Received: 6 June 2021 | Revised: 30 December 2021 | Accepted: 22 February 2022

DOI: 10.1111/coin.12516

ORIGINAL ARTICLE

Computational Intelligence WILEY

A deep reinforcement learning based decision-making approach for avoiding crowd situation within the case of Covid'19 pandemic

Wejden Abdallah[1] | **Dalel Kanzari**[2] | **Dorsaf Sallami**[2] | **Kurosh Madani**[3] | **Khaled Ghedira**[4]

[1]National School of Computer Sciences (ENSI), LARIA Laboratory, University of Manouba, Manouba, Tunisia

[2]Higher Institute of Applied Sciences and Technology, University of Sousse, Sousse, Tunisia

[3]LISSI/EA 3956 Laboratory, Senart Institute of Technology, Paris-East University (UPEC), Lieusaint, France

[4]Central University of Tunis, Tunis, Tunisia

Correspondence
Wejden Abdallah, National School of Computer Sciences (ENSI), LARIA Laboratory, University of Manouba, Manouba, Tunisia.
Email: wejden.abdallah@ensi-uma.tn

Abstract

Individuals' flow's fluidifcation in the same way as the thinning of the population's concentration remains among major concerns within the context of the pandemic crisis situations. The recent COVID-19 pandemic crisis is a typical example of the aforementioned where on despite of the containment phases that radically isolate the population but are not applicable persistently, people have to adapt their behavior to new daily-life situations tempering Individuals' stream, avoiding tides, and watering down population's concentration. Crowd evacuation is one of the well-known research domains that can play a pertinent role to face the challenge of the COVID-19 pandemic. In fact, considering the population's concentration thinning within the slant of the "crowd evacuation" paradigm allows managing the flow of the population, and consequently, decreasing the probable number of infected cases. In other words, crowd evacuation modeling and simulation with the aim of better-exploiting individuals' flow allow the study and analysis of different possible outcomes for designing population's concentration thinning strategies. In this article, a new decision-making approach is proposed in order to cope with the aforesaid challenges, which relies on an independent Deep Q Network with an improved SIR model (IDQN-I-SIR). The

machine-learning component (i.e., IDQN) is in charge of the agent's movements control and I-SIR (improved "susceptible-infected-recovered" individuals) model is responsible to control the virus spread. We demonstrate the effectiveness of IDQN-I-SIR through a case-study of individuals' flow's management with infected cases' avoidance in an emergency department (often over-crowded in context of a pandemic crisis).

KEYWORDS

COVID'19, crowd situation, decision making, deep reinforcement learning, multiagent reinforcement learning, SIR model

1 | INTRODUCTION

A pandemic is a crisis in which humanity risks the spread of a deadly virus. In some cases, we are vulnerable to infection through contact with infected or suspicious people. Humanity has a long history of pandemics and epidemics, which have invaded populations often causing many deaths. The severity of the virus may decrease as populations develop some degree of immunity. New species of human viruses are still being identified and, like many pandemics over the years, the coronavirus pandemic (COVID-19) has been a major shock to the world as a new respiratory virus in late 2019 and early 2020, with more than 100 million infections reported to date in less than a year.

The well-known novel virus attracts major public attention. Many researchers of different specialties have been motivated to understand and analyze the situation according to several parameters and objectives. Some research works,[1,2] have analyzed the spread of the virus in some countries based on some parameters such as population density, temperature, humidity, and so forth. Others have[3] analyzed the mortality rate. Another article[4] studied the relationship between the duration of containment, the number of infected persons and deaths due to Covid-19, and the economic growth of countries. Other articles[5,6] focus on predicting the number of deaths cases. Many strategies have been adopted to limit the spread of the virus by several countries, such as containment, city lockdown, traffic stops, and social distancing. However, the global spread of the new coronavirus has led to a substantial disruption of global economic activity,[7] that forced governments to suspend their closure measures.

Waiting for the vaccine to fight this pandemic, and given the global economic situation, the majority of governments have decided to suspend containment and adopted health protocols like, avoiding gathering to minimize the contamination in article, street, shops, and public area to live and adapt this virus. The most serious is the appearance of new infections, in some countries only a few days after relaxing social restrictions. In this context, governments are trying to keep people active regularly by protecting from the virus to find a balance between their own needs to resume basic activities when the world feels increasingly dangerous. Indeed, the main argument is how to behave and make decision to limit discreet mutable and rapid spread of the virus. To deal with, access to public spaces has been restricted, as various mitigation measures have been taken around the world to reduce physical interactions between people.[8] Participate widely in spiritual

and cultural activities, tourism and indoor entertainment in a crowded environment, in the time of an emergency, disorderly evacuation will not only reduce the effectiveness of the evacuation but also will lead to collisions and muddying and cause secondary damage.[9] Within current situation of COVID'19, the gathering of people will result in a high number of contacts, and subsequent infection, and consequently, the number of people requiring hospitalization and number of death will raise. Therefore, using evacuation strategies are suitable to make public spaces more fluid by respecting social distancing and sanitary protocols like disinfectant gel, wearing masks, avoiding hugging, and keeping a minimum distance of one meter.

Increasing number of researches on "crowd situations" as complex systems with collections of individuals study and model its physical, biological, social, and cultural characteristics.[10] Generally, the degree of the crowds depends on the crisis caused by a man-made or natural disaster, such as a fire, earthquake, terrorist incident, and pandemic, and so on. Within this, the perception of environmental risk often stresses the crowd and evokes chaotic disorder[11] as well as many lives may be lost if there are no effective strategies to avoid the crowd, to evacuate and save lives consequently. Developing an "avoiding crowd system" consider as a challenge because it involves complex interrelated parameters, diverse individuals and environments, as well as lack of direct evidence. Evacuation modeling, simulation and technics such as social force model, cellular automaton, metaheuristic, optimization, machine learning for studying crises have been used to analyze various possible outcomes as different scenarios unfold, typically when the complexity of the scenario is high.[12]

Therefore, this article represents a new strategy based on reinforcement learning (RL) to study the dynamics of the environment and follow its evolution, aiming to avoid crowds and preserve safety. This article focuses on different aspects of agent-based modeling and simulation systems and proposes a decision-support system that avoids crowds and mitigates during the pandemic. Among various techniques, Reinforcement Learning approaches have proven to be exceptionally effective for a variety of complex tasks and there has recently been a growing interest in applying RL to sequential decision-making tasks in the real world. Hence, RL as a suitable technic allows continuous learning and well adaptable to the real problems.

With this context, we propose an approach for decision-making during a crisis that offers a safe solution and decision for avoiding crowds. We are interested in the postlock-down pandemic phase, in particular COVID-19 crisis. To achieve this goal, we design and implement a multiagent system capable of reaching the location of targets with a minimum of movement, avoiding crowds, collisions between agents, obstacles placed in the environment, and certainly contaminated agents. This method is based on a fully decentralized multiagent system where the agents are independent. The essential characteristic of this system is that its participants are required to act and perform autonomous tasks whose impact is perceived in a shared environment with the other participants. Decision making process depends on deep reinforcement learning results to get the optimal path for the destination, as well as the states of the neighbors in question obtained by using an improved SIR Model to identify susceptible, infected, and recovered agents. Furthermore, a parameter of the crowd in each case could be calculated and considered in the decision process to better avoid gathered paths or destinations.

As an example of a real-life situation, we demonstrate the effectiveness of this approach within a case of individuals' flow's management with infected cases' avoidance in an emergency department. After the appearance of coronavirus, the emergencies became crowded, and even people are panic to go there to get treatment for another illness concerning catch the virus. The environment is model in the form of a grid transforming emergency department information into barriers, entry or exit emergency, and agents (healthy, infected, and recovered). In the time an agent goes

to the emergency department for emergency treatment, he must avoid to be infected by the virus and leave in heal specially by avoiding the crowd. It is observed that the modeled environment can provide an overview of the critical areas in a crowd avoidance situation. In addition, deep reinforcement algorithm can provide useful and concert information.

This article is organized in five sections as follow: Section 2 gives an overview of the related works and Section 3 describes the overall architecture of the proposed approach. In Section 3.3, the requirements for the environment and techniques to simulate the approach are given in detail. Section 4 represent the results of the simulation and discussions. Finally, Section 5 concludes the article and gives future perspectives.

2 | RELATED WORKS

Concerning avoiding crowds and safety, many works are dealing with crowd evacuation. Many techniques and methodologies are used. This article focuses on works using RL in a crowd situation. This section is an overview of the diverse approaches of multiagent systems with reinforcement learning (MARL). First, we focus on general application in various fields for crowd simulation. Then, we focus on works using RL for epidemic spread control situations. Finally, we give a comparative study. In Reference 13, the author proposed a method to integrate a central agent's joint action into a multiagent environment. The key idea was to use tate-aware online principal component analysis to embed the action into a space of reduced dimension using an encoding-decoding algorithm. This technique enabled the agent to perform better in terms of both performance and scale than an agent without embedding. It has made the learning in a lower dimension of joint action policies, which can be decoded back to the original space. Using a simulation system for the wind farms, they showed that a combination of Deep Q-Learning and embedding action can achieve faster convergence.

Fetzer et al.[7] suggested a method of route planning, for the evacuation of crowds, based on the learning of multiagent reinforcement, where the crowd has grouped and the leader picked. To store empirical knowledge about the learning process, a bulletin board had introduced to the multiagent reinforcement-learning algorithm, and the navigation agent transferred information between the leader and this bulletin board.

Through incorporating visual parameters, Wang et al.[14] enhanced the original concept of the social force model (SFM). In fact, for reinforcement learning (Q learning algorithm) the intersection of the pedestrian trajectory extracted from the real video has first used as the State space. This method employed a two-layer control mechanism, the upper layer leader used the algorithm-based decision process to select the path, and the bottom group individuals used the improved model of social force to evacuate.

Pageaud et al.[15] have proposed a multiagent, multilevel solution called clustered Deep Q-Network (CDQN) to resolve both problems, experience replay in nonstationary environments and credit environments and credit assignment issues, with a hierarchical approach where high-level agents handle low-level learning agent clusters and organize them efficiently to enhance urban policies. The high-level agent population, which uses a custom trust, score assignment to manage low-level agent clusters. Low-level agents learn the concept of action-value using individual local rewards and productive replay of experiences. The multilevel, multiagent environment enables collaboration even without contact and without interaction or input from other agents at the low level.

Zhang et al. addressed the multiagent reinforcement-learning problem with networked agents in Reference 16. In particular, they considered the decentralized environment where each agent makes an environment where each agent makes individual decisions and receives local incentives while sharing information over the network with neighbors to achieve maximum average return across the network. Wu et al.[17] also proposed two decentralized actor-critic algorithms. Besides, the agents have been allowed to have different reward functions from different tasks, but each agent can only experience their reward. They were interested in the collaborative environment where the agents have a shared objective of optimizing the global average gain for all agents in the field together. Many techniques have been studied to develop a framework for controlling epidemic spread control. We present an overall summary of the major contributions already made in this section.

The method in Reference 18 presented the application of RL to develop context-dependent outbreak response policies to minimize foot-and-mouth disease outbreaks. The authors of Reference 18 demonstrated that regulation based on the subsequent context-dependent policies, which adapt interventions to the specific outbreak, results in smaller outbreaks than static policies. To convert the complex machine-readable policies into simple heuristics that can be tested by human decision-makers, they explain two strategies that use RL and MC control to establish state-dependent response policies in the sense of a livestock outbreak, based on the dynamics of the 2001 foot-and-mouth outbreak (FMD); They only considered the initial stages of an outbreak in the first case study, where the state space is relatively limited, and developed state-based RL policies using Deep Q-Learning. The goal was to end the outbreak as soon as possible, with minimal costs, defined in the action-value feature by the immediate reward. For the second case study, to control interventions (ring culling or ring vaccination), they applied reinforcement learning to summarize state space. The main aim was to reduce an FMD outbreak period and to help ensure a human-readable policy.

Khadilkar et al.[19] suggested Deep Q Network, for determining the optimal lock-down strategy for each node in a network, given the characteristics of the disease (infectiousness, gestation time, symptom length, the likelihood of death) and network properties (density, tendency to move). The macroscopic model used in this analysis is the network model. Here the network displays India and each node presents a city. The open node allows people to access to/from other open nodes in the network. The simulation of the virus propagation with SEIRD (susceptible-exposed-infected-recovered-deceased) model. They did not model real individuals and their movements).

Yanez et al.[20] paid special attention to designing environments to represent the epidemic problem. They described the various components needed to build epidemic control environments. They addressed mathematical models such as SIR and SEIR, which seek to minimize the impact of a disease spread by preventing the spread of disease, various potential state representations, and specifying the incentive feature to reduce the number of people infected during the epidemic.

In Reference 21, Liu developed a microscopic approach using Q learning algorithm to model epidemics, which can explicitly take into account the effects of individual decisions on disease spread. They proposed an epidemic model of multiagent sharing the Q function, in which each agent may be: either contaminated or susceptible. The likelihood of a state change for a multiagent scheme is also implemented according to certain assumptions. The model also had two levels of activity: normal level of activity and reduced level of activity. Liu[21] found "no action" cases where all agents tend to follow the normal level of activity and concentrated on the reduced level of activity in three cases: "immediate isolation" case where the activity rates of infected agents fell instantly to the lower level, "delayed isolation" case the level of activity of contaminated agents

drops to a reduced level after several days, and where the level of activity of all agents drops instantly to a reduced level.

Kwak et al.[22] used deep reinforcement learning to allow agents to try to find public health strategies for controlling the spread of COVID-19. The research focused only on population health benefits without considering the negative impacts of economic and social effects.

In Reference 23, authors used RL to optimize mitigation policies that minimize the economic impact without exceeding hospital capacity. They proposed a novel agent-based pandemic simulator capable of modeling fine-grained interactions between people at specific locations in a community.

Padmanabhan et al.[24] proposed a decision support system capable of incorporating different interventions to minimize the impact of widespread respiratory infectious pandemics, including the recent COVID-19. They took into account pandemic characteristics, health system parameters and socio-economic aspects.

2.1 | Comparative study

An overall analysis of the previously developed works demonstrates the effectiveness of Reinforcement Learning in modeling crowd and epidemic spread control. We compare existing works using some parameters (Table 1). In epidemic modeling, the previous works focus mostly on macroscopic models or microscopic models for multiagent methods. In addition, for crowd simulation in other fields, the most widely used methods are the centralized approach, which requires grouped crowd as in Reference 7 and a decentralized method with networked agents.[25] However, the multiagent system should respond to the social distancing behavior in an infected area. For epidemics simulation, the commonly used epidemiology model is SIR and its variations as in Reference 25 who used SEIRD model. Additionally, some other works simulated the propagation with assumptions, like Reference 21. However, the use of these models is in order to get closer to reality in the simulation of the spread of the virus.

Almost all the works dedicated to simulating an outbreak system focus on the lock-down or isolation phase. Nonetheless, the postlock-down phase and how to mitigate the spread of the pandemic merit consideration, especially when in order to maintain activities the radical isolation of the population is not applicable persistently.

3 | PROPOSED APPROACH

Throughout this section, we describe the proposed approach in detail.

3.1 | Global agent architecture

The proposed approach is based on a global agent architecture IDQN I-SIR presented in Figure 1. This architecture is composed of three main different modules:

- IDQN: Independent Deep Q Networks model: the role of this module is to control the agent movements,

TABLE 1 Comparison of previous work-related studies

Works	Goal of each work	Multiagent methods	Algorithm	Crisis situation epidemic	Case study
19	Optimizing lock-down policies for epidemic control	Macroscopic model	DQN	SEIRD	Lock-down phase
21	Model epidemic to follow the spread of the disease	Microscopic model	Q learning	Model according to certain assumptions	Normal and reduced levels of activities
20	Focus on how to design environments to represent the problem of epidemics and finding optimal interventions	No agents	No algorithm	Built epidemic control environments: SIR and SEIR	In general
18	Illustrate the application of RL to the development of context-dependent outbreak response policies to minimize outbreaks of foot and mouth disease	No multiagent systems	Deep network	The virus spread depends on probability	Outbreak period and to help ensure a human readable policy
22	Propose an architecture to control spread of COVID'19	Multiagent systems	DQN	COVID'19	In relation to the lockdown and travel restrictions
23	Optimizing damage mitigation policies; The goal is to minimize the economic impact without exceeding hospital capacity	Agents	RL	COVID'19	
24	Decision-making system; incorporate different parameters to minimize the impact of widespread respiratory infectious pandemics	Agents	RL	COVID'19 and others pandemics	

- I-SIR: Improvement of SIR model to control the virus spread,
- Final decision.

The inputs of the IDQN-ISIR architecture are a matrix of obstacles, a matrix of goals, a matrix of agents and numbers of As, Ai, Ar that are respectively: safe agents, infected agents, and recovered agents. The outputs of the architecture are action to do, Status of the agent, crowd probability, and the state.

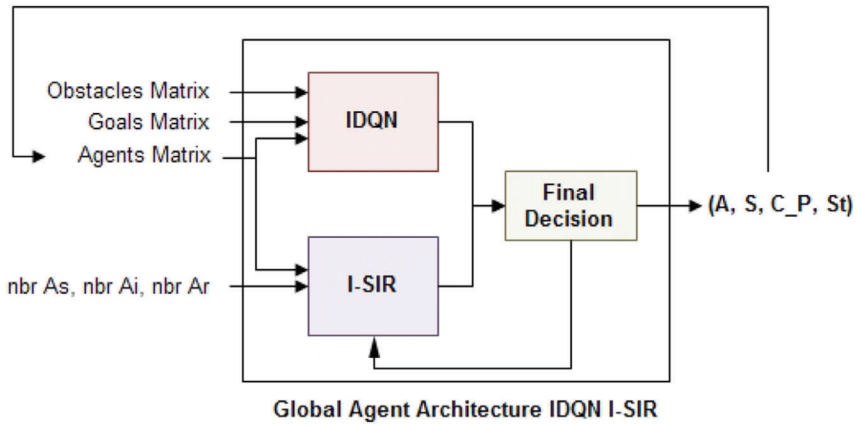

FIGURE 1 Global agent architecture IDQN I-SIR based on independent Deep Q Learning (IDQN), improved SIR model (I-SIR), and the block final decision

Each agent receives the information from outside from the environment and the moderator agent. The agent reasons according to these three modules and makes the final decision, which is the output. The modules are linked to help the agent to achieve its objective. In the next section, IDQN I-SIR architecture is presented with more details.

3.2 | Detailed agent architecture

Figure 2 shows the detailed IDQN I-SIR architecture, and we present the internal workings of each model of this architecture. The agent reasoning is based on two main blocks: IDQN and I-SIR, to find the shortest path avoiding crowds and contaminated agents.

3.2.1 | IDQN based shortest path reasoning

In this article, we used the independent Deep Q Learning for many reasons: DQN is recommended for environments that have a discrete space for action and a unique process. Besides, DQN is a practical tool for studying decentralized learning of Multiagent systems in a complex environment.[25]

For a given state, the DQN returns a vector of possible actions. These actions define the movements that the agent could take to maximize rewards. As the aim of the agent is to achieve its goal, these actions are stored in a way that the best action leads to the shortest path. The agent should find the shortest path to the goal considering the obstacles and other agents (safe, infected, or recovered) in the environment.

The neural network receives the state of the agent at t. After the process of IDQN, the outputs are a set of Q values of $Action_i$ that takes it to another $State_i$ at $t + 1$. This result of the IDQN model is then injected into the ISIR Model to evaluate the impact of the current State i on the spread contamination.

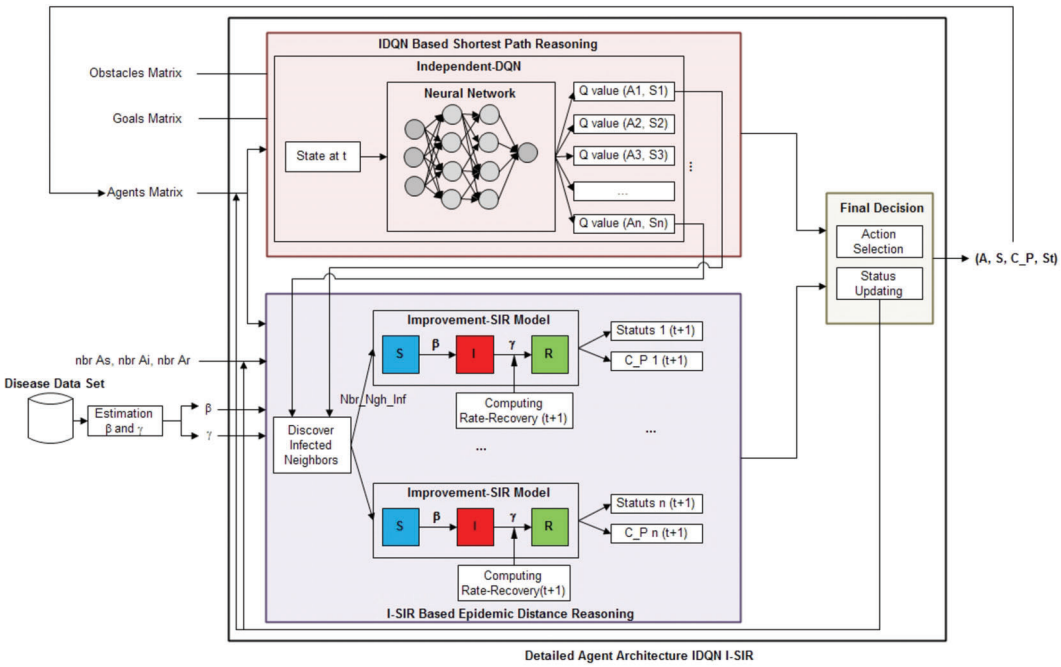

FIGURE 2 Detailed agent architecture IDQN I-SIR with detailed IDQN based shortest path reasoning, I-SIR based epidemic distance reasoning, and final decision to select action and update status

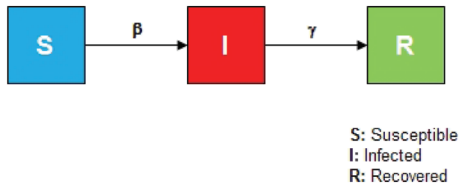

FIGURE 3 SIR model chart flow[26]

3.2.2 | I-SIR based epidemic distance reasoning

The SIR or the Kermack–Mckendrick-model is our reference in this article (Figure 3). SIR Model aims to explain the rapid rise and fall in the number of infected patients observed in epidemics. It assumes that the population size is fixed, an incubation period of the infectious agent is instantaneous, and the duration of infectivity is the same as the length of the disease. The studied population is divided into three compartments labeled S, I, and R. Susceptible individuals (S) become infected through contact with infectious individuals (I), and infectious individuals can recover (R) at a fixed rate.[26] A system of ordinary differential equations of the SIR model is given by these equations[26] (1):

$$\frac{dS}{dt} = -\beta SI; \frac{dI}{dt} = \beta SI - \gamma I; \frac{dR}{dt} = \gamma I. \tag{1}$$

The SIR model has two variables: the transmission rate β and the recovery rate γ.

An individual has on average β interacts with randomly chosen others will be infected. On the other hand, the recovery rate γ indicates infected individuals will recover or die from a set average rate γ.

In the present article, we improve the cited model to be more reliable to the COVID'19 case by introducing an infection rate I (I-SIR). The transmission process is based on the number of the agent's neighbors and their status (S, R, or I). That is to say, an agent is infected if the sum of infection rates of its neighbors exceeds a threshold.

Moreover, the traditional SIR model neglects the time-varying property of γ (to move from infected status to recovered status). Therefore, we propose a period of rehabilitation as a function of time. Therefore, the agent will recover over time as in reality.

In I-SIR model, the inputs are from the output of IDQN. For every Q value (A_i, S_i) of an agent, I-SIR searches the infected neighbors around it to compute the output status i at (t+1) and the crowd probability at (t+1). So at each move, an agent must know the status of its neighbors and the crowd probability.

3.2.3 | Final decision

The final decision is based on the results of the two models IDQN and I-SIR. In the proposed module, an agent must choose the best action. IDQN gives all the possible actions, I-SIR calculates the best action that leads to an environment with less contamination. After using the best action, the agent status will be updated. The outputs of this model are action, status, crowd probability, and state.

3.3 | Multiagent system architecture

Our system is composed of many agents. Each agent has its architecture to achieve its goal. Figure 4 is a global multiagent system architecture. It consists of an agent IDQN I-SIR and a moderator agent.

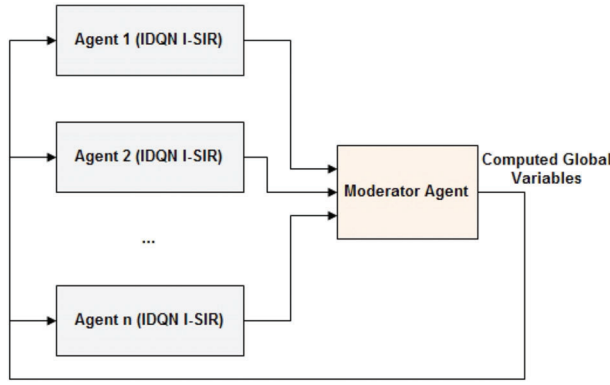

FIGURE 4 Multiagents system architecture IDQN I-SIR; agents with IDQN I-SIR architecture and moderator agent to moderate global variables

The role of the moderator is to receive the outputs from other agents and reinject the necessary information to help agents to get the right information, which means shared global variables. For example the number of A_s, A_i, and A_r. The moderator agent calculates the global variables such as agent state and crowd probability to be used by other agents.

3.4 | RL environment design

The environment design is one of the most crucial parts when experimenting with RL. The main function of the environment is to simulate the inputs; an agent would receive and modify itself according to the outputs of this agent.

Many reinforcement-learning libraries contain simulations and game environments to train reinforcement learning-based agents. However, we did not find an environment that satisfies our objective. Therefore, we built our environment that has the same structure as OpenAI gym[27] environments, so it can be used easily in the same manner.

3.4.1 | The general environment design

For the intents and purposes of this article, we aimed for a minimalist and efficient environment, that of an emergency department. The design consists of a limited two dimensional square grid-based environment in which the agent must perform its task while interacting with the different elements of the environment. Figure 5 represents the environment used for our method.

Each tile in the grid contains exactly zero or one object, and the agents can only be on an empty tile or a tile containing a goal. Each object has an associated discrete color. The agents A are blue, the goals G are in green and the obstacles X are yellow. The positions of agents are random.

3.4.2 | Process for epidemic control

We make use of the traditional SIR model, just to estimate the values of β and γ with a real date.

Process for epidemic control

Suppose there are N agents in the environment. Initially n_0 agents are infected. The agents are indexed from 1 to N. Each agent A_i has its own:

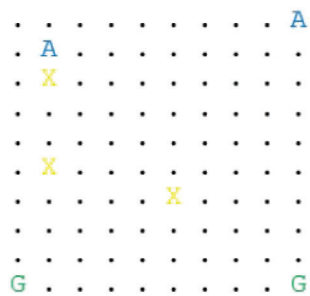

FIGURE 5 Grid environment two-dimensional 10*10

- State S_i: $S_i = 0$ healthy (susceptible), $S_i = 1$ infected, and $S_i = 2$ recovered,
- Infection rate β_i,
- Recovery rate γ_i, initialized to zero and once the agent is infected this rate is updated each time step in order to be more realistic.

The model is evaluated in discrete time. The time interval is set to be one day (same as the COVID-19 dataset). The evolution of the infection rate for consecutive days depends on agents' actions. We define the state transition probability for an agent as follows:

- $P(S_i(t+1) = 1 | S_i(t) = 0) = \sum_{j \in J} \beta_j > \beta$,
- $P(S_i(t+1) = 2 | S_i(t) = 1) = \gamma_i(t+1) > \gamma; \gamma_i(t+1) = \epsilon + \gamma_i(t)$,

where J is a set of infected neighbors according to a defined perimeter. β and γ are the optimal values explained in the next paragraph.

In other words, a "healthy" agent, at step t, transit to the infection state if the sum of infection rate of his infected neighbors is higher than β. In addition, an infected agent can recover if his recovered rate achieved γ.

Estimation of the optimal values for β and γ

For the simulation with COVID-19 data, we have used the data source available in Reference 28 the Center for Systems Science and Engineering at Johns Hopkins University. It offers us the number of cases: confirmed, deaths, and recovered. The initial uninfected population should be equal to the number of agents in the simulation environment.

To fit the model to the data we need two things:

- A solver for the differential equations,
- An optimizer to find the optimal values for our two unknown parameters.

In fact, we can fit the SIR model to our data by finding the values for β and γ that minimize the residual sum of squares between the observed cumulative incidence (from dataset) and the predicted cumulative incidence (predicted by our model). The loss function used in the optimization process was the root mean squared error (RMSE).

3.4.3 | Action space-state space

Each IDQN I-SIR agent has five possible actions to perform, namely: UP, DOWN, LEFT, RIGHT, clearly to move up, move down, move left and move right, and STAY when any other action leads to an obstacle or the limits of the environment, the agent did not move but it is still considered as an executed step. Each position in the environment is considered as a state.

3.4.4 | Reward function

What makes reinforcement learning versatile and powerful is the reward philosophy. Therefore, the reward system design acquires great relevance and importance.

After completing a step, each agent receives a reward signal that is computed based on the outcome of its last action. However, providing an individual reward function can slow down the learning process and undermine the performance of the decentralized MARL system as the individual rewards that agents receive do not reflect the performance of the other agents involved in the task. To address this issue, the rewards that the agents receive at each step are full informative team rewards.

Thus, we consider a reward function (2) that is made up of three main terms:

$$R_{total} = R_{obs} + R_{dist} + R_{virus}, \tag{2}$$

where:

- R_{obs} is a constant penalization given in situations when an agent is in front of an obstacle, to tell him not only to avoid that obstacle but to recognize his place also.

$$R_{obs} = \begin{cases} -10 & \text{if an agent-obstacle overlay} \\ 0 & \text{else} \end{cases}.$$

- R_{dist}: at each time step, all the agents receive a penalty of:

$$R_{dist} = -\sum_{a=1}^{n} min_{g \in G} ||pos_a^t - pos_g||_1, \tag{3}$$

  where: n = number of agents.
  G = goals set.

  With this penalty, each agent will follow the path with the minimum distance so the shortest path.

- R_{virus} this reward would control the epidemic spread with the following equation:

$$R_{virus} = -10N_{inf}^t + 10N_{rec}^t, \tag{4}$$

  where N_{inf}^t and N_{rec}^t are respectively the number of affected agents and the number of recovered agents at time step t.

With this shared reward function, the performance of each agent in the system contributes to the reward that they get after acting. Thus, it is in their best interest if all the agents receive higher as it directly affects their reward signals.

3.5 | Multiagent setting

Due to the complexity and dimension of the state space in this problem, we needed DRL (deep reinforcement learning). Therefore, we trained the agents in the environment described above using the independent Deep Q-Learning (IDQN). Each agent is equipped with two neural networks, online and target, to approximate its own state-action Q-function and a memory to store the experiences. The agents are trained in prioritized experience replay (PER) memory. In

addition to IDQN, we considered independent Double Q-learning (IDDQN) and dueling IDDQN, to compare these three algorithms.

3.5.1 | Structure of the MLP

A multilayer perceptron (MLP) is the most typical feedforward artificial neural network (ANN). An MLP consists of at least three layers of nodes: an input layer, a hidden layer and an output layer. The only parts that require some more careful thought are how we will represent our input and out layer, because the hidden layer is designed with the trial and error method until its performance is good enough.

The input layer of the MLP

The independent agents have to use their observation of the environment to create the input layer of the multilayer perceptron. To shape the input layer, the agents have to process their observation matrix first, as in Figure 6, and create three matrices, each representing one of the three components involved in the task.

The three matrices are created for goals, agents, and obstacles by encoding the observed matrix such that each cell occupied by the respective component is encoded as one and all other cells are encoded as 0. Therefore with these three matrices, the agent knows the position of the other components and the empty tiles on the board. The matrices are flattened and used for constructing the input layer by the MLP. The agents also receive their (x,y) coordinates on the board, and use two nodes to represent them.

Figure 7 illustrates an example of the process of creating the input layer of the MLP used for a decentralized MARL agent given an arbitrary observation matrix.

The output layer of the MLP

A neural network will approximate, given a state, the different Q-values for each action, so, the last layer will simply produce an output vector of Q-values, one for each possible action. Therefore, the output layer is determined by the number of actions the agent can perform, in our case five actions (UP, DOWN, LEFT, RIGHT, STAY). For the dueling DQN, we consider two streams of fully-connected layers where the first stream has a single output for each possible action, while the second stream has one single output. The outputs of these two streams are then combined.

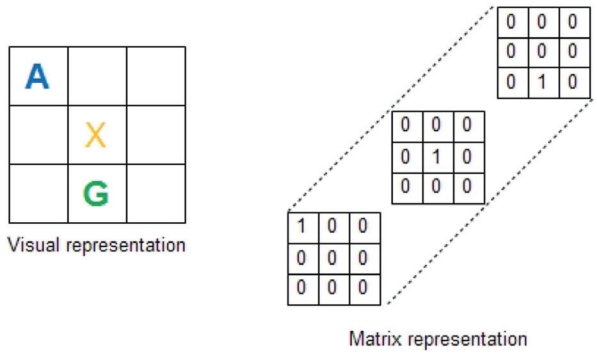

FIGURE 6 Visual representation versus matrix representation

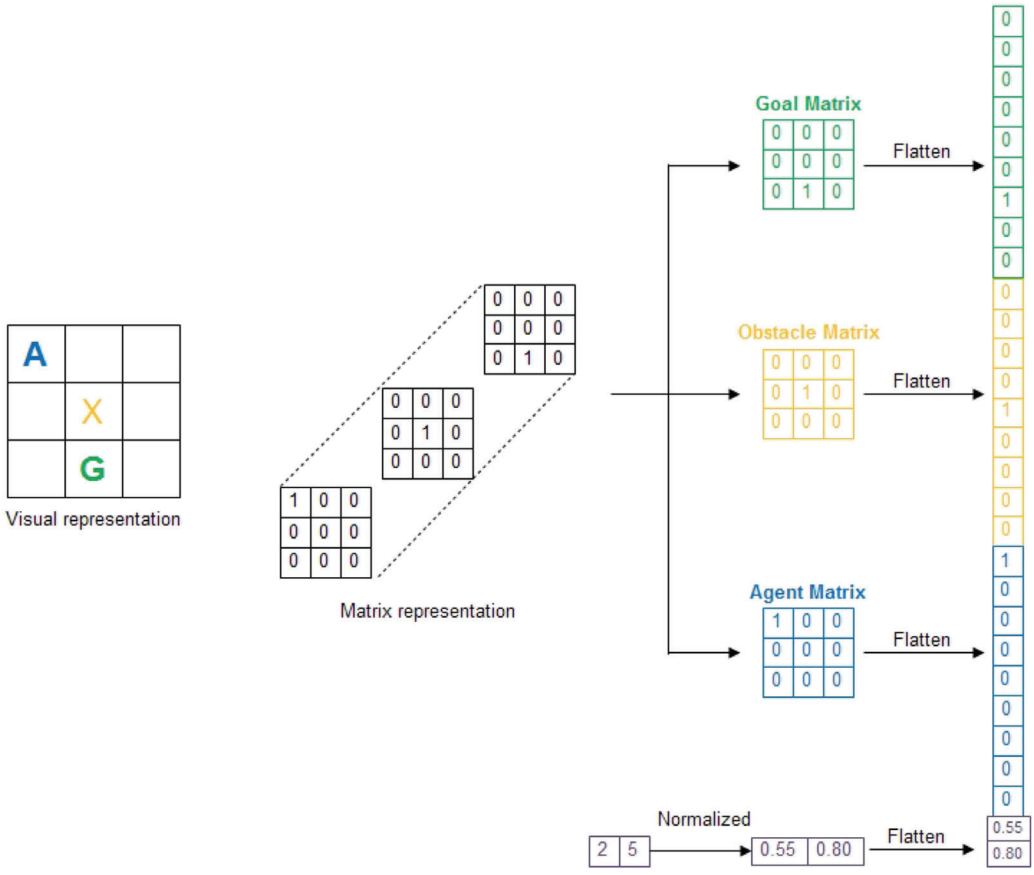

FIGURE 7 The process of creating an input layer of the MLPs

Current State	Action	Reward	Next State	Terminal	Priority

FIGURE 8 Structure of transition sets stored in the experience replay memory

3.5.2 | Experience replay

For the sake of the agent's learning process, a prioritized experience replay[29] was designed. The information stored in the memory as a transition consists of all the information related to the agent's current state and next state. This information is the current state, the action taken to transition from the current state to the next state, the reward obtained due to performing this transition, the next state, a value determining if the simulation has terminated or not, and the assigned priority. Figure 8 shows a simple representation of it.

3.5.3 | Simulation cycle

The process that the system performs to simulate an episode is described in Figure 9. The simulation cycle is executed as many times as the number of episodes set for an experiment.

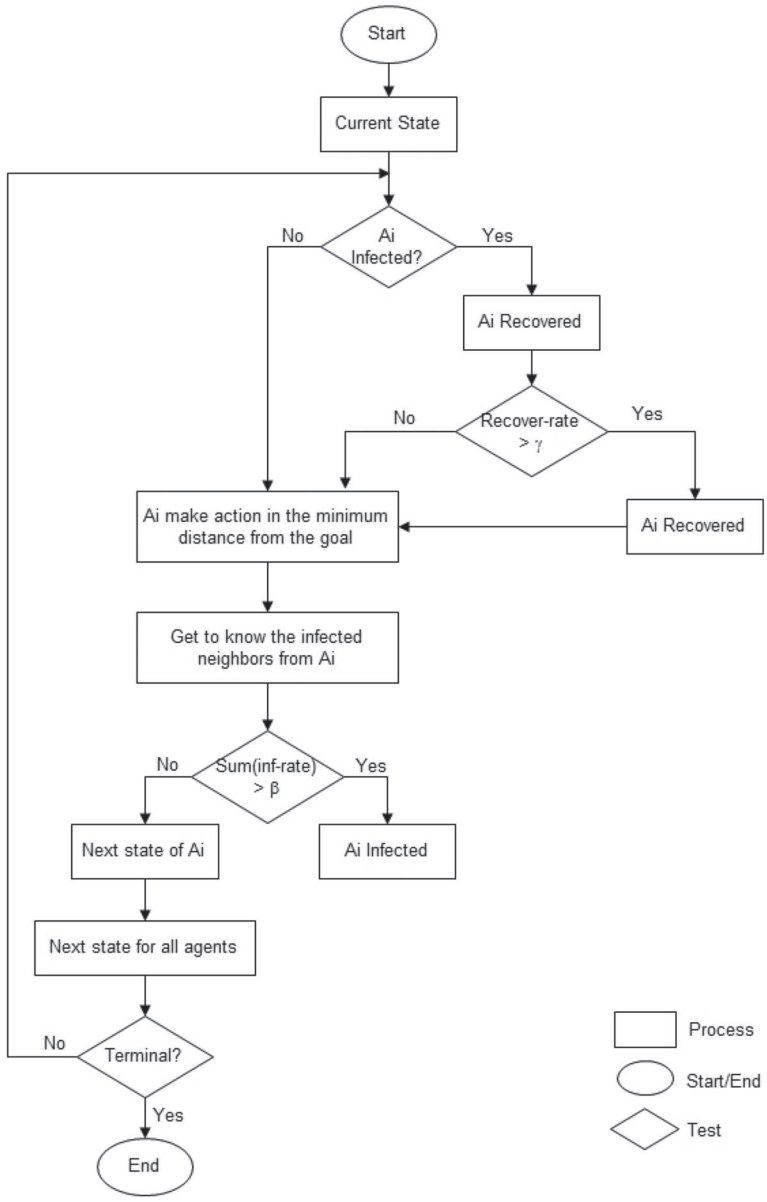

FIGURE 9 Navigation agent decision flow chart

4 | EXPERIMENTS AND RESULTS

The case study is to simulate the propagation of the virus in the environment case of the emergency department where agents take precautions (avoid a collision; select the path with minimal steps). The objective is to validate our architecture in a real case.

The simulation environment is a grid of size 10*10. As a first step, we have chosen five agents. After getting β and γ values, we train our model on the environment including one infecting agent and four "healthy" agents. In order to demonstrate how this infected agent could contaminate the other agents; we exploit the number of Susceptible, Infected, and recovered agents in each episode

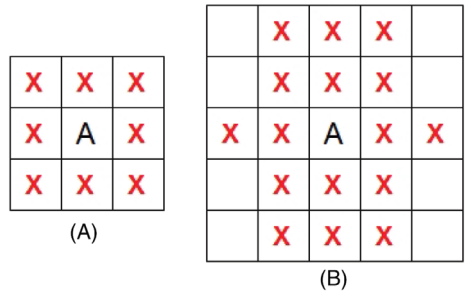

FIGURE 10 (A) Agent with eight neighbors, (B) agent with 16 neighbors

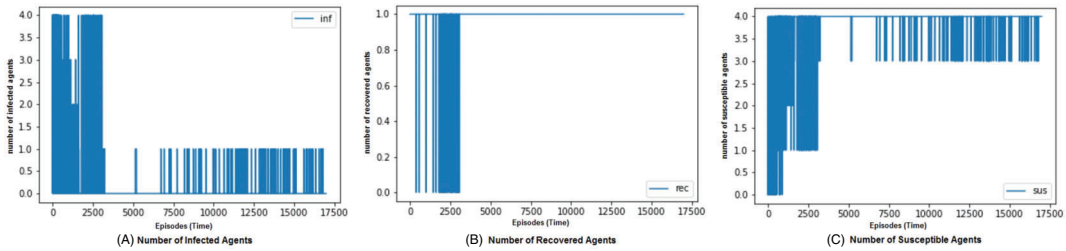

FIGURE 11 Simulation result based on the first scenario; an agent with eight neighbors. The figures show the number of agents according to time; (A) shows the number of Infected agents, (B) illustrates the number of Recovered agents, and (C) for susceptible agents. As a result, the number of infected agents decreases with time

during the training. As discussed in our model, the spread of the virus depends on the agents' neighbor, which is why we carried out two experiments where the perimeter of considering that one agent is neighboring another is different. In the first scenario, we consider that an agent could be infected by his eight neighbors, as in Figure 10A. With the second experiment, we expand the neighborhood radius to 16 as in Figure 10B.

As a result, Figures 11 and 12 present, for each case, the number of infected, recovered, and susceptible agents per episode. These figures reveal that the first episodes are characterized by a large number of contaminated agents, a low number of susceptible agents, and no recovered agents. Nevertheless, at the end of the training, there was only one infected agent per episode. These findings prove how effective our approach is since the agents learned to avoid infected agents. Comparing the two scenarios, the number of infected increases considering a larger perimeter.

Figures 13 and 14 show the evolution of the number of infected agents between the first and the second scenario respectively.

IDQN versus IDDQN versus IDuelingDQN

To better evaluate our approach, we compare the efficiency of the different algorithms Deep Reinforcement Learning in the case of crowd crisis: independent DQN (IDQN), independent double DQN (IDDQN), and independent dueling DQN (IDuelingDQN). IDuelingDQN:[30,31] is a network architecture from the original DQN that decouples the Q-value estimation in two streams. One stream estimates how good it is to be in state V(s) and the other stream estimates the advantage of

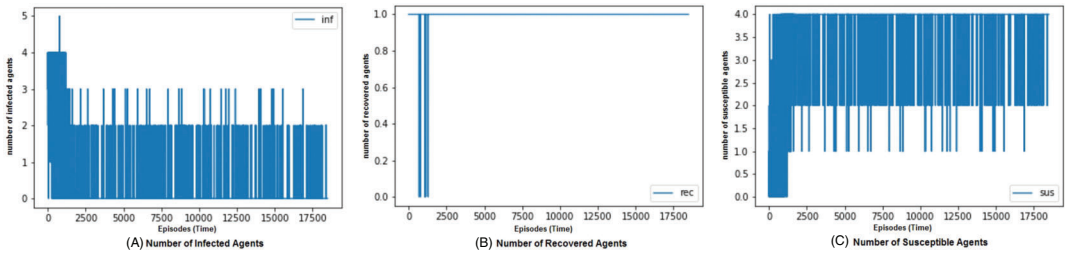

FIGURE 12 Simulation result based on the second scenario; an agent with 16 neighbors. The figures show the number of agents according to time; (A) shows the number of Infected agents, (B) illustrates the number of Recovered agents, and (C) for susceptible agents. As a result, the number of infected agents decreases with time

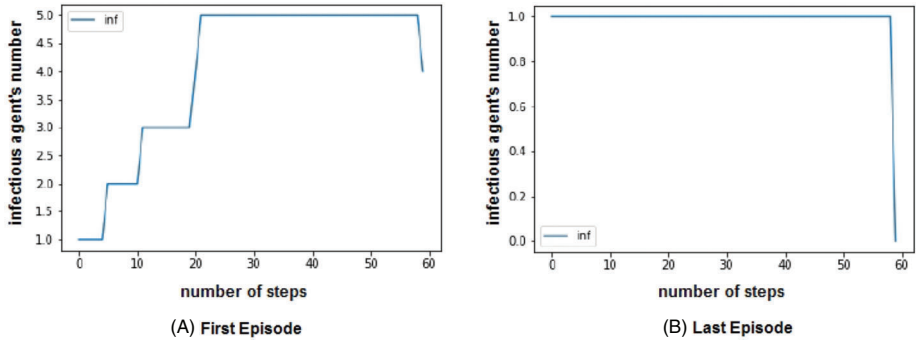

FIGURE 13 (A,B) show the number of infectious agents from the first episode to the last one. These results are those of the first simulation scenario—agent's neighbors are eight. At the end of the simulation, and applying the proposed architecture, the number of infected agents has decreased, and agents have reached their exit gate

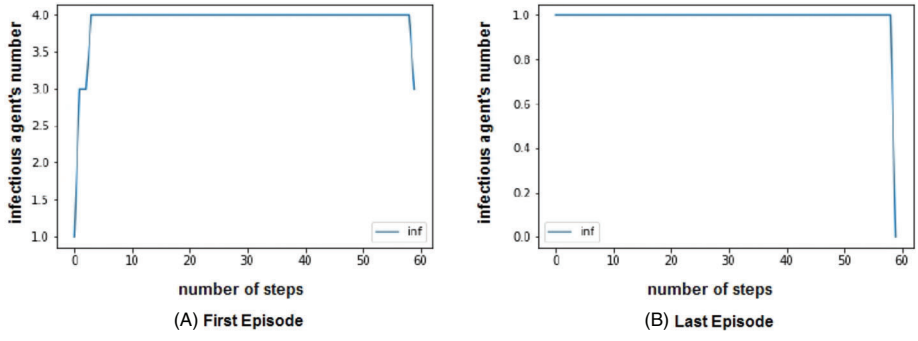

FIGURE 14 (A,B) show the number of infectious agents from the first episode to the last one. These results are those of the first simulation scenario—agent's neighbors are 16. At the end of the simulation, and applying the proposed architecture, the number of infected agents has decreased, and agents have reached their exit gate

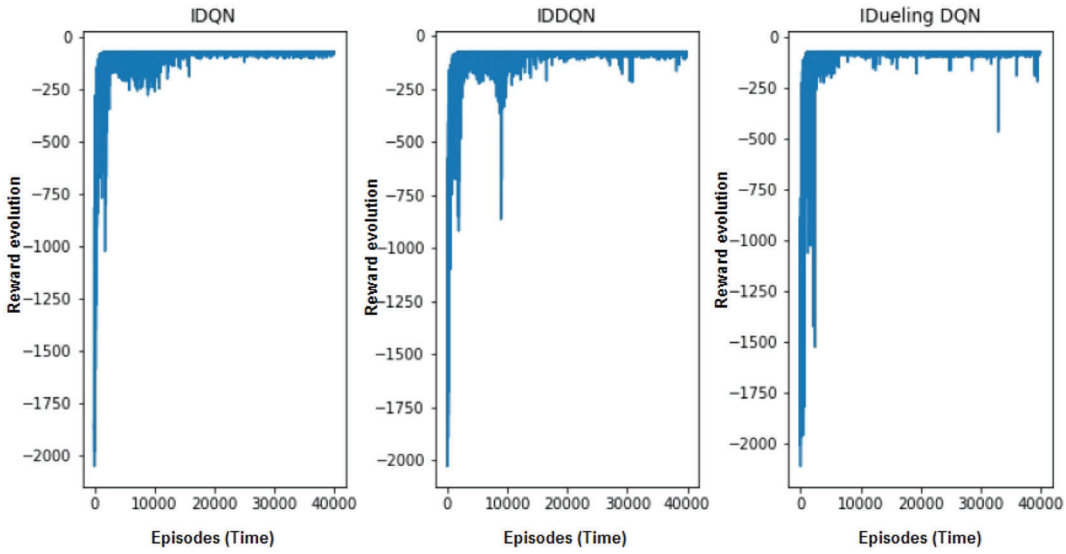

FIGURE 15 Reward evolution over time of the proposed approach for each algorithm: independent DQN (IDQN), independent double DQN (IDDQN), and independent dueling DQN (IDuelingDQN)

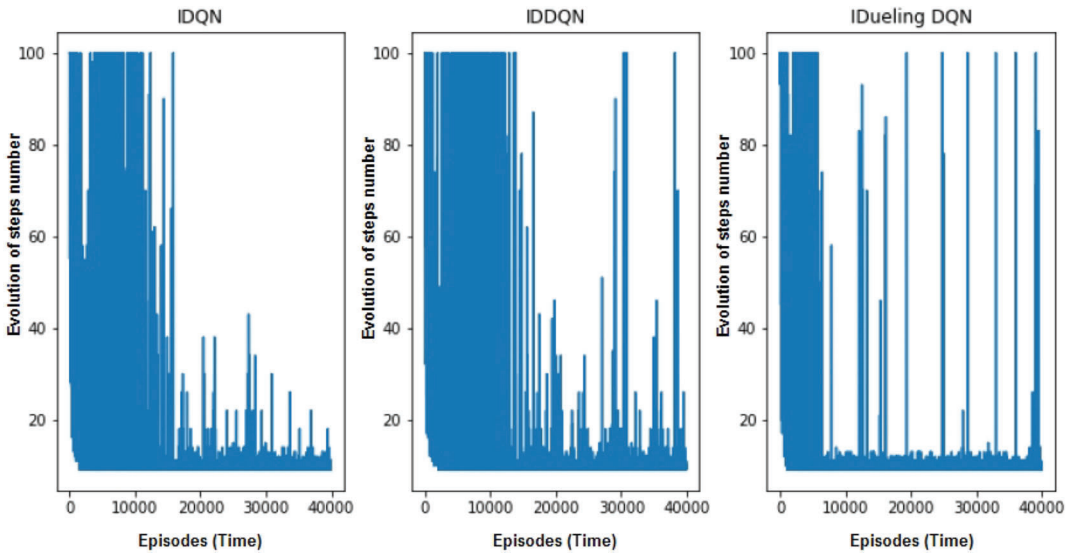

FIGURE 16 Steps number evolution over time of the proposed approach for each algorithm: independent DQN (IDQN), independent double DQN (IDDQN), and independent dueling DQN (IDuelingDQN)

TABLE 2 Performance according to each algorithm

	IDQN	IDDQN	IDuelingDQN
Max-reward	−72	−72	−72
Average-reward	−90.22	−90.34	−89.73
Min-reward	−2049	−2028	−2110
Min-steps	9	9	9
Average-steps	12.09	11.91	11.25
Episodes with missed goals	553	486	554
Episodes with goals achieved in minimal steps	30,610	30,872	32,467

taking action in that state A(s,a). Then the two streams will be combined through a special aggregation layer to get an estimate of Q(s,a). The outcome of that architecture is that the state value can be learned separately, without getting confused by the influence of the action advantage.

The three algorithms were evaluated in the same configuration and complexity of environment; 10×10 grid, two agents, and two goals. As a result, Figure 15 represents the reward evolution, and Figure 16 represents the evolution of the number of the steps. Table 2 shows more metrics to make the comparison.

The first observation is that with the independent dueling DQN algorithm, the agents obtained the highest reward and achieved the goals in a minimal number of steps in fewer episodes. Unlike the other algorithms where the agents had more than 10,000 episodes in order to reach the goals with the same steps. However, we can notice that with the independent dueling DQN algorithm even though the agents achieved the goals in minimal steps, they kept making further steps as in the independent double DQN algorithm. However, the independent DQN agents, after several episodes making the maximum number of steps, succeed in making fewer steps.

These findings can be explained by the difference between the architectures of the algorithms. DQN agents do not have enough knowledge about the best action to take and take the full Q value at the beginning of the training. As a result, certain suboptimal actions were obtaining higher values so the time to learn optimal policy increased, which is why in the first episodes we observed that these agents took more effort. However, the dueling DQN agents learn the state value separately, without getting confused by the influence of the action advantage. For this reason, we found that dueling DQN agents have tried many steps in the last episodes even when they founded the best actions from the beginning of the training.

5 | CONCLUSION

In this article, a decision-making architecture in a crisis case of COVID-19 was presented. The architecture integrates reinforcement-learning methodology. The environment was created in a grid form with some obstacles to make the task more complex. The basis of this process is the Q learning algorithm combined with the approximation of the Q function by a multilayer perceptron. To control the spread of the virus we used the SIR mathematical model with some improvements such as taking into account neighbors in the virus transmission and using a real data set to obtain real parameters to model the spread of the virus more realistically.

Experiments results, in the environment of an emergency department, prove that DQN gives us interesting results to take the shortest path and control the spread of the virus. The number of infected persons at the end is very negligible. These results are the mix between different parameters of DQN, I-SIR model, and the crowded parameter.

With this approach we have given a reasoning model to move in a place avoiding obstacles based on IDQN. The obstacles can be objects or infected agents. Our decision making approach used the SIR model which can be applied to any pandemic. It is a way to control the spread of the virus in the postcontainment phase. It is a way to control the spread of the virus in the postcontainment phase.

With the proposed system and sufficient computing power and time, we suggest increasing the complexity of the environment. It is interesting to completely change the configuration of the world and to use new configurations that can give more importance to the multiplicity of objectives in the environment. For example, we can build a maze with several entry points along the boundaries of the grid so that agents can start their process using the layout of the obstacles. Depending on the starting point of the agents, the difficulty of reaching each entry point will not only be based on distance but also determined by the challenge of moving toward them in the maze.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in CSSEGISandData at https://github.com/CSSEGISandData/COVID-19.

ORCID

Wejden Abdallah https://orcid.org/0000-0002-5102-6003

REFERENCES

1. Diao Y, Kodera S, Anzai D, Gomez-Tames J, Rashed EA, Hirata A. Influence of population density, temperature, and absolute humidity on spread and decay durations of COVID-19: a comparative study of scenarios in China, England, Germany, and Japan. *One Health*. 2021;12:100203.
2. Buhat CAH, Torres MC, Olave YH, et al. A mathematical model of COVID-19 transmission between frontliners and the general public. *Netw Model Anal Health Inform Bioinform*. 2021;10(1):1-12.
3. Aburto JM, Kashyap R, Schöley J, et al. Estimating the burden of the COVID-19 pandemic on mortality, life expectancy and lifespan inequality in England and Wales: a population-level analysis. *J Epidemiol Commun Health*. 2021.
4. Coccia M. The relation between length of lockdown, numbers of infected people and deaths of Covid-19, and economic growth of countries: Lessons learned to cope with future pandemics similar to Covid-19 and to constrain the deterioration of economic system. *Sci Total Environ*. 2021;775:145801.
5. Valvo PS. A bimodal lognormal distribution model for the prediction of COVID-19 deaths. *Appl Sci*. 2020;10(23):8500.
6. Jarndal A, Husain S, Zaatar O, Al Gumaei T, Hamadeh A. GPR and ANN based prediction models for COVID-19 death cases; 2020:1-5; IEEE.
7. Fetzer T, Hensel L, Hermle J, Roth C. Coronavirus perceptions and economic anxiety. *Rev Econ Stat*. 2020;1-36.
8. Ronchi E, Lovreglio R. EXPOSED: an occupant exposure model for confined spaces to retrofit crowd models during a pandemic. *Saf Sci*. 2020;130:104834.
9. Zhao Y, Liu H, Gao K. An evacuation simulation method based on an improved artificial bee colony algorithm and a social force model. *Appl Intell*. 2021;51(1):100-123.
10. Xiang N, Zhou Z, Pan Z. Using SIR model to simulate emotion contagion in dynamic crowd aggregation process. *Int J Performability Eng*. 2018;14(1):134.
11. Mao Y, Yang S, Li Z, Li Y. Personality trait and group emotion contagion based crowd simulation for emergency evacuation. *Multimed Tools Appl*. 2020;79(5):3077-3104.

12. Zia K, Ferscha A. An agent-based model of crowd evacuation: combining individual, social and technological aspects; 2020:129-140.
13. Shah T. State aware principal action space embedding for centralized MARL; 2020.
14. Wang Q, Liu H, Gao K, Zhang L. Improved multi-agent reinforcement learning for path planning-based crowd simulation. *IEEE Access*. 2019;7:73841-73855.
15. Pageaud S, Deslandres V, Lehoux V, Hassas S. Multiagent learning and coordination with clustered Deep Q-Network; 2019:2156-2158.
16. Zhang K, Yang Z, Liu H, Zhang T, Basar T. Fully decentralized multi-agent reinforcement learning with networked agents; 2018:5872-5881; PMLR.
17. Wu Y, Mansimov E, Liao S, Grosse R, Ba J. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation; 2017. arXiv preprint arXiv:1708.05144.
18. Probert WJ, Lakkur S, Fonnesbeck CJ, et al. Context matters: using reinforcement learning to develop human-readable, state-dependent outbreak response policies. *Philos Trans R Soc B*. 2019;374(1776):20180277.
19. Khadilkar H, Ganu T, Seetharam DP. Optimising lockdown policies for epidemic control using reinforcement learning. *Trans Ind Nat Acad Eng*. 2020;5(2):129-132.
20. Yanez A, Hayes C, Glavin F. Towards the control of epidemic spread: designing reinforcement learning environments; 2019:188-199.
21. Liu C. A microscopic epidemic model and pandemic prediction using multi-agent reinforcement learning; 2020. arXiv preprint arXiv:2004.12959.
22. Kwak GH, Ling L, Hui P. Deep reinforcement learning approaches for global public health strategies for COVID-19 pandemic. *PLoS One*. 2021;16(5):e0251550.
23. Kompella V, Capobianco R, Jong S, et al. Reinforcement Learning for Optimization of COVID-19 Mitigation policies; 2020. arXiv preprint arXiv:2010.10560.
24. Padmanabhan R, Meskin N, Khattab T, Shraim M, Al-Hitmi M. Reinforcement learning-based decision support system for COVID-19. *Biomed Signal Process Control*. 2021;68:102676.
25. Zhang K, Yang Z, Başar T. Decentralized multi-agent reinforcement learning with networked agents: recent advances; 2019. arXiv preprint arXiv:1912.03821.
26. Blackwood JC, Childs LM. An introduction to compartmental modeling for the budding infectious disease modeler. *Lett Biomath*. 2018;5(1):195-221.
27. Tensor flow website, Google Brain team; 2020. Accessed June 15, 2020. https://www.tensorflow.org/
28. Novel Coronavirus (COVID-19) cases. Accessed June 16, 2020. https: //github.com/CSSEGISandData/COVID-19
29. Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay; 2015. arXiv preprint arXiv:1511.05952.
30. Sewak M. *Deep Reinforcement Learning*. Springer; 2019.
31. Wang Z, Schaul T, Hessel M, Hasselt H, Lanctot M, Freitas N. Dueling network architectures for deep reinforcement learning; 2016:1995-2003; PMLR.

How to cite this article: Abdallah W, Kanzari D, Sallami D, Madani K, Ghedira K. A deep reinforcement learning based decision-making approach for avoiding crowd situation within the case of Covid'19 pandemic. *Computational Intelligence*. 2022;38(2):416-437. doi: 10.1111/coin.12516