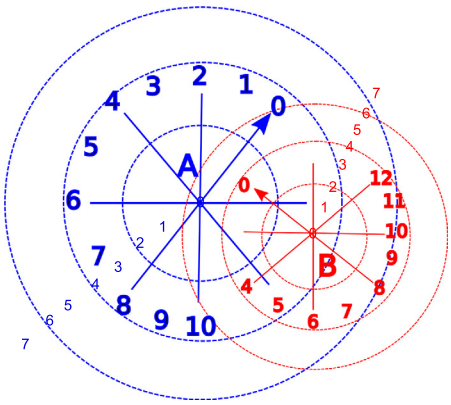


Fig. 1. $\mathcal{EOPRA}_4$ relation $A\ {}_4\angle^{1}_{13}\ {}^{5}_{3}\ B$. Adapted from Homem et al. [28].

This work considers that a *successful episode* occurs when an agent achieves the final state (i.e., in the robot-soccer case, when the agent scores a goal). Complementary, an *unsuccessful episode* occurs when the goal is not achieved (for instance, when the ball is intercepted by the opponent agent or when it goes out of the bounds of the field, in the soccer scenario).

2.3. Qualitative spatial reasoning

Qualitative Spatial Reasoning (QSR) is a subfield of Knowledge Representation in AI that aims at representing and reasoning about spatial relations between elementary spatial entities, without relying on quantitative methods [13]. Some of the most common QSR formalisms found in the literature (cf. Ligozat [38], Cohn and Renz [13]) are, for instance: the Region Connection Calculus (RCC) [48] that represents mereotopological relations between spatial regions by means of a connectivity relation; the Region Occlusion Calculus (ROC) [49], that uses RCC to model occlusion relations between two bodies; the Cardinal Direction Calculus (CDC) [21] or the *Star Calculus* [50] which model the relative directions between pairs of objects; and the Oriented Point Relation Algebra ($\mathcal{OPRA}$), that represents objects in the space in terms of oriented points [42].

Due to the large variety of distinct QSR formalisms, these methods find applications in a number of domains, such as robot navigation and self-localization [57], geographic information systems [20], cognitive linguistics [66,53] among others [13,70]. The present paper builds upon our previous work where an extension of the $\mathcal{OPRA}$, the Elevated Point Relation Algebra $\mathcal{EOPRA}$ [42], was applied to represent the agents in the robot-soccer domain [27,28]. $\mathcal{EOPRA}$ assigns an intrinsic orientation to objects and defines a qualitative distance based on an elevated point in the domain, that could be defined as the height of the observer [42]. The relative distance between objects is set in terms of this elevation as a distance around the observer on the 2D-plane defining a proximity region. In a humanoid-robot soccer domain, the height of the robots can be considered as elevated points [15].

In $\mathcal{EOPRA}$, the world can be discretised according to two parameters m and n, representing respectively the granularity of the distance (m) and of the orientation (n). For instance, considering the granularity $m = n$ for notational simplicity, Fig. 1 shows an example of an $\mathcal{EOPRA}_4$ relation between two elevated points (A and B), where $A_4\angle^{1}_{13}\ {}^{5}_{3}B$ represents that both A and B are discretised into 16 orientation relations ($4m$) and 8 distance relations ($2m$). For the relative orientation, A is in the sector 1 of B and B is in the sector 13 of A; and for the relative distance, A is in the sector 5 of B and B is in the sector 3 of A [28]. In this work, these sectors are grouped into regions, reducing the number of boundaries.

Besides $\mathcal{EOPRA}$, the present paper uses an important tool of qualitative spatial reasoning, the Conceptual Neighbourhood Diagram (CND) [23]. A CND can be defined as a graph representing, at each of its nodes, a single relation between spatial entities, whereby the edges connect a pair of conceptual neighbours. In other words, an edge represents that the transition from the pair of relations connected is smooth (i.e. there is no other relation of the set that represents the transition from one relation to the other). In this work, a CND is used as a tool to measure the distance between cases and it is an essential element in the case retrieval process.

In our previous work, the cases were hand-coded and the focus was on the retrieval and reuse processes [27,28]. So, in order to verify the learning phase, we have analysed the behaviour of the RL method with discretised states [29]. Finally, we have integrated a learning phase to the CBR system [30]. The present paper reports our most recent progress on integrating CBR, QSR and RL (into the QCBRL algorithm), while also extending our previous work with the implementation of case-base maintenance.

3. Related work

The development of hybrid systems integrating CBR and other AI methods has contributed with interesting previous related research. Bianchi et al. [9] proposed a hybrid method that uses CBR as an heuristic to accelerate RL methods (CB-HAQL). According to the authors, when a similar case is retrieved, an heuristic $H(s, a)$ is calculated with the sequence of

actions of the case, up to the number of actions stored in that case. Bianchi et al. [8] also proposed a case-base approach for transferring knowledge. The method works in three stages, where the first stage runs an RL algorithm to compute the optimal policy that populates the case; the second stage maps actions from a source domain to a target domain using a Neural Network; the third stage uses CB-HAQL [9] to solve the problem in the target domain. In contrast, the QCBRL does not execute a first step to populate the case base with an optimal policy. There is no guarantee of optimality, but the case-base maintenance corrects and improves the knowledge base.

More recently, Celiberto et al. [12] extended the work presented in [9] to be applied in a transfer-learning setting for real robots. Bianchi et al. [10] proposed a new Q(λ) algorithm for the same problem. The work presented in this paper differs from these previous approaches by creating the cases at runtime, without waiting for the RL algorithm stabilisation. Besides, we do not consider a transfer learning setting in this paper.

One of the first applications of CBR in real robot-soccer domain was proposed by Ros et al. [54], where CBR retrieves the most similar case to a given situation and the robots perform coordinated actions. In that paper, a Cartesian coordinate system is used to represent the positions of objects in a field. In contrast, the present work discretises the robot environment according to a qualitative spatial reasoning formalism. Besides, a new retrieval method is proposed that uses a CND to compute a similarity measure between the new problem and the cases in the case base.

The work presented in Auslander et al. [5] also integrates RL to CBR in a team-based first-person shooters game. This work stores the Q-table for each created case and, when a case is retrieved, the reward function updates the associated Q-table. In the present paper, the cases are composed of the problem description, the solution description and the *trust value*, and there is a single Q-table, updated only when an RL iteration occurs.

Floyd et al. [19] proposed a CBR approach to imitate human soccer players. The case base is populated with cases created through the *log* file of actual soccer matches, where each case represents a spatial configuration of the game.

Inspired by recent work on deep reinforcement learning [41,39], Hausknecht and Stone [26] analysed different approaches implementing Deep Q-Networks (DQN) and applied them to a robot-soccer player attempting to score at a non-defended goal. Only one approach resulted in an average of 100% of goals scored, while the others DQN approaches obtained an average of 99%, 98%, 96%, 94%, 84% and 80% goals. This indicates that DQN is an interesting strategy that can improve the results of HFO. The DQN agent, however, takes about three days to accomplish the training phase, whereas the QBCRL cycle proposed in the present paper learns new cases at runtime.

Knowledge representation and reasoning are important features in decision-making problems and they influence the ability of the agent to solve a problem, as argued in Tenorth and Beetz [65] and Ramirez-Amaro et al. [47]. Tenorth and Beetz [65] propose distinct levels of abstractions to maintain the original information, facilitating reasoning about symbolic representations. Ramirez-Amaro et al. [47] applies *rules* and a reasoning engine that generates new representations about the world and new relations between representations, creating new nodes on demand. Freire and Costa [22] and Koga et al. [34] investigate the state discretisation (or state abstraction) in RL methods. The authors defined that the states s (and the actions a) of RL methods can be represented by an abstract state (and an abstract action).

With a focus on decision-making problems, Agostini et al. [2] presents a framework for robotic applications that integrates planning and learning processes. The planner generates a plan that is executed and monitored by the system, but if a suitable plan is not found, a human teacher informs the action to be executed.

Kuipers [36] presents the Spatial Semantic Hierarchy (SSH), a formalism that models the knowledge of large-scale space in terms of multiple representations. The SSH representation uses qualitative and quantitative information and organises the spatial knowledge as a hierarchy of levels. This allows the agent to learn and solve problems even with partial knowledge. In our work, the vision system can be expressed as one level of the hierarchy, that receives the quantitative distance and orientation of the objects, converts them into qualitative regions and, by using the CND, measures qualitatively the distance from the cases to the problem.

Another way to categorise qualitative regions is the construction of fuzzy interval algebra, as described by Ligozat [38]. This formalism can be used when the frontiers between the two qualitative regions cannot be exactly determined. Considering a robot domain, Schiffer et al. [58] presents an application example with service robots that represents qualitative positional information of the objects with fuzzy sets. The authors have discretised the world according the room size. In contrast, in the present paper we have discretised the world according the humanoid robot's size.

Recently, Yang et al. [72] presented a new framework that integrates RL to symbolic planning, where the symbolic plans guide the agent's task and the agent's experience feeds back to the system, improving the plan.

Orduña Cabrera and Sànchez-Marrè [43] use a case-base stochastic learning method and propose a dynamic adaptive case base. The cases are clustered by similarity and new cases are assigned to a specific cluster, reducing the retrieval time. In contrast, in the present paper, the case base is built dynamically and cases are modelled following a QSR formalism.

Much effort in AI research is devoted to the development of intelligent robots endowed with skills of locomotion, localisation, image recognition, planning (among others) in order to operate in the human space. Within the myriad of open issues in this field, the present paper is concerned with the modelling and the development of autonomous robotic agents with individual abilities allowing them to efficiently solve problems in a collaborative manner. This context motivates the use of the robot-soccer domain as application test bed for groups of intelligent (humanoid) robots, as this is a dynamic environment in which each robot in a team must choose the best individual action in order to improve the overall team's performance. This can be solved as a machine learning task in which each agent needs to learn a sequence of actions that results in scoring a goal. This has been the motivation of several developments in the field, such as the work of Kuhlmann

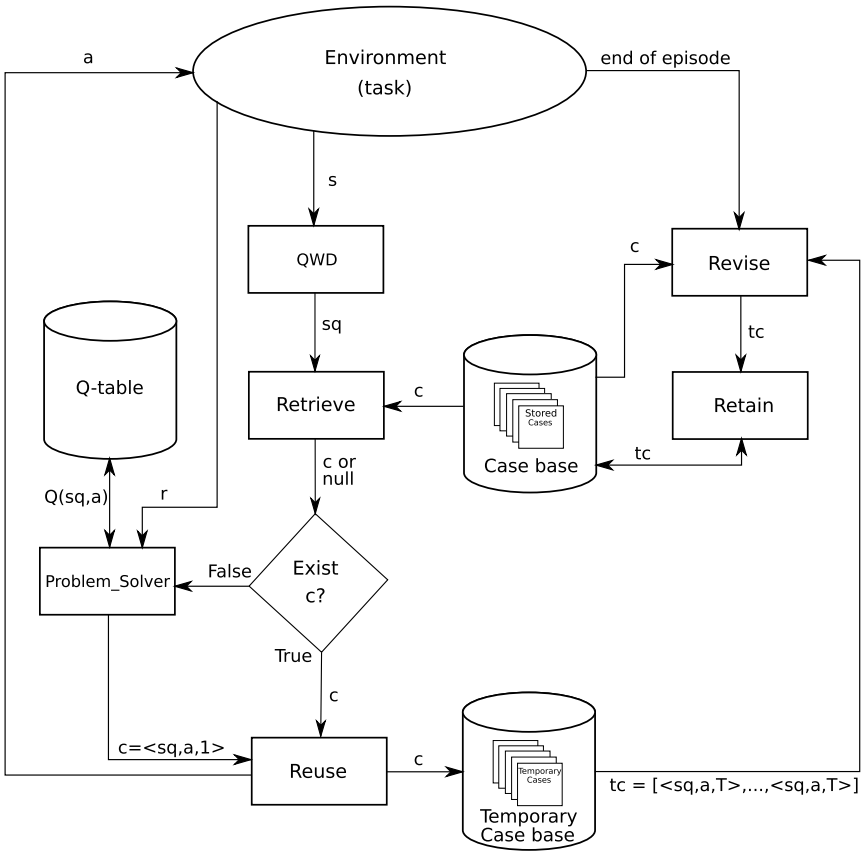


Fig. 2. QCBRL method.

and Stone [35] and Stone et al. [62] on the *Keepaway* problem, Barrett and Stone [6] and Hausknecht et al. [25] on the *Half Field Offense* problem, Floyd and Esfandiari [18] and Fabro et al. [16] on the robot-soccer simulation problem. The research presented in these papers motivated the development of the Qualitative Case-Based Reasoning and Learning (QCBRL) method proposed in this paper, integrating CBR with qualitative spatial reasoning and reinforcement learning. QCBRL is introduced in the next section.

4. Qualitative case-based reasoning and learning

This section describes the method proposed in this work: Qualitative Case-Based Reasoning and Learning (QCBRL), including a description of the qualitative spatial case modelling; of the $\mathcal{EOPRA}$ CND; of the CBR and RL integration; and, of how the case-base maintenance is performed.

The main hypothesis of this work is that the integration of CBR, QSR and RL methods allows the agent to learn the best action for each situation, running a partial RL method and that the use of qualitative relations facilitates a faster retrieval than that of numerical CBR systems (such as Ros et al. [54]). In the method proposed in the present paper, the use of a partial RL method with case-base maintenance outperforms more traditional methods, removing poor cases and excluding the need for repeated RL iterations until the algorithm convergence.

Fig. 2 and Algorithm 2 represent the QCBRL method, where the following abbreviations are used: CB is the case base; tc is the temporary case base; s is a state; sq is a qualitative state; a is an action; c is a case; QWD is the *Qualitative World Discretisation* process; $Retrieve$ is the *Retrieval* process; $Problem\_Solver$ is the *Problem Solver* process; $Reuse$ is the *Reuse process*; and $Revise$ and *Retain* are, respectively, the *Revision* and the *Retention* process. The algorithm assumes that the environment with which the agent interacts, or where it performs tasks, informs when an end of episode occurs by means of a flag that indicates whether the agent had solved (or not) the problem.

The QCBRL method starts with an empty case base. For each episode, the temporary case base (tc) is emptied. While an end of episode does not occur, the agent observes the environment and acquires numeric sensor data[1] (line 6). The *Qualitative World Discretisation* process (QWD) discretises the numerical data into qualitative relations of orientation and

[1] Sensor data can be obtained by camera, sonar, inertial measurement unit (IMU), GPS or any other device in the simulator or in the real robots.

Algorithm 2 QCBRL method.

```
1: function QCBRL
2:     CB ← ∅
3:     repeat (for each episode)
4:         tc ← ∅
5:         while (an end of episode does not occur) do
6:             Observe the state s
7:             sq ← QWD(s)
8:             c ← Retrieve(sq, CB)
9:             if c = NULL then
10:                a ← Problem_Solver(sq)
11:                trust_value ← 1
12:                c ← < sq, a, trust_value >
13:            end if
14:            Reuse(c, tc)
15:        end while
16:        Revise(tc, CB)
17:        Retain(tc, CB)
18:    until some stopping criterion is reached
19: end function
```

distance. The process considers the front of the agent and the object's position with respect to the agent and returns a qualitative state (line 7). The *Retrieval* process measures the similarity between the new problem and the cases in the case base and retrieves the most similar case (lines 8). If no similar case is found, the *Problem Solver* process is executed returning an action associated with that qualitative state (line 10), the trust value gets a confidence level of 100% (line 11) and line 12 stores a case (a new case composed of the problem, the new solution and the trust value triple). The *Problem Solver* process also waits for a reward from the environment to perform the necessary updates. The *Reuse* process reuses the case (line 12) and stores the reused case in the temporary case base. At the end of the episode, the *Revision* process verifies if the reused case successfully solved the problem (line 14) and updates the *trust value* of the reused cases. Then, the *Retention* process performs the case-base maintenance, storing new cases and discarding cases with low *trust value*. Fig. 3 illustrates how QCBRL discretises the world and represents the qualitative objects' position. In the soccer field, w.r.t. robot B1 (Fig. 3(a), left), QCBRL finds the ball and discretises the objects' position in the environment, representing the qualitative relations in the CND (Fig. 3(a), right). The position of the objects is defined as a new problem (Fig. 3(b)). Finally, QCBRL selects from the case base the most similar cases to this problem (Fig. 3(c)).

The QCBRL method works as an episodic task; therefore, the system can retrieve or learn one or more cases until the problem is solved. QCBRL uses the concept of n-step on the *Retention* process. Thus, when the problem is solved, *Retention* updates the *trust value* of the retrieved cases or stores the new cases. These processes, and the related data flows, are schematised in Fig. 2. The next sections describe these processes in more detail.

4.1. Qualitative case representation

Inspired by the work of Ros et al. [54], a case (C) is defined as the problem description (P), the solution description (A) and the *trust value* (T), represented by:

$$C = (P, A, T). \tag{3}$$

The problem description (P) corresponds to the qualitative spatial relations between an agent and the objects in an environment, given by the qualitative distances and directions of the objects, from the agent's point of view. P is given by:

$$P = \{Ag : [Obj_1, Obj_2, \ldots, Obj_v]\}, \tag{4}$$

where v is the total number of objects an agent can perceive in its environment, Obj_i ($i \in \{1, \ldots, v\}$) is the qualitative relation between the agent Ag and the object i. In the robot-soccer domain, the objects can be the ball, the robots or domain features (such as the goal posts).

The solution description (A) describes the action (or a sequence of actions) that the agent should perform to solve the problem (or part of the problem). A can be defined as:

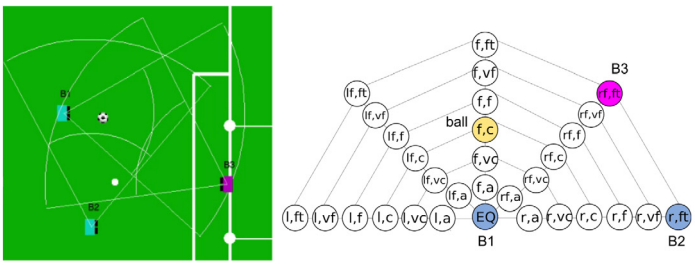
$$A = \{Ag : \{a_1, a_2, \ldots, a_p\}\}, \tag{5}$$

where p is the total number of actions in the case and a_j ($j \in \{1, \ldots, p\}$) is the action the agent Ag should perform.

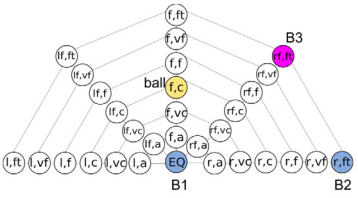
The *trust value* (T) refers to a confidence level, i.e., how much QCBRL trusts that the case can solve the problem. T is defined as:

$$T \in [0, 1] \tag{6}$$

where T is a real value, updated during the *Revision* process, as described in Section 4.6.

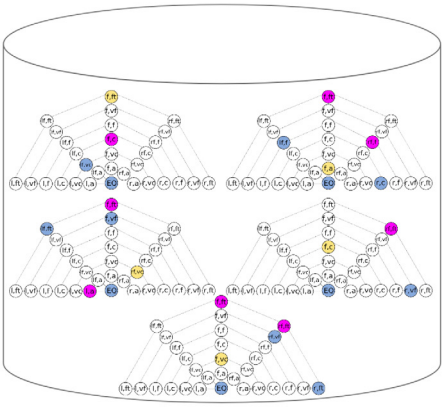


(a) Qualitative world discretisation w.r.t. robot B1



New Problem

(b) New problem



Case base

(c) Stored cases

Fig. 3. Qualitative world discretisation.

In contrast to the work described in Ros et al. [54], the case scope K is not necessary in the qualitative case representation since, in the present paper, objects are located in qualitative regions.

4.2. Qualitative world discretisation

Following our previous work [28], we consider that the viewpoint orientation is at the front of the agent and the granularity parameter is the same for the distance and orientation, i.e., $m = n = 6$. With respect to the direction, we obtained 24 sectors, that were grouped into 8 regions: *left(l)*, *right(r)*, *front(f)*, *back(b)*, *left-front(lf)*, *right-front(rf)*, *left-back(lb)* and *right-back(rb)*. For the elevated point and distance, we obtained 12 distance sectors, that were grouped into 6 regions: *at(a)*, *very close(vc)*, *close(c)*, *far(f)*, *very far(vf)* and *farthest(ft)*. Fig. 4 shows the distance (Fig. 4a) and orientation (Fig. 4b) regions created.

The granularity of $\mathcal{EOPRA}$ is domain dependent and was empirically defined in this paper based on the experiments previously conducted. In our initial tests we observed that if the discretisation is too rough (due to a coarse granularity), information is lost and QCBRL creates just a few cases. On the other hand, considering a larger number of regions (a fine granularity), the qualitative model approximates a numerical model and too many cases are created.

A CND for the spatial relations considered in this work is shown in Fig. 5. This diagram can be read as, for instance, if the agent is located at the centre of the CND (at the *EQ* region) and an object is placed on the *left* of and *far from* the agent, there are four relations in the diagram between the object and the agent. The regions (*lf,f*), (*l,vf*), (*lb,f*) and (*l,c*) are the direct neighbours in the CND with respect to the object-observer location and, therefore (assuming continuity of motion), these are the only possible regions the object can be placed with respect to the agent on the next step.

When the qualitative world discretisation receives a numerical state (s), it uses two algorithms. The first implements the qualitative *direction* discretisation: if s is an angle between the agent and an object, it is converted into the qualitative orientation region of s, and sq is returned. The second implements the qualitative *distance* discretisation: if s is a distance value (in *meters*), between the agent and an object, it returns sq that is the qualitative distance region where the object is located.

4.3. Retrieval

In a CBR system, the case retrieval process corresponds to measuring the similarities between the new problem and those stored in the case base, retrieving the most similar case to be used as a solution to the problem.

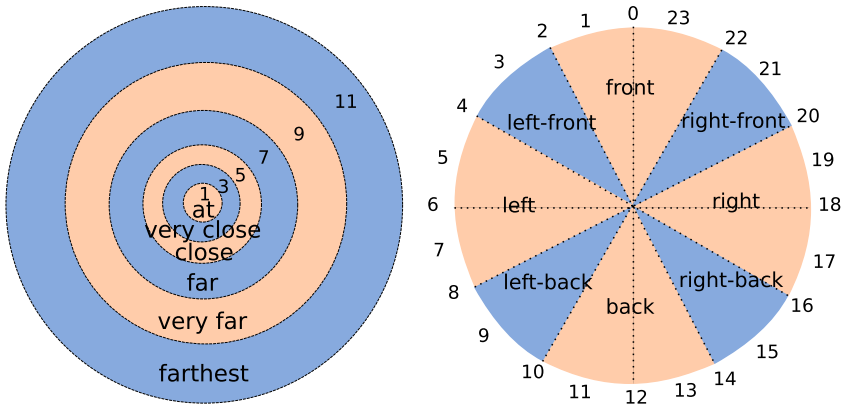


Fig. 4. a) Qualitative distance; b) qualitative orientation regions [28].

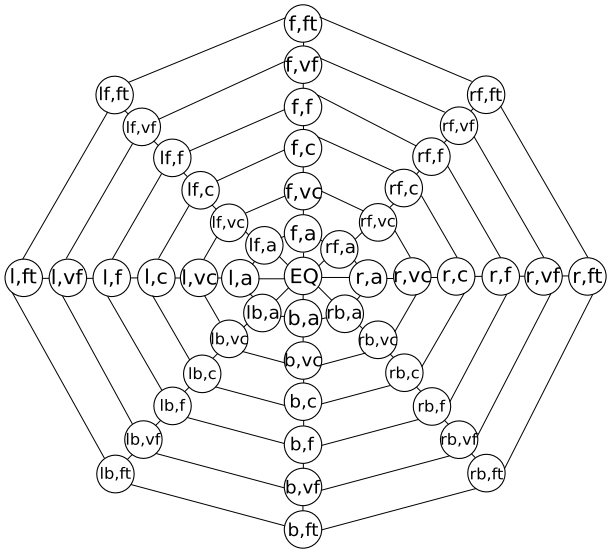


Fig. 5. CND of the proposed $\mathcal{EOPRA}_6$ representation [28].

As described in our previous work [28], the *Retrieval* process uses the distance between the objects in the CND to compute the similarity between the new problem and the cases in the case base. This is done by creating a similarity matrix for the $\mathcal{EOPRA}_6$, with the minimal distance (path) between each 49 relations computed using any algorithm to find the shortest path between nodes in a graph, such as Dijkstra's algorithm [14]. The minimum CND distance function is represented by $Dmin_\phi(X_1, X_2)$, for two spatial relations X_1 and X_2, that maps these regions to the minimum node-to-node distance in the CND [28]. The qualitative distance function ($Dist_Q(p, c)$) is defined as:

$$Dist_Q(p, c) = \sum_{i=1}^{v} Dmin_\phi(Obj_i{}^c, Obj_i{}^p), \tag{7}$$

where v is the total number of objects the agent can perceive, $Obj_i{}^c$ is the qualitative position of each object i in the case and $Obj_i{}^p$ is the qualitative position of each object i in the problem.

Then, a qualitative similarity function ($Sim_Q(p, c)$) can be defined as:

$$Sim_Q(p, c) = \frac{CND_{MaxDist} \times v - Dist_Q(p, c)}{CND_{MaxDist} \times v}, \tag{8}$$

where v is as defined as in Equation (7) and $CND_{MaxDist}$ is the maximum distance between two nodes in the CND. The result is normalised, so the similarity is bounded by 0 and 1.

Algorithm 3 represents the proposed retrieval method based on the CND distance measure, where *sim_candidates* is a list of cases whose similarity values is above a *threshold*. For each case on the case base, Algorithm 3 calculates the similarity and, if an equal case is found, this case is retrieved and the function ends. Otherwise, if similar cases are found (cases

Algorithm 3 Retrieval process.

```
1: function RETRIEVE(Qualitative state sq, Case base CB)
2:     sim_candidates ← ∅
3:     for each case c ∈ CB do
4:         sim_value ← Sim_Q(sq, c)
5:         if sim_value = 1 then
6:             return c
7:         else
8:             if sim_value > threshold then
9:                 sim_candidates ← {sim_value, c} ∪ sim_candidates
10:            end if
11:        end if
12:    end for
13:    if sim_candidates = ∅ then
14:        return NULL
15:    end if
16:    sort((sim_value, trust_value), sim_candidates)
17:    return first(c, sim_candidate)
18: end function
```

Algorithm 4 SARSA(λ) algorithm adapted from Sutton and Barto [64].

```
1: function PROBLEM_SOLVER(qualitative state sq)
2:     choose a from sq using policy derived from Q (ε-greedy)
3:     return action a; and wait until observed r
4:     if received reward r then
5:         observe sq'
6:         choose a' from sq' using policy derived from Q (ε-greedy)
7:         δ ← r + γ Q(sq', a') − Q(sq, a)
8:         e(sq, a) ← e(sq, a) + 1
9:         for all sq, a do:
10:            Q(sq, a) ← Q(sq, a) + αδe(sq, a)
11:            e(sq, a) ← γλe(sq, a)
12:        end for
13:        sq ← sq'; a ← a'
14:    end if
15: end function
```

with *sim_value* greater than a threshold value), the most similar case with the greatest *trust value* is retrieved with its corresponding action. If no similar case is found then a *null value* is returned.

4.4. Problem solver

The *Problem Solver* process aims to propose a solution to the problem when there is no similar cases available. The quality of the proposed solution is directly proportional to the quality of the new case created. In this work, given the similarity between the problem description (P) in CBR and the definition of state in the set of states S in RL, the method uses a partial RL algorithm in the *Problem Solver* process. By partial RL algorithm we mean that, instead of executing a full RL algorithm (iterating episodes until the Q-table converges to the optimal policy), the RL algorithm iterates until a first successful episode occurs.

Following the work of Stone et al. [63] and Hausknecht et al. [25], we have used the episodic SARSA(λ) that is an extension of SARSA that uses a decay rate (λ) for the *eligibility trace*, updating all the recently visited state-action values. As presented above, in this work the *Problem Solver* process executes SARSA(λ) until a successful episode occurs and a non-optimal case is stored as a new case. Since SARSA(λ) is not executed until its convergence, we name it *partial SARSA(λ)*. Algorithm 4 represents the proposed *Problem Solver* method based on SARSA(λ). The action is selected based on a policy derived from Q (line 2) and shared with the *Reuse* process to be executed (line 3). The process continues and waits until the reward is received (line 3). When the reward is received, the process executes a learning update (lines 5-13) using the observed information.

Other strategies can be executed to solve a new problem. For instance, running the SARSA(λ) algorithm before QCBRL starts, or randomly choosing actions. Section 5 presents the tests comparing our proposal with these strategies.

4.5. Reuse

The case reuse process consists of adapting the retrieved case to a given problem, or as proposed in our previous work [28], adapting the problem to the retrieved case. Instead, in this work the *Reuse* process is simplified and no adaptation is required. The agent performs the action of the retrieved case (obtained from the *Retrieval* process) or the action of the new (temporary) case (obtained from the *Problem Solver* process). Each case obtained from the *Problem Solver* process receives a

Algorithm 5 Reuse process.

```
1: function REUSE(case c, temporary case base tc)
2:     perform a of the case c
3:     insert(c,tc)
4: end function
```

Algorithm 6 Revision process.

```
1: function REVISE(temporary case base tc, case base CB)
2:     for each case c ∈ tc do
3:         if (the episode ended successfully) AND (c ∈ CB) then
4:             increment trust value of the case c
5:         else
6:             if (the episode ended unsuccessfully) AND (c ∈ CB) then
7:                 decrement trust value of the case c
8:             end if
9:         end if
10:    end for
11: end function
```

Algorithm 7 Retention process.

```
1: function RETAIN(temporary case base tc, case base CB)
2:     for each case c ∈ tc do
3:         if (the episode ended successfully) AND (c ∉ CB) then
4:             insert(c, CB)
5:         else
6:             if (the episode ended successfully) AND (c ∈ CB) then
7:                 update(c, CB)
8:             end if
9:         end if
10:    end for
11:    for each case c ∈ CB do
12:        if trust value of c < threshold then
13:            remove(c, CB)
14:        end if
15:    end for
16: end function
```

trust value equal to 1 ($T = 1$), and all the reused cases are stored in a temporary case base until the end of the episode. Algorithm 5 represents the *Reuse* process.

4.6. Revision

The *Revision* process verifies if the problem was solved. If it is a retrieved case, the *Revision* process updates the *trust value*, i.e., it increments 0.1 (up to 1) in T, increasing the trust that QCBRL has on this case to solve the problem. Otherwise, it decrements 0.1 (up to 0). These values were obtained empirically. Algorithm 6 represents the *Revision* process.

4.7. Retention

The *Retention* process performs the case-base maintenance. When a successful episode ends, the new cases stored in the temporary base are inserted as new in the case base and the cases that are already stored in the base are updated with a new *trust value*. When an unsuccessful episode end occurs, all the cases with a *trust value* less than a threshold are removed from the case base. Algorithm 7 represents the *Retention* process. As there is no evidence that the cases stored in the case base follow an optimal policy, this procedure performs a case-base maintenance, correcting and improving the knowledge base.

4.8. An example of the QCBRL method on the robot soccer domain

Let's consider an offensive robot in a soccer domain running the QCBRL method. The robot starts without prior knowledge, that is, QCBRL starts with an empty case base. The robot finds the ball, walks toward it and dominates it. The robot observes the soccer field and obtains the qualitative relations of the objects itself (the problem description, generated by the *Qualitative World Discretization* process). The robot retrieves a similar case. If no case is found, the *Problem Solver* process proposes an action (chooses an action using a policy derived from Q-learning). The *Reuse* process reuses the action and stores the case in a temporary case base. The *Problem Solver* process receives a reward and executes learning updates. If the episode does not finish, a new step of the episode is started and the QCBRL cycle continues, repeating the process execution. When the episode ends, the *Revision* process verifies if a successful end of episode occurred and, in this case, the *Retention*

process stores the new (temporary) cases in the case base. Otherwise, no new cases are learned. The temporary case base is emptied when a new episode is started.

If a new episode starts, the QCBRL repeats the process: the robot walks and takes possession of the ball; it observes the soccer-field and obtains the qualitative relations of the objects. The *Retrieval* process retrieves a similar case. If no case is found, the *Problem Solver* proposes an action and the sequence described above is repeated. Otherwise, if a similar case is found, the *Reuse* process reuses the action of the retrieved case and stores the case in a temporary case base. If the episode does not finish, a new step of the episode is started and the QCBRL cycle continues, repeating the process execution. When the episode ends, depending on the performed steps, the temporary case base may have cases retrieved and also new (temporary) cases. When an episode end occurs, the *Revision* process updates the *trust value* of the cases retrieved, according to the episode result (in case of success it increments the *trust value*, otherwise, it decrements it). Finally, if a successful end of episode occurs, the *Retention* process inserts the new (temp) cases in the case base and updates the retrieved cases in the case base. In order to perform the case-base maintenance, cases with a *trust value* that are less than a threshold are removed from the case base.

A new episode starts and the cycle repeats. The more cases stored in the case base, the less *Problem Solver* execution is necessary.

The next section presents tests and results evaluating the performance of QCBRL, comparing it to a traditional RL method.

5. Tests and results

This section presents the tests performed with QCBRL, comparing them with the results from traditional RL methods. The tests aim to show that QCBRL can act as a generic and scalable problem solving method that can be applied to spatial problems in which the object positions can be represented as qualitative relations with respect to an elevated point. The results obtained in the simulated Half Field Offense domain and with real humanoid robots are described in Section 5.1. Section 5.2 presents the results of running QCBRL in gridworld domains. Finally, a more global discussion of the tests performed is presented in Section 5.3.

5.1. Tests in the robot-soccer environment

5.1.1. Initial considerations

Tests were performed on simulated and real robot-soccer environments using an Intel NUC i5 with 8GB SDRAM, running Ubuntu 14.04. In both, real and simulated, domains two high-level actions were considered: $dribble()$ and $shoot()$, as well as four high-level states: $dist(Ag, G)$ and $ang(Ag, G)$ representing, respectively, the distance and the angle of the agent (Ag) relative to the goal centre (G); $ang(Ag, OG)$ represents the largest open goal (OG) angle and $dist(Ag, Op)$, the agent distance to its nearest opponent (Op).

Based on Moratz and Wallgrün [42], the qualitative distance regions with respect to an observer's viewpoint were defined in the following way: *at* refers to an object placed closer than 0.33 meters, *very close* refers to an object placed between 0.33 and 0.66 meters, *close* represents an object placed between 0.66 and 1.00 meter, *far* refers to an object placed between 1.00 and 1.50 meters, *very far* is related to an object placed between 1.00 and 3.00 meters, and *farthest* refers to an object more than 3.00 meters distant from the observer.

Qualitative orientation regions were defined with respect to the robot's front, and they are the following: *front* refers to an object placed between $]-30, 30[$ degrees, *left-front* refers to an object placed between $[30, 60]$ degrees, *left* refers to an object placed between $]60, 120[$ degrees, *left-back* refers to an object placed between $[120, 150]$ degrees, *back* refers to an object placed between $]150, 210[$ degrees, *right-back* refers to an object placed between $[210, 240]$ degrees, *right* refers to an object placed between $]240, 300[$ degrees and *right-front* refers to an object placed between $[300, 330]$ degrees. The *left-front*, *left-back*, *right-back* and *right-front* regions were considered transitory regions so they have smaller angular extents than *left*, *back*, *right* and *front* regions.

Despite the fact that the agent in the RoboCup Soccer Server Simulator is represented in the 2D plane, this work assumes that the agent observes the environment in a similar way to that of our humanoid robots. That is, the simulator uses the same parameters that are used by a humanoid robot, such as the robot's height to model distance relations in $\mathcal{EOPRA}$. Besides, the spatial relations defined in this domain are based on the robot having a local vision, located at its front (thus, not assuming a global viewpoint).

5.1.2. Tests in the simulated robot-soccer environment

The tests in the simulated environment were performed on the Half Field Offense (HFO) task [61] using an adaptation of the original code described in Hausknecht et al. [25]. HFO is a subtask of RoboCup 2D Soccer Simulator [52] in which the soccer match is performed on a half field.

HFO allows the implementation of a team of offensive players, where each player is capable of moving, kicking, dribbling or passing the ball, in order to score a goal against the team of defensive players. HFO also allows the implementation of a team of defensive players whose goal is to intercept the ball, avoiding the attack.

Inspired by the work proposed in Hausknecht et al. [25], instead of executing a full SARSA(λ) algorithm (iterating episodes until the Q-table converges to the optimal policy) a partial SARSA(λ) was used in this work. In this way, SARSA(λ) is set to iterate until the first successful episode occurs, assuming that there is no-guarantee of optimality.

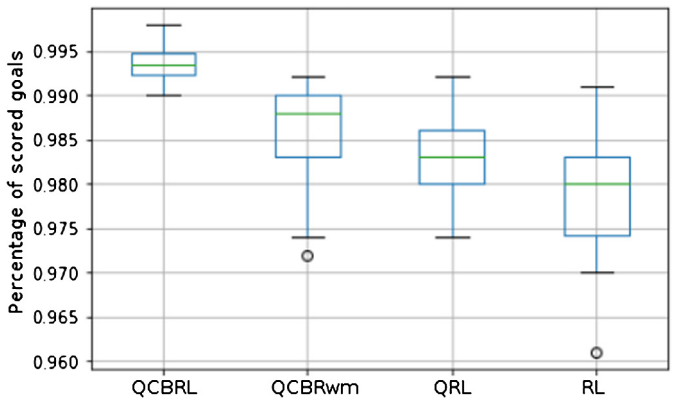


Fig. 6. HFO 1v0: average percentage of scored goals.

It is important to clarify that the SARSA(λ) algorithm applied in the HFO problem in Hausknecht et al. [25] follows the episodic RL proposed for the Keepaway domain [63]. RL routines were placed at the beginning of an episode, at each episode step and at the end of the episode. This is justified in Stone et al. [63] since the RoboCup Simulator has the control over the state perceptions and the action choices that are presented to the agent, which is a distinct process to what traditionally occurs in RL problems, in which the agent has the action control and requests the next state and reward from the environment.

The QCBRL algorithm was executed with the following parameters: exploration/ exploitation $\epsilon = 0.10$; discount factor $\gamma = 0.95$; learning rate $\alpha = 0.10$; eligibility trace $\lambda = 0.9375$; case similarity *threshold* $= 10\%$; and case removal of 0.7 (acceptable minimum *trust value*). The reward function was defined as: a reward of $+100$ at the end of a successful episode; -500 at the end of any unsuccessful episode; and -0.1 for each action the agent performs during the episode.

The tests were performed in three distinct scenarios: HFO 1v0, in which one agent attempts to score in an undefended goal; HFO 1v1, in which one agent has to score against a single goalkeeper; and HFO 1v2, in which one agent attempts to score against one defender and one goalkeeper. The results obtained by QCBRL were compared to the traditional SARSA algorithm defined in Hausknecht et al. [25] (RL), to the qualitative RL proposed in Homem et al. [29] (QRL) and to the Qualitative Case-Based Reasoning system without case-base maintenance (QCBRwm) described in Homem et al. [30]. In this work, we have used only two-high level actions (*drible* and *shoot*): the robot moves to get the ball possession, so it can drible or kick the ball. We have used the implementation of Hausknecht et al. [25] for single agent only, where only these two actions are employed.

In order to ensure that QCBRL running a partial RL does not result in a random selection of actions and to investigate if a two-step strategy, composed of a complete RL method and the QCBRL, outperforms our proposal, we performed three other tests: (1) run $1,000$ QRL episodes, obtaining an optimal Q-table, then run $1,000$ QCBRL episodes, where the learned cases came from the optimal Q-table (this is the two-step strategy and this test was named QRL+QCBRL); (2) run $1,000$ QCBRL episodes, with the *Problem Solver* proposing random actions (named Random QCBRL); and (3) run $1,000$ episodes with Random Actions. The results are presented as the average of 30 independent trials of $1,000$ episodes for each scenario and for each algorithm tested. The boxplot [40] used to represent these results is composed of a box, two whiskers and possible outliers. The box show (from bottom up) the first quartile, the median values and the third quartile. The whiskers show the minimum value within $1.5 \cdot IQR$ of the lower quartile, and the maximum value within $1.5 \cdot IQR$ of the upper quartile, where IQR is the distance between upper and lower quartiles. The empty circles represent possible outliers.

Fig. 6 shows the average percentage of scored goals with respect to each approach executed on HFO 1v0. As we can see, all the qualitative methods outperformed the traditional algorithms, scoring more goals. Table 1 presents the results obtained by each method.

QCBRL scored an average percentage of $0.994 \pm 3.697e^{-06}$ goals, QCBRwm scored an average of $0.986 \pm 3.429e^{-05}$ goals, QRL scored an average of $0.983 \pm 2.727e^{-05}$ goals and RL scored an average of $0.979 \pm 4.365e^{-05}$ goals. The analysis of variance test (ANOVA) was applied and the results obtained show that the QCBRL is statistically better than the other methods, with a confidence level of 99%.

QCBRL stored an average of 25.63 cases and QCBRwm stored an average of 17.63 cases. By performing the case-base maintenance, QCBRL performs more iterations of the RL algorithm than QCBRwm. This increases the average number of goals performed and the average number of stored cases. QCBRL has also reduced the average number of timesteps to score a goal, obtaining an average of 84.21 timesteps.

Fig. 7 presents the average percentage of scored goals comparing distinct methods on the *Problem Solver* process: QCBRL, QRL+QCBRL, random action selection with QCBRL (Random QCBRL) and a pure random action selection method (Random Actions).

Fig. 8 shows the average percentage of scored goals on the HFO 1v1 domain. QCBRL outperforms QCBRwm, QRL and RL algorithms, scoring an average percentage of $0.854 \pm 1.170e^{-04}$ goals, while QCBRwm scored an average of $0.831 \pm 5.367e^{-04}$

Table 1
Average values of each method on HFO 1v0.

1v0	Goals	Out of bounds	Out of time	Goals CBR	Goals RL	Duration of episode
QCBRL	993.60	4.97	1.43	548.23	445.37	84.21
QCBRwm	986.30	10.97	2.73	879.70	106.60	85.67
QRL	982.80	15.80	1.40	-	982.80	83.83
RL	978.93	18.70	2.37	-	978.90	84.68

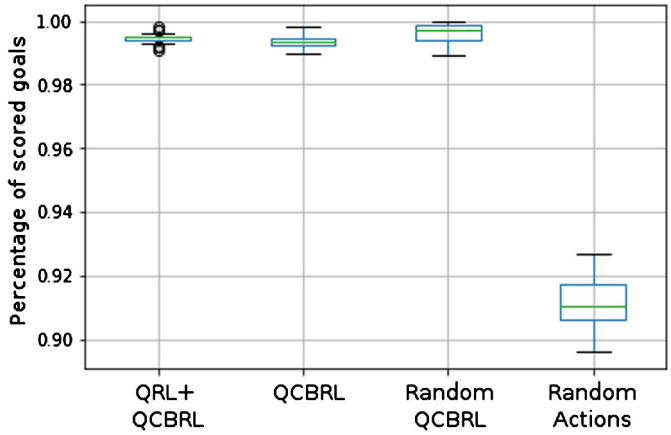


Fig. 7. HFO 1v0: *Problem Solver* methods.

Table 2
Average values of each method on HFO 1v1.

1v1	Goals	Captured	Out of bounds	Out of time	Goals CBR	Goals RL	Duration of episode
QCBRL	853.90	109.37	34.76	1.97	67.33	786.57	94.69
QCBRwm	831.03	130.77	35.40	2.80	815.43	15.60	94.98
QRL	812.40	146.37	39.13	2.10	-	812.40	95.30
RL	782.87	174.13	40.33	2.67	-	782.87	99.06

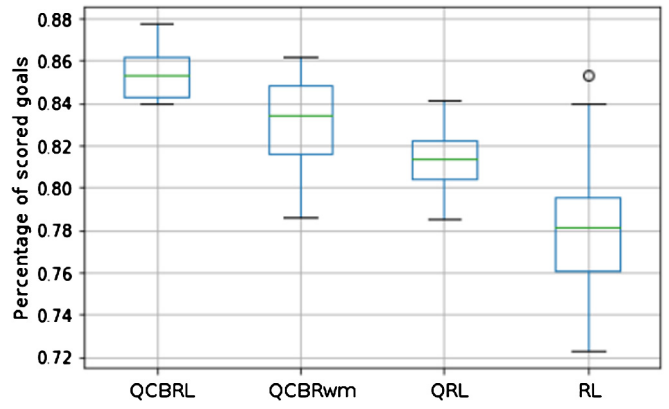


Fig. 8. HFO 1v1: average percentage of scored goals.

goals, QRL scored an average of $0.812 \pm 1.798e^{-04}$ goals and RL scored an average of $0.783 \pm 9.944e^{-04}$ goals. Table 2 presents the results of each method. QBCRL outperforms the other methods with a confidence level of 99%.

QCBRL stored an average of 50.96 cases and QCBRwm stored an average of 35.03 cases. QCBRL has also reduced the average timesteps to the goal, obtaining an average of 94.69 timesteps.

Fig. 9 presents the average percentage of scored goals comparing distinct *Problem Solver* methods: QCBRL, QRL+QCBRL, Random QCBRL and Random Actions. In contrast to the previous test, in HFO 1v1 the random methods (Random QCBRL and Random Actions) do not present interesting results. QCBRL (running partial RL algorithm) and QRL+QCBRL outperform the other methods. No significant difference can be seen between QCBRL and QRL+QCBRL. These are the first indications that

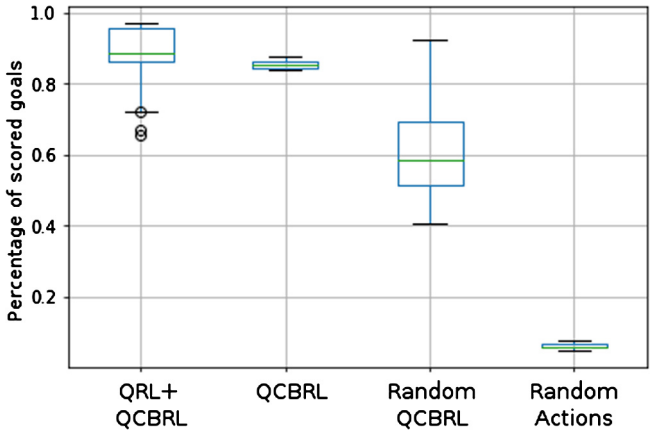


Fig. 9. HFO 1v1: *Problem Solver* methods.

Table 3
Average values obtained by each method on HFO 1v2.

1v2	Goals	Captured	Out of bounds	Out of time	Goals CBR	Goals RL	Duration of episode
QCBRL	523.30	255.63	204.57	16.50	48.03	475.27	114.71
QCBRwm	486.77	303.90	193.87	15.46	470.47	16.30	109.01
QRL	475.63	305.13	203.00	16.24	-	475.63	115.69
RL	406.30	336.77	251.03	5.90	-	406.30	118.71

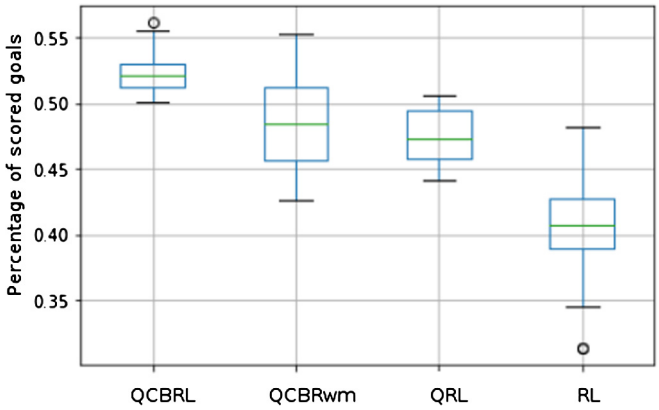


Fig. 10. HFO 1v2: average percentage of scored goals.

running partial RL does not generate a random policy, i.e., even with a non-optimal policy, QCBRL results are as good as performing a QRL+QCBRL.

The results of running HFO 1v2 (Fig. 10 and Table 3) show that QCBRL outperforms QCBRwm, QRL and RL algorithms, scoring an average percentage and standard deviation of, respectively, $0.523 \pm 2.460e^{-04}$ goals, against an average of 0.487 ± 0.001 goals running QCBRwm, an average of $0.476 \pm 3.953e^{-04}$ goals running QRL and an average of 0.406 ± 0.002 goals a running traditional RL algorithm.

ANOVA shows QCBRL is better than the other methods with a confidence level of 99%. QCBRL stored an average of 63.37 cases and QCBRwm stored an average of 39.17 cases. However, QCBRL has obtained an average of 114.71 timesteps to score a goal, while QCBRwm has obtained an average of 109.01 timesteps.

Finally, Fig. 11 presents the average percentage of scored goals comparing distinct methods for the *Problem Solver* process: QCBRL, QRL+QCBRL, Random QCBRL and Random Actions method. Similar to the previous test, in HFO 1v2 the random methods (Random QCBRL and Random Actions) do not result in interesting results; both QCBRL and QRL+QCBRL outperform the other methods. No significant difference was observed between QCBRL and QRL+QCBRL.

As can be seen in the results, the *Problem Solver* process supports the QCBRL system, and QCBRL can be an interesting strategy to learn cases at runtime. This can be useful when no previous learning phase is possible and the agent must learn without prior knowledge, for instance, in real-world domains. When a previous step can be performed, executing QRL and then QCBRL may yield good results.

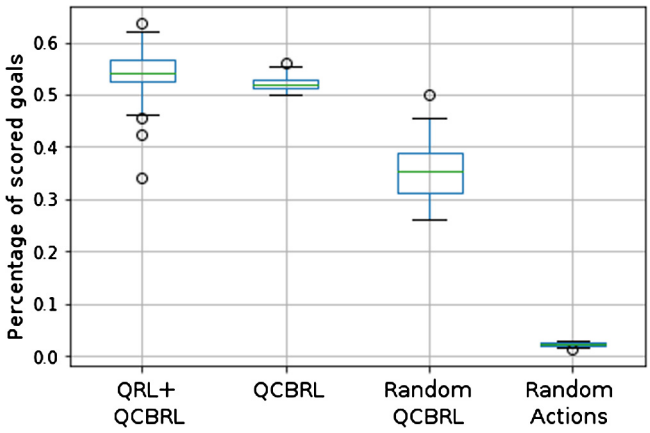


Fig. 11. HFO 1v2: *Problem Solver* methods.

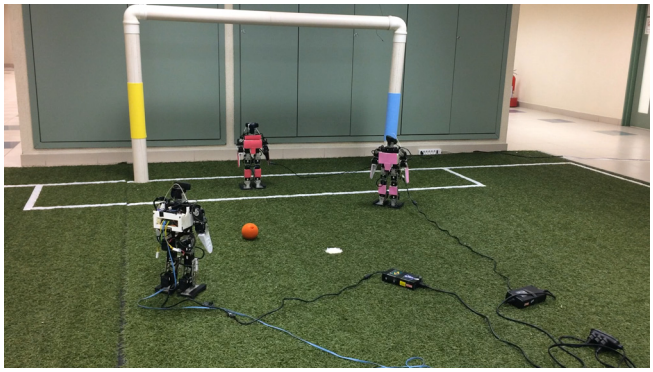


Fig. 12. Humanoid-robot scenario. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

It is worth mentioning that the higher variance observed in the QCBRL results occurs because the episode starts with random states, creating distinct conditions to those that generated the optimal Q-table used, that was obtained before running QCBRL. Besides, the agent running QRL also performs exploration, which also increases the variance.

Next, a case study of running QCBRL on a real humanoid-robot scenario is presented.

5.1.3. Case study with real robots

As the simulation tests presented promising results, QCBRL was tested in a real robot scenario, aiming to evaluate the performance of the proposed method in a real robot without prior knowledge.

The robots used during this test are based on the DARwIn-OP robot [24]. Four identical humanoid robots were used, that have 20 degrees of freedom and their motors were directly controlled by the on-board computer. The software architecture implemented in these robots is the Cross Architecture [45], a hybrid architecture that combines reactive and hierarchical aspects.

The same QCBRL method used in HFO was implemented in the real robots. Thus, instead of QCBRL directly performing an action on RoboCup 2D Soccer Server (as in the simulated environment), the action was performed by the robot. A referee software was also developed to control the end of an episode. It also indicates the success or failure, and, in this case, what type of failure occurred in the episode.

The HFO 1v2 challenge was replicated with the real robots in the scenario shown in Fig. 12. The white robot (on the left) is the offensive agent, the magenta robot is the defender and the red robot (in the goal) is the goalkeeper.

The states and actions used in HFO are the same in all (simulated and real) tests. The goalposts' colours help the robot to perceive the positions of each object in the field and to create or reuse the cases.

In this test, the robot learns without assuming prior knowledge. Fig. 13 illustrates the sequence of actions performed by the robot running QCBRL with partial SARSA(λ). The white robot starts with an empty case base and an arbitrary Q table. The partial RL method is executed in the first episodes. When the referee software verifies that the agent scored a goal, the cases are stored in the case base. Finally, the white robot is relocated and a new episode begins. After a successful episode, QCBRL retrieves and reuses a similar case, instead of executing the partial RL method.

Five trials of 10 episodes were run and the white robot scored an average of 4.4 ± 1.14 goals. In an average of 2.8 ± 1.3 episodes the ball was kicked out of bounds and an average of 2.8 ± 1.3 were considered as *near misses*, that is, situations

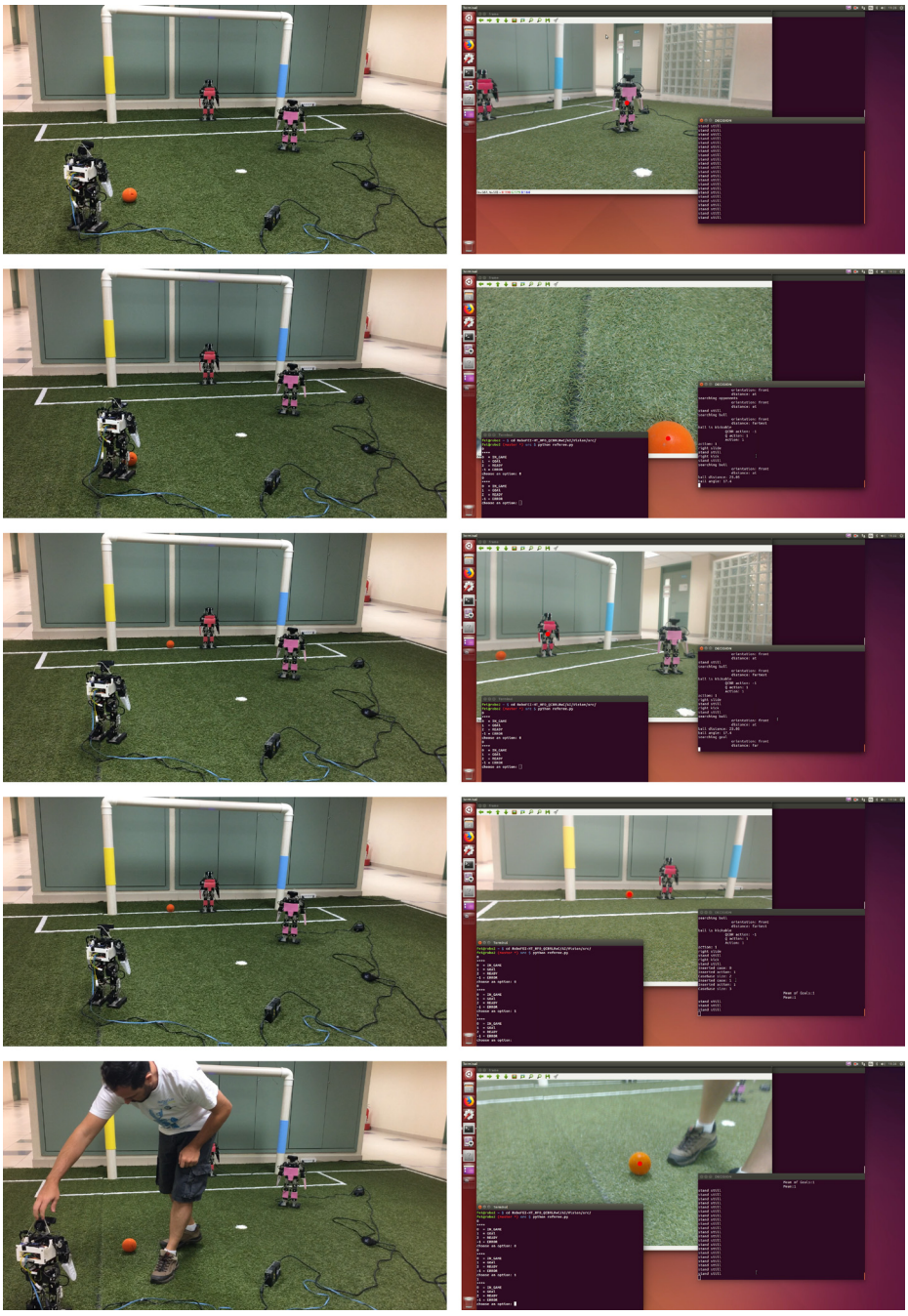


Fig. 13. Case study of QCBRL with humanoid robots.

where the robot kicked the ball in the direction of the goal, but it passed near the goalpost, or stopped before crossing the goal line. QCBRL stored an average of 20.8 cases. The goals scored in this case study are different from the tests performed in simulation environment, since the robots (simulated and real) implement different control and behaviour strategies. The number of goals scored and the number of stored cases could also become higher by improving some aspects of the humanoid-robot, such as the control of walk or kick, or the vision process. However, the qualitative relations of the learned cases on the QCBRL are similar to those created on the simulation (HFO 1v2). Finally, the robot was able to learn, retrieve and reuse the cases in order to score a goal.

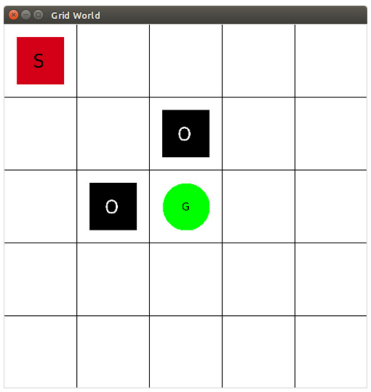


Fig. 14. Gridworld 5×5 scenario.

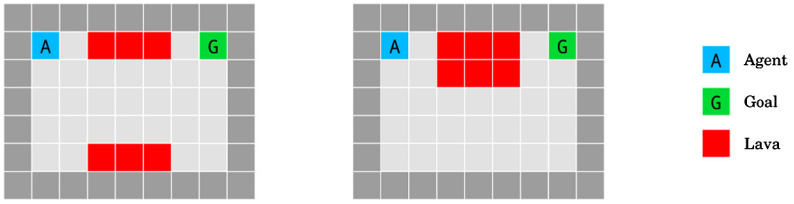


Fig. 15. The Lava world environment [37].

5.2. Tests in gridworld domains

The tests in this section aim at showing that QCBRL can be applied to different domains, such as the standard gridworld and the safety gridworld domains.

5.2.1. Initial considerations

The standard gridworld environment is a common testbed for AI methods, where the cells of the grid correspond to the states of the environment. For each cell, there are four possible actions: move north, south, east and west. The performed action takes the agent to the nearest cell in the grid, in the direction relative to the action [64]. Gridworld is an static environment and, when running RL algorithms, the agent learns the optimal policy, which makes it an important environment to test and compare AI (specially RL) methods. Fig. 14 presents the gridworld 5×5 scenario, where S indicates the source, G indicates the goal and O indicates the obstacles.[2]

Reusing a policy learned in a simulation in real environments may lead to failures under minor changes on the environment. The lava world domain [37] is a safety gridworld that aims to allow the investigation of the agent's adaptability when the testing conditions are distinct from the training conditions. In the lava world, the agent has to get to the goal state without stepping into lava cells, which results in a negative reward and the end of the episode. However, the location and distribution of lava cells differ from training to test sets. For instance, the agent may learn the optimal policy in the training environment shown in Fig. 15 (left), but it is tested on randomly generated test environments (such as that shown in Fig. 15 (right)).

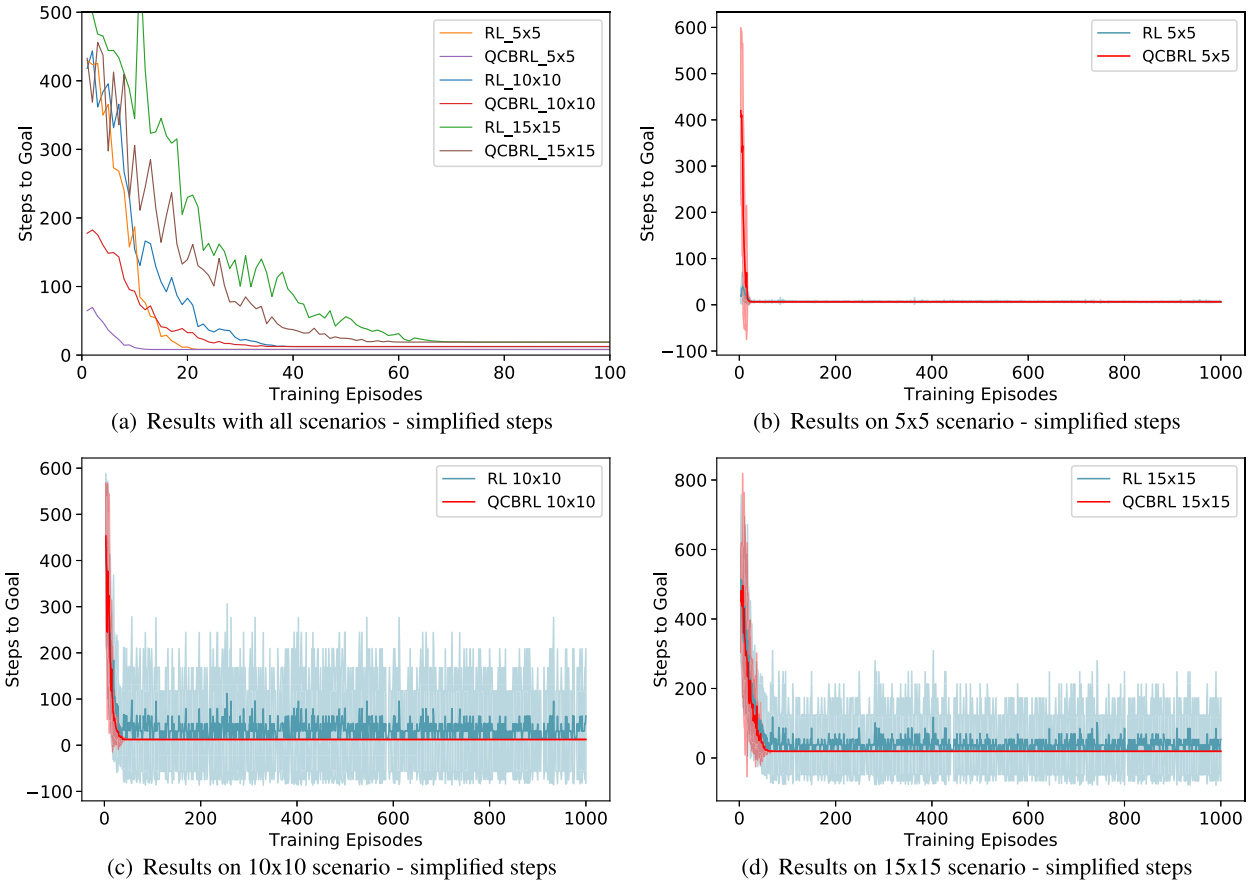
5.2.2. Standard gridworld

Similar to our previous tests, instead of executing a full RL algorithm (iterating episodes until the Q-table converges to the optimal policy), a partial RL was used in these tests, but this time, using the original Q-learning code of Sutton and Barto [64]. Q-learning algorithm is set to iterate until the first successful episode occurs, when QCBRL creates a case with the states and actions performed by the agent.

Q-learning and QCBRL algorithms were executed with the following parameters: exploration/exploitation $\epsilon = 0.10$; discount factor $\gamma = 0.9$; learning rate $\alpha = 0.10$; case similarity *threshold* = 10%; and case removal of 0.7 (acceptable minimum *trust value*). The reward function was defined as: a reward of $+100$ at the end of a successful episode; -100 at the end of an unsuccessful episode; and 0 for each action the agent performs during the episode.

The tests in the standard gridworld domain were performed on three cases: gridworld 5×5, gridworld 10×10 and gridworld 15×15. The results obtained by QCBRL were compared to Q-learning. Each experiment was repeated 30 times

[2] Adapted from https://github.com/rlcode/reinforcement-learning.



(a) Results with all scenarios - simplified steps

(b) Results on 5x5 scenario - simplified steps

(c) Results on 10x10 scenario - simplified steps

(d) Results on 15x15 scenario - simplified steps

Fig. 16. Results of running Q-learning and QCBRL in the gridworld environment. The lines mark the steps to goal averaged over 1,000 episodes of 30 trials and the filled areas correspond to the standard deviation.

and 1,000 iterations per episode. Aiming to validate QCBRL performance, at the end of each iteration the agent performed a greedy policy without exploration and learning; the number of steps to the goal at the iteration was considered.

Fig. 16 represents the average steps to achieve the goal state with respect to each gridworld scenario. These results show that QCBRL outperforms Q-learning in all scenarios tested (Fig. 16(a)). Considering the gridworld 5×5, QCBRL converges earlier than Q-learning, near to the 15th episodes (Fig. 16(c)). In the gridworld 10×10, the algorithms converged after the 35th episode (Fig. 16(e)); and, in the gridworld 15×15 QCBRL converged after to the 60th episode, while Q-learning converged after the 70th episode, as shown in Fig. 16(g).

The ANOVA test was applied and the results obtained show that the QCBRL is statistically better than Q-learning, with a confidence level of 99% to 5×5, 95% to 10×10 and 90% to 15×15 to gridworld scenarios.

5.2.3. Safety gridworld - the lava world environment

Similar to our previous experiments, Q-learning and QCBRL algorithms were executed with the following parameters: exploration/exploitation $\epsilon = 0.10$; discount factor $\gamma = 0.9$; learning rate $\alpha = 0.10$; case similarity *threshold* = 10%; and case removal of 0.7 (acceptable minimum *trust value*). The reward function was defined as: a reward of $+100$ at the end of a successful episode; -100 at the end of an unsuccessful episode; and 0 for each action the agent performs during the episode.

Each experiment was repeated 30 times and 1,000 iterations per episode and at the end of each iteration the agent performed a greedy policy without exploration and learning. Fig. 17 shows the results of QCBRL and Q-learning running on the lava world, considering the number of steps to the goal. Figs. 17(a) and 17(c) show the results of both training and test phases, where Fig. 17(c) plots the average and standard deviation of the number of steps to the goal. Figs. 17(b) and 17(d) show a simplified view and the test phase is considered. Fig. 17(d) plots the average and the standard deviation of the number of steps to the goal.

During the training phase, that corresponds to the first 500 episodes, both Q-learning and QCBRL learned an optimal policy. When running the test phase, where the location of lava cells differ from training phase, QCBRL learned a new safety policy for the test environment configuration, while Q-learning performs misleading policies and let the agent to fall into the lava lake (receiving a punishment of -100 and a value of 500 steps to the goal). The ANOVA test was applied and

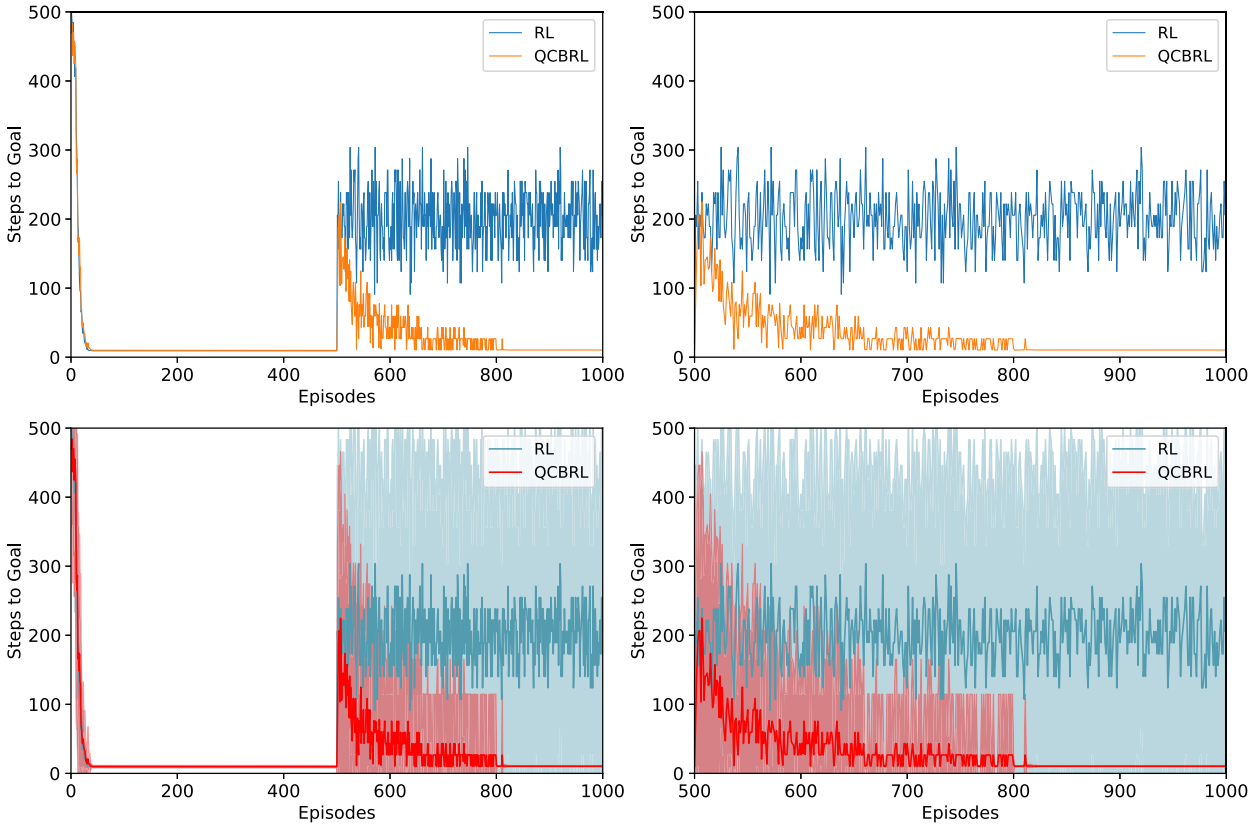


Fig. 17. Results of running Q-learning and QCBRL in the lava world environment. The lines mark the steps to goal averaged over 1,000 episodes of 30 trials and the filled areas correspond to the standard deviation.

the results obtained show that the QCBRL is statistically better than Q-learning with a confidence level of 99% to the lava world scenario. A comparison with the work described in Leike et al. [37], that trained two deep RL algorithms in this environment, will be considered in future research.

5.3. Discussion

The results obtained with the robot-soccer domain show that the QCBRL algorithm outperforms other algorithms tested in terms of the average number of scored goals. The qualitative relations and qualitative case modelling facilitated the similarity measurement and the case retrieval. Furthermore, starting without assuming prior knowledge, new cases were learned by combining a RL method with a CBR theory.

Despite the fact that the *Problem Solver* process proposes a case without guarantee of optimality, QCBRL performs the case-base maintenance and no significant difference can be observed comparing it with the case learned from an optimal Q-table.

More specifically, in the simulated robot soccer domain, QCBRL outperforms recent results of CBR methods without case-based maintenance as well as the results obtained by the traditional SARSA method. The results comparing distinct methods with respect to the *Problem Solver* process show that the cases learned by the partial SARSA(λ) are good solutions, although they can be non-optimal.

In the real robot environment, the agent was able to learn, retrieve and reuse cases in order to score a goal. The qualitative world discretisation and the goalposts' marks allowed the creation of qualitative states similar to the simulation environment. By improving some aspects of vision and localisation of the humanoid-robots, the cases learned in the simulation can be reused in real robots.

In the gridworlds considered, QCBRL outperformed Q-learning algorithm. The agent was able to learn optimal and safety policies, and reuse the cases in order to perform fewer steps to the goal (w.r.t. Q-learning) or to perform a safety path to the goal.

QCBRL is a powerful modelling tool whose performance was verified to be superior to numerical CBR and RL methods. When running QCBRL in a domain where the agent must learn at runtime, i.e., without performing a previous learning step, the agent learns faster than most other approaches since it stores the first successful case (sequence of states/actions) for

that state, that is reused when similar cases occur. In robot soccer, this strategy improves the average number of scored goals and, in the gridworld domain, it reduces the average steps for the agent to reach the goal.

6. Conclusion

This work introduced and analysed the algorithm called QCBRL, a case-based reasoning method assuming a qualitative spatial representation of the domain, implementing the complete CBR cycle. QCBRL have proved to be a generic (scalable) modelling method for solving problems in domains where the positions of the objects can be represented as qualitatite relations with respect to an elevated observer. The main contributions of this work are: the combination of CBR with RL, that allowed the agent to learn new qualitative cases at runtime; the execution of a partial RL method with a case-based maintenance; the use of a QSR method to model cases; and the implementation of a retrieval algorithm that measures qualitative similarity of cases.

The QCBRL cycle was implemented and evaluated in simulated and real robot scenarios, as well as in two gridworld domains. The agent running QCBRL was able to learn new cases, retrieve and reuse past cases as a solution to the new problem, and to perform the case-base maintenance, forgetting poor cases. Results show that the case-base maintenance process improved the performance of the agent, outperforming recent results in the literature. In particular, it is worth observing that the methods using a qualitative representation of the environment outperformed traditional numerical methods, whereas QCBRL obtained the best results overall. Moreover, the *Problem Solver* process running partial RL has obtained results that show a similar performance to that obtained when running a learning phase before applying QCBRL.

Future work shall consider the implementation of a Deep Q-Network method, integrating it with QCBRL, in order to transfer and reuse the cases learned in a simulator to real robots. Another important issue to consider in a future work is the Q-table update in each step of the *Reuse* process, in order to take advantage of the state/action pair to accelerate the RL method.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

Paulo Eduardo Santos acknowledges support from FAPESP-IBM Proc. 2016/18792-9. Ramon Lopez de Mantaras has been partially supported by the Generalitat de Catalunya Grant number 2017 SGR 172, by the CSIC Grant PIE 201750E090, and by the European Project “HumanE-AI-Net” H2020-ICT-48-2020-RIA-952026. Anna Helena Reali Costa acknowledges support from CNPq Proc. 425860/2016-7 and 307027/2017-1. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil (CAPES) - Finance Code 001.

References

[1] A. Aamodt, E. Plaza, Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Commun. 7 (1) (1994) 39–59.
[2] A. Agostini, C. Torras, F. Wörgötter, Efficient interactive decision-making framework for robotic applications, Artif. Intell. 247 (2017) 187–212.
[3] S.V. Albrecht, P. Stone, Autonomous agents modelling other agents: a comprehensive survey and open problems, Artif. Intell. 258 (2018) 66–95.
[4] K.D. Asis, J.F. Hernandez-garcia, G.Z. Holland, R.S. Sutton, Multi-step reinforcement learning: a unifying algorithm, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), 2018, pp. 2902–2909.
[5] B. Auslander, S. Lee-Urban, C. Hogg, H. Muñoz-Avila, Recognizing the enemy: combining reinforcement learning with strategy selection using case-based reasoning, in: Advances in Case-Based Reasoning, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 59–73.
[6] S. Barrett, P. Stone, Cooperating with unknown teammates in complex domains: a robot soccer case study of ad hoc teamwork, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015, pp. 2010–2016.
[7] A.K. Baughman, W. Chuang, K.R. Dixon, Z. Benz, J. Basilico, Deepqa jeopardy! Gamification: a machine-learning perspective, IEEE Trans. Comput. Intell. AI Games 6 (1) (2014) 55–66.
[8] R.A. Bianchi, L.A. Celiberto, P.E. Santos, J.P. Matsuura, R. Lopez de Mantaras, Transferring knowledge as heuristics in reinforcement learning, Artif. Intell. 226 (C) (2015) 102–121.
[9] R.A.C. Bianchi, R. Ros, R. Lopez de Mantaras, Improving reinforcement learning by using case based heuristics, in: L. McGinty, D.C. Wilson (Eds.), Case-Based Reasoning Research and Development, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 75–89.
[10] R.A.C. Bianchi, P.E. Santos, I.J. da Silva, L.A. Celiberto, R. Lopez de Mantaras, Heuristically accelerated reinforcement learning by means of case-based reasoning and transfer learning, J. Intell. Robot. Syst. 91 (2) (2018) 301–312.
[11] H. Burkhard, Case completion and similarity in case-based reasoning, Comput. Sci. Inf. Syst. 1 (2) (2004) 27–55.
[12] L.A. Celiberto, R.A.C. Bianchi, P.E. Santos, Transfer learning heuristically accelerated algorithm: a case study with real robots, in: 2016 XIII Latin-American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2017, pp. 311–316.
[13] A.G. Cohn, J. Renz, Chapter 13 qualitative spatial representation and reasoning, in: F. van Harmelen, V. Lifschitz, B. Porter (Eds.), Handbook of Knowledge Representation, in: Foundations of Artificial Intelligence, vol. 3, Elsevier, 2008, pp. 551–596.
[14] T.H. Cormen, C. Stein, R.L. Rivest, C.E. Leiserson, Introduction to Algorithms, 2nd ed., McGraw-Hill Higher Education, 2001.
[15] C.H. Dorr, L.J. Latecki, R. Moratz, Shape similarity based on the qualitative spatial reasoning calculus eopram, in: Spatial Information Theory - 12th International Conference, COSIT 2015, Santa Fe, NM, USA, October 12-16, 2015, Proceedings, 2015, pp. 130–150.
[16] J.A. Fabro, L. Reis, N. Lau, Using reinforcement learning techniques to select the best action in setplays with multiple possibilities in robocup soccer simulation teams, in: 2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol, 2014, pp. 85–90.

[17] D.A. Ferrucci, E.W. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J.W. Murdock, E. Nyberg, J.M. Prager, N. Schlaefer, C.A. Welty, Building Watson: an overview of the deepqa project, AI Mag. 31 (3) (2010) 59–79.
[18] M.W. Floyd, B. Esfandiari, A case-based reasoning framework for developing agents using learning by observation, in: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, 2011, pp. 531–538.
[19] M.W. Floyd, B. Esfandiari, K. Lam, A case-based reasoning approach to imitating robocup players, in: Proceedings of the Twenty-First International FLAIRS Conference, 2008, pp. 251–256.
[20] A. Formica, M. Mazzei, E. Pourabbas, M. Rafanelli, Approximate answering of queries involving polyline–polyline topological relationships, Inf. Vis. 17 (2) (2018) 128–145.
[21] A.U. Frank, Qualitative spatial reasoning with cardinal directions, in: H. Kaindl (Ed.), 7. Österreichische Artificial-Intelligence-Tagung / Seventh Austrian Conference on Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 1991, pp. 157–167.
[22] V. Freire, A.H.R. Costa, Comparative analysis of abstract policies to transfer learning in robotics navigation, in: Knowledge, Skill, and Behavior Transfer in Autonomous Robots, 2015 AAAI Workshop, AAAI Press, Austin, Texas, USA, 2015, pp. 9–15.
[23] C. Freksa, Temporal reasoning based on semi-intervals, Artif. Intell. 54 (1–2) (1992) 199–227.
[24] I. Ha, Y. Tamura, H. Asama, J. Han, D.W. Hong, Development of open humanoid platform DARwIn-op, in: SICE Annual Conference 2011, 2011, pp. 2178–2181.
[25] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, P. Stone, Half field offense: an environment for multiagent learning and AdHoc teamwork, in: Proceedings of the AAMAS 2016 - Workshop on Adaptive Learning Agents, Singapore, 2016.
[26] M. Hausknecht, P. Stone, Deep reinforcement learning in parameterized action space, in: Proceedings of the International Conference on Learning Representations (ICLR), 2016, pp. 1–17.
[27] T.P.D. Homem, D.H. Perico, P.E. Santos, R.A.C. Bianchi, R.L. de Mantaras, Qualitative Case-Based Reasoning for humanoid robot soccer: a new retrieval and reuse algorithm, in: Lecture Notes in Computer Science, vol. 9969, Springer, Cham, Atlanta, Georgia, USA, 2016, pp. 170–185.
[28] T.P.D. Homem, D.H. Perico, P.E. Santos, R.A.C. Bianchi, R.L. de Mantaras, Retrieving and reusing qualitative cases: an application in humanoid-robot soccer, AI Commun. 30 (3–4) (2017) 251–265.
[29] T.P.D. Homem, D.H. Perico, P.E. Santos, A.H.R. Costa, R.A.C. Bianchi, Improving reinforcement learning results with qualitative spatial representation, in: 2017 Brazilian Conference on Intelligent Systems (BRACIS), IEEE, 2017, pp. 151–156.
[30] T.P.D. Homem, D.H. Perico, P.E. Santos, A.H.R. Costa, R.A.C. Bianchi, R.L. de Mantaras, A hybrid approach to learn, retrieve and reuse qualitative cases, in: 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), 2017, pp. 1–6.
[31] S. Kalyanakrishnan, Y. Liu, P. Stone, Half field offense in robocup soccer: a multiagent reinforcement learning case study, in: G. Lakemeyer, E. Sklar, D.G. Sorrenti, T. Takahashi (Eds.), RoboCup 2006: Robot Soccer World Cup X, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 72–85.
[32] J. Kendall-Morwick, D. Leake, A study of two-phase retrieval for process-oriented case-based reasoning, in: Successful Case-based Reasoning Applications-2, Springer Berlin Heidelberg, Berlin, Heidelberg, 2014, pp. 7–27, https://doi.org/10.1007/978-3-642-38736-4_2.
[33] B. Khajotia, D. Sormaz, S. Nesic, Case-based Reasoning model of CO2 corrosion based on field data, in: 2007 NACE Conference Papers (NACE International), Houston, Texas, USA, 2007, pp. 1–14.
[34] M.L. Koga, V. Freire, A.H.R Costa, Stochastic abstract policies: generalizing knowledge to improve reinforcement learning, IEEE Trans. Cybern. 45 (1) (2015) 77–88.
[35] G. Kuhlmann, P. Stone, Progress in learning 3 vs. 2 keepaway, in: D. Polani, B. Browning, A. Bonarini, K. Yoshida (Eds.), RoboCup 2003: Robot Soccer World Cup VII, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 694–702.
[36] B. Kuipers, The spatial semantic hierarchy, Artif. Intell. 119 (1) (2000) 191–233.
[37] J. Leike, M. Martic, V. Krakovna, P.A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, S. Legg, AI safety gridworlds, CoRR 2017 arXiv:1711.09883 [abs].
[38] G. Ligozat, Qualitative Spatial and Temporal Reasoning, John Wiley & Sons, Inc., Hoboken, NJ, USA, 2013.
[39] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, CoRR 2015 arXiv:1509.02971 [abs].
[40] R. McGill, J.W. Tukey, W.A. Larsen, Variations of box plots, Am. Stat. 32 (1) (1978) 12–16, http://www.jstor.org/stable/2683468.
[41] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.
[42] R. Moratz, J.O. Wallgrün, Spatial reasoning with augmented points: extending cardinal directions with local distances, J. Spat. Inf. Sci. 5 (5) (2012) 1–30.
[43] F. Orduña Cabrera, M. Sànchez-Marrè, Environmental data stream mining through a case-based stochastic learning approach, Environ. Model. Softw. 106 (2018) 22–34.
[44] S.K. Pal, S.C.K. Shiu, Foundations of Soft Case-Based Reasoning, John Wiley & Sons, Inc., 2004.
[45] D.H. Perico, I.J. Silva, C.O. Vilão Junior, T.P.D. Homem, R.C. Destro, F. Tonidandel, R.A.C. Bianchi, Newton: a high level control humanoid robot for the robocup soccer kidsize league, in: F.S. Osório, D.F. Wolf, K. Castelo Branco, V. Grassi Jr., M. Becker, R. Romero (Eds.), Robotics, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 53–73.
[46] D. Precup, R.S. Sutton, S. Dasgupta, Off-policy temporal difference learning with function approximation, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 417–424.
[47] K. Ramirez-Amaro, M. Beetz, G. Cheng, Transferring skills to humanoid robots by extracting semantic representations from observations of human activities, Artif. Intell. 247 (August) (2017) 95–118.
[48] D.A. Randell, Z. Cui, A.G. Cohn, A spatial logic based on regions and connection, in: Proceedings 3rd International Conference on Knowledge Representation and Reasoning, Morgan Kaufmann, 1992.
[49] D.A. Randell, M. Witkowski, M. Shanahan, From images to bodies: modelling and exploiting spatial occlusion and motion parallax, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001, 2001, pp. 57–66.
[50] J. Renz, D. Mitra, Qualitative direction calculi with arbitrary granularity, in: C. Zhang, H.W. Guesgen, W.K. Yeap (Eds.), PRICAI 2004: Trends in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 65–74.
[51] M.M. Richter, R.O. Weber, Case-Based Reasoning: A Textbook, Springer Publishing Company, Incorporated, 2013.
[52] RoboCup, The RoboCup Soccer Simulator, https://sourceforge.net/projects/sserver/, 2018.
[53] E. Rodrigues, P.E. Santos, M. Lopes, Pinning down polysemy: a formalisation for a Brazilian Portuguese preposition, Cogn. Syst. Res. 41 (C) (2017) 84–92.
[54] R. Ros, J.L. Arcos, R.L. de Mántaras, M.M. Veloso, A case-based approach for coordinated action selection in robot soccer, Artif. Intell. 173 (9–10) (2009) 1014–1039.
[55] G.A. Rummery, M. Niranjan, On-Line Q-Learning Using Connectionist Systems, Technical Report, Cambridge University Engineering Department, 1994.
[56] S.J. Russell, P. Norvig, Artificial Intelligence - A Modern Approach, 3. internat. ed., Pearson Education, 2010.
[57] P.E. Santos, M.F. Martins, V. Fenelon, F.G. Cozman, H.M. Dee, Probabilistic self-localisation on a qualitative map based on occlusions, J. Exp. Theor. Artif. Intell. 28 (5) (2016) 781–799.
[58] S. Schiffer, A. Ferrein, G. Lakemeyer, Reasoning with qualitative positional information for domestic domains in the situation calculus, J. Intell. Robot. Syst. 66 (1–2) (2012) 273–300, https://doi.org/10.1007/s10846-011-9606-0.

[59] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, D. Hassabis, Mastering the game of go without human knowledge, Nature 550 (2017) 354.
[60] S.P. Singh, R.S. Sutton, Reinforcement learning with replacing eligibility traces, Mach. Learn. 22 (1–3) (1996) 123–158.
[61] T. Srinivasan, K. Aarthi, S.A. Meenakshi, M. Kausalya, CBRRoboSoc: an efficient planning strategy for robotic soccer using case based reasoning, in: 2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06), IEEE, 2006, pp. 113–118.
[62] P. Stone, G. Kuhlmann, M.E. Taylor, Y. Liu, Keepaway soccer: from machine learning testbed to benchmark, in: A. Bredenfeld, A. Jacoff, I. Noda, Y. Takahashi (Eds.), RoboCup 2005: Robot Soccer World Cup IX, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 93–105.
[63] P. Stone, R.S. Sutton, G. Kuhlmann, Reinforcement learning for robocup soccer keepaway, Adapt. Behav. 13 (3) (2005) 165–188.
[64] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, 2nd ed., The MIT Press, Cambridge, MA, 2018.
[65] M. Tenorth, M. Beetz, Representations for robot knowledge in the KnowRob framework, Artif. Intell. 247 (2017) 151–169.
[66] M. Vasardani, L. Stirling, S. Winter, The preposition at from a spatial language, cognition, and information systems perspective, Semant. Pragmat. 10 (2017) 1–34.
[67] C.J.C.H. Watkins, Learning from Delayed Rewards, Ph.D. thesis King's College, Cambridge, UK, 1989.
[68] C.J.C.H. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (3) (1992) 279–292.
[69] I. Watson, F. Marir, Case-based reasoning: a review, Knowl. Eng. Rev. 9 (4) (1994) 327–354.
[70] D. Wolter, J.O. Wallgrün, Qualitative spatial reasoning for applications: new challenges and the SparQ toolbox, in: Qualitative Spatio-Temporal Representation and Reasoning: Trends and Future Directions, IGI Global, 2012, pp. 336–362.
[71] A. Yan, L. Qian, C. Zhang, Memory and forgetting: an improved dynamic maintenance method for case-based reasoning, Inf. Sci. 287 (Complete) (2014) 50–60.
[72] L. Yang, M. Shi, Q. Zheng, W. Meng, G. Pan, A unified approach for multi-step temporal-difference learning with eligibility traces in reinforcement learning, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), 2018, pp. 2984–2990.
[73] C. Zeyen, G. Müller, R. Bergmann, Conversational retrieval of cooking recipes, in: Proceedings of ICCBR 2017 Workshops, vol. 2028, Trondheim, Norway, 2017, pp. 237–244.