Computer Systems Science & Engineering
DOI: 10.32604/csse.2022.024056
Article

Tech Science Press

QL-CBR Hybrid Approach for Adapting Context-Aware Services

Somia Belaidouni[1,2], Moeiz Miraoui[3,4,*] and Chakib Tadj[1]

[1]Ecole de Technologie Supérieure, Montréal, QCH3C1K3, Canada
[2]Faculty of Exact and Applied Sciences, University of Oran 1, Oran, 31000, Algeria
[3]Higher Institute of Sciences and Technologies, University of Gafsa, Gafsa, 2100, Tunisia
[4]AL-Lith Computer College, Umm Al-Qura University, Al-Lith, 28434, Saudi Arabia
*Corresponding Author: Moeiz Miraoui. Email: mfmiraoui@uqu.edu.sa
Received: 01 October 2021; Accepted: 11 November 2021

Abstract: A context-aware service in a smart environment aims to supply services according to user situational information, which changes dynamically. Most existing context-aware systems provide context-aware services based on supervised algorithms. Reinforcement algorithms are another type of machine-learning algorithm that have been shown to be useful in dynamic environments through trial-and-error interactions. They also have the ability to build excellent self-adaptive systems. In this study, we aim to incorporate reinforcement algorithms (Q-learning) into a context-aware system to provide relevant services based on a user's dynamic context. To accelerate the convergence of reinforcement learning (RL) algorithms and provide the correct services in real situations, we propose a combination of the Q-learning and case-based reasoning (CBR) algorithms. We then analyze how the incorporation of CBR enables Q-learning to become more efficient and adapt to changing environments by continuously producing suitable services. Simulation results demonstrate the effectiveness of the proposed approach compared to the traditional CBR approach.

Keywords: Context-aware service; smart space; auto-adaptation; reinforcement learning; Q-learning; supervised learning; CBR

1 Introduction

There is an increasing demand for adaptive systems that dynamically change their behavior in real-time in response to changes in user preferences and contexts. Such systems come in many different forms, but they generally follow the same process to respond to user needs. This process is called an adaptation loop [1]. This loop begins by capturing and analyzing changes in context, then using an adaptation mechanism to determine the appropriate service. Finally, the system executes the correct service to meet user expectations. Adaptive systems are context-aware systems that *sense* changes in their environments and *respond* by changing their behavior and/or structure appropriately [2]. Such systems are able to adapt their services in response to feedback from the surrounding environment. Context-aware systems are examples of adaptive systems, which must (a) model, process, and manage their environmental information, and (b) adapt according to changes in their environments [3]. The origin of the term "context awareness" is attributed to [4], who

asserted that context sensitivity is the ability to discover and react to changing situations. The main goal of context-aware systems is to provide users with relevant information and/or services based on their current context. Therefore, the notion of context is critical, and several authors have proposed different definitions [5–10]. The most-cited definition describes context as any information that can be used to characterize the situation of an entity (person, object, or physical computer). In general, we can consider context as any information that is relevant to the user and their surroundings. The main difficulty when dealing with context is how to properly adapt services in view of the high dynamicity of contextual values. A change in one contextual feature may generate changes in many other contextual ones. That is why context-aware systems must be adaptable to highly dynamic user situations and context models must be able to capture the dynamic interactions between contextual features.

Several studies on context-aware adaptive systems have addressed the problem of service adaptation in context-aware systems. The majority of previous methods were inspired by supervised learning models, such as Bayesian statistics, neural networks, and naive Bayes' classifiers. Reinforcement algorithms are another type of machine-learning algorithm that have been shown to be useful in dynamic environments through trial-and-error interactions. They also have the ability to build excellent self-adaptive systems [11]. However, this approach has not been widely used in context-aware systems. Our goal in this study is to utilize reinforcement algorithms in a context-aware system to respond to user needs.

Reinforcement learning (RL) represents a class of problems in which an autonomous agent acting in a given environment improves its behavior by progressively maximizing a calculated function based on a succession of scalar responses (rewards), which are received from the environment [12]. In other words, instead of learning from examples provided by an external supervisor, RL is accomplished by directly interacting with an environment. Therefore, incorporating RL into context-aware systems to select and adapt services according to user situations should enable the system to dynamically execute the best services. However, one of the shortcomings of RL is that the convergence of the value function and learning time can be computationally prohibitive for very complex and dynamic systems, such as smart spaces. Applied reinforcement algorithms can classify an entire group of services and specify different situations that can occur for each service, but they are not capable of adjusting or adapting services based on the current context. One method to speed up the convergence of RL algorithms is to utilize previous domain knowledge stored as a case base. This method incorporates case-based reasoning (CBR) techniques into existing reinforcement algorithms to facilitate the modification and selection of optimal services. This study investigates the combination of RL techniques and CBR algorithms to select the best services and adapt them according to contextual changes.

The remainder of this paper is organized as follows. Related work on context-aware reasoning systems is summarized in Section 2. In Section 3, an overview of the background knowledge required for the reinforcement approach is introduced. This is followed by an overview of CBR algorithms in Section 4. In Section 5, our hybrid approach is described in detail. In Section 6, we present a simulation scenario to objectively demonstrate how our approach performs and discuss the obtained results. Finally, conclusions are presented in Section 7.

2 Structure

Several studies have been performed on the adaptation of services in smart spaces and considerable effort has been invested to create and test different reasoning algorithms that are able to adapt and produce suitable services according to changes in an environment. Yuan et al. [13] introduced the Context Aware Real-time Assistant (CARA) system for smart-home environments. The system adopted a context-aware hybrid-reasoning framework by means of case-based reasoning and fuzzy rule-based analogy interpretation of sensor data within a wider context to perform reasoning with all available knowledge for

situation assessment and perform actions based on the results of the reasoning process. Adaptive control of home environments (ACHE) [14] is an adaptive-house model that controls the comfort systems in a home, such as lightning, ventilation and air, and water heating. The objective of ACHE is to use reinforcement algorithms, specifically Q-learning, to predict inhabitant actions and adjust systems to decrease energy consumption. Additionally, authors incorporated neural networks to predict which zone(s) would become occupied in the next timeframe. Wang et al. [15] proposed a novel learning-classifier system based on the co-evolution eXtended classifier system (XCS) to perform context-aware mobile service adaptation. The main concept in their system is to map user contexts onto classifier conditions and mobile services onto classifier actions. The generation of adaptation rules is then transformed to provide mobile service adaptation through the matching and competition of classifiers. Mandato et al. [16] proposed an approach to modeling and implementing context-aware adaptive software systems. They mainly considered explicit user preferences and implicit user circumstances. This context enables users to gain more control over the services they access. They proposed a component model (CAMP) that incorporates explicit support for the definition of system functionality, context, and management. Ni et al. [17] proposed a context-dependent task approach to manage pervasive services. They adopted an implementation of the case-based reasoning (CBR) method to recognize tasks, which facilitated task-oriented system design in smart home environments. Lum et al. [18] proposed a content adaptation system that can determine the optimal content version for presentation and the best strategy for deriving and generating that version. Their system's most crucial component was the decision engine, which utilized decision trees to determine the optimal content for presentation by focusing primarily on user preferences, intended target device capabilities, and network conditions. Miraoui et al. [19] presented a hybrid approach for context-aware service adaptation in a smart living room based on naïve Bayes', fuzzy logic, and CBR. Kabir et al. [20] focused on the use of two effective learning algorithms: the back propagation neural network and temporal differential (TD) class of RL to predict the demand of home users and proactively provide the proper services. Ali et al. [21] analyzed the prerequisites for user-centered prediction of future activities and presented an algorithm for autonomous context aware user activity prediction. They proposed combining the fuzzy-state and Q-learning algorithms to predict or anticipate a future situation in an assistive environment. Hong et al. [22] proposed an agent-based framework for providing personalized services using context history via context-aware computing. They mainly focused on context history to derive preference rules and recommend personalized services.

3 Reinforcement Learning (RL) and the Q-learning Algorithm

RL algorithms have been applied successfully to the online learning of optimal control policies in Markov decision processes (MDPs) [23]. RL approach is based on maximizing the sum of rewards received by an agent when selecting an action to act in an environment by learning from its previous experience. The reward is classically a continuous function between −1 and 1, where 1 corresponds to satisfaction, −1 to disapproval, and 0 to no opinion. Q-learning falls under the class of RL algorithms. It uses the temporal difference method to solve problems by estimating a value function called the Q-value for each state-action pair. For separate cases, the Q-learning algorithm assumes that the state set S and action set A can be divided into discrete values. At a given step t, the agent observes the state (context) $c \in C$ and then chooses a service (action) $s \in S$. After executing the action, the agent receives a reward r that reflects how desirable that action was (in a short-term sense). The state will change to the next state c_{t+1} based on the action s_t. The agent will choose the next action s_{t+1} according to its prior knowledge. This process is illustrated in Fig. 1. The Q-learning principle largely follows a Markov decision process that is defined by C, S, R, and P as:

• C is the set of all possible states or contexts in the environment.

• S is the list of possible actions or services to execute.

• R is the reward function that indicates the opportunity cost to choose action a in situation s.

• P is the transition function modeling the environment. P(c, s, c′) is the probability of being in a situation c′ when applying a service s in a situation c [24].

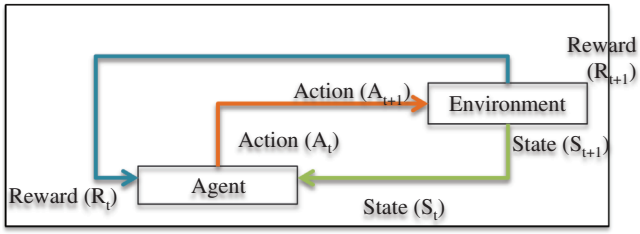


Figure 1: Reinforcement approach

This process forms the Q-matrix. The learning algorithm executes best the possible *action* in a particular *state* to reach the goal state, which is assigned by the agent(*s*). Finally, a Q-matrix is obtained through a finite number of iterations using learning parameters. The maximum value of Q is calculated by considering all possible actions at a particular state.

4 Case Based Reasoning (CBR)

CBR has become one of the most successful applied methods of artificial intelligence (machine learning). The CBR mechanism uses knowledge regarding previous situations (cases) to solve new problems by finding a similar past case and reusing it in a new problem situation. The fundamental concept is the assumption that similar problems have similar solutions, meaning the CBR algorithm retrieves previously solved problems similar to the current problem and attempts to modify the previous solution to fit the current problem. The CBR process relies on three main operations: retrieval, adaptation, and case memorization [25,26]. Adaptation is the heart of the CBR process and is performed by the inference engine. The case-based inference engine of a CBR system solves new problems by retrieving and adapting previous problem-solving experiences [27]. Adapting services in CBR involves obtaining a context description, measuring the similarity of the current context to previous contexts stored in the case base along with their known solutions, retrieving one or more similar cases, and attempting to reuse the solution(s) of the retrieved case(s), possibly after adapting them, as depicted in Fig. 2. Other steps that are typically found in CBR systems are the evaluation of the proposed solution, revision of the solution (if required in light of its evaluation), and retention (learning) of new cases (if the system has learned to solve a new problem). In our study, we used the case definition proposed by [28,29], which is composed of three parts: the context description (C), service description (S), and case scope (P), which are formally described using three-tuples: case (C, S, P). The context description C corresponds to a sensed user situation in which the case can be used. In general, it is a combination of various captured contexts surrounding the user. For example, in a smart environment; a context description may include the user's localization, activity, or time or battery charge level of the user's mobile phone. The service S describes the solution or action that must be executed to respond to user needs. Case retrieval is the most important process in a CBR system and is considered to be the most basic component [30]. It is typically driven by a similarity measure between new context and solved problems in the case base. When a new situation occurs, case retrieval indicates how similar a context (problem) and case are.

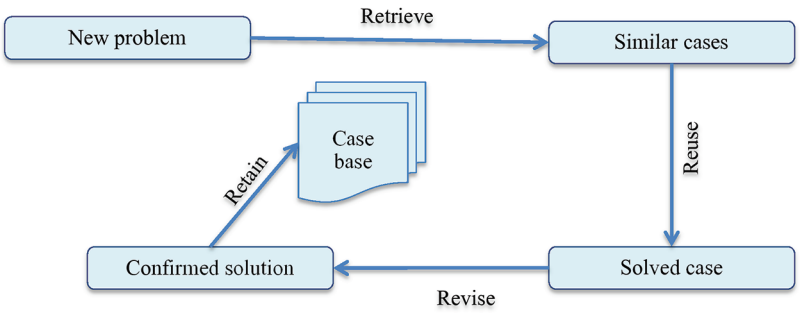


Figure 2: CBR Cycle

In CBR, the global approach to measuring the similarity between cases is primarily based on calculating local similarities between attributes, which can be customized for each case base [30]. In most cases, the similarity function is defined by the distance between the attributes of the new context and those of the contexts in the case base. This function is expressed by Eq. (1):

$$\text{Similarity}\left(\text{context}_{\text{new}}, \text{contex}_{\text{told}}\right) = \sum_{i=1}^{n} w_i \times \text{sim}\left(a_i^{\text{context}_{\text{new}}}, a_i^{\text{contex}_{\text{told}}}\right), \quad (1)$$

where $\text{context}_{\text{new}}$ is the new captured context, $\text{context}_{\text{old}}$ is the context stored in case base, a_i is the attribute of either the new context or old context, and w_i is the weight of attribute a_i. The sum of the similarities of all attributes is calculated to provide a measure of the overall similarity between the old context in the case base and the new context.

5 The Proposed Approach (QL-CBR)

A smart environment is composed of different devices communicating in order to provide proactively adapted services to users (resp. inhabitant). Each device can provide different services and each service has different forms according to changes in the environment. The extent of service changes and their forms depends on many factors, including the user environment and their preferences. The adaptation process consists of adjusting services or their forms depending on the sensed context. In order to provide Q-learning with the capability of adapting services to changes in context, we propose a novel approach called Q-learning case-based reasoning (QL-CBR), which enhances the Q-learning algorithm with the abilities to classify contexts and their appropriate services and adapt services to the current situation. The process of Q-learning adaptation using CBR is illustrated in Fig. 3. This process consists of a Q-learning phase, retrieval phase, and adaptation phase.

5.1 Q-Learning Phase

First, we consider an autonomous agent interacting with a smart environment via perception and action. For each interaction, the agent senses the current context and chooses an action (service) to perform. A reward (r) is then given to the agent to indicate the desirability of the resulting situation. The best services can then be identified through a trial-and-error process. The goal of this state is to find the most suitable service according to the sensed context. Assume that there are K contexts $\{C_1, C_2, \ldots, C_K\}$. Each context C is composed of A_m attributes $\{A_1, A_2, \ldots, A_M\}$ and can perform S_N services $\{S_1, S_2, \ldots, S_N\}$.

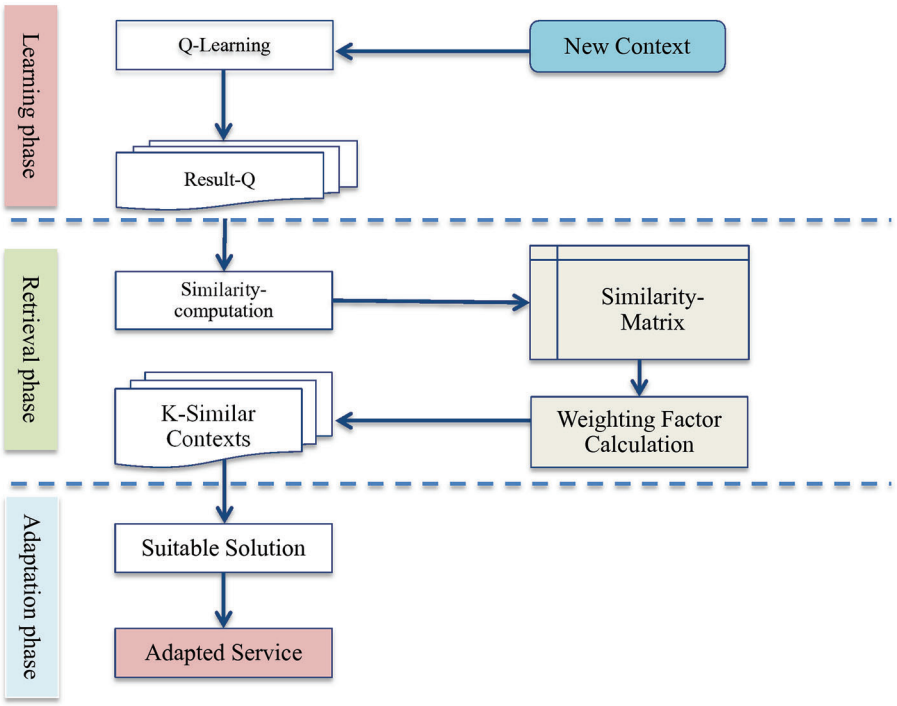


Figure 3: Process of QL-CBR

The rows of the Q-matrix are composed of different attributes that represent various states, as shown in Tab. 1. In other words, each line is a combination of A_k attributes that represent one context that requests the execution of an adequate service. To set this service as a goal, we associate a reward value with each attribute (i.e., link between nodes). The different attributes that lead to the goal have an instant reward of 100. The others have zero reward. At a particular step t, the agent observes the context C_t and then chooses a service S_t. After executing the service, the agent receives a reward r_{t+1}, which reflects how desirable that service is. The context will then change into the next context C_{t+1}. The agent will then choose the next service S_{t+1} according to the best acquired knowledge. The goal of Q-learning is to learn a policy π by learning the service values. The policy π is a rule that the agent follows in selecting actions with the highest value in each state.

Table 1: Composition of the Q-matrix

Contexts	C1	C2	…	C_K	S_1	…	S_N
C1	C_{11}	C_{12}		C_{1M}	..	..	..
C2	C_{21}	C_{22}		C_{2M}	..	..	..
⋮	⋮		⋮	⋮	⋮		⋮
C_K	C_{K1}	C_{K2}	…	C_{KM}	..	…	..

The Q-learning update rule is:

$$Q_{t+1}(c_t, s_t) \leftarrow Q_t(c_t, s_t) + \alpha[r_{t+1} + \gamma \max Q(c_{t+1}, s_t) - Q_t(c_t, s_t)], \quad (2)$$

where γ is the discount factor ($0 \leq \gamma < 1$) and α is the learning rate.

When the Q-matrix approaches a state of convergence, we conclude that our agent has learned the most optimal paths to the goal service. Tracing the best sequences of attributes is as simple as following the links with the highest values at each state. At the end of the learning process, the Result-Q contains all possible contexts with their appropriate services to form the principal rules that are used in the next phase.

5.2 Retrieval Phase

In this stage of the procedure, we use the retrieval phase of the CBR algorithm to retrieve similar cases to a captured new context ($context_{new}$). For this purpose, we use WordNet [31], which is a lexical database based on semantic similarity measures. WordNet organizes an entire word set (synset) into a hypernym tree that can be used for reasoning based on the similarity between two words. The Result-Q must be visited to retrieve all rules with a similar profile to $context_{new.}$ This similarity is determined by calculating the weights between attribute rules and those of $context_{new}$ as follows: Suppose that our captured context is $context_{new}$ = (A_1, A_2, …, A_M) and the first rule in Result-Q is defined by Rule1 = (A r1, A r2, …, A rM). Then, the similarity between them is represented by W (newcontext1, Rule1) = (w1, w2, …, wz), where w_i indicates the weight value of the attributes Ai and A rM. This value corresponds to the degree of similarity between two contexts.

Next, searching for the most suitable result is performed by calculating Euclidian distances, which is repeated for each rule in the Q-Result, to find the K-nearest cases using a distance measure and selecting the class of the majority of these K cases as the appropriate solution.

5.3 Adaptation Phase

The adaptation phase represents the process of transforming the service of the most similar retrieved case from Result-Q into an appropriate solution for the new context. At this stage of the process, we use the minimum distance between the retrieved contexts to resolve and determine the appropriate service. Once an adequate case is selected, if the result coincides completely with the old context, the appropriate service is the same as the result and will be considered as a default service. However, if some attributes only coincide partially, the appropriate service is only the most-similar rule and will be considered as adaptable.

6 Application Scenario and Simulation

For an experiment to analyze the proposed approach, suppose that there is a context-aware scenario for a user named Jack, who is an employee of the Gaz Company and is studying for his master's degree in Biochemistry. He spends his days at work and attends the university in the evening to attend or revise his courses. At night, he returns home. Jack often uses his cellular phone for his work or studies to meet his needs. In this scenario, we consider the cellular phone as the service-providing device. In our work, we focus on incoming call notifications as services that can be changed according to the current context. The adaptation process in the first step consists of collecting the set of data that change the form of the service. This data includes fives attributes that represent the context of the scenario namely: day, time, localization, battery, and activity.

The second step consists of specifying the set of possible values for each context element as follows:

- Day type (weekday, weekend);
- Time (morning, afternoon, evening, night);
- Localization (home, company, restaurant, university class, university library);
- Battery Charge Level (high, low);
- Activity (working, eating, studying, resting, sleeping).

In the default state, the cellular phone notifies Jack of incoming calls by using ringtones. However, in some situations, it can notify Jack of incoming calls using other methods, including silent notification, audio calls, vibration, and video calls. Tab. 2 lists the scenario contexts and services according to context changes.

Table 2: scenario's contextual information and services according to changes

Device	Form of context		Current service	Modified form of context	Adapted-service
Cellular phone	Context-type	Attribute-id	"Audio call" Or "Vibrator" Or "Video call"	Activity = "meeting"	Silent
	Day type	weekday Weekend		Activity = "sleeping" Localization = "universityclass" Charge Level = "low"	
	Time	Morning Afternoon Evening Night	"Audio call" Or "Video call" Or "Silent"	Localization = "car" Localization = "home" Activity = "studying" Localization = "university library"	Vibrator
	Location	Home Company Restaurant University class University library	"Audio call" Or "Silent" Or "Vibrator"	Activity = "eating" Activity = "resting" Day type = "weekend"	Video call
	Battery	High Low	"Silent" Or "Vibrator" Or "Video call"	Day type = "weekday" Localization = "company" Activity = "resting"	Audio call
	Activity	Working Eating Studying Resting Sleeping			

Silent: when the user is sleeping. In our case, we assume that the user is usually sleeping at night.

Audio call: we assume that the user receives audio call during his free time or when he is at home or at university.

Vibrator: we assume that the user can receives calls using vibrator form of his cellular when he is studying either in the university library or at home.

Video-call: we assume that the user can receive incoming calls notification using video call form when he is eating or resting anywhere (home, restaurant). We need also to specify the contradictory rules to eliminate meaningless context configurations that are:

- At week-end, jack has not studies or work;
- In case Jack drives his car, he cannot eat, work or sleep;
- In the morning of the week-day, the localization cannot be university–class or university-library;
- In the night, the localization cannot be the company;
- At university, Jack cannot sleep or work.

The final step consists of using the trained model to choose the most appropriate services according to the current context. For the implementation of the application scenario, we used the Java 1.8 platform. Our goals were to find a good tradeoff between all parameters, obtain the best performance for all schemes, and evaluate the schemes objectively. The experimental settings for these algorithms were set as follows: The reward was r = 100 for each attribute that could reach the target. Otherwise, the reward was zero. The discount factor was $\gamma = 0.8$ and the learning rate was $\alpha = 0.01$.

Fig. 4 illustrates the tree of relationships between attributes and services. We note that user profiles and services under the root begin with the day type attribute and end with services. It should also be noted that the different combinations of attributes form a context that reaches an appropriate service and that a single service can be reached by different contexts.

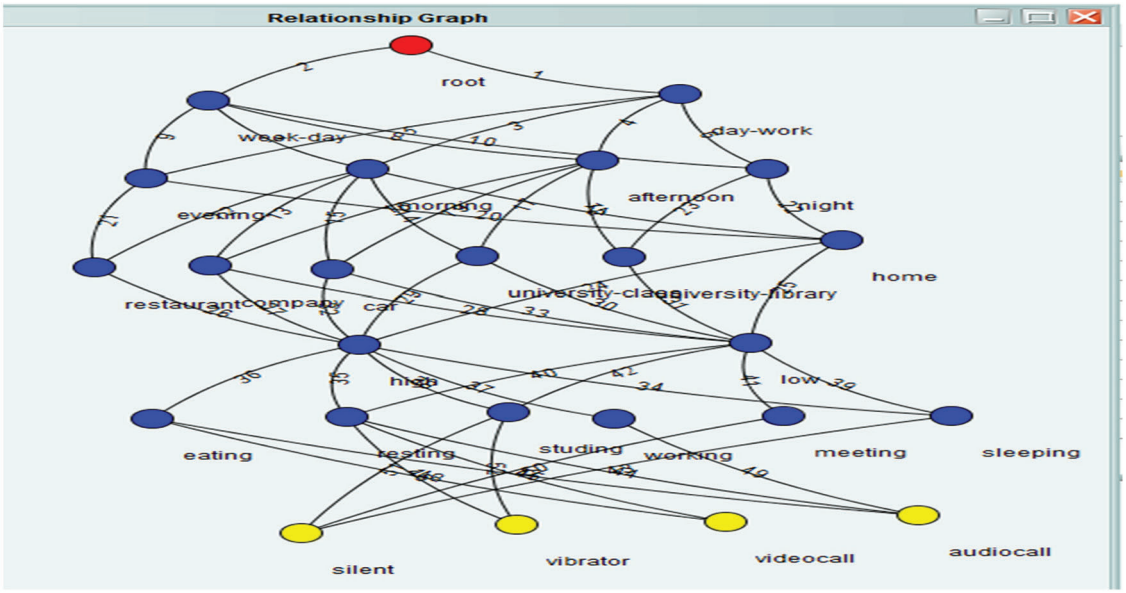


Figure 4: Relationship Graph

Fig. 5 presents the final matrix of the Q-learning algorithm. At this level, the agent can converge to the goal (adequate service) in an optimal manner. It can trace the sequence of appropriate attributes by finding the service that maximizes Q for this state.

```
------------------------------------------------------------------
-------------Matrice Q-Learning Finale -------------------------
Q Matrix values:
380,    380,    433,    433,    476,    476,    371,    371,    317,    313,    311,
371,    314,    312,    313,    371,    309,    364,    391,    369,    384,    380,
380,    380,    476,    333,    476,    476,    371,    371,    317,    312,    314,
371,    304,    295,    300,    376,    304,    364,    393,    384,    384,    387,
380,    380,    333,    333,    376,    333,    471,    371,    317,    313,    311,
371,    314,    312,    312,    376,    304,    364,    392,    353,    364,    387,
380,    380,    376,    333,    333,    376,    371,    417,    417,    311,    296,
471,    314,    312,    313,    364,    304,    364,    389,    369,    371,    375,
380,    380,    376,    333,    376,    376,    471,    371,    317,    313,    314,
471,    304,    312,    300,    376,    309,    364,    393,    384,    384,    387,
380,    346,    376,    376,    376,    376,    471,    471,    417,    313,    311,
471,    304,    312,    313,    376,    300,    364,    394,    369,    376,    375,
380,    380,    376,    333,    376,    376,    317,    371,    317,    313,    311,
371,    314,    311,    312,    376,    300,    464,    393,    353,    384,    387,
380,    380,    333,    333,    376,    333,    317,    371,    317,    313,    311,
371,    314,    312,    313,    371,    304,    464,    394,    375,    371,    387,
380,    380,    376,    333,    376,    376,    371,    317,    317,    313,    314,
371,    304,    311,    313,    397,    304,    364,    391,    384,    371,    387,
380,    380,    333,    333,    376,    376,    371,    371,    317,    313,    314,
371,    304,    312,    313,    376,    300,    364,    392,    375,    371,    387,
380,    380,    376,    333,    376,    376,    371,    371,    317,    313,    314,
371,    304,    312,    313,    376,    304,    364,    394,    384,    380,    387,
380,    380,    376,    333,    376,    376,    371,    371,    317,    313,    311,
371,    304,    312,    313,    371,    309,    464,    394,    375,    384,    380,
380,    380,    376,    333,    376,    376,    371,    371,    317,    289,    314,
371,    304,    312,    313,    376,    304,    364,    393,    380,    384,    375,
```

Figure 5: Final matrix of Q-learning

Fig. 6 presents the Q-learning results. We can deduce that the Q-learning classified all situations with the appropriate services. For example, the silent service was identified for five different contexts.

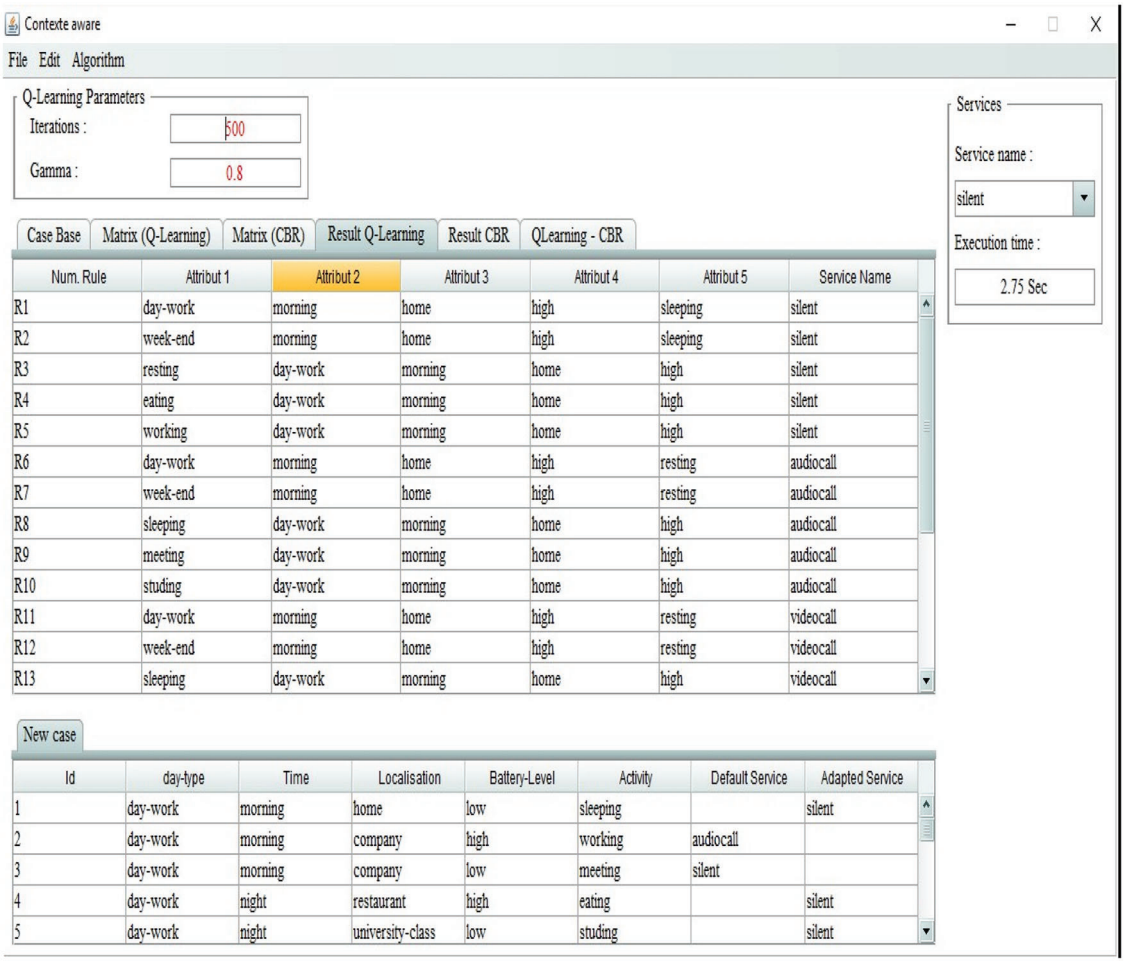
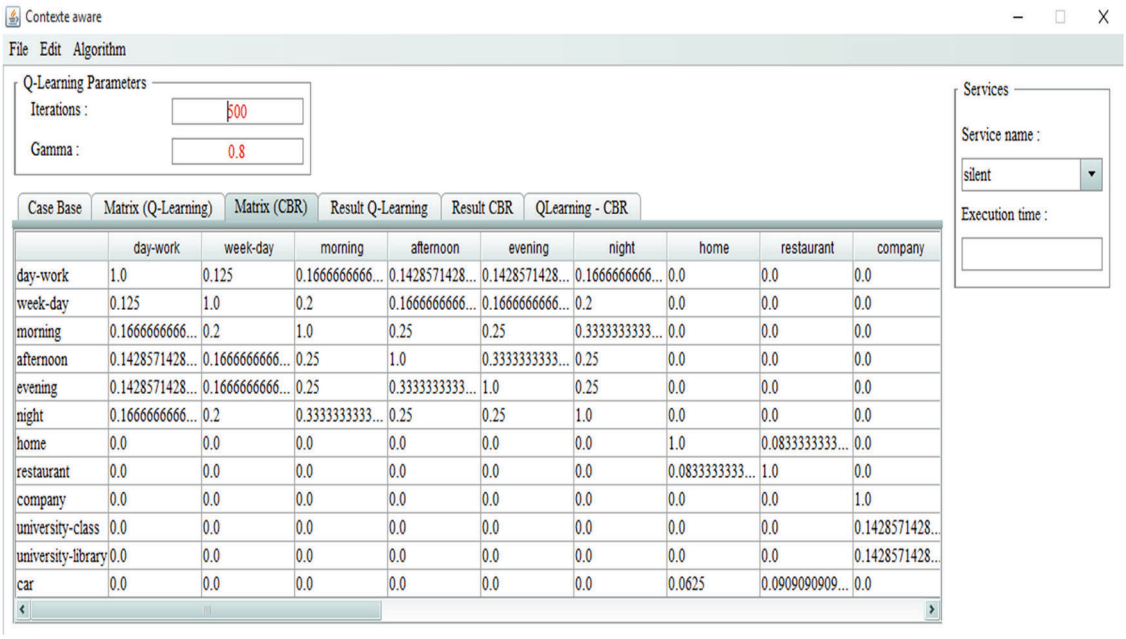


Figure 6: Results of Q-learning

Following the Q-learning process, the CBR algorithm begins by calculating the similarity between the attributes of the new context and the case base using WordNet measures (Fig. 7). For example, let A_1 = « night » and A_2 = « night ». We then obtain *sim* (A_1, A_2) = 1. If A_1 = « week-day » and A_2 = « week-end », then *sim* (A_1, A_2) = 0.125. If A_1 = 'company' and A_2 = « morning », then *sim* (A_1, A_2) = 0. At the end of this stage, we obtain the most-similar situations.



	day-work	week-day	morning	afternoon	evening	night	home	restaurant	company
day-work	1.0	0.125	0.1666666666...	0.1428571428...	0.1428571428...	0.1666666666...	0.0	0.0	0.0
week-day	0.125	1.0	0.2	0.1666666666...	0.1666666666...	0.2	0.0	0.0	0.0
morning	0.1666666666...	0.2	1.0	0.25	0.25	0.3333333333...	0.0	0.0	0.0
afternoon	0.1428571428...	0.1666666666...	0.25	1.0	0.3333333333...	0.25	0.0	0.0	0.0
evening	0.1428571428...	0.1666666666...	0.25	0.3333333333...	1.0	0.25	0.0	0.0	0.0
night	0.1666666666...	0.2	0.3333333333...	0.25	0.25	1.0	0.0	0.0	0.0
home	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0833333333...	0.0
restaurant	0.0	0.0	0.0	0.0	0.0	0.0	0.0833333333...	1.0	0.0
company	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
university-class	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1428571428...
university-library	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1428571428...
car	0.0	0.0	0.0	0.0	0.0	0.0	0.0625	0.0909090909...	0.0

Figure 7: Results of similarity calculations

This study aims to verify the performance of combining Q-learning and the CBR algorithm. The fundamental advantage of combining Q-learning and the CBR algorithm is the ability to adapt services to users according to their context and provide the most suitable services.

In order to verify the reliability of our proposed method, we examined the results of services requiring reformulation. If an adapted service had the same form when using the CBR algorithm, then the service was considered to be well adapted. Otherwise, it was considered to be poorly adapted. The experimental results of service adaptation via QL-CBR and CBR are presented in Figs. 8 and 9, respectively.

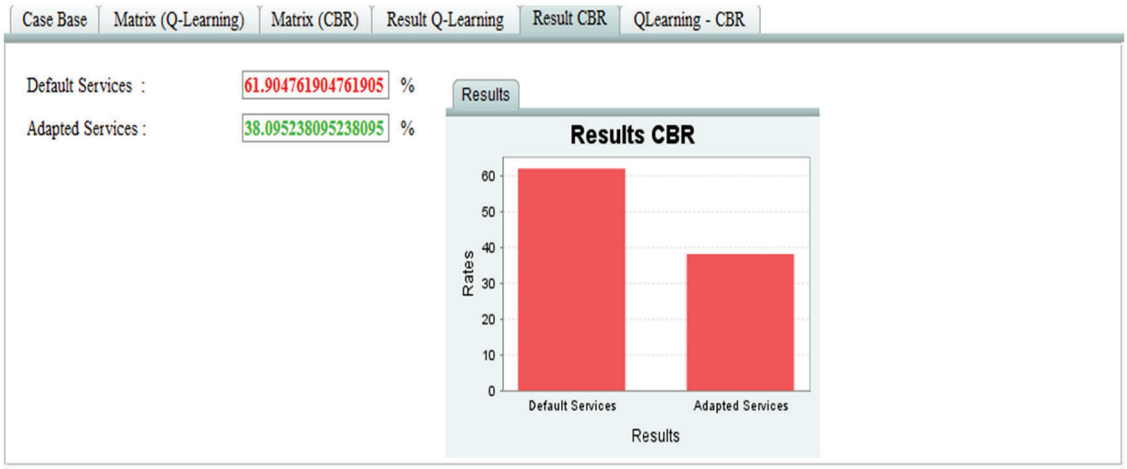


Figure 8: Results of CBR

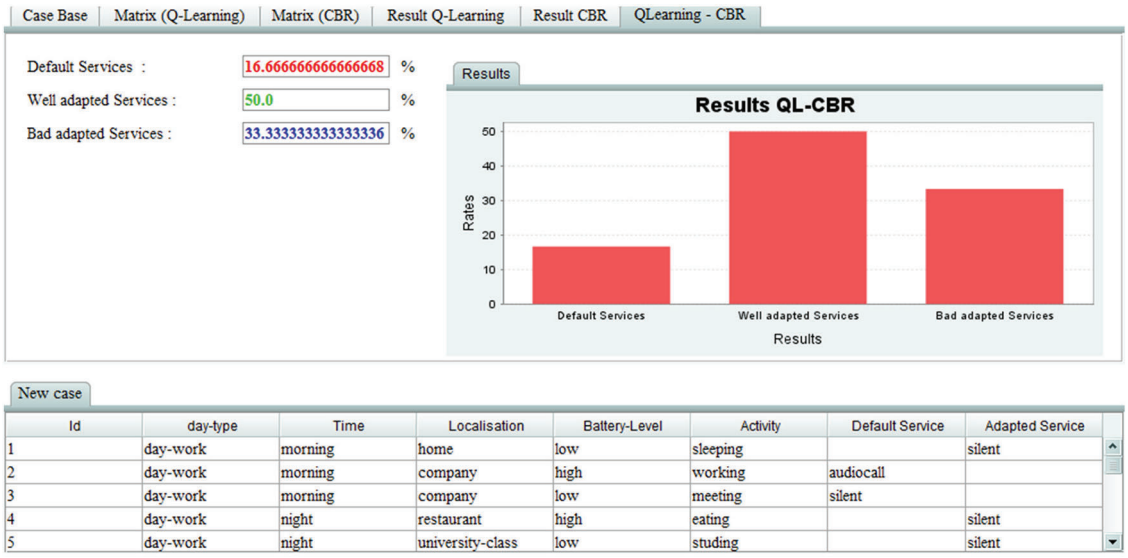


Figure 9: Result of QL-CBR

It can be seen clearly that success rate is 50% compared to CBR which has 38.09% service adapted. Moreover, the well-adapted service is 50% against 33,33% of bad adapted one. That means the percentage of well-adapted service has largely exceeded the bad-adapted one in QL-CBR, which proves the effectiveness of this method.

Fig. 10 presents the results of QL-CBR after 1,000 iterations. Overall, combining Q-learning and CBR seems to yield superior performance in terms of service adaptation. We noticed that the rate of poorly classified services decreased and that of well-adapted services increased with the number of iterations. We also noticed that the overall rate of adapted service increased with the number of iterations. For example, at 100 iterations, the ratio of well-adapted services was approximately 40% (Fig. 10). However, at 1,000 iterations, this ratio reached 90%.

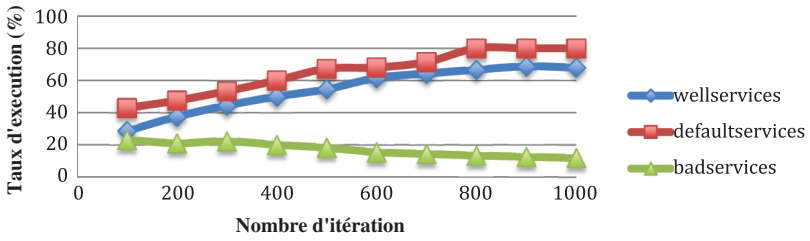


Figure 10: Simulation results of QL-CBR

7 Conclusion

In this paper, we introduced an RL algorithm for context-aware systems for adapting services. Our proposed approach consists of combining Q-learning and CBR, which facilitates the consideration of the current context for service adaptation. The entire process can be divided into three phases of operation. In the first phase, we introduce the Q-learning algorithm, which has the capability to classify all situations with appropriate services. It should be noted that this result is used as the base for the next phase, which incorporates the CBR algorithm to retrieve the most-similar contexts to a new context. The final phase involves adapting one or more solutions, if necessary, to fit the new situation. The obtained results are

very encouraging and indicate that our approach is able to provide reasonable services. One major remaining problem is long execution time. More generally, when using reinforcement algorithms, particularly Q-learning, in context-aware systems to respond to user needs, services must be effectively adapted prior to execution. However, the deployment of such a system in a real-world operational environment is a challenging task that still requires further investigation. Our future work will consist of testing additional reinforcement algorithms with time constraints in the learning phase and performing adaptation tasks for more than one.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest regarding the work reported in this paper.

References

[1] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee *et al.,* "Software engineering for self-adaptive systems: A research roadmap," in *Self-Adaptive Systems*, B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, J. Magee *et al.* (eds.), Vol. 5525. Berlin: lecture Notes in Computer Science, Springer, pp. 1–26, 2009.

[2] A. Colman, M. Hussein, J. Han and M. Kapuruge, "Context-aware and adaptive systems," in *Context in Computing*, P. Brezillon, A. J. Gonzalez (eds.), Chapter 5. Berlin: Springer, pp. 63–82, 2014.

[3] M. Hussein, J. Han and A. Colman, "Context-aware adaptive software systems: A system-context relationships-oriented survey," in Technical Report# C3–516_01, Swinburne University of Technology," 2010.

[4] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *IEEE Network*, vol. 8, no. 5, pp. 22–32, 1994.

[5] B. N. Schilit, N. Adams and R. Want, "Context-aware computing applications," in *Proc. of the 1994 First Workshop on Mobile Computing Systems and Applications*, NW Washington, DC, USA, pp. 85–90, 1994.

[6] D. Abowd, A. K. Dey, R. Orr and J. Brotherton, "Context-awareness in wearable and ubiquitous computing," *Virtual Reality*, vol. 3, pp. 200–211, 1998.

[7] P. Brézillon and J. C. Pomerol, "Contextual knowledge and proceduralized context," in *Proc. of the AAAI-99 Workshop on Modeling Context in AI Applications*, Orlando, Florida, USA, 1999.

[8] U. Meissen, S. Pfennigschmidt, A. Voisard and T. Wahnfried, "Context-and situation- awareness in information logistics," in *Current Trends in Database Technology*, W. Lindner, M. Mesiti, C. Türker, Y. Tzitzikas, A. I. Vakali (eds.), Vol. 3268. Berlin: lecture Notes in Computer Science, Springer, pp. 335–3444, 2004.

[9] A. S. M. Kayes, J. Han and A. Colman, "PO-SAAC: A purpose-oriented situation-aware access control framework for software services," in *proc. of 26th Int. Conf. on Advanced Information Systems Engineering, CAiSE 2014*, Thessaloniki, Greece, Springer, pp. 58–74, 2014.

[10] M. Miraoui and C. Tadj, "A service-oriented definition of context for pervasive computing," in *Proc. of the 16th Int. Conf. on Computing*, Mexico City, Mexico, IEEE computer society press, 2007.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, pp. 322, 1998.

[12] M. Dorigo and H. Bersini, "A comparison of Q-learning and classifier systems," in *Proc. of the 3rd Int. Conf. on Simulation of Adaptive Behavior: From Animals to Animats 3*, Cambridge, Mass, USA, MIT Press, pp. 248–255, 1994.

[13] B. Yuan and J. Herbert, "Context-aware hybrid reasoning framework for pervasive healthcare," *Personal and Ubiquitous Computing*, vol. 18, pp. 865–881, 2014.

[14] M. C. Mozer, "Lessons from an adaptive house," in *Smart Environments: Technologies, Protocols, and Applications*, D. Cook, R. Das (eds.), New York: J. Wiley & Sons, pp. 273–294, 2004.

[15] S. Wang, Z. Zheng, Z. Wu, Q. Sun, H. Zou *et al.,* "Context-aware mobile service adaptation via a Co-evolution extended Classifier System in mobile network environments," *Mobile Information Systems*, vol. 10, pp. 197–215, 2014.

[16] D. Mandato, E. Kovacs, F. Hohl and H. Amir-Alikhani, "CAMP: A Context-Aware Mobile Portal," *IEEE Communications Magzine*, vol. 40, no. 1, pp. 90–97, 2002.

[17] H. Ni, X. Zhou, D. Zhang, K. Miao and Y. Fu, "Towards a task supporting system with CBR approach in smart home," in *Proc. of the 7th Int. Conf. on Smart Homes and Health Telematics: Ambient Assistive Health and Wellness Management in the Heart of the City*, Tours, France, Springer, pp. 141–149, 2009.

[18] W. Y. Lum and F. C. Lau, "A context-aware decision engine for content adaptation," *IEEE Journal of Pervasive Computing*, vol. 1, no. 3, pp. 41–49, 2002.

[19] M. Miraoui, S. El-Etriby, C. Tadj and A. Z. Abid, "A hybrid modular context-aware services adaptation for a smart living room," *Intelligent Automation & Soft Computing*, vol. 24, no. 2, pp. 299–308, 2018.

[20] M. H. Kabir, M. R. Hoque, H. Seo and S. H. Yang, "Machine learning based adaptive context-aware system for smart home environment," *International Journal of Smart Homes*, vol. 9, no. 11, pp. 55–62, 2015.

[21] F. M. Ali, S. W. Lee, Z. Bien and M. Mokhtari, "Combined fuzzy state Q-learning algorithm to predict context aware user activity under uncertainty in assistive environment," in *Proc. of the Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, Phuket, Thailand, pp. 57–62, 2008.

[22] J. Hong, E. H. Suh, J. Kim and S. Kim, "Context-aware system for proactive personalized service based on context history," *Expert Systems with Applications*, vol. 36, no. 44, pp. 7448–7457, 2009.

[23] X. Xu, D. Hu and H. He, "Accelerated reinforcement learning control using modified CMAC neural networks," in *Proc. of the 9th Int. Conf. on Neural Information Processing*, Singapore, pp. 2575–2578, 2002.

[24] R. Bianchi, C. Ribeiro and A. Costa, "Accelerating autonomous learning by using heuristic selection of actions," *Journal of Heuristics*, vol. 14, no. 2, pp. 135–168, 2008.

[25] J. L. Kolodner, "Educational implications of analogy: a view from case-based reasoning," *American Psychologist*, vol. 52, no. 1, pp. 52–57, 1997.

[26] C. K. Riesbeck and R. C. Schank, *Inside Case-Based Reasoning*, 1st. ed., Hillsdale (New Jersey): Lawrence Erlbaum & Associates, Psychology Press, pp. 448, 1989. [Online]. Available at: https://doi.org/10.4324/9780203781821.

[27] H. Ahmad, "Computer fault diagnosis system using case-based reasoning," *International Journal of Latest Research in Science and Technology*, vol. 3, no. 2, pp. 1–6, 2014.

[28] R. Ros and J. L. Arcos, "Acquiring a robust case base for the robot soccer domain," in *Proc. of the 20th International Joint Conf. on Artificial Intelligence*, M. Veloso (eds.), India: AAAI Press, pp. 1029–1034, 2007.

[29] R. Ros, J. L. Arcos, R. L. De Mantaras and M. Veloso, "A case-based approach for coordinated action selection in robot soccer," *Artificial Intelligence, elsevier*, vol. 173, no. 9–10, pp. 1014–1039, 2009.

[30] M. M. Richter and R. O. Weber, Basic CBR elements. In: *Case-Based Reasoning: A Textbook*. Vol. Chapter 2. Berlin: Springer, pp. 17–40, 2013.

[31] C. Fellbaum, *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, pp. 423, 1998. [Online]. Available at: https://doi.org/10.7551/mitpress/7287.001.0001.