Multi-Agent Reinforcement learning Approach to IoT Coordination

Multi-Agent Reinforcement learning Approach to IoT Coordination

Radia Belkeziz, Zahi Jarir, Ilyass El Kassmi

Cadi Ayyad Universiy, Faculty of Sciences Semlalia, Boulevard Prince Moulay Abdellah, Marrakech 40000

E-mail: radia.belkeziz@ced.uca.ma, jarir@uca.ma, ilyass.elkassmi@ced.uca.ma

Abstract. Nowadays, the IoT is evolving at a very fast pace and has proven its usefulness in several areas by creating better applications and services. However, more flexible approaches proving a well-defined architecture meeting the general requirements and building blocks of IoT are still needed despite the results obtained in the literature. In this paper, we focus on the IoT coordination challenge which represents a fundamental property allowing things to collaborate and make a decision when an appropriate change is detected in its environment. This contribution proposes an agent-based approach coupled with Q-learning which is a reinforcement learning technique, to compensate for coordination in its entirety, namely objective coordination and subjective coordination. To illustrate this approach, an evacuation use case is presented.

1. Introduction

Nowadays Internet of Things or IoT is a blazing technology and is gaining ground over time, especially when it is merged for example with artificial intelligence and big data techniques. It has proven its usefulness in several areas through building better quality applications and services. Among the definition proposed in the literature, IoT is defined as a network, integrating communication technologies, formed by multitudes of connections between physical objects, places, concepts, which make it possible to think and create a large number of applications involved in the daily life: health, household appliances, agriculture, connected vehicles etc. To take advantage of this technology, several architectures, research projects, frameworks, tools etc. have been suggested. However despite the obtained results, IoT challenges evolve and become more and more complex, especially when the required IoT system or process recommends some properties such as processing of sensitive, heterogeneous, massive and real-time data, being aware, responsive, having the advantage to make decision, etc. In this paper we focus on IoT coordination challenge that represents a fundamental property enabling things to collaborate and make decision when an appropriate change is detected in its environment. In our opinion to meet this challenge, an entire process of organizing objects, tasks and services, while taking into account the environment and its changes is needed. In the literature, a generic IoT architecture dealing with coordination is still absent. This is because coordination affects both what is happening in the system and its environment. This is the reason why these architectures are application oriented and in general cannot be used to build other applications. In general, coordination is approached through two mechanisms: orchestration or choreography. These are the two classic ways of coordinating services / actions of connected objects. Another way to

approach coordination is to base the architecture on existing coordination models. However, coordination is a whole process that must be taken into account following the general steps to set up an IoT system and its building blocks which are identification, sensing, communication, computation, services and semantics [1]. Most of the time, to approach coordination, rules are established in advance according to the use case. The question that remains is whether these rules will stay true over time; alternatively, is there a need to intervene each time to modify them and is there a way for this modification to be automatic? All these observations led us to propose a solution which respects the coordination process and whose modeling is based on Multi-agents System (MAS). We chose to use MAS for two reasons. The first is trivial: MAS is a research axe of AI and the second, we believe that coordination falls under the distributed resolution of problems which consists in organizing the cooperation of connected objects by sharing knowledge by relying on each individual's capacities and knowledge. In MAS coordination should be handled at two different yet linked levels: objective coordination and subjective coordination. Objective coordination refers to the interaction between agents, defined by each one's rules considering the environment, while subjective coordination refers to the overall behavior of the system and considering the objective coordination.In the same time, machine-learning techniques have undergone a great evolution and are also taking place in the technology used in a daily basis. Applying these machine learning techniques in the coordination process could lead to more advanced applications and especially to making the IoT systems more autonomous with less to none human intervention. Relatively new, the concept of AI applied to IoT has nevertheless opened up many perspectives in the IoT domain. AI brings more to the IoT by giving a learning capacity to the connected objects, and, allows to structure the data resulting from the IoT and thus they transform into information of value which can improve the operation of the system or to be exploited by decision-making processes. The reminder of this article is organized as follows. Section 2 is dedicated to related work and presents some interesting contribution regarding IoT services coordination and multi-agent coordination. In section 3, we describe our proposed approach based on both multi-agent and Qlearning techniques. Before concluding in section 5, an evacuation case study is discussed in section 4.

2. Related work

2.1. IoT service coordination

An event-driven situation-aware dynamic IoT services coordination approach is presented in [2] to meet the heterogeneity of the different physical objects and integrate them into the context-aware IoT infrastructure and the on-demand distribution of situation-aware information and dynamic, event-driven service coordination are implemented. This solution uses the advantages of both SOA (service-oriented architecture) and EDA (Event-Driven architecture) and integrates a situational event pattern and a situational event-driven service coordination behavior model based on event-condition-action trigger mechanism.

A goal-driven service orchestration approach for IoT environment is resented in [3]. It can orchestrate the service processes dynamically based on the context and users' goal. A reasoning mechanism is constructed as an orchestrator and a semantic metadata specification of RESTful is defined for IoT services.

A semantic metadata specification of RESTful is defined for IoT services in [4]. A remotely configurable MQTT client implementation is handled mixing both orchestration and choreography to organize the services. Authors in this paper have discussed the pros and cons of each architecture and have came up with the fact that both should be used and that they complete each other. They found that a mixed architecture based on both orchestration and choreography is better than choosing between the two mechanisms as it makes it possible to create new logical interactions.

2.2. Multi-agent coordination

Authors deal with learning in reactive multi-agent systems [5], in an environment where several agents have to coordinate their actions in order to solve a task together. Authors introduce two algorithms ACE (Action Estimation) and AGE (Action Group Estimation) for the delayed reinforcement learning of sequences of action sets. The agents collectively learn to estimate the goal relevance of their actions. According to the estimates, agents coordinate their actions and generate the proper action sequence. Both ACE and AGE algorithms follow the same working cycle that consists of repeated execution of action determination, competition and credit assignment. The difference between the two is at the competition level as the agents do not compete on carrying out individual actions but groups of actions for the AGE.

Authors started from two situations in [6]. The first is when the agent is capable of achieving by himself its sub-goal but needs a coordinated course of action to organize the interactions. The second situation is when an agent needs assistance from other ones to achieve its goal. Therefore, they came up with algorithms based on classical planning. Agents use them to generate their own plans and plans that are compatible with other agents' plans to avoid harmful interactions. The Agent-based COoperating Smart Object (ACOSO) middleware is presented in [7. It supports the development of smart object-based IoT systems. Each smart object is abstracted into a cooperating agent. The agent is characterized by its set of tasks that represent both agent lifecycle management operations and specific operations that define smart objects distinct services. A Task Management System is used to coordinate the operations and a Communication Management System is utilized to realize smart objects interactions and coordination mechanisms and paradigms.

2.3. classification of coordinaion approaches

		B. Cheng et al. (2017)	S. Mayer et al. (2016)	S.Cherie ret al. (2018)	G. WeiB (1993)	D. Yannis et al. (2006)	C. Savaglio et al. (2016)
Context-awareness		✓	✓	✓	✓	✓	✓
Decison-making		✓ remote monitoring center	✗	✗	✓ ACE/AGE for selecting actions	✓ Agent planning to coordinate actions	✓ Knowledge base Management System
Model		EDSOA	REST	SOA	Agent-based	Agent-based	Agent-based
Coordination mechanisms	Orchestration	✓	✓	✓	✗	✓	✓
	Choreography	✗	✗	✓	✗	✗	✗

Table1. Classification of coordination approaches

All the established requirements have to met in order to achieve coordination. However Table1 shows that these requirements are not satisfied. In this work, we have built our coordination process following IoT building blocks. It includes data collection, data processing, interpretation and decisiong-making. Then, we came up with a three-layer architecture: object layer, MAS layer and Coordination layer based on the coordination process. We have modeled the connected objects according to multi-agent systems. Coordination is based on context and decision-making. The context is defined by policies linked to environmental parameters. As for decision-making, it is based on machine learning techniques, particularly reinforcement learning. The coordination approach is defined such that it can assimilate the two mechanisms orchestration and choreography.

3. Proposed approach

3.1. Coordination process

Coordination as we have noted before, is a complex challenge that requires an entire organization. We have therefore established that to approach this challenge in its entirety, it was necessary to follow the general steps to build an IoT system [1].

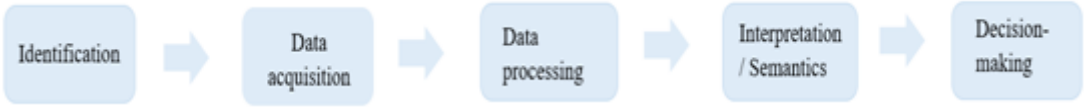

Figure1. Coordination process

We can summarize the main tasks of the coordination process as described in figure 1. These tasks are:

• The identification makes it possible to recognize each object of the network by giving it a unique identity.

• The data acquisition is the fact that each object collects data by querying its environment. In some cases, this can cause communication between objects in the form of sending/receiving messages, events or notifications.

• Data processing is a series of processes for manipulating data to extract information or produce knowledge.

• The interpretation / semantics is to give meaning to the information or knowledge acquired to be able to operate the steps to follow.

• Decision-making is the ultimate step as it enables to identify the system's actions that have to be taken in order to achieve its goal.

3.2. Proposed Architecture

The proposed architecture is a three layer architecture as presented in Figure2. It includes the object layer, the Multi-agent layer and the coordination layer

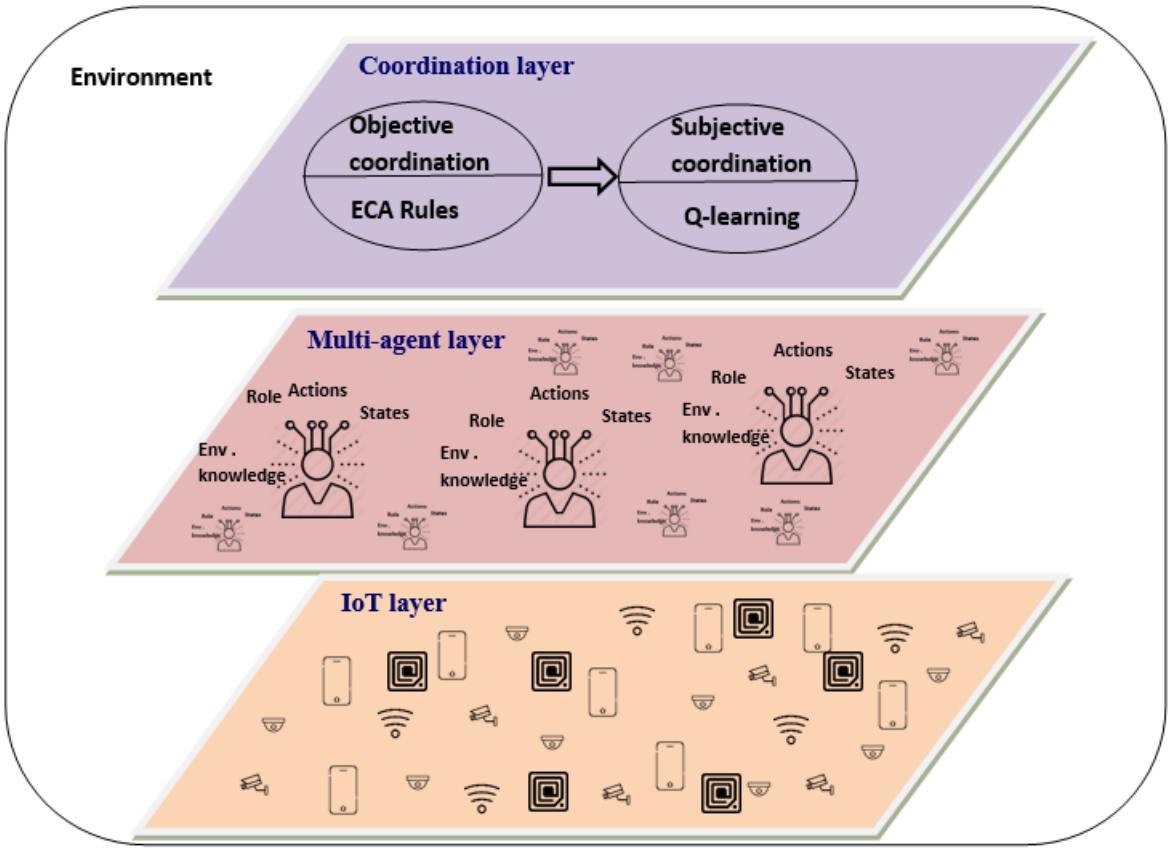

Figure2. An Agent-based architecture for IoT Coordination

In the following, we present the architecture in details.

3.2.1. Environment:

The environment where the system operates is defined by its set of states noted S and includes a set of connected objects (things) that interrogate the environment and monitor it. They may be equipped with sensors/actuators and in some cases may have processing features or storage capabilities. Objects perform their tasks and can collaborate and communicate throughout executing their tasks. All collaboration and communication are triggered by events. The environment is also defined by several parameters which represent the characteristics of the environment for which the values are defined for each environment's state. We specify that the environment can go through several states over time that we can group in two general categories normal and abnormal.

- Normal: the normal state of the environment is that of everyday life when there is no problem that arises.
- Abnormal: the state of the environment changes to abnormal when one or several environment parameters change, when an agent does not respond or is down etc... In the case of an abnormal state of the environment, three outcomes are possible:
- Call for coordination: Coordination is triggered when there is an opportunity for the IoTs to maintain the purpose of the system and to collaborate to achieve it
- Human intervention: there is a need for human intervention when the system is no longer able to function properly or when there is a case of extreme emergency.
- Call for a single agent that can to handle the problem.

To define whether the state of the environment is normal or abnormal, we have opted for the

Event-condition-action mechanism. The condition defines the state and the action defines the behavior to adopt towards this state. A condition can take the form of: an evaluation of the value of one or more parameters
or/and
listening for events triggered by other objects.

-Environment policies:
Here, we present the notations: op() : Operation that indicates equality or inequality between the parameters of the environment established for the policies and the data captured by the agents. $(=; \leq ; \geq)$.
P_a: environment parameter.
C: Condition which determines the state of the environment.

A policy is a condition on an environment parameter. A condition is an operation on an environment parameter noted C= $op(P_a)$. Parameters are the characteristics of the environment for which the values are defined for the normal state of the environment and which are known by the agents. We note them:
$P_a = \{p_a^1, p_a^2, \ldots, p_a^n\}$
p_a^n: nth environment parameter
$C = \{op(p_a^1), op(p_a^2), \ldots, op(p_a^n)\}$

3.2.2. IoT layer:
This layer holds all physical connected objects operating within the system.

3.2.3. Multi-agent layer:
At this level, we have chosen for the objects' abstraction to work with MAS. Each physical object is abstracted into an agent. Each Agent A_i is defined by:
An individual goal G_i
A set of roles. An Agent can have one or several roles. There are four roles:
-Data-collection: the agent interrogate its environment for collecting data.
-Data-storage: the agent stores information or collected along with its state.
-Data-processing: The agent process the collected data in order to extract information.
-Data-interpretation: interpretation of processed data.
-Decision-making: agent will be responsible for selecting the most optimal activity, deciding which action will be taken under the established condition.

A set of states $[St]_i = \{[St]_i^1, [St]_i^2, \ldots, [St]_i^n\}$. Agent changes state every time it performs an action or receives an event from another agent. States can be useful when storing agents' performances.
A set of actions $a_i = \{a_i^1, a_i^2, \ldots, a_i^m\}$ i means the i^th agent.
Knowledge of the environment E^i, which is the part of the environment's general states that the agentA_i knows about. It takes into consideration one or several environment conditions.
Each abstracting agent has access to the abstracted object components (i.e sensing and actuating components, computation, communication, and storage components). Thus, the sensed data are processed at this level in order to extract needed information.

3.2.4. Coordination layer:
The system we are proposing is a faultless system. Our goal is to find a solution for coordi-

nation. A faultless system assumes that the security challenge is resolved and that there is no risk regarding data.
Coordination is triggered when the state of the environment is abnormal and objects can collaborate to achieve a goal. The coordination between our agents is triggered based on events. An event is instantiated when an agent detects a change in a condition or receive an external event. This leads to a reorganization of tasks at the agent level in order to achieve goals. It is important in a multi-agent system to differentiate between objective coordination and subjective coordination.
Objective coordination concerns the description of the agents' environment, the interactions between agents and their environment and between agents themselves. While subjective coordination concerns the overall attitude of the system that ensures that the whole community of agents acts in a coherent manner. The relation between the two is that subjective coordination is based on objective coordination and assumes that it exists. In our case, to ensure objective coordination, we rely on the ECA paradigm. In terms of subjective coordination we will use Q-learning a reinforcement learning algorithm which will be based on the outcome of ECA.
To approach coordination, generally two mechanisms are used: orchestration or choreography. We describe objective and subjective coordination for both these mechanisms.

- Objective coordination:
- Orchestration is a coordination mechanism that consists of a central controller that coordinates interactions, which is responsible for calculations and decision-making. The collaborating agents all refer to this same controller to coordinate all of their interactions. In our case, the agents only do the perception work to control the various parameters of the environment. The central controller is then responsible for initiating communications and deciding on the various actions to be taken to achieve the overall goal.In the case of orchestration, the actions that agents must perform are decided by a single decision-making agent. The decision is centralized at the level of this agent and all the other agents have the roles of data-collection/data-processing.In this case, the central controller triggers the orchestration by sending an event to the agents involved in the already defined coordination. The data-collection/data-processing agents establish the conditions by evaluating environment parameters. Based on all these conditions, the decision-making agent decides on the action(s) to accomplish. It means that the decision-making agent have to collect all conditions before determining action(s) to undertake.
-Choreography is the coordination mechanism that is performed from a decentralized point of view. The overall behavior of the process is based on that of some or all of the parties. These parts, each follow and autonomously, its own act. The choreography is triggered by the agent which detects the anomaly in the environment by sending an event to all the other agents (broadcast). Then, the agents who are concerned evaluate their conditions to make a decision or collaborate with other agents by sending data through events to be able to make a decision.

-Subjective coordination:
The ECA's outcome can identify a single policy or a set of possible policies to follow. The Overall behavior is based on choosing the most optimal policy. Therefore, we decided to use Q-learning, a model-free reinforcement learning technique. This technique finds the optimal policy by maximizing the expected value of the reward for the following agent states, starting with the current state. By accomplishing an action a_i^k, an agent moves from a state $[St[_i$ to a state $[St]_j$. This provides the agent a reward and the agent's goal is to maximize its total reward. The Q-learning algorithm has a function that calculates the quality of a state-action. It is named Q and its formula is as follows: $Q(state, action) = R(state, action) + gamma * Max[Q(nextStep, allActions)]$ Before learning begins, Q is initialized to a fixed value (chosen by the programmer).

Q represents the memory of what the agent has learned. A reward matrix R is set. Rows represent the states and columns represent actions. The instant rewards are placed in the matrix. Then at each time t an action is selected for the chosen state$a_i^k[[St]_i^k]$, the reward R(state, action) is observed and Q is updated. It ends when the final state St is found.
Gamma is the learning rate. It is set between 0 and 1. If it is near 0, it means that the agents only considers immediate rewards so the learning is slow. If it is near to 1 it means that the learning is quick as the agent considers future rewards with greater weights.
For both types of coordination, the orchestration and choreography mechanisms can be altered depending on the situation. When it is a normal course of the case, without incidents, the mechanism can be either orchestration or choreography, depending on the choice made in advance.
However, when an incident occurs:
- If the mechanism used is orchestration, in this case we switch to choreography, for more flexibility.
- With the choreography mechanism, the use of a clustering algorithm can be practical, since the communication between the objects is done in a more reliable way and requires less power.
- Clustering allows the system to adapt to the changes. Several approaches for clustering exist. The choice will be made according to the situation and the criteria defined. The grouping could be carried out according to the distribution of objects or data. Clustering brings advantages to the system by enabling scalability, good management in real time and enables energy efficiency and robustness. This will be the subject of future work.

4. Use case

Evacuation is the moving of people away from an area that contains a threat, a hazard to a safe area. In general, the current evacuation plans are presented as static maps hung in different parts of the building. This practice creates several disadvantages: i) the evacuees all rush to the pre-selected evacuation routes at the same time, this results in a congested crowd; ii) these pres-elected static routes can not predict whether it includes hazards or obstacles; iii)no calculation of the different possible scenarios in time and in real time; iv)no situation- nor context-awareness; v)no distinction of evacuees regarding their abilities, ages, behaviors. The key to evacuation is to secure people without any trouble by making sure not to create a chaotic atmosphere. To achieve this, several parameters must be taken into consideration as it is presented in [8]. These parameters are counted for two factors: the factor of the environment ie the building, and the human factor ie the evacuees. For the building factor, the most important parameters are the number of emergency exits, the capacity (maximum amount of people) of each area as well as the capacity of the passages between areas and doors capacity. The capacities dynamically change over time, and re-computing these data allows understanding possible route scenarios and thus avoiding congestion. For the evacuees, pedestrians are characterized by several parameters such as dynamics, personal choice and vulnerability. People are in different areas of the building and their goal is to reach outside/safe area by moving from an area to another.
For our use case, we can view the building as a graph $G = (V, A)$ where V is the set of cells x that represent areas and A is the set of arcs that represents the passages between areas. We will base our notations on the work presented in [8].

t: unit of time slot where $t \in 0, 1, \ldots, T$

x_a^t: state of node x where a is the number of people that occupy x at t.

x_n : n is the maximum number of people that can occupy the area x at any time.

$[(x, y)]_b^t$: state of the passage between area x and area y where b is the number of people that traverse x to y at t.

$[(x, y)]_c[(y, x)]_c$: c is the maximum capacity of the passage between x and y i.e, the maximum number of people that can traverse (x,y) at any time.

4.1. case of an orchestration:

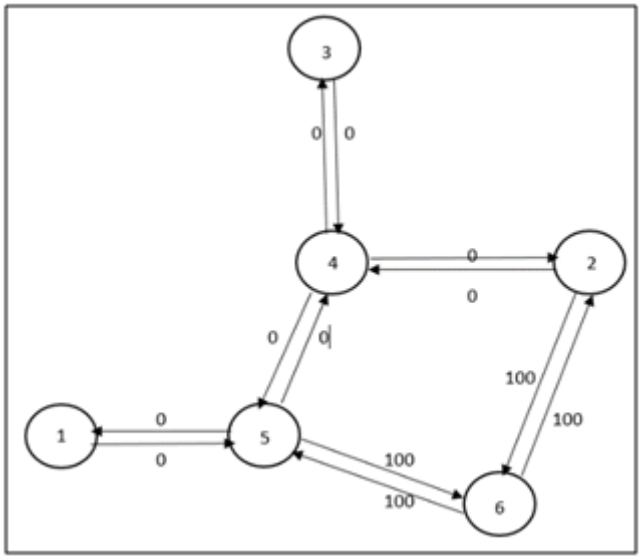

Figure3.Graph representing the building and its areas

We are working on a space with 5 areas and a single exit (number 6). The next step is to assign a reward to each passage and we do the following:
- Passages of areas that are directly linked will have a reward of 0.
- Passages of areas that directly lead to the exit are assigned 100 as a reward.
- Exit 6 iban en double ou quelque chose.
This is the representation of the static evacuation route.
In Q-learning the end goal is to reach the state with the highest reward. We can put the state diagram depicted in figure, and the instant reward into a reward matrix R.

$$R = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Figure4. Initialization of marix R from static route

-1 means that there isn't any passage between areas. The next step is to add another matrix Q whose rows represent the current state and columns represent the possible actions leading to the following state. The formula for calculating Q was presented before. Q portrays the memory of the agent's learning experience. The computing is done by the orchestrator. The reward matrix at the input of the algorithm is that defined by the static evacuation plan.
When agents receive the event$e_v orch$ they evaluate the following conditions (objective co-ordintion):

- Agents in areas
$x_a^t - x_a^{t-1} = \sum_{(x,y)\in A}((x,y)_b^{t-1} - \sum_{(y,x)\in A}((y,x)_b^{t-1}$; $t \in T$, $t > 0$, $x \in V$, $y \in V$
$0 \leq x_a^t \leq x_n$; $x \in V$, $t \in T$

- Agents in passages
$0 \leq (x,y)_b^t + (y,x)_b^t \leq (x,y)_c(y,x)_c$; $(x,y),(y,x) \in A$

Then each agent sends the evaluation to the orchestrator as we note in the following :
$c_i : 0 \leq x_a^t \leq x_n$ and $x_a^t - x_a^{t-1} = \sum_{(x,y)\in A}[(x,y)]_b^{t-1} - \sum_{(y,x)\in A}(y,x)_b^{t-1}$
$\rightarrow e_i(c_i)op$
$c_j : 0 \leq x_c^t \leq x_m$ and $x_c^t - x_c^{t-1} = \sum_{(x,y)\in A}[(x,y)]_d^{t-1} - \sum_{(y,x)\in A}(y,x)_d^{t-1}$
$\rightarrow e_j(c_j)op$
...
$c_p : 0 \leq x_e^t \leq x_l$ and $x_e^t - x_e^{t-1} = \sum_{(x,y)\in A}[(x,y)]_f^{t-1} - \sum_{(y,x)\in A)}(y,x)_f^{t-1}$
$\rightarrow e_p(c_p)$
$\otimes\{c_i, c_j, \ldots, c_p\} \vdash a_Dec \rightarrow e_v dec(a_Dec)$

For example an agent in a certain area where people needs to be evacuated evaluates its condition and the agent in the passage that leads to the area also evaluates its condition, if $0 \leq x_a^t \leq x_n$ is true and $0 \leq (x,y)_b^t + (y,x)_b^t \leq (x,y)_c(y,x)_c$ is true, the reward will be 0. If one of these parameters is false the reward will be -1.
Each time the conditions are re-evaluated the matrix is updated. Then after, the matrix Q is calculated and the decision on the most adequate path is given.
a_Dec is the decision-making actions that are the result of the Q-learning algorithm calculation.

4.2. case of a choreography
In the case of a choreography, the Q-learning algorithm is used according to the agent's own point of view. That is, the matrix R represents the agent in question and all its available actions. Each agent can have in addition the role of Q-learning agent. The Q-learning algorithm is then used to select the adequate available action to take next and after that the agent informs other ones with whom he collaborates about the selected action.
• Example
The difference between the first implementation and the second is that the matrix R is defined for one agent and its knowledge of the environment. For example for the state 5 the matrix R is as follows:
$R = ([-1, 0, 0, -1, 0, -1)]$
The matrix R is updated each time other collaborating agents send the evaluations of their conditions or the decisions of actions to take.
$e_i \rightarrow c_j : 0 \leq x_a^t \leq x_n$ and $x_a^t - x_a^{t-1} = \sum_{(x,y)\in A}(x,y)_b^{t-1} - \sum_{(y,x)\in A}(y,x)_b^{t-1}$
$\rightarrow e_j(c_j)op$
$e_i \rightarrow c_k : 0 \leq x_c^t \leq x_m$ and $x_c^t - x_c^{t-1} = \sum_{(x,y)\in A}(x,y)_d^{t-1} - \sum_{(y,x)\in A}(y,x)_d^{t-1}$
$\rightarrow e_k(c_k)op$
...
$e_i \rightarrow c_n : 0 \leq x_e^t \leq x_l$ and $x_e^t - x_e^{t-1} = \sum_{(x,y)\in A}(x,y)_f^{t-1} - \sum_{(y,x)\in A}(y,x)_f^{t-1}$
$\rightarrow e_n(c_n)$
$(e_j(c_j), e_k(c_k), \ldots, e_n(c_n)) \otimes c_i \vdash a_i Dec$
OR
$c_j \oplus (e_k(c_k), e_l(c_l), \ldots, e_n(c_n)) \vdash a_j Dec \rightarrow e_j(a_j Dec)op$
$c_k \oplus (e_j(c_j), e_l(c_l), \ldots, e_n(c_n)) \vdash a_k Dec \rightarrow e_k(a_k Dec)op$
...

$(e_j(a_jDec), e_k(a_kDec), \ldots, e_n(a_nDec)) \otimes c_i \vdash a_iDec$

a_Dec is the decision-making actions that are the result of the Q-learning algorithm calculation. Other definitions are the same and the calculation is the same.

5. Conclusion

IoT coordination is the core of any IoT system. This article proposed a coordination approach based on the specificity of multi-agent systems and techniques of artificial intelligence, namely Q-learning, for the two coordination mechanisms: orchestration and choreography. An architecture was presented along with its different layers based on the IoT coordination process that was defined. The choice to integrate the multi-agent system was taken to exploit their advantages, namely to add intelligence within the object itself, to take advantage of the ability to process heterogeneous data, coming from a complex and distributed environment. However, by also combining the techniques of machine learning, more specifically reinforcement learning, it helps to define coordination as a whole. To illustrate all of this, an evacuation case study was presented. For futur work we aim to explore more the clustering techniques to take advantage fully of the choreography part of coordination. As we mentioned, clustering enables more scalability, good management in real time and enables energy efficiency and robustness, in addition to some aspects of security.

6. References

[1] Belkeziz R and Jarir Z 2020 *IJSSMET* **11** 99-115
[2] Cheng B, Wang M, Zaho S, Zhai Z, Zhu D. and Chen J 2017 *IEEE/ACM Trans. Netw.* **25** 2082-95
[3] Mayer S, Verborgh M, Koyatsch M and Mattern F 2016 *IEEE T Autom. Sci. Eng.* **13** 1247-55
[4] Cherrier S and Langar R 2018 *GIIS*
[5] WeiB G 1993 *13th Int. Conf. AI*
[6] Dimopouos Y, and Moratis P 2006 *IEEE WIC ACM Int. Conf. Intell. Agent Technol.*
[7] Savaglio C, Fortino G and Zhou M 2016 *IEEE 3rd Forum IoT.*
[8] Choi W, Hamacher H, Tufekci S 1988 *Europ. J. Operat. Resear.* **35** 98-110