2017 31st International Conference on Advanced Information Networking and Applications Workshops

# Design and Implementation of a Simulation System Based on Deep Q-Network for Mobile Actor Node Control in Wireless Sensor and Actor Networks

**Tetsuya Oda*, Ryoichiro Obukata†, Makoto Ikeda*, Leonard Barolli*, Makoto Takizawa‡,**

*Department of Information and Communication Engineering,
Fukuoka Institute of Technology (FIT),
3-30-1 Wajiro-Higashi, Higashi-Ku, Fukuoka 811–0295, Japan
Email: oda.tetsuya.fit@gmail.com, makoto.ikd@acm.org, barolli@fit.ac.jp

†Graduate School of Engineering,
Fukuoka Institute of Technology (FIT),
3-30-1 Wajiro-Higashi, Higashi-Ku, Fukuoka 811–0295, Japan
E-mail: obukenkyuu@gmail.com

‡Department of Advanced Sciences,
Hosei University,
3-7-2, Kajino-machi, Koganei-shi, Tokyo 184-8584, Japan
Email: makoto.takizawa@computer.org

***Abstract*—A Wireless Sensor and Actor Network (WSAN) is a group of wireless devices with the ability to sense physical events (sensors) or/and to perform relatively complicated actions (actors), based on the sensed data shared by sensors. This paper presents design and implementation of a simulation system based on Deep Q-Network (DQN) for mobile actor node control in WSANs. DQN is a deep neural network structure used for estimation of Q-value of the Q-learning method. In this work, we implement the proposed simulating system by Rust programming language. We describe the design and implementation of the simulation system, and show some simulation results to evaluate its performance.**

***Keywords*-Wireless Sensor and Actor Networks, Deep Q-Network, Simulation System, Mobile Actor Node Control, Rust.**

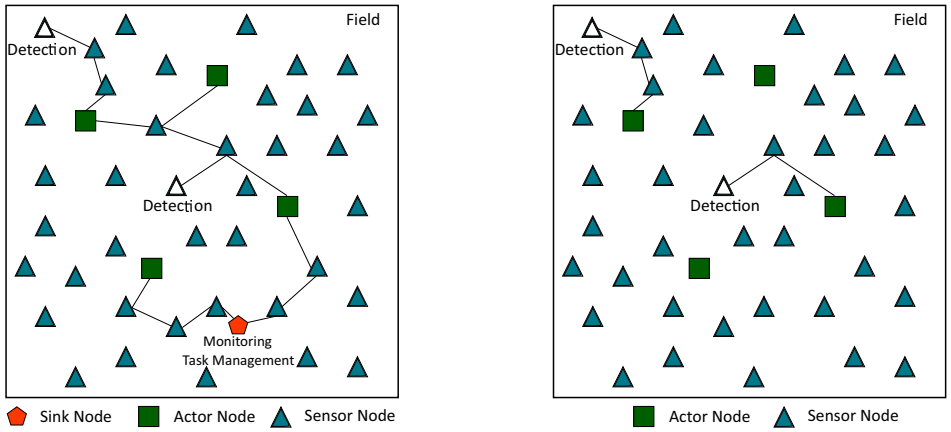
## I. Introduction

Wireless Sensor and Actor Networks (WSANs) have emerged as the-state-of-the-art technology in data gathering from remote locations by interacting with physical phenomena and relying on collaborative efforts by a few resource rich devices and a large number of low cost devices [1]. In WSANs sensors are low cost resource constrained devices that have limited energy, communication and computation capabilities. Once deployed, sensor nodes collect the information of interest from their on board sensors, perform local processing of these data including quantization and compression. The actor nodes, on the other hand are resource rich nodes that are equipped with better processing power, higher transmission power, more energy resources and may contain additional capabilities such as mobility.

The concept of a mobile actor node is relatively new in the area of WSAN [2]. The main advantage of having a mobile actor node is that more effective action can be taken, since the actor can get as close as possible to the event location. Additionally, in a densely connected network the presence of a mobile actor eases the problem of overburdened forwarding nodes by balancing the load among all the nodes in the network. In a sparsely connected network, a mobile actor node can bridge the connectivity between groups of isolated nodes. But, there is no guarantee that the existing mobility models will work efficiently for a mobile actor node in a sparsely connected network. An example application is coastal monitoring where sensor nodes may be deployed to monitor water temperature, wave characteristics, water level or meteorological conditions. When an event occurs in this environment, a possible action taken by the actor node could be of collecting a water sample for further analysis.

One of the main advantages of WSANs is the ability to exploit node mobility for various purposes [3]. Several performance metrics including dependability, connectivity, deployment convenience, coverage, energy, and accuracy can be improved by moving and relocating various nodes in these networks [4], [5]. Our focus in this paper is connectivity which is crucial in WSANs. In WSANs, the connectivity of actor node is needed to organize collaborative tasks. Mobility helps improving dependability of these networks in several ways. For instance, failed nodes can be replaced with the other spare nodes by moving the spare nodes to their locations. Similarly, if the network is partitioned, mobility can be exploited to restore connectivity by moving one or multiple nodes to the selected locations. Recently, the aforementioned dependability issues have been studied extensively in the context of WSANs [6], [7]. The main

978-1-5090-6231-7/17 $31.00 © 2017 IEEE
DOI 10.1109/WAINA.2017.67




(a) Semi-Automated (b) Fully-Automated

Figure 1. WSAN Architectures.

focus of these works was to deal with individual failures and restore the connectivity of the actor nodes or the connectivity of the WSN with the sink node.

There are many critical issues arising in WSANs. Based on the specific application, different objectives can be taken into account such as energy consumption, throughput, delay, coverage, etc [8], [9]. Also many schemes have been proposed in order to optimize a specific Quality of Service (QoS) parameter. With the focus on the self-organizing capabilities of nodes in WSAN, we propose a movement-assisted technique for nodes self-deployment. Specifically, we propose a simulation system based on Deep Q-Network (DQN) as a controller for actor node mobility and implement by Rust programming language. We describe the implementation and show the simulation results of the simulation system.

The rest of the paper is organized as follows. In Section II, we describe the basics of WSANs including architectures, research challenges and mobile control for actor node. In Section III, we present the overview of DQN. In Section IV, we show the description and design of the simulation system. Simulation results are shown in Section V. Finally, conclusions and future work are given in Section VI.

## II. WSAN

### A. WSAN Architecture

The development of smart sensors and the spread of Micro Electro Mechanical Systems (MEMS) have contributed in a major way to the recent expansion of WSANs [10]. WSANs are normally composed of a large number of nodes, which are low power devices generally equipped with one or more sensing units (composed of sensors and, eventually, analog to digital converters), an actuator, a processing unit that includes a limited memory, a radio transceiver, and the power supply.

The main functionality of WSANs is to make actors perform appropriate actions in the environment, based on the data sensed from sensors and actors [11]. When important data has to be transmitted (an event occurred), sensors may transmit their data back to the sink, which will control the actors' tasks from distance or transmit their data to actors, which can perform actions independently from the sink node. Here, the former scheme is called Semi-Automated Architecture and the latter one Fully-Automated Architecture, as seen in Fig. 1(b) and Fig. 1(a), respectively. Obviously, both architectures can be used in different applications. Fully-Automated Architecture, which is considered in this paper, emerges the need to develop new sophisticated algorithms, in order to provide appropriate coordination between nodes of WSAN. On the other hand, it has advantages, such as *low latency*, *low energy consumption*, *long network lifetime*, *higher local position accuracy*, *higher reliability* and so on.

### B. WSAN Challenge

Some of the key challenges in WSAN are related to the presence of actors and their functionalities [11].

*Deployment and Positioning*: WSAN are heterogeneous networks [12], where actors and sensors have different processing powers, mobility abilities and functionalities. Thus, at the moment of node deployment, algorithms must consider to optimize the number of sensors and actors and their initial positions based on application [13], [14].

*Architecture*: The main functionality of WSANs is to make actors perform appropriate actions in the environment, based on the data sensed from sensors and actors [15]. When important data has to be transmitted (an event occurred), sensors may transmit their data back to the sink, which will control the actors' tasks from distance or transmit their data to actors, which can perform actions independently from the sink node.

*Real-Time*: The purpose of using WSANs in most of the applications is mainly related to their ability to react independently to situations where human intervention is physically difficult or time-restricted [16]. In other words, there are a lot of applications that have strict real-time requirements. In order to fulfill them, real-time limitations must be clearly defined for each application and system.

*Coordination*: Unlike WSN, where sensors coordinate with each-other to send data to the sink, in WSAN, sensor-actor coordination occurs as well, because all sensed data controls actor's behavior. Also, actor-actor coordination is important in cases when actors collaborate on performing tasks together. In order to provide effective sensing and acting, a distributed local coordination mechanism is necessary among sensors and actors [17].

*Power Management*: Similar to energy-constrained WSNs, in WSANs sensors have limited power supplies, which limits the network lifetime. Actors have more powerful power supplies but their functionalities are more sophisticated, so they spend more energy when completing complicated tasks. Thus, WSAN protocols should be designed with minimized energy consumption for both sensors and actors [18].

*Mobility*: In WSANs, nodes, especially actors can be mobile. For example, robots used in industrial monitoring sites or flying drones aver a disaster recovery area [19]. Therefore, protocols developed for WSANs should support the mobility of nodes [20], where dynamic topology changes, unstable routes and network isolations are present.

*Self Healing*: One of the main problems in mobile Self-Organizing Networks (SON) is the high probability of node isolations during network runtime. An actor failure may lead to partitioning the network and thus hinder the fulfillment of the application requirements. Many works have been done on connectivity restoration, by using actors ability to move without using much energy [6]. Actors may also be specialized to carry extra energy supplies, in order to charge sensors or other actors in the network.

*Scalability*: Smart Cities are emerging fast and WSAN, with its practical functions of simultaneous sensing and acting, are a key technology. The heterogeneity is not limited and most of the systems will continue to grow together with cities. In order to keep the functionality of WSAN applicable, scalability should be considered when designing WSAN protocols and algorithms. Data replication, clustering and so on, can be used in order to support growing networks [14].

## III. DQN

The algorithm for training Deep Q-learning is presented in Algorithm 1. The DQN defined the tasks between the agents and the environments [21], [22] in Atari 2600 games emulator [23]. Because Q maps history-action pairs to scalar estimates of their Q-value, the history and the action have been used as inputs to the neural network by some previous

**Algorithm 1** Deep Q-learning with Experience Replay

Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights
**for** $episode = 1$, $M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  **for** $t = 1$, $T$ **do**
    With probability $\epsilon$ select a random action $a_t$
    otherwise select at = $\max_a\ Q^*\ (\phi(s_t), a; \theta)$
    Execute action at in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \max_a(\phi_{j+1}, a_j; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a'_j; \theta))^2$ according to equation 3
  **end for**
**end for**

approaches [24], [25]. The environment was set as $\xi$. At each step, the agent selected an action $a_t$ from the action sets of the game and observed a displayed image $x_t$ from the current screen [26]. The change of the game score $r_t$ was regarded as the reward for the action. For a standard reinforcement learning method, we can complete all of these game sequences $s_t$ as Markov decision process directly, where sequences of actions and observations $s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t$. Also, it uses a technique known as experience replay [27] in which it store the agent's experiences at each time-step, $e_t = (s_t, a_t, r_t, s_{t+1})$ in a data-set $D = e_1, \ldots, e_N$, pooled over many episodes into a replay memory. Defining the discounted reward for the future by a factor $\gamma$, the sum of the future reward until the end would be $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$. $T$ means the termination time-step of the game. After performing experience replay, the agent selects and executes an action according to an $\epsilon$-greedy policy. Since using histories of arbitrary length as inputs to a neural network can be difficult, Q-function of DQN instead works on fixed length representation of histories produced by a function $\phi$. The target was to maximize the action-value function $Q^*(s,a) = max_\pi E[R_t | s_t = s, a_t = a, \pi]$, where $\pi$ is the strategy for choosing of best action. From the Bellman equation, it is equal to maximize the expected value of $r + \gamma Q^*(s', a')$, if the optimal value $Q^*(s', a')$ of the sequence at the next time step is known.

$$Q^*(s', a') = E_{s' \sim \xi}[r + \gamma \underset{a'}{max} Q^*(s', a') | s, a] \qquad (1)$$

Not using iterative updating method to optimal the equation, it is common to estimate the equation by using a function approximator. Q-network in DQN was such a neural network function approximator with weights $\theta$ and $Q(s, a; \theta) \approx Q^*(s, a)$. The loss function to train the Q-network is:

$$L_i(\theta_i) = E_{s,a \sim \rho(.)}[(y_i - Q(s, a; \theta_i))^2]. \qquad (2)$$

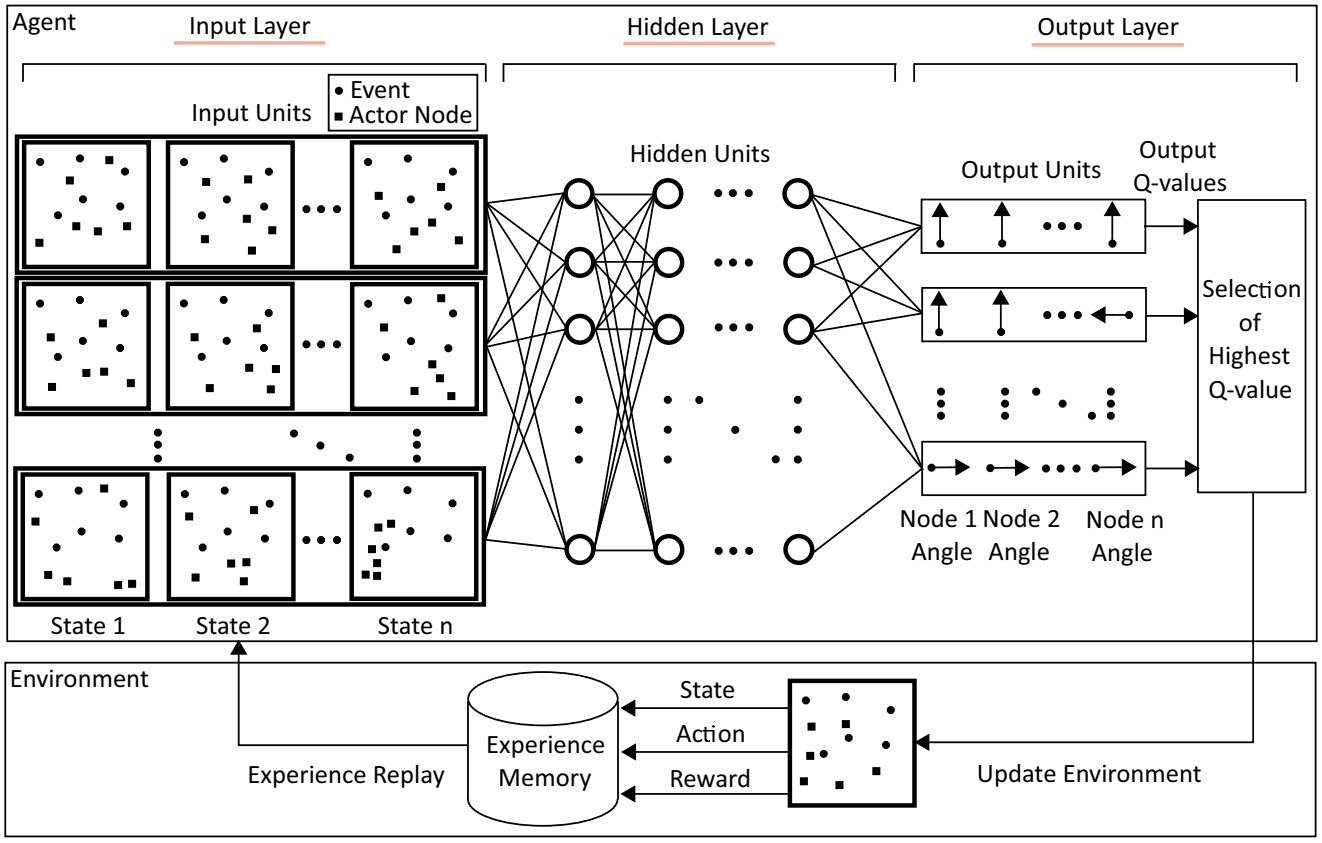



Figure 2. The structure of DQN based mobile actor node control simulation system.

The $y_i$ is the target, which is calculated by the previous iteration result $\theta_{i-1}$. $\rho(s, a)$ is the probability distribution of sequences $s$ and $a$. The gradient of the loss function is shown in Eq. (3):

$$\nabla_{\theta_i} L_i(\theta_i) = E_{s,a\sim\rho(.);s'\sim\xi}[(y_i - Q(s, a; \theta_i))\nabla_{\theta_i} Q(s, a; \theta_i)]. \quad (3)$$

## IV. Design and Implementation of Proposed Simulation System

In this section, we present design and implementation of proposed simulation system based on DQN for mobile actor node control in WSANs. The simulation system structure is shown in Fig. 2. The proposed simulating system is implemented by Rust programming language [28], [29]. Rust is a system programming language focused on three goals: safety, speed, and concurrency [30]. Rust supports a mixture of programming styles: imperative procedural, concurrent actor, object-oriented and functional.

We consider tasks in which an agent interacts with an environment. In this case, the mobile actor node moves step by step in a sequence of actions, observations and rewards. We took in consideration the mobility and connectivity of actor nodes.

For a mobile actor node are considered 5 patterns (forward, back, left, right, stop). The actor nodes have mobility, networking, sensing and actuation mechanisms. In order to decide the reward function, we considered the Number of Connected Actor Nodes (NCAN) and Number of Sensing Events (NSE) parameters. The reward function $r$ is defined as follows:

$$r = \begin{cases} 5 \times (NCAN + NSE) & (if \quad NCAN \geq 1) \\ -5 \times (NCAN + NSE) & (if \quad NCAN = 0). \end{cases} \quad (4)$$

The initial weights values are considered as Normal Initialization [31]. The input layer is using the position of events and actor nodes, mobile actor node patterns and total reward values in Experience Memory. The hidden layer is connected with 256 rectifier units in Rectified Linear Units (ReLU) [32]. The output Q-values are mobile actor node patterns.

## V. Simulation Results

The simulation parameters are shown in Table I. The simulations are done for 4 actor nodes and 24 events for normal distribution. One episode has 100 iterations.

In Table II, we show the simulation total reward of episodes. There are best, median and worst episodes. The

Table I
SIMULATION PARAMETERS.

| Parameters | Values |
|---|---|
| Field size | 32 × 32 |
| Types of actor node mobile | forward, back, left, right, stop |
| Number of episode | 100000 |
| Number of iteration | 100 |
| Number of hidden layers | 3 |
| Number of hidden units | 15 |
| Initial weight value | Normal Initialization |
| Activation function | ReLU |
| Action selection probability ($\epsilon$) | 0.999 - ($t$ / Number of episode) ($t$ = 0, 1, 2, . . ., Number of episode) |
| Learning rate ($\alpha$) | 0.04 |
| Discount rate ($\gamma$) | 0.9 |
| Experience memory size | 300 × 100 |
| Batch size | 32 |
| Positions of events | Normal distribution |
| Initial positions of actor nodes | Center |
| Number of events | 24 |
| Number of actor nodes | 4 |

Table II
RESULTS OF TOTAL REWARD.

| Episode | Total reward |
|---|---|
| Best episode | 7590 |
| Median episode | 5265 |
| Worst episode | -1525 |

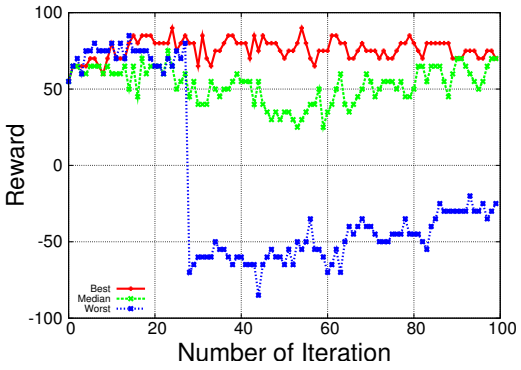



Figure 3. Results of reward.

best episode shows that all actor nodes are connected with other actor nodes and have covered all events. In Fig. 3, we show the simulation results of reward vs. number of iteration. The higher reward value means that the actor node can move and keep connection with other actor nodes. In Fig. 4 is shown visualization interface of implemented simulation system.

## VI. CONLUSION

In this work, we designed and implemented a simulation system based on DQN for mobile actor node control in WSANs. We presented the implementation of the proposed simulation system and have shown also the interface in a simulation scenario.

In the future, we would like to make extensive simulations for different simulation scenarios.

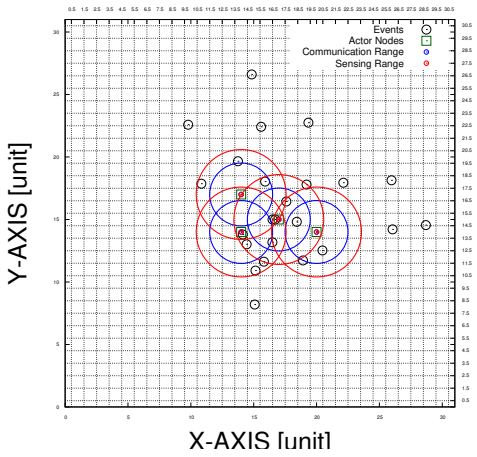



Figure 4. Visualization interface.

## REFERENCES


[1] I. F. Akyildiz, I. H. Kasimoglu, “Wireless Sensor and Actor Networks: Research Challenges”, Ad Hoc Networks, Vol. 2, No. 4, pp. 351-367, 2004.

[2] S. S. Krishnakumar, R. T. Abler, “Intelligent Actor Mobility in Wireless Sensor and Actor Networks”, IFIP WG 6.8 First International Conference on Wireless Sensor and Actor Networks (WSAN-2007), pp. 13-22, 2007.

[3] M. Y. Sir, I. F. Senturk, E. Sisikoglu, K. Akkaya, “An Optimization-based Approach for Connecting Partitioned Mobile Sensor/Actuator Networks”, IEEE Conference on Computer Communications Workshops, pp. 525-530, 2011.

[4] M. Younis and K. Akkaya, “Strategies and Techniques for Node Placement in Wireless Sensor Networks: A Survey”, Ad-Hoc Networks, Vol. 6, No. 4, pp. 621-655, 2008.

[5] H. Liu ,X. Chu, Y.-W. Leung, R. Du, “Simple Movement Control Algorithm for Bi-connectivity in Robotic Sensor Networks”, IEEE Journal on Selected Areas in Communications, Vol. 28, No. 7, pp. 994-1005, 2010.

[6] A. Abbasi, M. Younis, and K. Akkaya, “Movement-Assisted Connectivity Restoration in Wireless Sensor and Actor Networks”, IEEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 9, pp. 1366-1379, 2009.

[7] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, “Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility”, IEEE Transactions on Computers, Vol. 59, No. 2, pp. 258-271, 2010.

[8] C. Costanzo, V. Loscri, E. Natalizio, T. Razafindralambo, “Nodes Self-deployment for Coverage Maximization in Mobile Robot Networks Using An Evolving Neural Network”, Computer Communications, Vol. 35, No. 9, pp.1047-1055, 2012.

[9] Y. Li, H. Li, and Y. Wang, “Neural-based Control of a Mobile Robot: A Test Model for Merging Biological Intelligence Into Mechanical System”, IEEE 7th Joint International Information Technology and Artificial Intelligence Conference (ITAIC-2014), pp. 186-190, 2014.

[10] A. Llaria, G. Terrasson, O. Curea and J. Jiménez, “Application of Wireless Sensor and Actuator Networks to Achieve Intelligent Microgrids: A Promising Approach towards a Global Smart Grid Deployment”, Applied Science, Vol. 6, No. 3, pp. 61-71, 2016.

[11] E. Kulla, T. Oda, M. Ikeda, L. Barolli, “SAMI: A Sensor Actor Network Matlab Implementation”, The 18-th International Conference on Network-Based Information Systems (NBiS-2015), pp. 554-560, 2015.

[12] J. Kruger, D. Polajnar, and J. Polajnar, “An Open Simulator Architecture for Heterogeneous Self-Organizing Networks”, Canadian Conference on Electrical and Computer Engineering, 2006 (CCECE-2006), pp. 754-757, 2006.

[13] M. Akbas and D. Turgut, “Apawsan: Actor Positioning for Aerial Wireless Sensor and Actor networks”, IEEE 36th Conference on Local Computer Networks (LCN-2011), pp. 563-570, 2011.

[14] M. Akbas, M. Brust, and D. Turgut, “Local Positioning for Environmental Monitoring in Wireless Sensor and Actor Networks”, IEEE 35-th Conference on Local Computer Networks (LCN-2010), pp. 806-813, 2010.

[15] T. Melodia, D. Pompili, V. Gungor, and I. Akyildiz, “Communication and Coordination in Wireless Sensor and Actor Networks”, IEEE Transactions on Mobile Computing, Vol. 6, No. 10, pp. 1126-1129, 2007.

[16] V. Gungor, O. Akan, and I. Akyildiz, “A Real-Time and Reliable Transport (rt2) Protocol for Wireless Sensor and Actor Networks”, IEEE/ACM Transactions on Networking, Vol. 16, No. 2, pp. 359-370, 2008.

[17] L. Mo and B. Xu, “Node Coordination Mechanism Based on Distributed Estimation and Control in Wireless Sensor and Actuator Networks”, Journal of Control Theory and Applications, Vol. 11, No. 4, pp. 570-578, 2013.

[18] K. Selvaradjou, N. Handigol, A. Franklin, and C. Murthy, “Energy-Efficient Directional Routing Between Partitioned Actors in Wireless Sensor and Actor Networks”, IET Communications, Vol. 4, No. 1, pp. 102-115, January 2010.

[19] Y. Kantaros and M. M. Zavlanos, “Communication-Aware Coverage Control for Robotic Sensor Networks”, IEEE Conference on Decision and Control, pp. 6863-6868, 2014.

[20] T. Melodia, D. Pompili, and I. Akyldiz, “Handling Mobility in Wireless Sensor and Actor Networks”, IEEE Transactions on Mobile Computing, Vol. 9, No. 2, pp. 160-173, 2010.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, “Human-level Control Through Deep Reinforcement Learning”, Nature, Vol. 518, pp. 529-533, 2015.

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, “Playing Atari with Deep Reinforcement Learning”, arXiv:1312.5602v1, pp. 1-9, 2013.

[23] T. Lei, L. Ming, “A Robot Exploration Strategy Based on Q-learning Network”, IEEE International Conference on Real-time Computing and Robotics (RCAR-2016), pp. 57-62, 2016.

[24] Riedmiller, M. “Neural Fitted Q Iteration - First Experiences with a Data Efficient Neural Reinforcement Learning Method”, The 16th European Conference on Machine Learning (ECML-2005), Vol. 3720 of the series Lecture Notes in Computer Science, pp 317-328, 2005.

[25] S. Lange, and M. Riedmiller, “Deep Auto-Encoder Neural Networks in Reinforcement Learning”. The 2010 International Joint Conference on Neural Networks (IJCNN-2010), pp. 1-8, 2010.

[26] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and Acting in Partially Observable Stochastic Domains”, Artificial Intelligence, Vol. 101, No. 1-2, pp. 99-134, 1998.

[27] L. J. Lin, “Reinforcement Learning for Robots Using Neural Networks”, Technical Report, DTIC Document, 1993.

[28] “The Rust Programming Language”, https://www.rust-lang.org/.

[29] “GitHub - rust-lang/rust: A safe, concurrent, practical language”, https://github.com/rust-lang/.

[30] “’rust’ tag wiki - Stack Overflow”, http://stackoverflow.com/tags/rust/info/.

[31] X. Glorot, Y. Bengio, “Understanding The Difficulty of Training Deep Feedforward Neural Networks”, The 13-th International Conference on Artificial Intelligence and Statistics (AISTATS-2010), pp. 249-256, 2010.

[32] X. Glorot, A. Bordes and Y. Bengio, “Deep Sparse Rectifier Neural Networks”, The 14-th International Conference on Artificial Intelligence and Statistics (AISTATS-2011), pp. 315-323, 2011.