Multimedia Tools and Applications (2022) 81:39945–39961
https://doi.org/10.1007/s11042-022-13000-0

Effective deep Q-networks (EDQN) strategy for resource allocation based on optimized reinforcement learning algorithm

Fatma M. Talaat[1]

Received: 28 September 2021 / Revised: 12 January 2022 / Accepted: 27 March 2022 /
Published online: 5 May 2022

Abstract
The healthcare industry has always been an early adopter of new technology and a big benefactor of it. The use of reinforcement learning in the healthcare system has repeatedly resulted in improved outcomes.. Many challenges exist concerning the architecture of the RL method, measurement metrics, and model choice. More significantly, the validation of RL in authentic clinical settings needs further work. This paper presents a new Effective Resource Allocation Strategy (ERAS) for the Fog environment, which is suitable for Healthcare applications. ERAS tries to achieve effective resource management in the Fog environment via real-time resource allocating as well as prediction algorithms. Comparing the ERAS with the state-of-the-art algorithms, ERAS achieved the minimum Makespan as compared to previous resource allocation algorithms, while maximizing the Average Resource Utilization (ARU) and the Load Balancing Level (LBL). For each application, we further compared and contrasted the architecture of the RL models and the assessment metrics. In critical care, RL has tremendous potential to enhance decision-making. This paper presents two main contributions, (i) Optimization of the RL hyperparameters using PSO, and (ii) Using the optimized RL for the resource allocation and load balancing in the fog environment. Because of its exploitation, exploration, and capacity to get rid of local minima, the PSO has a significant significance when compared to other optimization methodologies.

Keywords Artificial Intelligence · Machine Learning · Reinforcement Learning · Critical care · Decision support systems · Particle Swarm Optimization (PSO) · Resource Allocation

✉ Fatma M. Talaat
fatma.nada@ai.kfs.edu.eg

[1] Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh, Egypt

1 Introduction

Owing to the high prevalence of complex diseases and dynamic shifts in patients' clinical conditions, clinical processes are dynamic in the health care domain. Using rule-based procedures defined by physicians based on evidence-based clinical recommendations or best practices, current treatment recommendation systems are enforced [2]. Furthermore, several comorbid conditions may not be considered in these protocols and guidelines [5]. Critically ill patients will benefit from deviation from existing treatment guidelines in an intensive care unit (ICU) and the personalization of patient care using means not based on rules [13].

Reference can be made to randomized controlled trials (RCTs), systematic evaluations, and meta-analyses if doctors need to change care for particular patients. However, RCTs may not be available or definitive for many ICU conditions. It is also possible that many patients admitted to ICUs may be too ill for inclusion in clinical trials [15]. In addition, just 9% of the ICU therapy guidelines are based on RCTs [17], and the vast majority of critical care RCTs have negative findings [28]. We need other approaches, including the use of large observational data sets, to help clinical decisions in ICUs [1]

Artificial intelligence (AI) systems [39] (using machines to simulate human cognitive functions) and Machine Learning (ML) techniques can then be fed a vast amount of information (using computer algorithms to perform clinical tasks without the need for explicit instructions). Diagnosis [21], treatment, and resource management [4] in the ICU can then be supported by AI and ML. One ML methodology, called Reinforcement Learning (RL) [22], is especially appropriate for ICU settings considering the complex nature of critically ill patients.

The main objective of the RL algorithm [7] is to train an agent that can maximize the cumulative future reward from the state-action pairs given the patients' state-action trajectories. When a new state is observed [11], the agent can perform an action, which could choose the action for the greatest long-term outcome (eg, survival). When the RL agent is well-trained [8], it is possible to pick the best action given the state of a patient, and we describe this process as acting according to an optimal policy [37].

We can consider the state as the well-being/condition of a patient [18]. The state of the patients could depend on static traits (eg, patient demographics including age, gender, ethnicity, pre-existing comorbidity) and longitudinal measurements (eg, vital signs, laboratory test results). An action is a treatment or an intervention that physicians do for patients (eg, prescription of medications and ordering of laboratory tests) [27]. The transition probability is the likelihood of state transitions, and it is viewed as a prognosis. If the well-being in the new state is improved, we assign a reward to the RL agent, but we penalize the agent if the patient's condition worsens or stays stagnant after the intervention.

This paper presented two main contributions, (i) Optimization of the RL hyperparameters using PSO, and (ii) Using the optimized RL for the resource allocation and load balancing in the fog environment. Figure 1 illustrates the block diagram of the proposed strategy. The main goal of the system is to achieve a low latency while improving the Quality of Service (QoS) metrics such as (the allocation cost, the response time, bandwidth efficiency, and energy consumption). Unlike other resource allocating techniques, ERAS employs a deep RL algorithm in a new manner. Accordingly, ERAS is a suitable algorithm in the case of real-time systems in FC which leads to load balancing.

The rest of the paper is organized as follows: Section 2 gives a background for some basic concepts of the proposed strategy. Section 3 introduces some of the recent previous efforts in

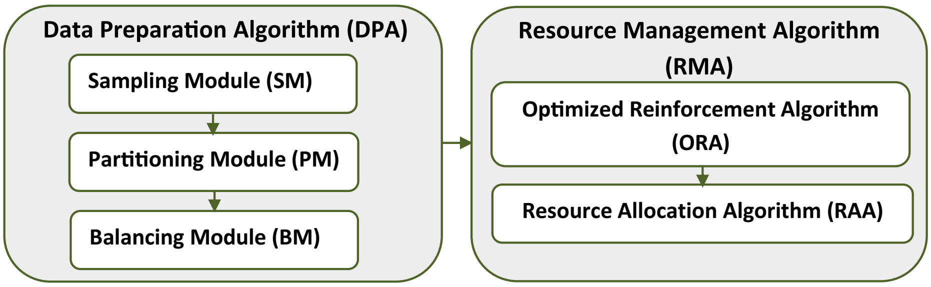
Data Preparation Algorithm (DPA)
Sampling Module (SM)
Partitioning Module (PM)
Balancing Module (BM)
Resource Management Algorithm (RMA)
Optimized Reinforcement Algorithm (ORA)
Resource Allocation Algorithm (RAA)

Fig. 1 Block diagram of ERAS

the field of RL techniques generally, Markov Decision Process, Analogies to Critical Care, Q-Learning for Neural Networks, and Particle Swarm Optimization (PSO). Section 4 introduces an Effective Resource Allocation Strategy (ERAS) for the Fog environment. Section 5 introduces the evaluation results and discussion. Our conclusion is discussed in Section 6.

2 Background and basic concepts

This section introduces some concepts in the field of Reinforcement Learning (RL), Markov Decision Process (MDP), Analogies to Critical Care, and Particle swarm intelligence.

2.1 Reinforcement learning (RL)

In formal terms, RL is a machine learning approach in which the software agent learns to carry out certain actions in an environment that results in maximum reward [37]. It does so by testing and leveraging the information it gains to optimize the reward through repeated trials. In each step, the RL agent receives evaluative feedback on the performance of its action, allowing it to improve the performance by trial and error of subsequent actions [18]. Mathematically, this sequential decision-making process is called the Markov decision process (MDP) [27].

2.2 Markov decision process (MDP)

Markov generally means that given the present state, the future and the past are independent [14]. For MDP, Markov means that the action outcomes depend only on the current state, not the history. The MDP is characterized by four main components: (i) A state which at each moment reflects the environment; (ii) An action that the agent takes at each time which affects the next state; (iii) a transition that provides an estimation of reaching various subsequent states that represent the environment in which an agent will interact; (iv) a reward function is the observed feedback given a state-action pair. The solution of the MDP is an optimized set of rules and is called the policy.

In some applications, including autonomous control, board games, and video games, RL has already emerged as an efficient method to solve complex control problems with large-scale, high-dimensional data [14, 19, 25]. RL has been shown to be capable of learning complex sequential decisions at the human level. For example, Alpha Go is an RL agent for playing the strategy board game Go. Alpha Go has learned a policy using the current state of the Go stones, and then decide what the next position for white/black stone on the board to maximize the opportunity of winning.

2.3 Analogies to critical care

Given the vast volume and granular nature of reported data, RL is well suited for critical care to provide sequential therapy recommendations, refine therapies, and improve outcomes for new ICU patients. By automatically testing different care options, RL also has the ability to extend our knowledge of current clinical protocols. The RL agent analyzes the patient trajectories and derives a policy, a tailored treatment plan that optimizes the likelihood of favorable clinical outcomes, through trial and error (eg, survival). Since this computerized method is an effort to emulate the thinking process of the human clinician, RL has also been named the AI clinician [33].

2.4 Q-learning for Neural networks

The Q-learning is a model-free RL algorithm that tells an agent what action to take under what conditions to learn the quality of actions. It requires no environmental model, and without adaptation, it can handle problems with stochastic transitions and rewards. It can be directly implemented in a Neural Network (NN). In Q-learning, there is no direct assignment of Q values such as the table-based view, instead, an error function is presented to calculate the difference between the existing Q value and the new value. Double DQN appears more robust to this more challenging evaluation, suggesting that appropriate generalizations occur and that the found solutions do not exploit the determinism of the environments. This is appealing, as it indicates progress towards finding general solutions rather than a deterministic sequence of steps that would be less robust.

2.5 Particle swarm optimization (PSO)

Particle swarm optimization is a global optimization approach for situations where the optimal solution is a point or surface in an n-dimensional space. In this space, hypotheses are drawn and seeded with a starting velocity, as well as a route for communication between the particles. Particles then flow across the solution space, with each timestep resulting in an evaluation based on some fitness criterion. Particles within their communication grouping are accelerated over time towards those particles with higher fitness levels. The fundamental benefit of this approach over other global reduction procedures like simulated annealing is that the enormous number of members that make up the particle swarm makes the process extremely resistant to local minima.

3 Related work

This section introduces some of the recent various RL models in subsection 3.1. It also introduces the relevant works focused on optimizing hyperparameters for various RL techniques in subsection 3.2.

3.1 RL models

This section introduces some of the recent various RL models. When the RL agent is well-trained, it is possible to pick the best action given the state of a patient, and we describe this process as acting according to an optimal policy. A policy is analogous to a clinical protocol. Nonetheless, a policy has advantages over a clinical protocol because it is capable of capturing more personalized details of individual patients. A policy can be represented by a table where it maps all possible states with actions.

Alternatively, a policy may also be interpreted by a Deep Neural Network (DNN), where the DNN model generates the highest likelihood of an intervention given the feedback of a patient's state. Using various RL algorithms, an optimal policy can be learned. Some widely applied RL algorithms include Q-Learning for Neural Networks [30], the Fitted-Q-Iteration (FQI) [30], Deep Q Network (DQN) [20], actor-critic network [29], and model-based RL [36]. More technical details about various RL models have been explained [9, 26]. The Comparison of some of the recent various RL models is shown in Table 1.

3.2 Optimization of RL hyperparameters

This section contains works that are linked to hyperparameter optimization in various Deep Reinforcement Learning (DRL) techniques.

The authors proposed Population-Based Bandits (PB2), a pioneering, proven, and effective population-based training (PBT-style) approach in [38]. In comparison to PBT, the approach allows for the detection of outstanding hyperparameter settings with fewer agents. With multiple RL trials, the authors show that PB2 can achieve exceptional results on a small computational budget. Authors in [35] recently presented an online hyperparameter adaption approach for DRL. An upgraded approach to population-based training was used to provide an effective online hyperparameter adaption method (PBT). The recombination operation, inspired by the GA, is introduced into the population optimization to speed up the population's convergence to the best hyperparameter configuration. The authors confirmed the usefulness of this strategy and found that it produced better outcomes than PBT, which is a traditional approach that they have used in the past.

In [31], Liessner et al. describe a model-based hyperparameter optimization for DDPG, which was found to be effective in industrial settings. In their work, the optimization is enhanced to include strictness on the available training time for the DDPG algorithm in the chosen domain. The authors demonstrated that DDPG hyperparameters could be tuned even when the time was limited.The authors employed a Genetic Algorithm (GA) to identify suitable HER+DDPG hyperparameters and optimized DDPG hyperparameters in [40]. The hyperparameters that require fewer epochs to learn superior task performance were discovered using GA. In robots manipulation duties, they used this strategy for reach, fetch, push, slide, place, pick, and open processes. Authors in [23] recommended using the Bayesian technique to modify hyperparameters in DRL. Chen et al. conducted the most comprehensive RL hyperparameters investigation, preferring to construct the AlphaGo algorithm via Bayesian optimization. Traditional approaches will never be able to produce these results. Bayesian optimization enhanced AlphaGo's chances of winning and helped

Table 1 Comparison of some of the recent various RL models

Algorithms	Basic Idea	Strength	Weakness
Q-Learning for Neural Networks [30]	In Q-learning, there is no direct assignment of Q values such as the table-based view, instead, an error function is presented to calculate the difference between the existing Q value and the new value. This is called off-policy learning.	Amazing result: Q-learning converges to optimal policy even if you're acting sub optimally	i. You have to explore enough (several ten thousands of episodes have to be done until an optimal or near optimal policy has been found) ii. You have to eventually make the learning rate small enough
Fitted-Q-Iteration (FQI) [30]	The basic concept behind NFQ is that the update is done off-line, with a whole range of transfer experiences rather than uploading on-line the neural value function (which leading to issues with Q-learning).	Overcomes the problems with the Q-learning. Using the whole training data leads to faster convergence and more reliability. The parameters for the supervised learning portion of the overall (RL) learning problem do not need to be modified. Advantages of NFQ are the following: (i) Offline; (ii) Model-free; and (iii) Works with random trajectories.	Complex algorithm
Deep Q Network (DQN) [20]	Deep Q-Networks combines Q-Learning with deep learning (Deep Neural Network (DNN)). The input for the DNN is the current state (s), while the output is the Q-values for all actions Q(s,a,w)≈ Q*(s,a). The loss function is used to calculate the difference between the actual output and the target output.	□ All the past experience is stored by the user in memory □ The next action is determined by the maximum output of the Q-network □ The loss function here is mean squared error of the predicted Q-value and the target Q-value – Q*. □ Both the target network and the experience replay dramatically improve the performance of the algorithm.	Using only previous setup will not help Q* to converge easily because of two reasons: (i) Samples correlation: samples that occur consecutively are mostly very same which doesn't add any diversity to the model. (ii) Non-stationary targets: targets already change as the policy improves (better rewards so better Q--values).
Double Q-learning [3]	In the original Double Q-learning algorithm, two value functions are learned by assigning each experience randomly to update one of the two value functions, such that there are two sets of weights, _ and _0. For each update, one set of weights is used to determine the greedy policy and the other to determine its value.	Double Q-learning can find an estimate for the maximum value of a set of stochastic values and it is shown that this sometimes underestimates rather than overestimates the maximum expected value.	The selection of the action, in the argmax, is still due to the online weights θt. This means that, as in Q-learning, we are still estimating the value of the greedy policy according to the current values, as defined by θt. However, we use the second set of weights θ't to fairly evaluate the value of this policy. This second set of weights can be updated symmetrically by switching the roles of θ and θ'.

Although there are many RA algorithms they have many limitations such as i) most of them depend on the response time to decide whether to assign a task to fog or to cloud which is not plausible. ii) They don't consider the task's requirements such as the priority of the task and the number of tasks. (iii) Calculating the capacity is difficult in some cases such as in the case with varying packet sizes. (iv) They may cause network bottleneck.

gather crucial data that can be used to create updated self-play agents using Monte Carlo Tree Search (MCTS). This practice, on the other hand, demands extensive testing and specialized knowledge. Furthermore, these techniques can only adapt to a single hyperparameter and cannot adapt to a variety of hyperparameters.

Similarly, authors in [32] developed a population-based training strategy (PBT). They suggested the OMPAC method, which is a different approach that focuses on the evolutionary mechanism. OMPAC was the first population-based technique to implement DRL's multiple hyperparameter adaptation. The population-based neural network training (PBT) strategy, which effectively employs a static computational budget to optimize a population of models and their hyperparameter configuration to achieve the optimal output, was also used by the authors in [6]. Machine translation, DRL, and GANs all performed well using the proposed method. On the other hand, PBT uses rudimentary stochastic perturbations to achieve hyperparameter adaptation, which is ineffectual in tracking changes in potential temporary ideal hyperparameter configuration. These techniques can only adapt to a single hyperparameter and cannot adapt to a variety of hyperparameters.

4 Effective resource allocation strategy (ERAS) for fog environment

This section introduces a new Effective Resource Allocation Strategy (ERAS) for the Fog environment. EARS is suitable for real-time systems such as healthcare applications. ERAS tries to achieve effective resource management in the Fog environment via real-time resource allocating. In the health care systems, the patient data is sent to the most appropriate server to handle it. This server is administrated by a specific healthcare organization. In the proposed system, the fog layer consists of two main modules as shown in Fig. 2, namely: (i) Data Preparation Algorithm (DPA), and (ii) Resource Management Algorithm (RMA).

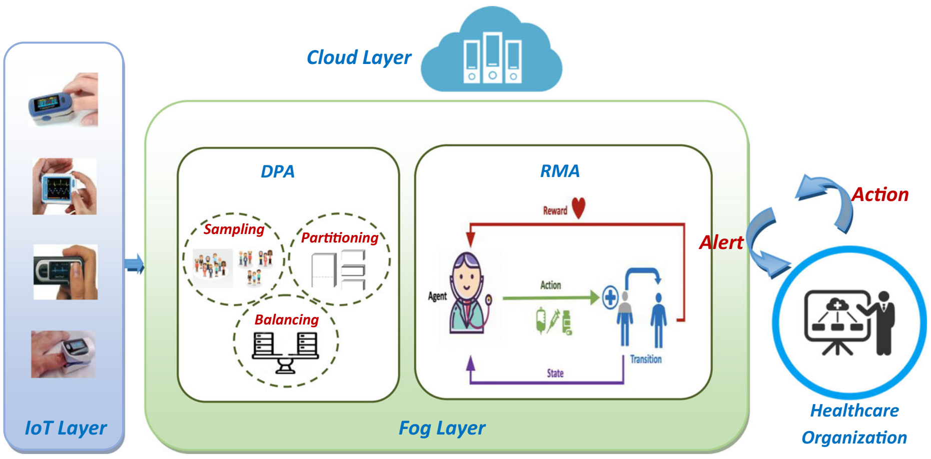

Fig. 2 Effective Resource Allocation Strategy (ERAS)

4.1 Data preparation algorithm (DPA)

One of the most challenging tasks in any machine learning project is data preparation. The reason for this is that each dataset is unique and tailored to the project. Nonetheless, there are enough similarities amongst predictive modeling projects that we can establish a rough sequence of phases and subtasks that you will most likely complete. The DPA is divided into three sub-modules, namely: (i) Sampling Module (SM), (ii) Partitioning Module (PM), and (ii) Balancing Module (BM).

i. Sampling Module (SM)

In SM, the incoming data is divided into subgroups (or strata) that share a similar characteristic using a stratified sampling algorithm. In stratified sampling, it may also be appropriate to choose non-equal sample sizes from each stratum. In SM, data is first sampled according to the location it comes from. Then it will be sampled according to its type. Data can be classified into three categorize: (i) Not or Low Critical: Examples are the logging of training activity, weight, or body posture. Such data can be examined by a doctor when needed. If the system fails to log some data points, the patient is still safe. ii) Critical Data: data in critical conditions. Examples are cardiac monitoring via ECG with automatic alarms once critical situations are detected [10]. The criticality requires fast response time, i.e., real-time response. Context Management merely observes patients, devices, or employees to figure out their context and help by improving planning or taking proper decisions [16]. However, data, in this case, is not urgent but it gains some degree of criticality due to real-time response need. (iii) Very Critical Data Control: The detected events are not only used to alert personnel, but also to control devices. This kind of data needs feedback and real-time response. An example is a device that regulates the amount of oxygen provided to a patient [12].

ii. Partitioning Module (PM)

PM splits the data into three samples. The model is built on the training set, and the model is applied to the testing set to establish its credibility. However, the testing set can then be used to further refine the model. If the performance of the model needs improvement, the parameters can be changed, and the model is then rebuilt using the training sample, after which the performance on the testing set is examined. The validation sample, which unlike the training and testing sets played no role in developing the final model, is then used to assess the model the model's performance against unseen data.

iii. Balancing Module (BM)

The BM achieves the balancing of the data by discarding (reducing) records in the higher-frequency categories. The reason is that when you boost the record you will run the risk of duplicating anomalies in the data.

4.2 Resource management algorithm (RMA)

RL is an Artificial Intelligence technique in which an agent takes an action in an environment to gain rewards. The agent receives the current environment state and takes an action accordingly. The taken action leads to a change in the environment state and then the agent will be informed of the change through a reward. The proposed RMA is based on Optimized Reinforcement Learning (RL) algorithm to achieve low latency for a fog environment. In RL, an agent learns to interact with the environment to achieve a reward. This section is divided into two subsections which are (i) Optimized Reinforcement Learning (ORL), and (ii) Resource Allocation Algorithm (RAA).

4.2.1 Optimized reinforcement learning (ORL)

RL is a machine learning technique that uses examples to develop optimum controllers, making it an ideal option for improving heuristic-based controllers utilized in the most popular and widely used optimization algorithms. RL techniques have the benefit over other approaches to optimization techniques in that they do not require previous knowledge of the basic system dynamics, and the system designer is free to set reward criteria that best match the controller performance desiderata. For example, in the event of time restrictions, a good incentive may be achieving the smallest loss in the shortest length of time. A simple variation of the Particle Swarm Optimization (PSO) algorithm employs a population of potential solutions (referred to as a swarm) (called particles). A few basic equations are used to move these particles about in the search space. The particles' motions are governed by their individual best-known position in the search space, as well as the best-known position of the entire swarm. When better places are located, they will be used to steer the swarm's movements. The procedure is repeated in the hopes that a suitable answer will be found someday, although this is not guaranteed. Particle Swarm Optimization (PSO) is used to optimize reinforcement learning. PSO has a significant significance when compared to other optimization methodologies, because of its exploitation, exploration, and capacity to get rid of local minima. The selected set hyperparameters to be optimized are (i) learning rate (lr), (ii) batch size (bs), (iii) discount factor (df), and (iv) reward (r). The overall steps of the ORL are shown in Algorithm 1.

4.2.2 Resource allocation algorithm (RAA)

In computing systems, resource allocation is the process of allocating available system resources to distinct activities that are ready to be done. This is a procedure that has a substantial impact on the system's overall performance. A list of activities or processes that are ready to be executed at a specific moment, as supplied by a system scheduler, is typically fed into resource allocation algorithms. The Resource Allocation Algorithm (RAA) learns to select the best server to execute the incoming request. Firstly, the input is the data in the NCT, and finally, the output is the balanced system with high performance and lowest latency. The overall steps of the RAA are shown in Algorithm 2.

ORL Algorithm

- ***Input:***
  - HPT containing initial values for the hyperparameters (lr, bs, df, r)
- ***Output:***
  - The optimal values for (lr, bs, df, r)
- ***Steps:***

```
 1:  Initialize gbest and r (gbest=Vi and r=r0)
 2:  Collect data from HPT
 3:  For each Fog value for r in HPT do
 4:      calculate the Fitness Value (FVi) using Fuzzy Logic via the three parameters stored in HPT
         which are: lr, bs, and df.
 5:      If (FVi > gbest) then:
 6:              gbest= FVi and r=ri
 7:      End If
 8:  Next
 9:  Update the FST
10:  Assign the new values to the HPT
```

Symbol	Meaning
gbest	global best value
r	reward
Vi	Initial value of gbest
r0	Initial value for r
HPT	HyperParameter Table
FVi	Fitness Value
lr	learning rate
bs	batch size
df	discount factor

Algorithm 2: ORL Algorithm

Resource Allocation Algorithm (RAA)

- ***Input:***
  - The data in the NCT
- ***Output:***
  - Balanced system with high performance and lowest latency.
- ***Steps:***

```
 1:  For each new incoming process ($P_i$) do
 2:      Create a Q-table containing two columns [State: Available Underloaded and Balanced Fog
         servers, Action: Selecting the best Fog Server to execute $P_i$] initialized to zero.
 3:      //Taking Action
 4:      The agent interacts with the environment (Fog servers) and makes updates to the state-action
         pairs in the Q-table Q[state, action].
 5:      The agent uses the Q-table as a reference and view all possible actions (all possible available
         Fog servers) for a given state. Then it selects the best FN based on the lowest latency.
 8:      // Updating the Q-table
 9:      Update the values in the Q-table
10:      //Update the data in SCT
11:      The LBA updates the data in the SCT (update the AW and the status for each FN)
12:  Next
```

Algorithm 2: Resource Allocation Algorithm (RAA)

5 Implementation and evaluation

FC runs applications in fog devices between cloud and end devices. This paradigm has used the benefits of cloud and edge for distributed data and low latency. IoT sensors which are located in the lower layer of the architecture, are responsible for receiving and transmitting data through the gateways to the higher layer. The actuators in the lowest level, are also responsible for system controls. FC provides filtering and analysis of data by edge devices. Each application of a fog network has different topology. The Resource Allocation Module (RAM) and the Effective Prediction Module (EPM) are implemented using python.

5.1 Mobile HEALTH dataset

The Mobile HEALTH (MHEALTH) [34] dataset contains vital signs and body motion recordings for 10 volunteers during several physical activities. Sensors placed on the subject's chest, right wrist, and left ankle are used to measure the motion experienced by diverse body parts, namely, acceleration, rate of turn, and magnetic field orientation. The main characteristics of MHEALTH Dataset are shown in Table 2.

In this paper, the MHEALTH is used to detect the possibility of a heart attack. Hence, a new extra column (column 24) is added to have the values of the probability of attack. Column 24 has three main values which are; (i) 1 for strong probability, (ii) 2 for average probability, (iii) 3 for low probability. In order to simplify the classification, only 560 instances are selected from the MHEALTH Dataset for training and 240 instances are selected for testing. A number of training dataset instances for each probability is shown in Table 3. A sample of MHEALTH Dataset is shown in Table 4.

5.2 Performance metrics

The performance of ERAS vs. previously mentioned LB algorithms can be compared by considering the following metrics shown in Table 5.

Table 2 MHEALTH Dataset Characteristics

Data Set Characteristics:	Attribute Characteristics:	Associated Tasks:	Number of Attributes:	Number of Instances:	Missing Values?
Multivariate, Time-Series	Real	Classification	23	161,280	N/A

Table 3 Used MHEALTH Dataset

Heart_Attack_Probability	1	2	3
No. of Training Dataset Instances	180	175	205

Table 4 A sample of MHEALTH Dataset

chest sensor (X axis)	chest sensor (Y axis)	chest sensor (Z axis)	electrocardiogram signal (lead 1)	……	Heart_Attack_Probability
−9.7409	0.68291	0.7562	−0.06698	……	1

Table 5 the Performance metrics to evaluate the proposed ERAS scheme

Metric	Definition	Notes
Makespan	It is the total execution time in which task get scheduled or completely executed. It can be called Completion Time (CT) CT: is the time at which process completes its execution.	Makespan always should be low [24].
Average Resource Utilization (ARU)	It is the complete utilization of each resource present in fog environment.	For better performance, ARU ratio should be high.
Load Balancing Level (LBL)		For better performance, LBL should be high (https://archive.ics.uci.edu/ml/datasets/MHEALTH+Dataset).

ARU can be calculated as in (1) and LBL can be calculated as in (2).

5.3 ERAS implementation

ERAS algorithm has been compared with LC, RR, WRR, and AWRR. The values of makespan are shown in Table 6 and in Fig. 3. The values of ARU are shown in Table 6 and in Fig. 4. The values of LBL are shown in Tables 7, 8 and in Fig. 5.

Table 6 Makespan analysis (in ms)

Number of tasks	LC	RR	WRR	AWRR	ERAS
50	95.33	96	93.13	87.95	84.87
90	169.34	170.98	167.78	164.66	150.75
130	232.48	235.01	230.21	225.81	220.71
150	280.74	285.11	279.85	275.01	270.89

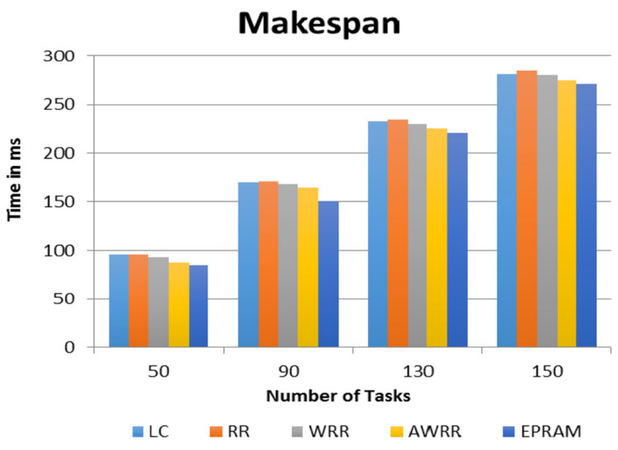

Fig. 3 Makespan for ERAS vs Previous LB algorithms

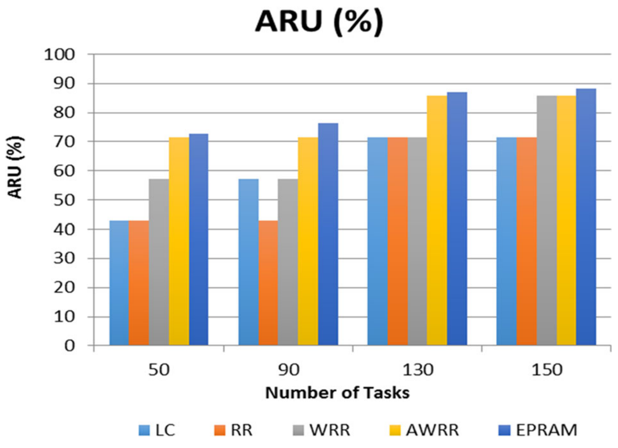

Fig. 4 ARU for ERAS vs Fig. 3 Makespan for ERAS vs Previous LB algorithms Previous LB algorithms

Table 7 ARU Analysis (%)

Number of tasks	LC	RR	WRR	AWRR	ERAS
50	42.85	42.85	57.14	71.42	72.58
90	57.14	42.85	57.14	71.42	76.28
130	71.42	71.42	71.42	85.71	86.89
150	71.42	71.42	85.71	85.71	88.11

Table 8 LBL Analysis (%)

Number of tasks	LC	RR	WRR	AWRR	ERAS
50	28.57	28.57	42.85	57.14	59.04
90	42.85	28.57	57.14	71.42	72.45
130	57.14	57.14	71.42	71.42	73.45
150	71.42	57.14	71.42	85.71	86.14

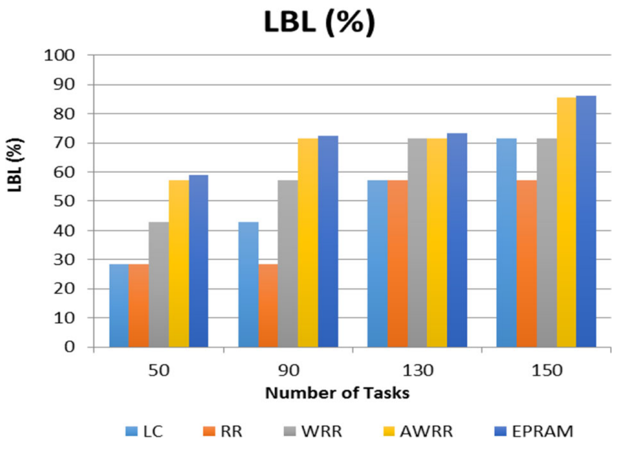

Fig. 5 LBL for ERAS vs Previous LB algorithms

Figure 3 explained that ERAS algorithm gives lower Makespan as compared to previous resource allocating algorithms. Figures 4 and 5 explained that ERAM algorithm gives better result as compared to previous resource allocating algorithms as it achieved the higher ARU and higher LBL. Hence all the above results have shown that ERAS algorithm performs better for makespan, ARU and LBL as compared to LC, RR, WRR, and AWRR.

The fog computing architecture is centralized but without traffic as each fog region has a master node (fog master) which is responsible for managing and controlling all requests in this region. There is no large requests (send and receive messages all the time) as the fog master has a huge database which contains information bout the whole nodes in its region. Hence, there is no need for send and receive messages all the time. It first checks its database for the required information.

$$\mathrm{ARU} = \frac{(BS + OL)}{\mathrm{FSs}} {*}100\% \tag{1}$$

$$\mathrm{LBL} = \frac{BS}{\mathrm{FSs}} {*}100\% \tag{2}$$

Where, BS: is the number of Balanced Fog Servers, OL: is the number of Overloaded Fog Servers, and FSs: is the number of all available Fog Servers.

6 Conclusions and future work

This paper presented a new Effective Resource Allocation Strategy (ERAS) for Fog environment, which is suitable for Healthcare applications. ERAS tries to achieve effective resource management in Fog environment via real-time resource allocating as well as prediction algorithm. ERAS is composed of two main modules, namely: (i) Data Preparation Algorithm (DPA), and (ii) Resource Management Algorithm (RMA). The DPA is responsible for sampling, partitioning, and balancing data to be in the appropriate form for analyzing and processing. The RMA learns to select the best server to execute the incoming request. The RMA uses Reinforcement Learning (RL) algorithm to achieve low latency. Comparing the ERAS with the state-of-the-art algorithms, ERAS achieved the minimum Makespan as compared to previous resource allocation algorithms, while maximizing the Average Resource Utilization (ARU) and the Load Balancing Level (LBL). For each application, we further compared and contrasted the architecture of the RL models and the assessment metrics. In critical care, RL has tremendous potential to enhance decision-making. In future work, we aim to test ERAS using various datasets. This paper presented two main contributions, (i) Optimization of the RL hyperparameters using PSO, and (ii) Using the optimized RL for the resource allocation and load balancing in the fog environment. Because of its exploitation, exploration, and capacity to get rid of local minima, the PSO has a significant significance when compared to other optimization methodologies. The selected set hyperparameters to be optimized are (learning rate (lr) – batch size (bs) - discount factor (df) – reward (r)). Optimized hyperparameters regularly produced better total rewards at test time than a commonly used reference hyperparameter suggested by an expert, according to the findings. The hyperparameters of RL algorithms have been shown to affect their efficiency.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Declarations

Conflicts of interests/competing interests We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

References

1. Al-Khafajiy M, et al 2018 Towards fog driven IoT healthcare: challenges and framework of fog computing in healthcare. Proceedings of the 2nd international conference on future networks and distributed systems
2. Almirall D, Compton SN, Gunlicks-Stoessel M, Duan N, Murphy SA (2012) Designing a pilot sequential multiple assignment randomized trial for developing an adaptive treatment strategy. Stat Med 31(17):1887–1902 [FREE Full text] [CrossRef] [Medline]
3. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: a brief survey. IEEE Signal Process Mag 34(6):26–38 [CrossRef]
4. Burke AE, Thaler KM, Geva M, Adiri Y (2019) Feasibility and acceptability of home use of a smartphone-based urine testing application among women in prenatal care. Am J Obstet Gynecol 221(5):527–528 [CrossRef] [Medline]
5. Chen Z, Marple K, Salazar E, Gupta G, Tamil L (2016) A physician advisory system for chronic heart failure management based on knowledge patterns. Theor Pract Log Prog 16(5–6):604–618 [CrossRef]
6. Chen Y, Huang A, Wang Z, Antonoglou I, Schrittwieser J, Silver D, et al (2018) Bayesian optimization in alphago. arXiv preprint arXiv:181206855
7. Choi E, Bahadori MT, Schuetz A, Stewart WF, Sun J. (2016) Doctor AI: predicting clinical events via recurrent neural networks. JMLR Workshop Conf Proc 2016 Aug;56:301–318 [FREE Full text] [Medline]
8. Dagan N, Elnekave E, Barda N, Bregman-Amitai O, Bar A, Orlovsky M et al (2020) Automated opportunistic osteoporotic fracture risk assessment using computed tomography scans to aid in FRAX underutilization. Nat Med 26(1):77–82 [CrossRef] [Medline]
9. Doya K, Samejima K, Katagiri K, Kawato M (2002) Multiple model-based reinforcement learning. Neural Comput 14(6):1347–1369 [CrossRef] [Medline]
10. Elfwing S, Uchibe E, Doya K, editors (2018) Online meta-learning by parallel algorithm competition. Proceedings of the Genetic and Evolutionary Computation Conference
11. Fan J, Wang J, Chen Z, Hu C, Zhang Z, Hu W (2019) Automatic treatment planning based on three-dimensional dose distribution predicted from deep learning technique. Med Phys 46(1):370–381 [CrossRef] [Medline]
12. Gia TN, Jiang M, Rahmani A-M, Westerlund T, Liljeberg P, Tenhunen H (2015) Fog computing in healthcare Internet of Things: A case study on ECG feature extraction, in Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Auto. Secur. Com-put., Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM), PP. 356–363
13. Hannes K, Leys M, Vermeire E, Aertgeerts B, Buntinx F, Depoorter A (2005) Implementing evidence-based medicine in general practice: a focus group based study. BMC Fam Pract 6:37 [FREE Full text] [CrossRef] [Medline]
14. Howard RA (1960) Dynamic programming and Markov process. MIT Press and Wiley, New York

15. Hutchinson A, Baker R (1999) Making use of guidelines in clinical practice. Br Med J 319(7216):1078 [FREE Full text] [CrossRef] [Medline]
16. Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, et al (2017) Population based training of neural networks. arXiv preprint arXiv:171109846
17. James JT (2013) A new, evidence-based estimate of patient harms associated with hospital care. J Patient Saf 9(3):122–128 [CrossRef] [Medline]
18. Javed AR, Sarwar MU, Beg MO, Asim M, Baker T, Tawfik H (2020) A collaborative healthcare framework for shared healthcare plan with ambient intelligence. Human-centric Computing and Information Sciences 10(1):1–21
19. Kiumarsi B, Vamvoudakis KG, Modares H, Lewis FL (2018) Optimal and autonomous control using reinforcement learning: a survey. IEEE Trans Neural Netw Learn Syst 29(6):2042–2062 [CrossRef] [Medline]
20. Komorowski M, Celi LA, Badawi O, Gordon AC, Faisal AA (2018) The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. Nat Med 24(11):1716–1720 [CrossRef] [Medline]
21. Laffey JG, Kavanagh BP (2018) Negative trials in critical care: why most research is probably wrong. Lancet Respir Med 6(9):659–660 [CrossRef] [Medline]
22. Laserson J, Lantsman CD, Cohen-Sfady M, Tamir I, Goz E, Brestel C, et al (2018) TextRay: Mining Clinical Reports to Gain a Broad Understanding of Chest X-Rays. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. 2018 Presented at: MICCAI'18; September 16–20; Granada, Spain. [CrossRef]
23. Liessner R, Schmitt J, Dietermann A, Bäker B editors (2019) Hyperparameter Optimization for Deep Reinforcement Learning in Vehicle Energy Management. ICAART (2)
24. Masip-Bruin X, Marín-Tordera E, Alonso A, Garcia J (2016) Fog-to-cloud computing (F2C): The key technology enabler for dependable ehealth services deployment, in Proc. Medit. Ad Hoc Netw. Workshop (Med-Hoc-Net), PP. 1–5
25. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, et al (2013) Playing atari with deep reinforcement learning. arXiv preprint 2013:- epub ahead of print(1312.5602) [FREE Full text]
26. Mnih V, Puigdomenech A, Mirza M, Graves A, Lillicrap T, Harley T, et al (2016) Asynchronous methods for deep reinforcement learning. Arxiv 2016:- epub ahead of print(1602.01783) [FREE Full text]
27. Montague PR (1999) Reinforcement learning: an introduction, by Sutton, RS and Barto, AG. Trends Cogn Sci 3(9):360 [CrossRef]
28. Nemati S, Ghassemi MM, Clifford GD (2016) Optimal medication dosing from suboptimal clinical examples: a deep reinforcement learning approach. Conf Proc IEEE Eng Med Biol Soc 2016:2978–2981 [CrossRef] [Medline]
29. Neural RM Fitted Q (2005) Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In: Proceedings of the European Conference on Machine Learning. 2005 Presented at: ECML'05; October 3–7; Porto, Portugal URL: https://doi.org/10.1007/11564096_32 [CrossRef]
30. Ng A, Coates A, Diel M, Ganapathi V, Schulte J, Tse B et al (2006) Autonomous inverted autonomous helicopter flight via reinforcement learning. In: Experimental Robotics IX (ed) New York. Springer, USA, pp 363–372
31. Parker-Holder J, Nguyen V, Roberts SJ (2020) Provably efficient online hyperparameter optimization with population-based bandits. Adv Neural Inf Proces Syst;33
32. Sehgal A, La H, Louis S, Nguyen H, editors (2019) Deep reinforcement learning using genetic algorithm for parameter optimization. 2019 Third IEEE International Conference on Robotic Computing (IRC): IEEE
33. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489 [CrossRef] [Medline]
34. Tentori M, Favela J (2007) Activity-aware computing in mobile collaborative working environments, in Proc. 13th Int. Conf. Groupw., Design Implement. (CRIWG), Berlin, Germany, PP. 337–353
35. Van Hasselt H. (2010) Double Q-learning, 2613–2621
36. van Hasselt H, Guez A, Silver D (2016) Deep Reinforcement Learning With Double Q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. 2016 Presented at: AAAI'16; February 12–17; Phoenix, Arizona, USA. [CrossRef]
37. Watanabe AT, Lim V, Vu HX, Chim R, Weise E, Liu J et al (2019) Improved cancer detection using artificial intelligence: a retrospective evaluation of missed cancers on mammography. J Digit Imaging 32(4): 625–637 [FREE Full text] [CrossRef] [Medline]
38. Wiering M, van Otterlo M (eds) (2012) Reinforcement learning: state-of-the-art. Springer-Verlag, Berlin, Heidelberg

39. Zhang Z, Hong Y, Liu N (2018) Scientific evidence underlying the recommendations of critical care clinical practice guidelines: a lack of high level evidence. Intensive Care Med 44(7):1189–1191 [CrossRef] [Medline]
40. Zhou Y, Liu W, Li B (2019) editors. Efficient Online Hyperparameter Adaptation for Deep Reinforcement Learning. International Conference on the Applications of Evolutionary Computation (Part of EvoStar): Springer