Mehran University Research Journal of Engineering and Technology
Vol. 40, No. 03, 645-662, July 2021
p-ISSN: 0254-7821, e-ISSN: 2413-7219
DOI: https://doi.org/10.22581/muet1982.2103.17

Reinforcement Learning Based Hierarchical Multi-Agent Robotic Search Team in Uncertain Environment

Shahzaib Hamid[1], Ali Nasir[2], Yasir Saleem[3]

RECEIVED ON 04.07.2019, ACCEPTED ON 16.12.2020

ABSTRACT

Field of robotics has been under the limelight because of recent advances in Artificial Intelligence (AI). Due to increased diversity in multi-agent systems, new models are being developed to handle complexity of such systems. However, most of these models do not address problems such as; uncertainty handling, efficient learning, agent coordination and fault detection. This paper presents a novel approach of implementing Reinforcement Learning (RL) on hierarchical robotic search teams. The proposed algorithm handles uncertainties in the system by implementing Q-learning and depicts enhanced efficiency as well as better time consumption compared to prior models. The reason for that is each agent can take action on its own thus there is less dependency on leader agent for RL policy. The performance of this algorithm is measured by introducing agents in an unknown environment with both Markov Decision Process (MDP) and RL policies at their disposal. Simulation-based comparison of the agent motion is presented using the results from of MDP and RL policies. Furthermore, qualitative comparison of the proposed model with prior models is also presented.

Keywords: Artificial Intelligence, Computational complexity, Intelligent Systems, Robotics.

1. INTRODUCTION

Artificial intelligence (AI) based robotics is gaining significant importance because of popularity in the field of Unmanned Search and Rescue (USR) operations. USR is of utmost importance when it comes to deal with both natural and human created disasters. During these disasters using a robot instead of humans and rescue dogs seems both reasonable and logical. After an unfortunate disaster of World Trade Center (WTC) in 2001, the feasibility and performance of robotic search teams have been studied [1]. In 2011, robots performed surveillance and rescue during earthquake in Japan. According to a report generated by US army corporation small unmanned aerial and ground vehicles along with underwater Remotely Operated Vehicles (ROVs) have played important role in 2010 Haiti disaster [2]. The impact of search and rescue missions on workers has been analyzed ten years later and shown multiple health problems due to weeks spent at WTC in 2001. This study included 27,449 rescue workers from WTC and extensive comorbidity has been reported in these workers regarding both physical and mental health problems and are under treatment [3]. Another report included 919 pulmonary tests with extensive chest scans and methacholine tests to see if extensive lung diseases have been captured by 13,000 WTC rescue workers. Reports included high percentage of obstructive airway diseases that decreased the lung functions of rescue workers [4]. Different reactive airway diseases with increased rate of asthma, pneumonias caused by exposure to dangerous environment is elaborated in [5]. Thus

[1] Department of Electrical Engineering, Superior University, Lahore, Punjab, Pakistan.
Email: shahzaib.hamid@superior.edu.pk (Corresponding Author)
[2] Department of Electrical Engineering, University of Central Punjab, Lahore, Pakistan. Email: a.nasir@ucp.edu.pk
[3] Department of Computer Engineering, University of Engineering and Technology, Lahore, Pakistan.
Email: yasir@uet.edu.pk

robotic usage must be increased to decrease the human life risk. A number of search topologies have been utilized for Urban Search and Rescue (USAR) process. Most of USAR robots used in real world are tele operated. However, researchers are working continuously to provide autonomy to the robots by developing better exploration and mapping methods so that they can help in victim manipulation. Although, for now this is being prohibited due to fear of harming victims [6].

Number of methods can be used for the control of multi-agent systems. Formation control is one of the methods which depends on interaction topologies and sensing capabilities of multi-agent systems. Sensing capabilities developed with the help of sensed variables and interaction topologies depend merely on control variables. Depending on these variables formation control can be divided into position-based control, distance-based control and displacement-based control. Agents achieve their desired formation with Position-based control by actively changing their position according to global coordinate system. Meanwhile, Displacement-based control takes agent to desired formation by relative positions of neighbor agents according to global coordinate system. Local coordinate systems are used to analyze inter agent distances for distance-based control [7]. Other methods like estimation based control, containment control, angle based control, flocking and pure distance based control can also be utilized to achieve formation control. However, due to changing of inter agent distances and absence of global coordinate system local distribution estimators are used [8]. Additionally, local controllers are designed according to local distribution estimators for system having both centralized and distributed control. Consensus problems are also solved by designing a control strategy with the help of local information for linear multi-agent systems. The protocol solves problems for directed agent networks with simple linear dynamics and synchronous feedback information [9]. Meanwhile, the rate of convergence of consensus protocols is also mandatory. Convergence rate for network agent with time delays and switching topology is analyzed. Performance of consensus protocol enhances as time delay robustness decreases. Moreover, communication cost also increases with better consensus performance [10].

2. RELATED WORKS AND MOTIVATION

Planning for deployment of agents in complex indoor area must include information processing, visual sensing and collaboration between them. Sharing can be accomplished by adding a communication layer into existing POMDPs, to add information gathered by the robot with the information gathered by neighbor robots for a specific action [11]. However, this research focused only on robot deployment but task allocation is also an important factor in Multi-Robot search teams. For a limited task length, dependent tasks are allocated through a distributed algorithm assuming maximum consensus for communication among robots. This also excludes the need of shared memory in the system [12]. Furthermore, the size of tasks allocated can be increased by using reduction techniques on POMDPs. The size of input POMDP is decreased leading to an enhanced runtime for POMDP solver. Output policy of the solver is re-evaluated to make sure it works for every input problem [13]. Forming coalitions between robots is necessary after task length becomes satisfactory. For tightly coupled multi-robot tasks, robots must be able to share information during execution assuming all robots are in the field of view of each other. IQ-ASyMTRe architecture resolves the problem of forming executable coalitions and also identifying situation during which feasibility of coalitions change dynamically [14]. However, in case of any problem during these coalition forming transferring all the available data when connection is re-established is not feasible. Hence, data fusion techniques and coordination topology are used which includes sharing belief states eliminating the cost and computation required to communicate all previous information. Data fusion can also be utilized for moving targets [15].

In many industrial applications optimal solution to control problems is a critical issue. Whenever, an original problem is reformulated new optimal solution is concluded according to the shifted dynamics. This can be accomplished with help of an accelerated policy iteration algorithm producing an approximate value function in limited time for Hamilton-Jacobi-

Bellman equations [16]. However, if the input constraints are unknown adaptive control is used due to uncertainty of the systems. Neural networks identify the dynamics of unknown system and a cost function is utilized to optimize the unknown input constraints [17]. Robust control is also used for continuous time non-linear systems in addition to adaptive control. For that purpose, robust control problem is converted to optimal control problem using cost function and a new critic network is developed. Adaptive dynamic programming method is used to solve the optimal problem with facilitation of neural networks [18]. Discrete time non-linear systems can also be solved with the help of policy iteration (PI). It is demonstrated by converging iterative performance index to optimal solution through policy iteration. Neural Network weight convergence is also changed according to the need of discrete time systems [19]. Non-linear discrete systems can also be analyzed with the help of value iteration. Traditional value iteration algorithms (VI) start from zero condition and have to be implemented infinite times to reach optimal control policy. But starting with arbitrary positive semi-definite function optimal conditions can also be achieved and also new termination criterion is developed to guarantee the effect of optimal control law [20]. Applications of PI and VI are numerous in the field of robotics including motion planning with probabilistic satisfaction applying POMDP for simpler tasks and probabilistic models for complex tasks [21]. Partial satisfaction can also be implemented by the help of PI having hard constraints for full satisfaction and soft constraints for partial satisfaction [22]. Probabilistic search strategies can be utilized to analyze the effectiveness of search decision making and also decrease the expected time until this decision is finalized [23].

A vast majority of learning algorithm are based on function learning from a typical training set. However, for real world problems one object may have different meanings. For that reason, each object is assigned a set of labels to express its properties. Thus, they are known as multi-label learning algorithms[24]. Learning can be accomplished through evolutionary computation, game theory, complex systems, agent modeling, reinforcement learning and robotics using team learning or concurrent learning. All the aforementioned areas are used for cooperative learning where agents work simultaneously to complete collective goals. Other approach is to decompose the problem, distribute to agents and construct a solution for specific problem. Both of these approaches are a part of distributed artificial intelligence [25]. Before learning starts the data set can be converted from continuous to discrete values hence moving to qualitative data from quantitative one [26]. Extreme learning machines (ELMs) are also used for topologies without dataset i.e unsupervised (US-ELM) and semi-supervised learning (SS-ELM). Both US-ELMs and SS-ELMs have the same efficiency and learning rate as those supervised ELMs. They can handle unknown data at test time directly and these algorithms are also suitable for multi-clustering [27]. MASs can learn on the basis of coordinated behavior between them. Independence of agent can be exploited for efficient coordinated learning (CL) without the information about location of coordination or whole knowledge about the environment. There exists a tradeoff between single agent learning process and coordinated learning process [28]. CL can also be depicted with the help of reinforcement learning. There also exists a tradeoff between exploitation and exploration for MARL technique. The approach adopted is similar to single agent model. It is also kept in mind other than the value of information which helps in making an exploration policy [29]. Cooperative RL can also implemented on hierarchical (HRL) framework for which learning is decentralized. Algorithm focuses on performing individual subtasks also defines the order of those subtasks alongside coordination between these agents. Cooperative subtasks are defined with coordination among agents to improve the performance of overall task[30].

Deep reinforcement learning has been used extensively in the recent era to solve a wide range of complex decision making problems which is depicted recently in the fields such as health care, robotics, smart grid and other practical application [31]. Deep reinforcement learning is thus revolutionizing the way research is seen in the field of artificial intelligence by developing a unique understanding of autonomous systems and the visual world. As an example take a robot whose policy is learned with the help of camera attached to it in the real world environment. Another example of learning based real environment is learning

to play video games by looking at pixels [32]. Most of the learning is done with data which is available beforehand thus there is need of methodologies which help decreasing the errors which are introduced in the extrapolation. For that off-policy learning method the agent has to be restricted with the help of batch constraint methodology thus decreasing the action space which makes the system behave more like an on-policy learning system [33]. Another problem that comes with it is the over fitting problem which can be taken into account. Agent normally over fit to large training sets which can be solved by using deep convolutional architectures to improve generalization. Methods such as regularization, augmentation and normalization are also used [34].

All of the aforementioned techniques are related to multi-agent systems but some do not present a viable method for agents to learn during exploration [1-17], also agents which do learning themselves cannot detect errors on their own and hence those systems are not fault tolerant [18-24]. For exploration, agents must also be equipped with making decisions in less time, so there should be limited computation time for each agent. All of these problems including the consensus between agents are important for agent movement in an unknown environment. This paper addresses these problems in the best way possible.

In this paper, we have extended the use of HRL on MAS by using Q-learning on robotic search teams. The proposed algorithm initializes by defining an MDP model for leader-follower approach. Furthermore, a Q-learning algorithm is proposed on MDP model to construct a Q-look-up table. Number of agents and search areas are flexible and can be altered according to need. Results for Q-learning depicts a better computational efficiency as well as the time to perform each task. This scheme makes local agents independent in taking actions towards the goals assigned by leader agent. Moreover, these assigned goals are completed by each local agent which is the main criteria for understanding how each agent is performing. Hence, this approach like other learning techniques makes sure that agent takes its own decisions depending on the algorithm and exhibits a number of advantages including fault tolerance, error detection, goal completion, standby agents and coordination between local agents and the leader agent. The explanation of all these factors is given as:

Goal completion:
Goal completion from the perspective of robotics AI means completion of a specific task, assigned to an agent, in minimal amount of time. It is the most important factor in this research work as its purpose is to directly depict the agent's success in an uncertain environment, which is the key objective of this work as aforementioned.

Fault tolerance:
Fault tolerance includes the impact of fault on any system as it is the amount of acceptable error. It is an essential factor in this research work because if the local agent does not responds according to the leader agent, it is considered faulty and is excluded from the system temporarily until a connection has been made or fault has been extracted. So in this work its main advantage is to create foundation for the error detection.

Error Detection:
This factor directly correlates with fault detection as in this system it ensures that all the local agents are working properly. If for some reason agents stop responding then it is considered as faulty and the specific error is detected subsequently. It can be due to multiple reasons including

- Fuel leakage or low fuel due to exploration
- Agent is in hazardous environment and is destroyed
- The connectivity between local and leader agent is lost.

Standby Agent:
The agents that become faulty due to the reasons explained above are considered as standby agents. In this work if leader is unable to confirm the location and update its lookup table because of faulty entries from the local agents, they are considered in standby. This factor helps leader agent to keep the policy updated without the probability of having any faulty entries.

Agent Coordination: leader and local agents. The main purpose of leader

This is a hierarchical model for coordination between agent in this work is to assign specific goals to the local agents. Until the goal completion, the local agent updates the policy as well as takes decisions on the basis of real world environment. Therefore, the coordination between these agents is of utmost importance as based on the local agents continuous feedback the leader agent updates its state table for each agent. If the agent is in hazardous place and is destroyed than the leader agent will not be able to coordinate with that local agent. During this scenario, the agent is considered to be on standby. Thus fault tolerance and error detection also depend on the coordination between the agents so agent coordination is a continuous process and is very important for communication between the leader and local agents in all above mentioned processes.

This research develops a methodology in which decision making policy is developed with the help of data coming from the environment. The camera mounted on the agent will bring data values to the agents in the hierarchical system and the policy will be updated according to the incoming data. Another advantage of this research is that agents which are present in the field can update the local policy on their own and later send the updated results to the main leader agent or control room. The main contribution of this work are explained above and are summarized as follows:

- Development of optimal policy for each local agent. This is achieved through cameras canvasing of the real environment
- Development of overall policy of the system through the data collection from all the local agents.
- Making the system fault tolerant by introducing factors such as standby agent, fuel level and agent coordination.

The rest of this paper is divided as follows. Section 3 of this paper depicts system constraints and methodology. Section 4 of this paper includes results and comparisons Section 5 presents the conclusion.

3. SYSTEM CONSTRAINTS AND METHODOLOGY

This research treats the problem of p-agents having to search through an area with m-partitions. Total area of these partitions is known, alongside with number of agents and number of partitions. But the status (hazardous/friendly) of partitions is not known and recognized with search. Basic purpose of this research is to facilitate agent's movement. Some important constraints which affect the movement are

- Agents interaction
- Partition status
- fuel status.

If an agent A is in partition 1 and wants to move to any partition 2 but another agent is already present in that partition 2, the agent A will restrict its movement and not move to partition 2. Hence agent movement is dependent on position. Another factor this system is dependent on is partition status. If an agent A wants to move to partition 2 but that partition is already declared hazardous by the agent which visited it before. In that scenario the agent will not move to partition 2. Third constraint under consideration is fuel status. Agent A will stop its movement to any partition if it identifies its fuel level as empty or close to empty.

In practice, such situations may occur when there is fire in a building. The rooms and halls can be considered as partitions and rooms where fire has spread may be considered hazardous whereas rooms where fire has not spread may be regarded as friendly. Also this topology can be used by Mars rovers in which case the partitions would be imaginary i.e. without physical walls or separations. Sample area for this research is depicted in Fig. 1.

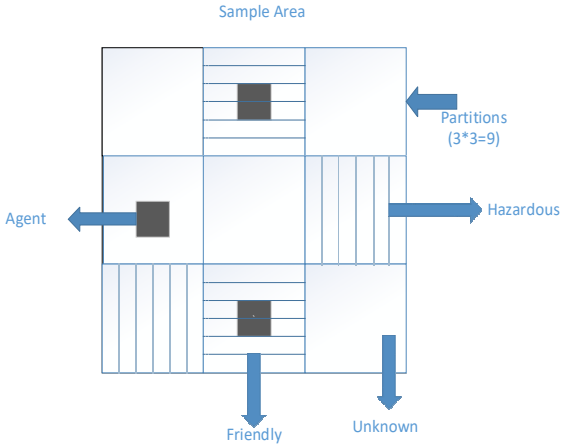


Fig. 1: Sample Area

3.1 MDP Model

Leader Agent MDP Model: Leader MDP model consists of four elements including states, actions, reward function and transition probability. Major tasks of the leader agent involve assigning the agent to different search area partitions and managing the operating modes of the agents. It has been assumed that the member agents are able to communicate the status of their power/fuel levels and fault(s) to the leader agent.

States: There are five variables which construct the MDP model for leader agent:

- Agent state.
- Goal status.
- Partition assigned.
- Partition status.
- Standby agent.

$$S = \{s_1, s_2, \cdots, s_n\} \quad (1)$$
$$s_i = \{A, B, C, G, Sb\}, i\epsilon\{1,2,\cdots,n\}$$
$$A = \{a_1, a_2, \cdots, a_p\}, a_j\epsilon\{0,1\}, j\epsilon\{1,2,\cdots,p\}$$
$$B = \{b_1, b_2, \cdots, b_r\}, b_j\epsilon\{0,1,2\}, j\epsilon\{1,2,\cdots,r\}$$
$$C = \{c_1, c_2, \cdots, c_p\}, c_j\epsilon\{1,2,\cdots,r\}, j\epsilon\{1,2,\cdots,p\}$$
$$G = \{g_1, g_2, \cdots, g_q\}, g_j\epsilon\{0,1\}, j\epsilon\{1,2,\cdots,q\}$$
$$Sb = \{sb_1, sb_2, \cdots, sb_p\}, sb_j\epsilon\{0,1\}, j\epsilon\{1,2,\cdots,p\}$$

Equation (1) depicts the state space starting from 1st state to nth state where each state is calculated based on five types of variables. Policies for this system can be computed offline so the computation limit is dependent on the processor.

First type of variable in the state is the status of an agent (A). Each agent can either be in a working state ($a_i = 1$) or a failed state ($a_i = 0$) and there are total p agents. In second variable b_i represents the i^{th} partition of the area being unknown ($b_i = 0$), friendly ($b_i = 1$), or hostile ($b_i = 2$). Third variable is agent assignment variable, represented by C, for the assignment of the i^{th} agent. An agent can be assigned to any of the partitions of the search area. Value of c_i identifies the partition to which i^{th} agent is currently assigned. Next variable is the goal variable (G) represented where g_i is the i^{th} goal. This variable represents whether the goal has been achieved ($g_i = 1$) or not ($g_i = 0$). The goal status is the most important variable which undertakes the concept that if a certain problem given to the agent is solved as an example if the agent is a robot on a rescue mission and it sees a person in danger and gives feedback to the control room that a person has been found and this will be considered as goal completed. In real life problem, goal could be to find a survivor or an object of interest. Fifth variable (Sb) is related to standby agent which shows if the agent is standby ($sb_i = 0$) or not ($sb_i = 1$). Therefore, if an agent is not responding to the leader agent it is considered as standby agent. The reasons for not responding can be multiple like its battery can drain out, its connection can be cut, the agent can also be destroyed in a hazardous environment due to dangerous conditions. Thus all of these problems can accumulatively generate an error in the system which makes the agent with that corresponding error standby. This property makes the system fault tolerant as it considers the fact that any of the agents can be on fault and the leader agent will keep on trying to connect with local agent and after a fixed amount of time will send another agent for better understanding of the problem.

Actions: In this paper, we have considered p agents and r partitions of the search area. Hence, in total there are p×r agent assignment actions. Here the assumption is that the agent assignment is done sequentially (one by one) but it is not a limitation. For example, if two agents can be assigned at a time then actions will be (total agent pairs*areas to be assigned including current partition as twice). Therefore, the computational complexity depends on hardware and varies linearly with change in actions. There are actions to put the agent on stand-by mode in case of fault of lack of electrical power/fuel. There is also an action NOOP (no-operation) for cases where no change in current assignment or mode of the agents is required. Actions are shown in (2).

$$M = \{\mu_{1,1}, \mu_{1,2}, \cdots, \mu_{p,r}, \mu_{sb1}, \mu_{sb2}, \cdots, \mu_{sbp}, NOOP\} \quad (2)$$

Reward Function: Our reward function has the following priority order:

- Goal achievement.
- Considering the standby agents.
- Avoiding more than two agents in a single partition.

- Avoiding hostile areas.

Defining the reward function is the basic problem during MDP model design. State and actions can be changed according to the situation and transition probabilities are defined according to the statistical data. However, reward function must be aligned to the objectives of the problem and with the outside environment otherwise, it will be of no practical use.

Proposed reward function is presented in (3). It consists of six terms each scaled by a positive coefficient (λ).

$$R(s) = \lambda_1 e^{\sum_{i=1:q} g_i} + \lambda_2 e^{\sum_{i=1:p} a_i} - \lambda_3 e^{\sum_{i=1:p} sb_i \times a_i} \dots + \lambda_4 e^{\sum_{j=1:r} \alpha_j} - \lambda_5 e^{\sum_{i=1:p} \sum_{j=1:r} \beta_{i,j}} - \lambda_6 e^{\sum_{i=1:r} (b_i = 0)}$$

$$\alpha_j = \begin{cases} 1 & ab_j,\ j \epsilon \{1,2,\cdots,r\} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$cb_j = \sum_{i=1}^{p} (c_i = j),\ j \epsilon \{1,2,\cdots,r\}$$

$$\beta_{i,j} = \begin{cases} 1 & \alpha_i > 1, \text{and } b_j = 2 \\ 0 & \text{otherwise} \end{cases}, \quad i \epsilon \{1,\cdots,p\}, j \epsilon \{1,\cdots.r\}$$

First expression in (3) depends on the goal status. More the goals completed more shall be the reward having maximum value when all goals have been achieved. Consequently, if the values for these goals are zero then reward function will be diminished. Second term of equation includes the status of agents in a specific area. Higher reward for more agents in working condition. This term supports the assignment of agents to friendly areas so that their failure probability is minimized. Third term penalizes the standby agents so that no agents are put on standby unless the agents have failed. Fourth term encourages efficient utilization of search agents by penalizing more than two agents in one search partition. If the number of agents cb_i in the i^{th} area are two or less then the reward parameter α_i will decrease. Fifth term in the reward function penalizes assignment of agents to the hazardous areas. Note that if one agent is present in this type of area, it will not change reward value (this is to support the achievement of goals since some goals are likely to be achieved in the hazardous partition). Sixth term penalizes the unexplored partitions.

Transition Probability: Transition probability from one state(s) to another state(s') is given by T(s,μ,s'). It is combination of all possible probabilities corresponding to all of the states for movement of agents in search area. These probabilities depend on a number of factors continuing to assignment or reassignment of these agents. Reassignment is a problem because agents must change their location according to some specific reason. However, there is no modeling defined for transition between states. So, for transition to happen an external algorithm is used to determine whether the agent position should be altered or not. Probability distributions of all these events can be calculated using the conditional independence relations among the variables involved in the problem. Such relations can be represented with the help of a Bayesian network. Bayesian network for leader agent is shown in (Fig. 2). Note that the standby variable (Sb) is not part of Bayesian network because it is deterministic and it does not affect any other random variable. On the other hand, agent assignment variable (C) is part of the Bayesian network despite being deterministic because it affects other random variables. Remaining variables i.e. agent status (A), search area partition status (B), and goal status (G) are random variables. Agent status depends upon existing agent status and assignment of agents. If an agent is assigned to a hazardous partition, then its probability of failure is expected to be higher compared to the one assigned to friendly partition. Partition status depends upon existing partition status and agent assignment because the probability of a partition being friendly or hostile depends upon whether it is unknown and whether a search agent is assigned to it. An unknown partition shall remain unknown if no agent is assigned to it. Finally, achievement of goals depends upon current status of goal achievement and assignment of agents. Goal achievement probability may be higher for agents assigned to certain "important" partitions.

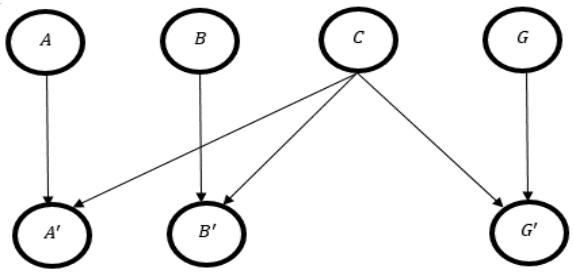


Fig. 2. Bayesian Network for Leader Agent

Practically, the conditional probabilities involved in the problem can either be learnt online (starting from initial offline guess), or may be available from statistical data. This paper does not discuss the methods of obtaining the probabilities rather it shows how to use them in an MDP model.

3.2 Member Agent MDP Model

States: States of the member agent depicts the variables involved in the problem.

$$S = \{s_1, s_2, \cdots, s_m\} \quad (4)$$

$$S_i = \{ps, cp, G, energy, cs, block, Bc, f\}$$

$$G = \{g_1, g_2, \cdots, g_q\}, g_j \epsilon [0,1], j \epsilon \{1,2, \cdots, q\},$$

$$Bc = \{bc_1, bc_2, \cdots, bc_x\}, bc_j \epsilon \{0,1,2\}, j \epsilon \{1,2, \cdots, x\}$$

$$energy \epsilon \{0,1, \cdots, y\}, cs \epsilon \{0,1, \cdots, z\}, ps \epsilon \{0,1\}$$

$$cp \epsilon \{1,2\}, block \epsilon \{1,2, \cdots, x\}, f \epsilon \{0,1\}$$

$$x, y, z \epsilon \{0,1,2, \cdots\}$$

There are eight variables in (4). First variable in this equation is partition status ps. Partitions are subdivided into blocks. Partition status is '0' if the partition is unexplored and is '1' if the partition is explored with no re-visits required. Next variable in this list is cp this variable refers to current partition. It shows whichever partition the agent is present in. Next variable is G which tells us about the status of the goals. If one of those goals is completed its state value will be changed to '1'. Next variable is energy which tells about the level or fuel present. Total fuel/energy capacity has been divided into y+1 discrete levels. Fifth variable included is the companion status cs, it tells about the status of companion agents present in partition 1 represented as '1' or in partition 2 represented as '2' or not in any partition represented as '0' (in general there could be z partitions). A sample structure of search area with two partitions (z = 2) and seven blocks (x = 7) is shown in Fig. 3. Other variable is block tells us the exact location where this agent is available. Seventh variable present is Bc which includes status of each individual block where unknown represented by '0', friendly represented by '1' and hazardous represented by '2'. Last state variable is fault variable shown as f, it will be '0' if there is no fault present and '1' for physical or logical faults.

Actions: Actions means the number of steps agent can take to move from one state to another. In most cases there are five basic types of actions given as

(i) Left
(ii) Right
(iii) Straight
(iv) Back
(v) Stay

All these actions are given in the equation below. Each action taken takes the agent to another block (except for stay action).

$$M = \{MovLeft,\ MovRight,\ MovStraight,\ MovBack,\ Stay\} \quad (5)$$

Any agent can move within its partition on a specific action until there assignment is changed. Some types of actions are given below (Fig. 3.)

(i) Taking left from block 2 takes agent to block 1.
(ii) Taking left from block 1 in partition 1 will not make agent move to any other block.
(iii) Right action from block 3 takes the agent to block 4.
(iv) Taking front from block 4 takes agent to block 2.
(v) Back action from block 5, 6 takes agent to block 7.

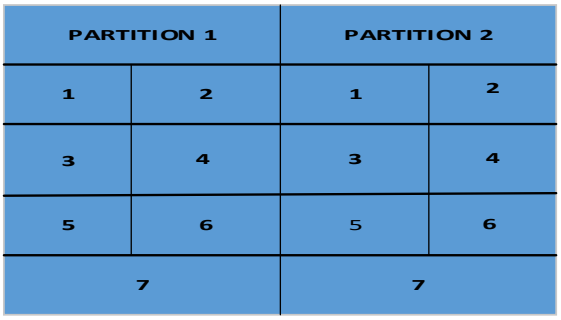


Fig. 3: Area Distribution

Reward Function: Reward function must be designed with caution. It is based on the MDP model created, but it can be changed according to the need of the search teams. For member agent reward function is based on these basic parameters.

(i) Agent should cover maximum area.
(ii) To make sure that not more than two agents resides in same partition.
(iii) Agent should be able to find if it is faulted and communicate it back to the leader agent.
(iv) Agent should be familiar with hazardous blocks in the partition.
(v) Agent should maximize the battery usage.

$$R(s) = \sum_{i=1}^{5} \alpha_i r_i, \ \alpha_i \in [0, \infty) \quad (6)$$
$$r_1 = \sum_{i=1}^{x} (bc_i > 1) + \beta_1 ps, \beta_1 \epsilon (0, \infty)$$
$$r_2 = (cp \oplus cs)$$
$$r_3 = energy$$
$$r_4 = \sum_{i=1}^{x} (block == i) \times (bc_i \neq 2)$$
$$r_5 = f$$

The first expression of r_1 in equation (6) motivates the agent to explore the search area (by exploring the partition in which it is currently present). As the block exploration is increased the reward function also increases with it. For this scenario value of α_1 is taken to be a large positive number so that exploration can be increased. Other priority is to make sure that agent avoids the blocks in which another agent is already present. For that r_2 is introduced. The value of this variable is also very important and increases the reward as a whole. Therefore, the value of its constant is also taken to be large positive number (e.g. 100). Third expression represents the level of fuel/battery present. Its value will alter according to fuel/battery being low, medium or full. For the missions where level of fuel/battery is most important, α_3 is taken to be about ten times larger than that of α_2 on the other hand if exploration is more important, then lower value of α_3 may be used. Full value of energy will take reward function to maximum value. Fourth expression is for avoidance of hazardous blocks; these blocks can have temperature value high or terrain may be unsustainable. Hence, there can be a number of reasons to avoid these blocks which leads to failure of agent. It is also a priority of leader agent to not assign agents in these blocks. Last expression is for the faults present in an agent. It will be maximized if there is no fault in the agent so its constant is taken to be a large positive value (e.g. 1000) which may lead to reduction in risk taken by the agent while performing search.

Transition Probability: Transition probability explains about the change from one state to another. Bayesian network for transition probabilities of member agent state is shown in (Fig. 4).

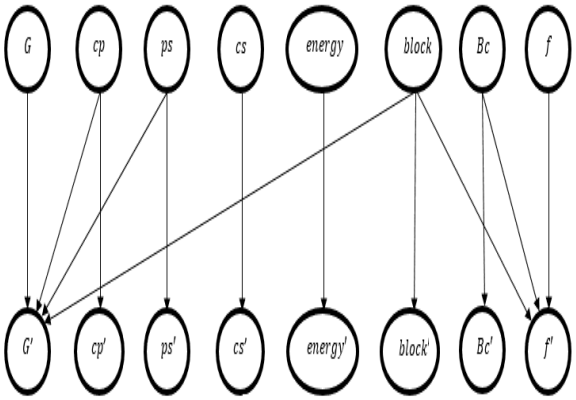


Fig. 4: Bayesian Network For Member Agent

Two most important probabilities in (Fig. 4) are that of goal achievement and occurrence of fault. Goal achievement depends upon the current location of the member agent (current partition and block), status of the partition in which the agent is located, and the existing status of the goal achievement. This probability guides the agent about which blocks have more potential of rendering the goals achieved. Fault occurrence probability depends upon the location of agent and status of the blocks (high fault probability in hazardous blocks). Other variables in the state space transition based on their current value or the action taken by the agent. Note that the actions are not included in the Bayesian network of (Fig. 4). However, consumption of energy does depend on action taken (but this dependence is often deterministic rather than stochastic).

3.3 RL Learning Model

There are different types of reinforcement learning which are differentiated on the basis of their learning method. Supervised learning has models and training sets upon which agent acts. However, unsupervised learning is quite opposite with no training set. But there is another type which has a feedback system in order to make decisions, it is known as reinforcement learning where positive or negative reinforcements are used for more effective and efficient decision making.

These are two types of such learning systems

(i) Utility Learning.
(ii) Q-Learning.

In utility-based learning agent learns about the utility function and selects actions that maximizes the expected utility. However, it must understand about the environment better or have the knowledge before even starting to evaluate actions and making decisions. In this paper, agent learns action-value function which is utility of taking action in a given state. So, there is no model required but the knowledge of environment becomes very limited.

Q-Learning: In Q-learning we Define action-value function. In addition, we assign expected utility by taking actions in states known as Q-values. It also allows decision making without use of any model.

It can also be used for iterative learning but need to learn model thus known as TD Q-learning. In the equation given below maximum value is computed with respect to Q values with all actions. With reward function given for next state and multiplying it with alpha which is learning rate we will get the new value of Q-function. This is represented in equation (7).

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + A \quad (7)$$
$$A = \alpha_t(s_t, a_t)\left[R(t+1) + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right]$$

Proposed Model: This proposed model is different from simple TD Q-learning equation. Instead of taking the maximum value out of all possible values, value function is defined which is multiplied with the discount factor.

Value Function: Value function V_{t+1} tells about the expected overall value of states under given policy. No type of reward is required to compute this value. But in turns it helps in making the reward functions. In the equation given below Q value of next state is computed on the basis of current value and next state value function.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + B \quad (8)$$
$$B = \alpha_t(s_t, a_t)[R(t+1) + \gamma V(t+1) - Q_t(s_t, a_t)]$$

In our proposed model There are three local agents and a single leader agent. A local agent starts learning and changes state based on the environment. So this is the state transition method in which agent is in state s1 and switches to state s2 based on the environment.

On the other hand, local agent also communicates its known state to leader agent so that it can help avoid other local agents from hazardous places. Thus leader agent learns on the basis of local agent changing and assigns other local agents on this basis.

4. RESULTS AND COMPARISONS

4.1 Qualitative Comparison

Upon the basis of certain techniques, a comparison is given for different properties in (Table 1). All the techniques mentioned in (Table 1) are used for the control of an agent in any area. The main qualities that this paper addresses are Uncertainty Handling, Agent Coordination, Area Classification and Learning Capability. The proposed model is this paper can handle all of these problems but other approaches lack in providing a solution for some of these problems.

A comparison between Reinforcement learning technique and previous MDP technique mentioned in [35] demonstrates an addition of learning topology and fault detection based on states in which agent is present. Proposed model also has enhanced efficiency

Table 1: Qualitative comparison between techniques

	Uncertainty Handling	Agent Coordination	Area Classification	Learning Capability
Formation Control	x	X	X	x
Data Fusion Control	x	√	X	X
Adaptive Control	√	X	X	X
MDP	√	X	√	X
Extreme Learning Machines	√	X	X	√
HRL(Q-Learning)	√	√	√	√

due to less dependency on the leader agent for decision making. Out of these models MDP modeling is conventional method and cannot be compared with any other method but this paper depicts an improved MDP version with addition of Q-learning.

4.2 Simulation Based Comparison

Let's assume that rescue robots are trying to explore through a building under fire with a goal to search for humans. Unknown constraints are if any room of the building has fire or smoke it is declared as hazardous. If room does not contain fire or smoke it is considered as friendly and if room is not explored yet it is considered as unknown. Building is divided into different search areas which are further divided into partitions. This is also demonstrated in (Fig. 5).

Building is divided into two areas with each area having two partitions. Out of these partitions Partition 1 of Area 1 and Partition 2 of Area 2 are not under fire or smoke thus they will be considered friendly. On the other hand, Partition 1 of Area 2 is under fire and is labeled as hazardous. Partition 2 of Area 1 is still unknown as it is not explored yet.

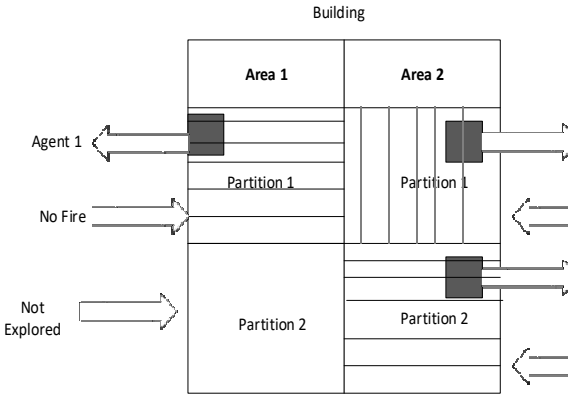


Fig. 5: Building Sample

All the policies are calculated on MATLAB 2013a installed on a system with windows 10 and with processor Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz 3.10 GHz.

Leader Agent: There are two types of simulations for agent policies. First simulation is for leader agent assigning other agents without the use of learning. This policy is taken from policy evaluation and all of the states are taken randomly. The resultant of these states are shown both by MDP policy making and Q-learning policy making.

Initial Conditions: Initial conditions for the simulations are given in Table 2.

State Diagram: As shown in (Table 3) for case 'A', all of the goals are unachieved and agent1 is assigned to partition 1, agent 1 and 2 are assigned to partition1 where as agent3 is assigned to partition 2. For case 'B', second goal is achieved with agent assignment same as case 'A', partitions for both these cases are unknown. In case 'C' goal status is changed to achieved with agent 1 and 3 assigned to partition 1 and agent 2 assigned to partition 2.

For next three cases 'D', 'E' and 'F' partition status is hazardous for first partition but unknown for second one. For each case 2 agents are assigned in partition number one. Goal status is also different for each state. As first being in unachieved state, for second goal number 2 is achieved and for third case both goals are achieved.

Last three cases have first partition in hazardous state and second partition is unknown. For all cases two agents are given to partition no 1, so result is determined. Goal status is also altered same as above.

MDP Policy: Each of the states given gave the results as shown in Table 4.

Learning Policy: For policy with learning, we have only taken references from the MDP policy but it is not necessary. These values can also be taken upon intuition or by choosing randomly. In the same states the result for learning policy are

- Case 'A' of (Table 5), Action 2 has been taken which takes agent 1 to partition 2 to fulfill the goals alternatively.
- Case 'B' recommends to take action no. 7 which takes agent 1 to a standby state because only one goal is remaining.
- Case 'C' recommends action 7 which means agent 1 is taken to standby state because both of the goals are achieved.

Table 2: Initial Conditions of Leader Agent				
	Status of Area 1	Status of Area 2	Status of Goal 1	Status of Goal 2
A	UNKNOWN	UNKNOWN	UNACHIEVED	UNACHIEVED
B	UNKNOWN	UNKNOWN	UNACHIEVED	ACHIEVED
C	UNKNOWN	UNKNOWN	ACHIEVED	ACHIEVED
D	UNKNOWN	FRIENDLY	UNACHIEVED	UNACHIEVED
E	UNKNOWN	FRIENDLY	UNACHIEVED	ACHIEVED
F	UNKNOWN	FRIENDLY	ACHIEVED	ACHIEVED
G	HAZARDOUS	UNKNOWN	UNACHIEVED	UNACHIEVED
H	HAZARDOUS	UNKNOWN	UNACHIEVED	ACHIEVED
I	HAZARDOUS	UNKNOWN	ACHIEVED	ACHIEVED

Table 3: States Status of Leader Agent for Initial Conditions					
	Agent State	Goal Status	Partition to which agent is assigned	Partition Status	Standby Agent
A	111	00	112	00	111
B	111	01	112	00	111
C	111	11	121	00	111
D	111	00	112	01	111
E	111	01	211	01	111
F	111	11	121	01	111
G	111	00	121	20	111
H	111	01	211	20	111
I	111	11	112	20	111

Table 4: States status of Leader Agent after implementing MDP Policy					
	Agent State	Goal Status	Partition to which agent is assigned	Partition Status	Standby Agent
A	111	00	121	10	111
B	111	00	112	01	111
C	111	10	112	01	111
D	111	00	121	10	111
E	111	00	122	10	111
F	111	10	112	01	111
G	111	00	121	11	111
H	111	00	121	11	111
I	111	10	112	01	111

- Case ‘D’ recommends action 2 and take agent 1 to partition 2 because partition 2 has a status of friendly.
- Case ‘E’ recommends action 7 to take agent 1 to standby because one goal is achieved.
- Case ‘F’ takes action 7 to take agent 1 to standby because both goals are achieved. At this point all agents could go to standby. Action is just taken randomly here.
- Case ‘G’ refers to action 2 which takes agent 1 to partition 2 because partition 1 is hazardous.
- Case ‘H’ takes action 7 to take agent 1 to standby mode because one of the goals is fulfilled.
- Case ‘I’ takes action 7 to take agent 1 to standby mode because partition 1 is hazardous and goals are completed.

Comparison: Both MDP and Q-learning models are implemented for leader agent with MDP being a more conventional and common approach. Comparison of MDP and Q-learning technique is demonstrated with practical results to ensure its reliability.

For the first case represented in Fig. 6, the agents are moved from one area to other taking agent 2 to area 2 and agent 3 to area 1 with the help of MDP process.

But with the help of Q-learning one agent remains in area 1 and two agent moves to area 2 because of the goal achievements.

For case C represented in Fig. 7, both of the goals are accomplished. However, one of the agents should move to standby in area 1. Through MDP modeling places of agents are swapped but neither of them moves to standby state. That is accomplished with the help of Q-learning which can also be seen in Table 5. Case E is similar to Case C with slight difference that one of the goals is achieved not both of them as represented in Fig. 8. Still, being a more optimized method Q-learning moves the agent in area 1 to standby rather than swapping positions like in MDP model.

The scenario shown in Fig. 9 is also similar in which agents are taken to standby with Q-learning but not with MDP process. Hence, the performance of Q-learning can be seen from the given cases in which agent does not moves from one place to another but goes to standby.

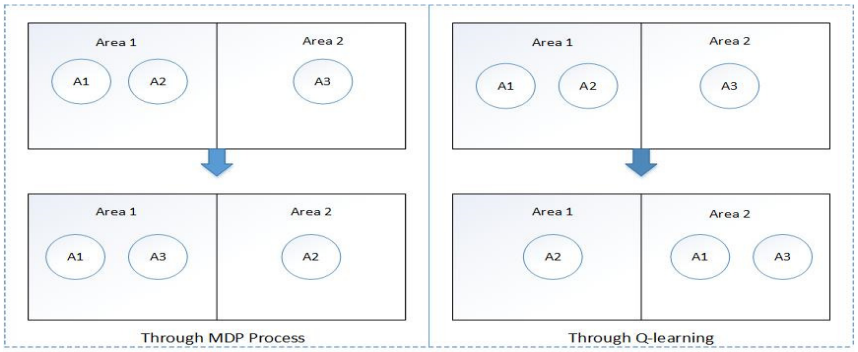


Fig. 6. Case A of Agents Movement

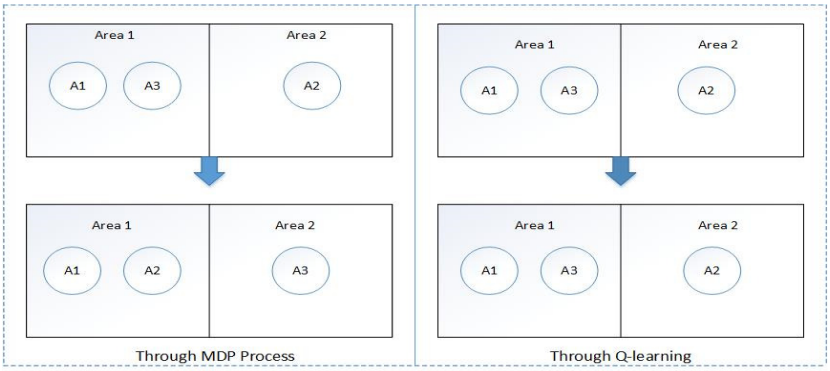
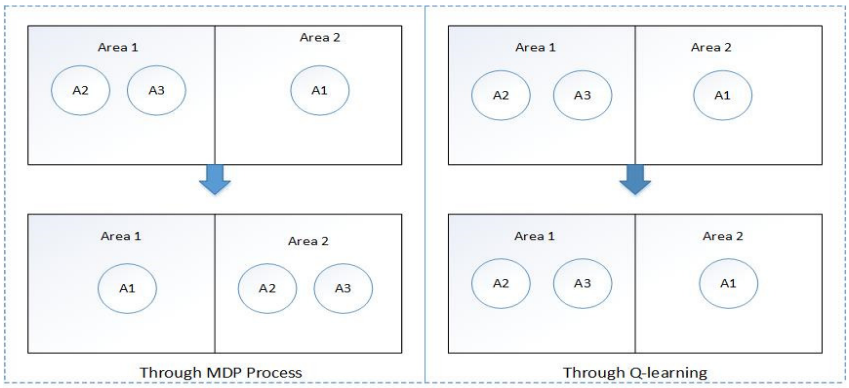


Fig. 7: Case C of Agents Movement

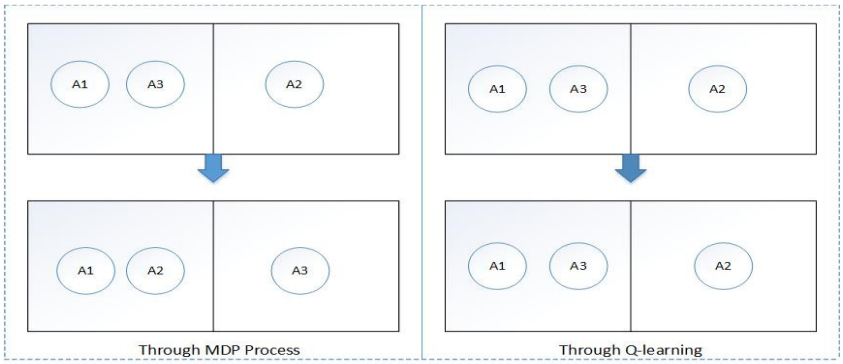
Table 5: States Status of Leader Agent after implementing Q-Learning

	Agent State	Goal Status	Partition to which agent is assigned	Partition Status	Standby Agent
A	111	00	212	00	111
B	111	01	112	00	011
C	111	11	121	00	011
D	111	00	212	01	111
E	111	01	211	01	011
F	111	11	121	01	011
G	111	00	221	20	111
H	111	01	211	20	011
I	111	11	112	20	011



Case E

Fig. 8: Case E of Agents Movement



Case F

Fig. 9: Case F of Agents Movement

4.3 Member Agent

Initial status for local agent taken by intuition are shown in Table 6 and states status for local agent are shown in Table 7.

MDP Result: For case 'A' represented in Table. 8, member agent the block number in which agent is present is changed from 1 to 7 with a difference that companion is present in the same partition.

For case 'B' represented in Table. 8, block number is again changed from 3 to 7, as can be seen from the table that both of these cases have a fault which is either physical or sensor. Hence, it does not matter to which block agent moves to.

For case 'C' represented in Table. 8, the agent moves from block number 4 to block number 7, with no fault present the result is changed block because companion is present in the same block.

For case 'D' represented in Table 8, there is again a fault present in the agent in the initial state, so moving to other blocks does not matter.

For case 'E' represented in Table. 8, the agent moves from block 1 to block 7 because block number 7 is friendly and number 1 is not.

For case 'F' represented in Table 8, agent moves to block 6 because it is the only block which is not hazardous.

Learning Result: The results computed at the end of learning policy gives the result shown in Table 9**.**

- Takes action move right, hence moves the agent to block 2 of partition 1.
- Takes action move right, hence moves the agent to block 4 of partition 1 because block 3 is hazardous.
- Takes no action because block is friendly and one of the goals is completed.

- Takes action no. 2 and moves to partition number 2 because block 7 of current partition is hazardous.
- Takes action no. 2 and moves to second block just for the sake of goals.
- Takes action no. 3 to take agent to block number 6 of same partition.

Table 6: Initial Status for Local Agent

Case ID	Partition Status (PS)	Current Partition (CP)	Companion Status (CS)	Status of Goal 1	Status of Goal 2
A	Unexplored	P1	Not Present	Unachieved	Unachieved
B	Unexplored	P1	Not Present	Unachieved	Unachieved
C	Unexplored	P1	Present	Achieved	Unachieved
D	Unexplored	P1	Present	Unachieved	Achieved
E	Unexplored	P1	Not Present	Unachieved	Unachieved
F	Explored	P2	Not Present	Achieved	Achieved

Table 7: States Status of Local Agent for Initial Conditions

Case ID	Partition Status	Current Partition	Goal Status	Energy	Companion Status	Block	bci	F
A	0	1	00	0	0	1	1111211	1
B	0	1	00	0	0	3	1121211	1
C	0	1	10	0	1	4	2111221	0
D	0	1	01	1	1	7	2221212	1
E	0	1	00	0	0	1	1121220	0
F	1	2	11	2	0	7	2222212	0

Table 8: States Status of Leader Agent after MDP Policy

Case ID	Partition Status	Current Partition	Goal Status	Energy	Companion Status	Block	bci	F
A	0	1	00	2	2	7	1111211	1
B	0	1	00	2	2	7	1121211	1
C	0	1	00	2	2	7	2111221	0
D	0	1	00	2	2	6	2221212	1
E	0	1	00	2	2	7	1121220	0
F	1	1	10	2	1	6	2222212	0

Table 9: States Status of Leader Agent after Q-Learning

Case ID	Partition Status	Current Partition	Goal Status	Energy	Companion Status	Block	Bci	F
A	0	1	00	0	0	2	1111211	1
B	0	1	00	0	0	4	1121211	1
C	0	1	10	0	1	4	2111221	0
D	0	2	01	1	1	7	0000001	1
E	0	1	00	0	0	2	1121220	0
F	1	2	11	2	0	6	2222212	0

Comparison: According to states of member agents, the MDP model is implemented and learning policy is also determined. This section presents a comparison between agent movement with Q-learning and agent movement with MDP model.

The corresponding Case in Fig. 10 takes block number 3 and 5 as hazardous. With the help of MDP this agent moves to block number 7 moving from block 4 and 6. By this route block 5 is also avoided completely. For Q-learning, agent moves from block 3 to block 4 taking the agent to safety by a single movement.

For the first case in Fig. 11, the only hazardous block is 5, the local agent is present in block number 1 in this scenario. With simple MDP model this agent is taken to block number 7, only taking into consideration some of the problems. But with Q-learning the agent moves to block number 2 after taking one right. This way agent does not travel long for non-hazardous block.

For Case F in Fig. 12, all of the blocks present in a block are hazardous leaving block number 6 or 4. MDP result shows the movement from this block to block number 6 of the same partition. The result from Q look up table shows that agent moves to partition no. 2 because already companions are present in the same partition.

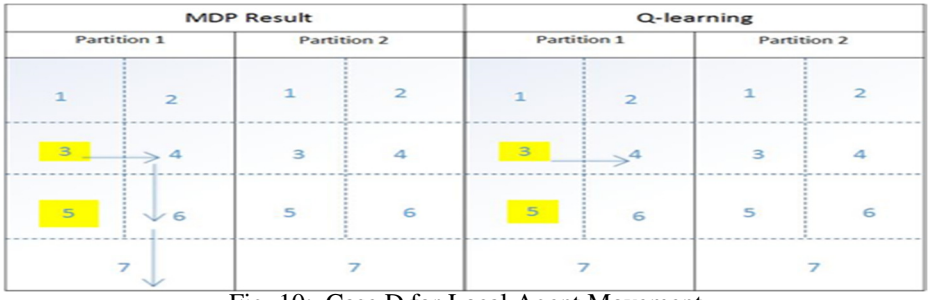


Fig. 10: Case D for Local Agent Movement

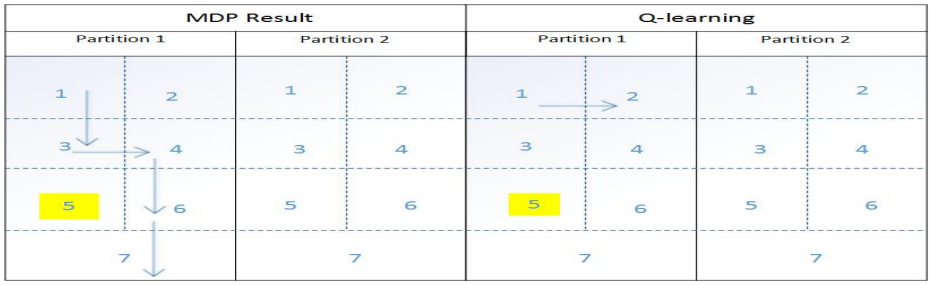


Fig. 11: Case A for Local Agent Movement

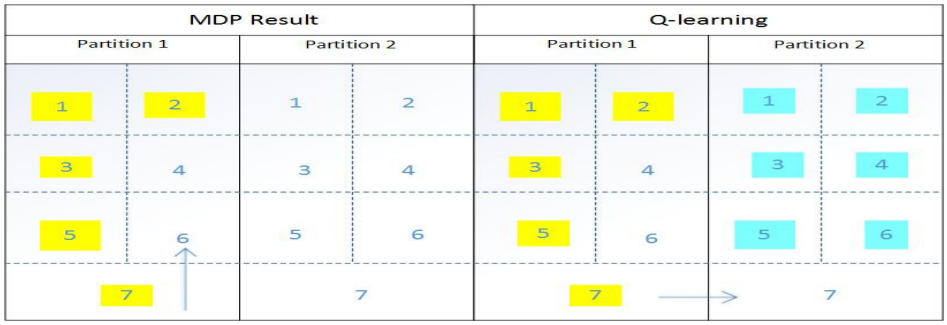


Fig. 11: Case F for Local Agent Movement

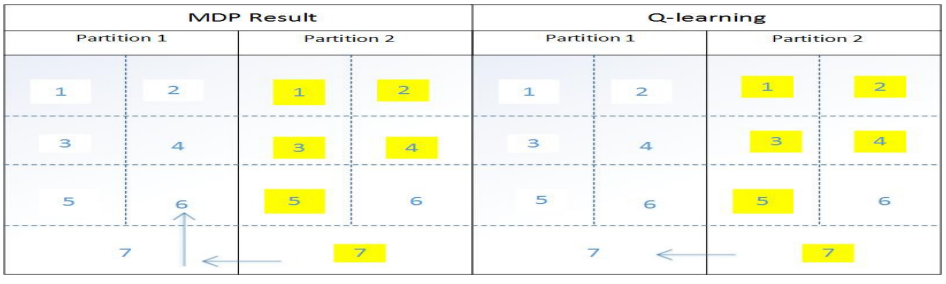


Fig. 12: Case B Local Agent Movement

The last scenario illustrated in Fig. 13, all of the blocks to be hazardous with companions present. So in MDP model, it takes agent to safety by moving to partition no. 1 but it takes another step to move agent to block no. 6. But on the other hand Q-learning only moves it to block number 7 of partition no 1.

5. CONCLUSION

In this paper, an effective hierarchical reinforcement learning algorithm is developed to find optimal Q-learning policy for robotic search teams. The proposed framework works with general dynamics in an unknown environment. The main advantage of using this methodology is the decision making on behalf of agents from the data which is coming from the outside sensors i.e. camera in case of our research. Therefore, all the policies that are being developed majorly depends on the camera output. Agent can take decision on basis of that output directly without informing the control room thus making a decentralized approach. The paper also presents a comparison between MDP policy and Q learning which depicts better computational time and efficiency of this architecture. Although, both MDP and Q-learning are correlated results depict a better performance of Q-learning algorithm. The main advantage of this approach is member agent can take action on its own by detecting present faults, communicating with other member agents, choosing between active and non-active position and moving between search areas. The number of goals, search areas and agents can be changed according to the demand. States, actions, reward function and transition probabilities are constructed or altered according to demand.

REFERENCES

1. Casper J., Murphy R. R., "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 33, pp. 367-385, 2003.
2. Guizzo E., "Japan earthquake: more robots to the rescue," IEEE Spectrum, March 2011.
3. Wisnivesky J. P., Teitelbaum S. L., Todd A. C., Boffetta P., Crane M., Crowley L., *et al.*, "Persistence of multiple illnesses in World Trade Center rescue and recovery workers: a cohort study", *The Lancet*, Vol. 378, pp. 888-897, 2011.
4. Weiden M. D., Ferrier N., Nolan A., Rom W. N., Comfort A., Gustave J., et al., "Obstructive airways disease with air trapping among firefighters exposed to World Trade Center dust", *Chest*, Vol. 137, pp. 566-574, 2010.
5. Udasin I., Schechter C., Crowley L., Sotolongo A., Gochfeld M., Luft B., et al., "Respiratory symptoms were associated with lower spirometry results during the first examination of WTC responders", *Journal of Occupational and Environmental Medicine*, Vol. 53, pp. 49-54, 2011.
6. Liu Y., Nejat G., "Robotic urban search and rescue: A survey from the control perspective", *Journal of Intelligent & Robotic Systems*, Vol. 72, pp. 147-165, 2013.
7. Oh K.-K., Park M.-C., Ahn H.-S., "A survey of multi-agent formation control", *Automatica*, Vol. 53, pp. 424-440, 2015.
8. Cao Y., Yu W., Ren W., Chen G., "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, Vol. 9, pp. 427-438, 2013.
9. Olfati-Saber R., Murray R. M., "Consensus problems in networks of agents with switching topology and time-delays", *IEEE Transactions on Automatic Control*, Vol. 49, pp. 1520-1533, 2004.
10. Wen G., Duan Z., Ren W., Chen G., "Distributed consensus of multi-agent systems with general linear node dynamics and intermittent communications", *International Journal of Robust and Nonlinear Control*, Vol. 24, pp. 2438-2457, 2014.
11. Zhang S., Sridharan M., Washington C., "Active visual planning for mobile robot teams using hierarchical pomdps", *IEEE Transactions on Robotics*, Vol. 29, pp. 975-985, 2013.
12. Luo L., Chakraborty N., Sycara K., "Provably-good distributed algorithm for constrained multi-robot task assignment for grouped tasks", IEEE Transactions on Robotics, Vol. 31, pp. 19-30, 2015.
13. Grady D. K., Moll M., Kavraki L. E., "Extending the applicability of POMDP solutions to robotic tasks", *IEEE Transactions on Robotics*, Vol. 31, pp. 948-961, 2015.
14. Zhang Y., Parker L. E., "IQ-ASyMTRe: Forming executable coalitions for tightly coupled multirobot tasks", *IEEE Transactions on Robotics*, Vol. 29, pp. 400-416, 2013.

15. Hollinger G. A., Yerramalli S., Singh S., Mitra U., Sukhatme G. S., "Distributed data fusion for multirobot search", *IEEE Transactions on Robotics*, Vol. 31, pp. 55-66, 2015.
16. Alla A., Falcone M., Kalise D., "An efficient policy iteration algorithm for dynamic programming equations", *SIAM Journal of Scientific Computing*, Vol. 37, pp. A181-A200, 2015.
17. Modares H., Lewis F. L., Naghibi-Sistani M.-B., "Adaptive optimal control of unknown constrained-input systems using policy iteration and neural networks", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 24, pp. 1513-1525, 2013.
18. Wang D., Liu D., Li H., "Policy iteration algorithm for online design of robust control for a class of continuous-time nonlinear systems", *IEEE Transactions on Automation Science and Engineering*, Vol. 11, pp. 627-632, 2014.
19. Liu D., Wei Q., "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 25, pp. 621-634, 2014.
20. Wei Q., Liu D., Lin H., "Value iteration adaptive dynamic programming for optimal control of discrete-time nonlinear systems", *IEEE Transactions on Cybernetics*, Vol. 46, pp. 840-853, 2016.
21. Lahijanian M., Andersson S. B., Belta C., "Temporal logic motion planning and control with probabilistic satisfaction guarantees", *IEEE Transactions on Robotics*, Vol. 28, pp. 396-409, 2012.
22. Lahijanian M., Maly M.R., Fried D., Kavraki L. E., Kress-Gazit H., Vardi M. Y., "Iterative temporal planning in uncertain environments with partial satisfaction guarantees", *IEEE Transactions on Robotics*, Vol. 32, pp. 583-599, 2016.
23. Chung T. H., Burdick J. W., "Analysis of search decision making using probabilistic search strategies", *IEEE Transactions on Robotics*, Vol. 28, pp. 132-144, 2012.
24. Zhang M.-L., Zhou Z.-H., "A review on multi-label learning algorithms", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, pp. 1819-1837, 2014.
25. Panait L., Luke S., "Cooperative multi-agent learning: The state of the art", *Autonomous Agents and Multi-Agent Systems*, Vol. 11, pp. 387-434, 2005.
26. Garcia S., Luengo J., Sáez J. A., Lopez V., Herrera F., "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 25, pp. 734-750, 2013.
27. Huang G., Song S., Gupta J. N., Wu C., "Semi-supervised and unsupervised extreme learning machines", *IEEE Transactions on Cybernetics*, Vol. 44, pp. 2405-2417, 2014.
28. Yu C., Zhang M., Ren F., Tan G., "Multiagent learning of coordination in loosely coupled multiagent systems", IEEE Transactions on Cybernetics, Vol. 45, pp. 2853-2867, 2015.
29. Chalkiadakis G., Boutilier C., "Coordination in multiagent reinforcement learning: a Bayesian approach", *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 709-716, 2003.
30. Ghavamzadeh M., Mahadevan S., Makar R., "Hierarchical multi-agent reinforcement learning", *Autonomous Agents and Multi-Agent Systems*, Vol. 13, pp. 197-229, 2006.
31. François-Lavet V., Henderson P., Islam R., "An introduction to deep reinforcement learning", *Trends in Machine Learning*, Vol. 11, No. 3-4, 2018.
32. Arulkumaran K., Deisenroth M.P., Brundage M., Bharath A.A., "Deep reinforcement learning: A brief survey", *IEEE Signal Processing Magazine*, Vol. 34, Vol. 6, pp. 26-38, 2017.
33. Fujimoto S., Meger D., Precup D., "Off-policy deep reinforcement learning without exploration", *Proceedings of the 36th International Conference on Machine Learning*, pp. 2052-2062, Long Beach, California, PMLR 97, 2019.
34. Cobbe K., Klimov O., Hesse C., Kim T., Schulman J., "Quantifying generalization in reinforcement learning", P*roceedings of the 36th International Conference on Machine Learning*, PMLR 97, 2019.
35. Nasir A., Salam Y., Saleem Y., "Multi-Level Decision MakinJ.,g in Hierarchical Multi-agent Robotic Search Teams", *The Journal of Engineering*, Vol. 1, No.1, 2016.