International Journal of Automation and Computing

17(3), June 2020, 364-377
DOI: 10.1007/s11633-019-1187-6

HDec-POSMDPs MRS Exploration and Fire Searching Based on IoT Cloud Robotics

Ayman El Shenawy[1] Khalil Mohamed[1] Hany Harb[2]

[1]Systems and Computers Engineering Department, Faculty of Engineering, Al-Azhar University, Cairo 11751, Egypt

[2]Information Technology College, Misr University for Science and Technology (MUST), Cairo 77, Egypt

Abstract: The multi-robot systems (MRS) exploration and fire searching problem is an important application of mobile robots which require massive computation capability that exceeds the ability of traditional MRS′s. This paper propose a cloud-based hybrid decentralized partially observable semi-Markov decision process (HDec-POSMDPs) model. The proposed model is implemented for MRS exploration and fire searching application based on the Internet of things (IoT) cloud robotics framework. In this implementation the heavy and expensive computational tasks are offloaded to the cloud servers. The proposed model achieves a significant improvement in the computation burden of the whole task relative to a traditional MRS. The proposed model is applied to explore and search for fire objects in an unknown environment; using different sets of robots sizes. The preliminary evaluation of this implementation demonstrates that as the parallelism of computational instances increase the delay of new actuation commands which will be decreased, the mean time of task completion is decreased, the number of turns in the path from the start pose cells to the target cells is minimized and the energy consumption for each robot is reduced.

Keywords: Multi-robot systems, hybrid decentralized partially observable semi-Markov decision process (HDec-POSMDPs), multi-robot systems (MRS) exploration and fire searching, cloud robotics, cloud computing.

1 Introduction

Robots are autonomous agents that interact with the surrounding environments directly to perform tedious tasks precisely and correctly using onboard software modules to facilitate the achievement of several tasks[1, 2]. A robot task can be classified into simple tasks (all reactive and real-time control operations such as an obstacle avoidance, guidance, etc.), complex tasks (such as path planning, localization, etc.), and intelligent tasks that assist humans (such as cleaners, delivery, etc.)[3–6].

The great importance of robots comes from the tremendous development that allow them to be adaptive, cooperative, and learn from their experience in performing future actions[7, 8]. This development is accommodated by a parallel improvement in the basic resources of robots, methods, and services[9–11]. Robot services may be a simple service (robotic cleaners, delivery), or complex service (painting, object recognition and detection, searching and rescuing, etc.)[9, 10, 12–17].

Searching for fire in complex environments is sometimes extremely dangerous mission, and subjects human life to risk if they try to perform these tasks due to various environmental problems such as high temperatures and limited communication. Therefore, these dangerous missions can be assigned to a single robot or multi-robot systems (MRS) that operates in hazardous situations to search for fire sources in unknown environments[18, 19]. The control architecture of fire searching robot is designed for gathering information about fires in the indoor environment[20]. Each robot participating in this task is equipped with temperature, flame and smoke sensors for fire detection. It is also has the capability to follow virtual path lines, avoid obstacles and detect fire source regions with a specific distance and accuracy[21, 22].

For MRS to perform run-time applications in an accurate manner and appropriate time, there must be an urgent need for modern algorithms that requires vast computing resources for environmental perception, decision-making, and navigation management[23]. MRS are bounded by the following limitations: 1) hardware/software limitations such as limited communication capacities, 2) limited storage resources used to store huge amount of data collected by robots and 3) limited computation capability used to complete heavy tasks. Therefore, it is difficult for MRS to perform tedious tasks, at the same time it is not possible to add further developments to MRS under these limitations, which poses serious challenges to the continuation of MRS[24].

To overcome such challenges, some researchers have been thinking about using powerful parallel computers that process and control data using a range of remote servers and provide complex services[8, 15, 23]. As a result,

Research Article
Manuscript received October 27, 2018; accepted May 9, 2019; published online July 15, 2019
Recommended by Associate Editor Min Tan

many benefits have been added to MRS: 1) the limitation problems of the onboard computing and storage were solved; 2) additional capabilities that are impossible for MRS to perform was provided; 3) new skills to generate, access and process huge amount of data were added; 4) trends such as a remote brain, big data were enhanced[25, 26].

Cloud computing is an internet-based computing model that enables magnification of ubiquitous, convenient, on-demand network access to a shared pool of configurable servers, memory storage, networks, and applications by interconnecting physical and virtual devices based on existing information and communication technologies. It can provide services rapidly with minimal management effort or service provider interaction. This means that not all measuring devices, computation, and memory storage are integrated into a standalone device[8, 25, 27–30].

Cloud computing provides three types of basic services: 1) Software as a service (SaaS): allows users to access applications across different devices ranging from simple devices like thermostats to a highly complex smart devices like robots[31–33]. 2) Platform as a service (PaaS): allows users to implement their applications without looking at the cloud infrastructure. 3) Infrastructure as a service (IaaS): is controlled by the user to make the hardware infrastructure available as a service such as Amazon's EC3/S3 and Microsoft Azure[34–36].

An integration can be made between MRS and cloud computing to exploit its resources, acquire intelligence from the hosted software programs remote servers[37–40], perform complicated computations, share knowledge with other robots and store large-scale knowledge in an efficient manner. This integration is considered a new trend in a robotics field that is commonly referred to as cloud robotics[1, 3, 41, 42]. The main concern of cloud robotics is to off-load heavy and expensive computational tasks to the cloud servers in secure and customizable environments for robots and other smart devices.

The use of cloud computing in robotic has solved many of problems faced by MRS: 1) The lack of local computing is amended and the task completion time is reduced[7, 35]; 2) The physical world is explored and faulty robots are replaced immediately with other robots of similar capabilities[3, 43]; 3) Robots are able to optimally cooperate with each other to execute complex tasks efficiently[2]; 4) Robots are provided with massive resources of information such as environmental maps[8, 44]; 5) Computational bandwidth was allocated efficiently and the energy consumption was reduced effectively[2].

In recent years, many researchers have presented a lot of studies related to the field of cloud robotics. First, at the PaaS layer, a set of frameworks was presented such as 1) RoboEarth: A framework that encodes the task structure, environment information and object descriptions as a formal knowledge base formed based on the robots observation[1, 45]. 2) Rapyuta: A framework that exploits the RoboEarth database for running applications within it[46–48]. 3) Cloudroid: A framework that outsources robot computation without any modification in the software package of the traditional robot[23, 49]. 4) Internet of things (IoT) cloud: a generic real-time framework that exceeds other frameworks in performance and encapsulates data from devices and processes it. It provides a simultaneous localization and mapping (SLAM) algorithm and improves the system performance[50]. In [51], the authors have implemented a collision avoidance algorithm for swarm robotics based on IoT cloud, it focuses on agent level parallelization.

Second, at the SaaS layer, Rahman et al.[52] developed a framework for smart factory maintenance and formulate it as a joint optimization problem. They have used a modified Ggenetic algorithm (GA) based decision-making scheme to find the near-optimal solutions. Chen et al.[37, 53] propose a framework to offload collected data from individual robots and robot clusters to a remote server. The problem is formulated as a joint quality of service (QoS) optimization of robotics stream workflow (RSW) in networked cloud robotics (NCR) and RSW is transferred into a mixed integer linear programming (MILP) problem and solved using a heuristic algorithm to manage the heterogeneity of NCR and the strict latency requirements of RSW. Miratabzadeh et al.[54] implement a reliable, scalable, and powerful software platform using OpenStack to add scalability to serve large independent and heterogeneous robots.

Third, at the IaaS layer, Chen et al.[26] presents a hybrid CloudStack platform to deploy and manage large networks of virtual machines, to provide safety and scalability to traditional approaches. Tian et al.[55] developed a software framework called Berkeley Robotics and Automation as a service that performs a robust grasp-planning system to increase the grasp reliability. Manzi et al.[5] increase the ability of robots skills and provide text-to-speech and speech recognition abilities for human interactions using a KuBo cloud robotics. Koubaa and Qureshi[56] propose a cloud-based object tracking real-time application called DroneTrack using unmanned aerial vehicles, it uses GPS to exchange its locations with the cloud.

Cloud robotics has presented a lot of benefits to MRS such as minimizing computation cost, scaling computation resources, providing support for large-scale systems and controlling robots in real-time[56]. The process of offloading robot applications to cloud servers is still difficult in practice due to the following reasons: 1) Robot applications software must be modified to be compatible with cloud environments; 2) There is no mature cloud system designed for robotics tasks; 3) Most robot applications interact with the environment, and there was a need to guarantee QoS when the robot is connected to servers[23].

Therefore, there is an urgent need for more work to enhance the existing cloud robotics frameworks and offload other complex robot applications such as an exploration of unknown environments, navigation, map building, co-ordinating MRS tasks, searching and rescuing, planning a collision-free 3D path and other space-oriented applications for unmanned aerial vehicles (UAV). The intended target is to enhance MRS performance according to some criteria such as energy, safety and time[57].

Environment exploration is considered as the base of most MRS and UAV applications[58]. Performing complex tasks using MRS implicitly includes the environment exploration and task coordination, then the remaining part of the task is performed. For example, to search for fire objects in an unknown environment, the robot must explore the environment using the generated map and then perform the search process. Also, task coordination and task decomposition join the execution of these applications[59].

In [60], a hierarchical architecture for task coordination and decomposition for MRS exploration and fire searching based on the hybrid decentralized partially observable semi-Markov decision process (HDec-POSMDPs) algorithm was used to decompose the global task into sub-tasks that can be executed by one or more individual robots in sequence or in parallel, and the coordination between robots was handled. The MRS is allowed to navigate, plan paths, avoid collisions between obstacles and robots, and finally build a grid map using the information collected by the individual robots. Performing this complex task by traditional MRS requires a lot of computational resources and takes a long time.

In this paper, the complex HDec-POSMDPs algorithm in [60] is reconfigured into a cloud-based HDec-POSMDPs algorithm. Its main tasks are split up into parallel computing blocks (get robot state, get a local map, avoid obstacles, assign goals and construct global map) to be run on different machines in a parallel form. The cloud-based HDec-POSMDPs is implemented on the top of the IoT cloud robotics framework to offload the massive computation from the robot onboard computing resources to the cloud framework and to achieve high computation efficiency. The input to the IoT cloud (laser scans, pose share, thermopile temperature, etc.) from robots is processed and the results are returned back directly as commands to robots through an interfacing device.

As a summary, the main contribution of this paper is:

1) The HDec-POSMDPs algorithm is reconfigured to a cloud-based HDec-POSMDPs.

2) The cloud-based HDec-POSMDPs is implemented on the IoT cloud robotic framework to explore a set of different environments using different sets of robot sizes.

3) Some preliminary results have been presented to evaluate the performance of the proposed cloud-based HDec-POSMDPs using different metrics such as task execution meantime, average energy consumption.

The rest of the paper is organized as follows: in Section 2, the architecture of the IoT cloud robotics framework is discussed. Section 3 addresses the implementation of the cloud-based HDec-POSMDPs MRS exploration and fire searching algorithm. In Section 4, the experimental results are discussed and we conclude and summarize this paper in Section 5.

2 Architecture of IoT cloud robotics framework

As shown in Fig. 1, the IoT cloud framework consists of three layers coordinated by Zookeeper and connected together by exchanging formally-defined messages using message broker[55]. The three layers are:

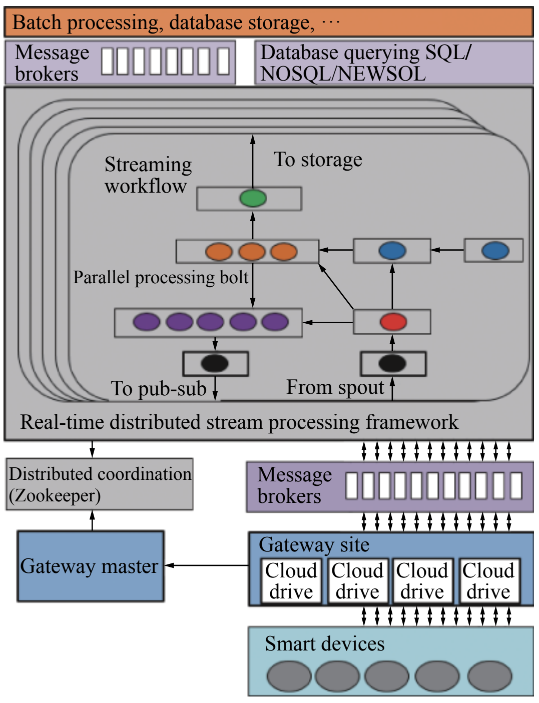


Fig. 1 Architecture of the cloud robotics framework

Front-end gateway layer: Maps between message brokers and smart devices. It connects all devices with message broker services using the cloud drivers and the gate master. Cloud drivers are used to convert the smart device information into messages which are processed via a remote server on the cloud.

Stream processing middle layer: This is used to process a huge amount of data gathered by numerous smart devices with highest-capacity rates using batch processing engines and an apache storm computation engine. A number of spouts are used to read the information from smart devices and bolts to process the data for this information respectively. This layer can be represented by a graph topology in which spouts and bolts represent the nodes and streams representing the connected edges of this topology.

Batch/Storage back-end layer: Stores data from the middle layer and provides batch processing and data mining services from different distributed databases.

3 Implementation of the cloud-based HDec-POSMDPs MRS exploration and fire searching

3.1 System overview

The overall design of the cloud based HDec-POSMDPs MRS exploration and fire searching algorithm is based on the IoT cloud robotics framework which is shown in Fig. 2. The system design consists of three main layers as follows:

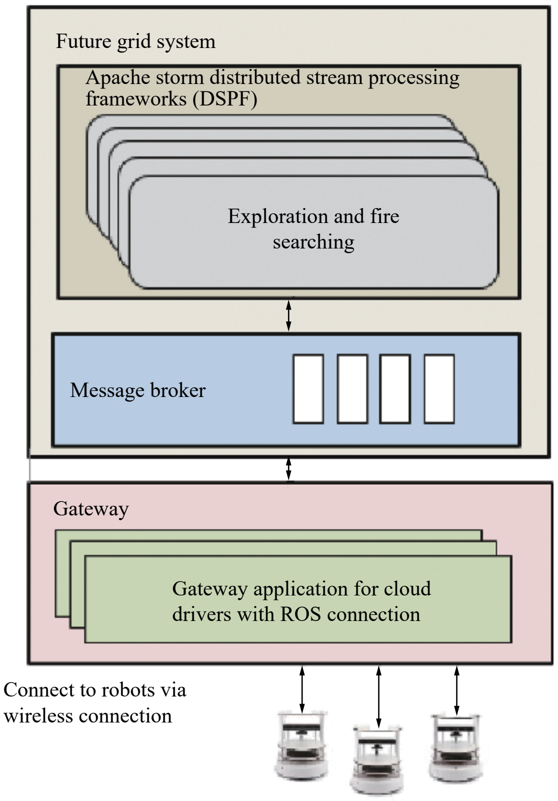

Future grid system
Apache storm distributed stream processing frameworks (DSPF)
Exploration and fire searching
Message broker
Gateway
Gateway application for cloud drivers with ROS connection
Connect to robots via wireless connection

Fig. 2 Overall design of exploration and fire searching

In the first layer, a set of homogenous Turtlebot robots form the MRS to perform the task of exploration and fire searching in an unknown environment. The second layer represents the front-end gateway. In this layer, robot operating system (ROS) based Turtlebot application programming interface (API) robots are run on a desktop machine as the device driver. The device driver generates a unique identifier (ID) for each robot to distinguish between different robots and define the cloud channels to establish a connection with those ROS. Robots are connected directly to the device driver without any interface to send the measured data (such as laser range, thermopile temperature, and odometer reading, etc.) to be processed and get control commands (such as moving, rotate and stop) after processing the measured data[50, 51].

In the third layer, the cloud computation engine and the message broker servers are deployed in virtual machines called Future Grid System of the Cloud framework[51]. This layer consists of a fixed number of input spouts and output bolts that are connected to a predefined cloud channel as shown in Fig. 3.

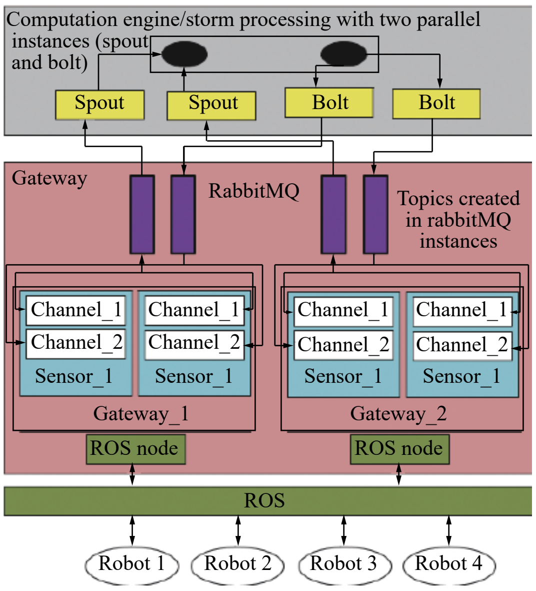


Fig. 3 IoT cloud driver with connected channel

The message broker receives a ROS message from the device drivers, converts it to a custom defined data type using a Rabbitmq cloud driver and sends it to the cloud computation engine through input spouts to be processed. Then, the cloud computation engine sends the control commands through output bolts to the message broker[50, 51]. The message broker sends these messages again to the device driver, and the device driver sends them to the robots to make their decision for moving, rotating, stopping, etc.

3.2 HDec-POSMDPs MRS exploration and fire searching

In this section, the HDec-POSMDPs MRS exploration and fire searching algorithm is summarized, the algorithm is implemented in the cloud computation engine layer of the IoT cloud framework.

As shown in the flowchart in Fig. 4, the HDec-POSMDPs MRS exploration and fire searching algorithm is divided into five primary stages as follows[60]:

Initialization: In this phase: 1) The startup location of each robot is initialized randomly with restrictions that the distance between them is less than the scope of the communication range; 2) The navigation of a robot includes building a map using a set of information acquired by a robot, determining the exact position and orientation of a robot in the environment at all times, and generating a collision-free trajectory from its current pose to

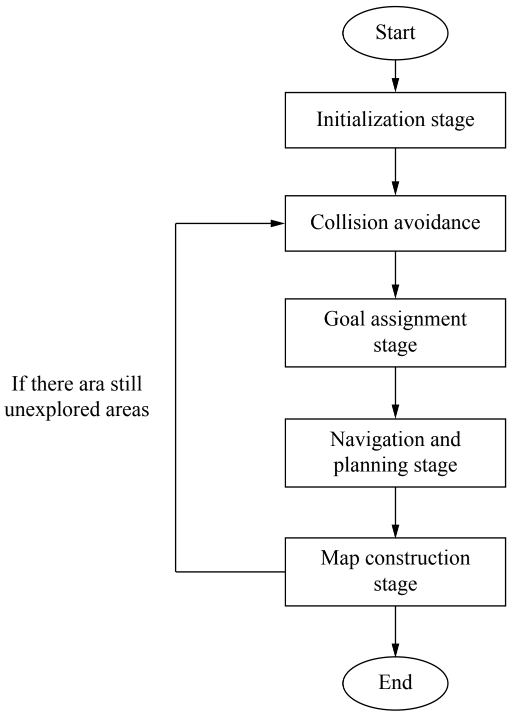


Fig. 4 Flowchart of the HDec-POSMDPs MRS exploration and fire searching approach

the desired target pose using enhanced A* (EA*) algorithm[59, 60].

Collision avoidance: During the navigation process, robots must avoid collisions with obstacles, other robots and fire sources by implementing some rules between the robot team. The robot tries to detect a frontier or a fire source cell during the navigation process. If the robot detects a frontier or a fire source cell, it will go to the next step, if not it will end the exploration process.

Goal assignment: In this phase, the robot determines its next location based on the discovered new and unexplored areas. Some factors are taken into consideration such as energy consumption, network connectivity and the overlaps between the robot′s sensor ranges[61, 62]. The candidate frontier cell is determined and assigned based on (Dec-POSMDPs). The best frontier cells for robots are determined. Each robot takes a decision about the nearest cell, if it is inside its region or not; based on a specific utility function. The process of fire searching is done in parallel with the frontier detection based on sensing the heat levels of objects allocated in the environment using the temperature sensors[63]. Once the fire source is detected, the map is updated and the robot moves to its goal.

Navigation and planning: In this phase, a path with minimum cost is determined by solving the multiple travelling salesman problem (MTSP)[59, 64] based on a cluster-first, route-second heuristic approach.

Map construction: As robots explore new areas, a new information is gathered, added to the current information, and a global map is created and broadcasted to other robots. The entire exploration process is repeated. Full details about the HDec-POSMDPs can be found in [59, 60].

3.3 Architecture design of cloud-based HDec-POSMDPs MRS exploration and fire searching

In order to implement the cloud based HDec-POSMDPs MRS exploration and fire searching algorithm, the spouts and bolts that connect the cloud computation engine and the message broker should be designed first.

As shown in Fig. 5, six spouts are defined to represent

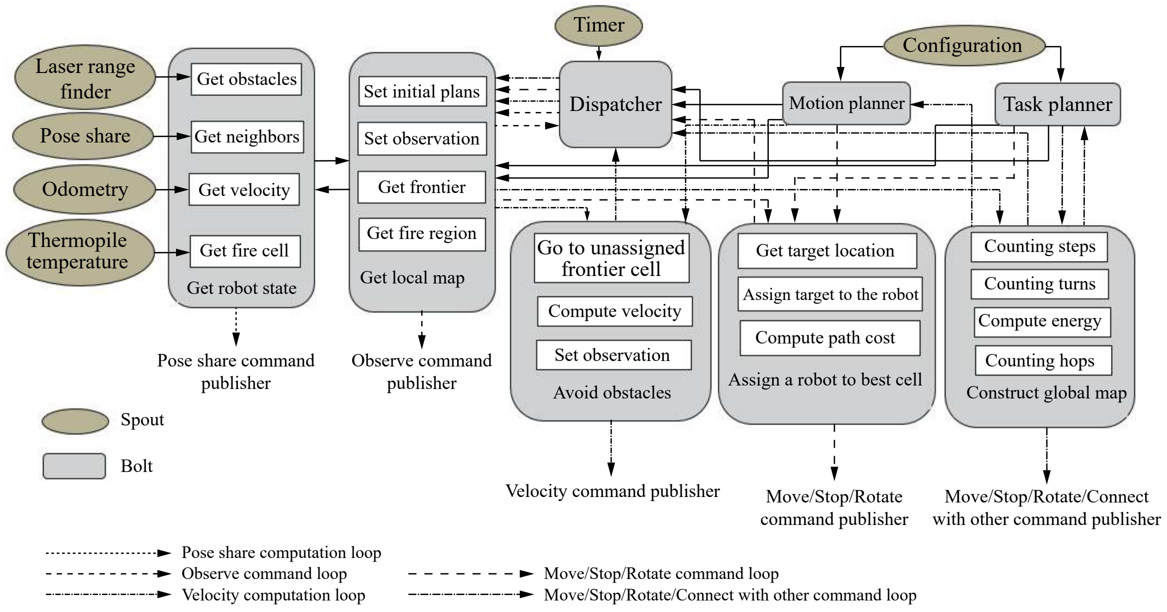


Fig. 5 Topology for exploration and fire searching application

all the required information to be entered to the computation module. The first four spouts represent the robot state (laser range finder, pose share, odometry and thermopile temperature spout). The fifth spout receives the robot configuration parameters (such as control frequency, min/max velocity, start and goal pose, etc.). The sixth spout is the timer spout which asks the dispatcher if it needs a new tuple or not.

The distribution of all computation tasks required for the cloud based HDec-POSMDPs MRS exploration and fire searching algorithm is divided into a number of bolts as shown in Fig. 5. During the task execution all robots must share their states periodically to a predefined cloud channel; therefore, they can share their newest state with each other. The cloud-based HDec-POSMDPs MRS exploration and fire searching computation module receive all the input data through the specified spouts, process it in specific bolts and publish the computed commands to the message broker respectively.

The proposed cloud-based HDec-POSMDPs is implemented using eight bolts as shown in Fig. 5 and the function of each bolt is discussed as following:

Motion planner bolt: The input for these bolts is sent through the configuration spout, to do the following:

1) Generate a set of global paths based on the start and goal poses of each robot.

2) Calculate the cost of all paths, and allow each robot to move through the shortest path using MTSP.

3) Generate a time state record, and feed it to dispatcher bolt that triggers control commands to the given period.

Task planner bolt: These bolts receive a tuple of data from the configuration spout, and do the following steps:

1) Determine the suitable cell location and assign it to the winning robot according to the minimum distance between the cell location and all the robot team.

2) Construct the global map by collecting the local maps generated by each robot.

3) Generate a custom-record including some information about robot locations, neighbors, connection, states, control frequency, min/max velocity, and local map. This information is sent to other bolts to assign a robot to best cell bolt or to avoid obstacles bolt, etc.

4) Generate a time state object to record the last time of controlling the robot or publishing the data of other teammates, and feed it to dispatcher bolt that triggers control commands to the given period.

5) Generate a local map object that contains information about robots′ local map.

6) Generate a pose share message which contains the basic shared knowledge between a robot team.

The dispatcher bolt: It receives a tuple of data from the timer spout every step to get the local map and publish the control commands.

Each of the following bolts is used to perform a specific task when receiving a tuple of data from motion and task planner bolts. After finalizing this task, a set of control commands is sent to the robot to take action, and another tuple of information is sent back to the dispatcher to report the termination of the task. The sent information includes the robot ID to ensure that the command control will be sent to the correct robot in MRS.

Get robot state bolt: Compute robot velocities, move it to unassigned frontier, count number of turns, and create a new robot state. The final state of the robot is published using a pose share messages and the related commands will be sent to message broker.

Get local map bolt: Uses the shared robot states between robots, sets the initial plans generated from start to goal cell, gets the frontier cell between explored and unexplored cell and detects the fire region in the map.

Avoid obstacles bolt: It contains three modules to observe the map and other teammate status, send the robot to unassigned or unexplored cells and computes the robot velocity to ensure the collision avoidance.

Assign a robot to best cell bolt: It contains three modules, the first one to determine the location of the target cell, the second one to assign robots to the best target, and the third one to compute the cost of the path.

Construct global map bolt: It contains four modules to count the number of steps to the goal, number of turns, and number of hops and compute the energy consumption during the task.

4 Experiments and results

In this paper, some experiments have been performed to check and verify the implementation of the proposed cloud-based HDec-POSMDPs MRS exploration and fire searching algorithm. The Simbad simulator is used to evaluate the performance of the applied algorithm by simulating a number of differential robots with many types of measuring devices like pose share, range finder sensor, thermopile temperature, odometry, etc.[50, 53].

The Gateway cloud driver and ROS are deployed together with the Simbad simulator on a local desktop computer while the computation engine and Rabbitmq message broker are deployed in IoT cloud robotics platform on a virtual machine with 4 GB memory storage and 2 CPU cores at 1.89 GHz. The hardware specification and configurations of 12 cores with six computation nodes are presented in Table 1. In our experiments, the maximum number of parallel instances for each computed bolt is limited to six instances to ensure the process parallelism, and for robot state bolt, it is limited to three instances in order to observe their effect on the performance of the task. The remaining components in the Apache Storm application Topology such as (Rabbitmq, ZooKeeper and Storm master node) have only one instance for each.

The information collected by each robot is published to ROS via Simbad simulator, and the cloud driver will

Table 1 System hardware specification

Specifications	VMs in cloud	Local host
CPU model	Intel Core i5 3437U	Intel(R) Core(TM) i7-3632QM
CPU Frequency/MHz	1 895.5 MHz	2 893 MHz
Cores	2	4
Thread per core	2	2
Memory/MB	4 GB	16 384
OS	Windows 10 Pro 64-bit	Windows 10 Pro 64-bit
Hypervisor	KVM	None

convert it into custom-records to be processed by the HDec-POSMDPs MRS exploration and fire searching algorithm.

The following metrics are used in the performance evaluation of our algorithm:

1) Delays coming from the computation-nodes (Mean-time) to complete the task for a different number of robots with a different parallelism.

2) Number of steps taken by each robot to accomplish the task.

3) Number of turns taken by a robot team when it faces an obstacle or other robot.

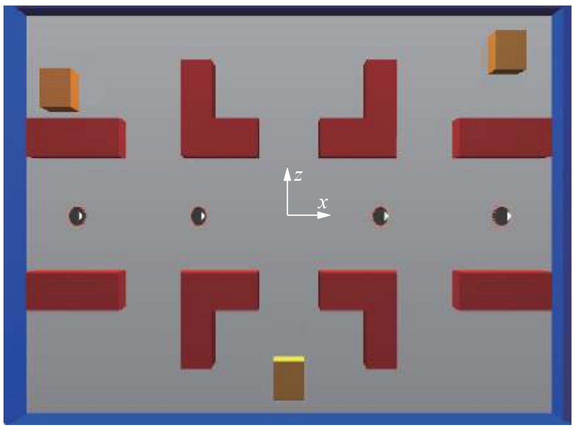
4) Average energy consumption by a robot.

5) Number of hops taken to accomplish the task.

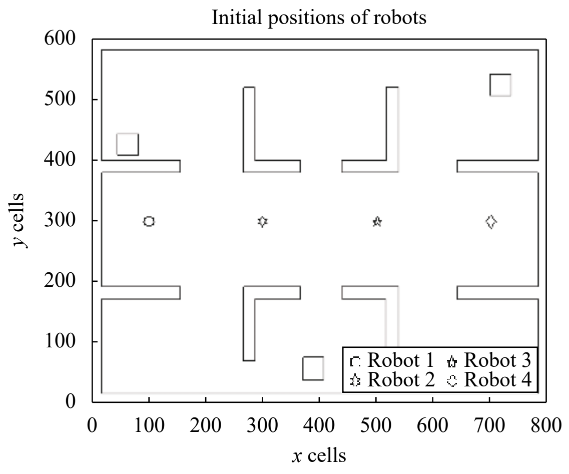
The number of parallelism for the four aforementioned bolts is set to six and the number of parallelism for getting the robot state bolt is set to three.

First, the proposed implementation is used to perform the environment exploration and fire searching for a selected environment which consists of 800 × 600 cells using four robots initialized on a single line of different locations within the environment, and with three fire sources (represents as cubes in Fig. 6 (a) and as a squares in Fig. 6 (b) distributed as shown in Figs. 6 (a) and 6 (b)). Simbad simulator does not have the capability of drawing the explored area and robot paths graphically. So all of these parameters are recorded during the exploration process into log files. And the recorded data is plotted and analyzed using another third party software (Matlab).

Fig. 6 (a) describe a moderate environment used for testing the cloud-based implementation of HDec-POSMDPs MRS exploration and fire searching algorithm, and its equivalent plot in Matlab is plotted in Fig. 6 (b) (Robots 1–4 are represented as a circle, pentagon, hexagon and diamond respectively, and fire sources are represented as squares, obstacles are represented as triangles of different lengths and widths). All the robots are starting to explore the environment and search for the fire sources distributed through the environment in a coordinated manner based on the implementation of the cloud-based HDec-POSMDPs.



(a) Test environment on Simbad simulator



(b) Test environment mapped on Matlab

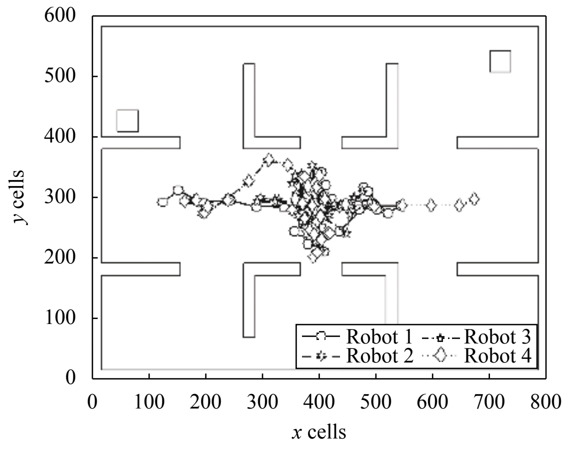
Fig. 6 Test environment

Figs. 7 (a) and 7 (b) shows the progress of the exploration process by all the participating robots after 300 time steps and 1 000 time steps, respectively.

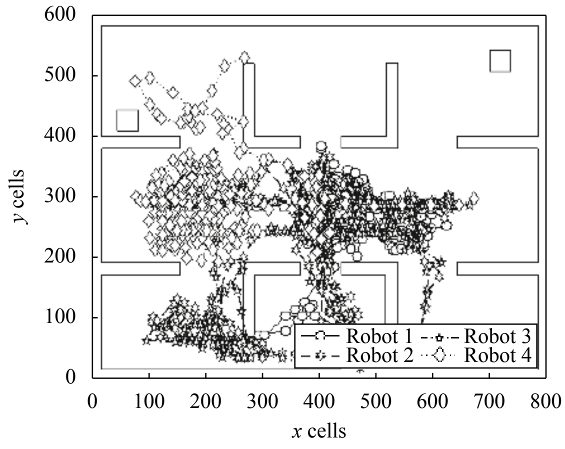
The plotted data in Fig. 7 indicates that the burden of the exploration process is divided between all the robots, not all the robots explore the same area. Also, there was a common area between all robots in which the robots need to communicate and publish the measured information.

The MRS exploration and fire searching process is considered as a cooperative process, in which each robot participates in the process by navigating through the environment and performing all the tasks explored in Section 3.2. Because each robot is behaving differently than the other robots based on its sensors reading and its decision. Figs. 8 and 9 shows the path generated by robots 2 after 300 time steps and 1 000 time steps, respectively.

We have tried to perform the same task using a different number of robots range from 2 to 10 robots. The command latency is around 50 ms for robots to finish commands effectively and frequencies are set to 20 Hz. Robots are deployed in a complex environment gener-



(a) Exploration path of all robots after 300 time steps



(b) Exploration path of all robots after 1 000 time steps

Fig. 7 Environment state

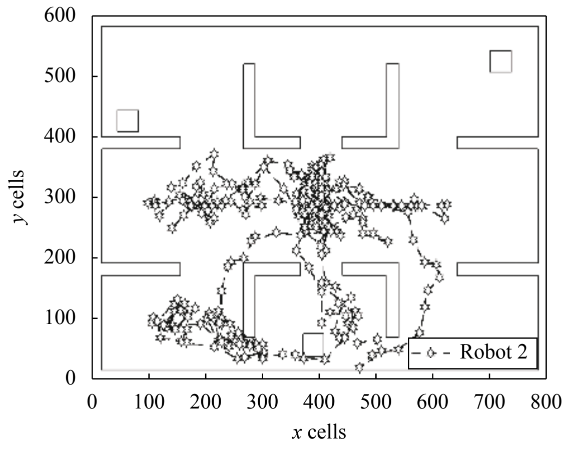
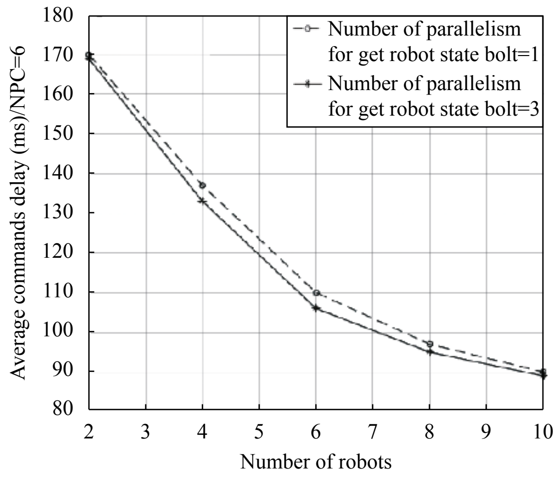
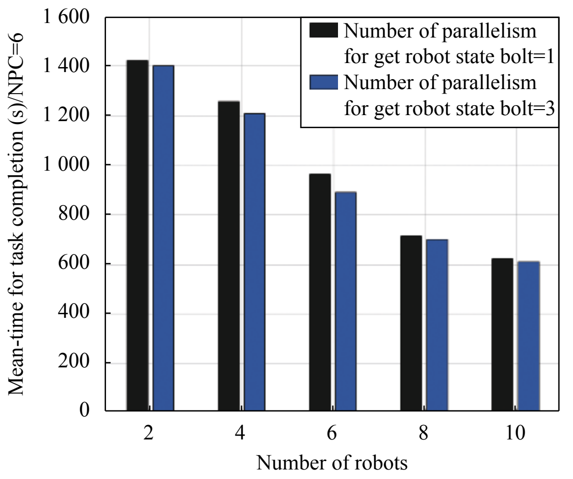


Fig. 9 Path followed by Robot 2 after 1 000 time steps



(a) Average commands delay versus number of robots



(b) Task completion average time versus number of robots

Fig. 10 Average commands delay and task completion meantime (NPC=6)

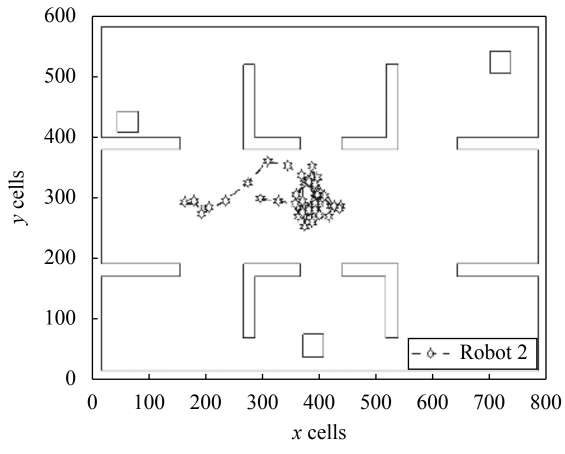


Fig. 8 Generated paths by Robot 2 after 300 time steps

ated by the Simbad robot simulator and modelled as an occupancy grid of a size 800 m × 600 m with 0.05 resolution and occupied by a set of randomly distributed obstacles. And the experimental runs are done for every 500 s as shown in Figs. 10 (a) and 10 (b) when the number of parallelism for each computation bolts is set to six and the number of parallelism for robot state bolt is set to three.

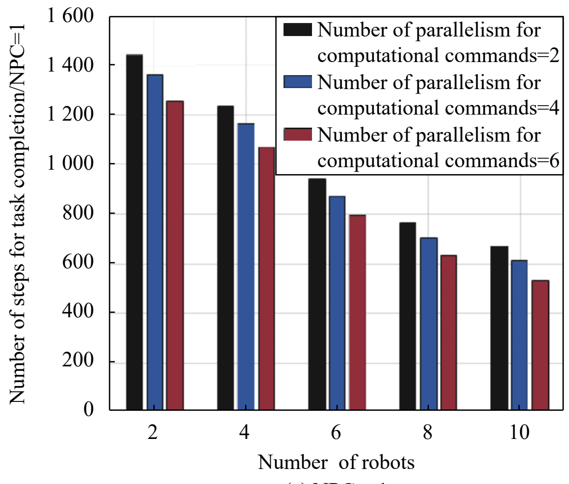
The plotted data in Figs. 10 (a) and 10 (b) indicates that as the number of robots starts to increase, the delay of other bolts is decreased, and finally causes the average completion time to decrease task time. Roughly speaking, when the number of robots increases, the collisions between the robots team will happen which in turn leads to increasing the delay of other bolts, but in our case there is a coordination and cooperation between the robot team which prevents the collision with robots themselves or with obstacles which leads to decreasing the average completion of task time.

Figs. 11 (a) and 11 (b) shows the number of steps that are required to complete the task when using different sizes of robot teams and the number of parallelism for each computation bolt varies from two to six and the number of parallelism for robot state bolt is changed one time to one and another to three to observe its effect on the system performance.

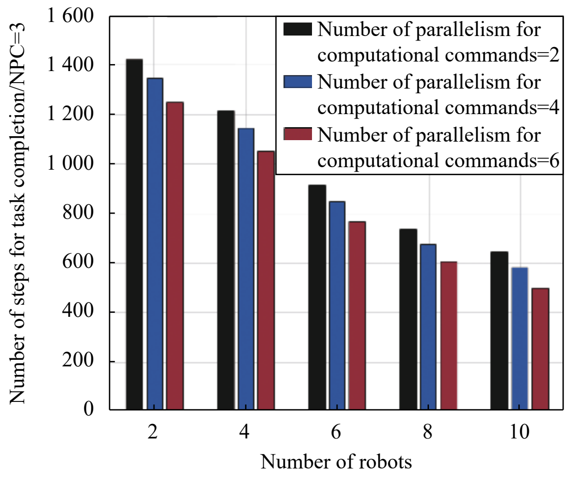
The results shown in Figs. 11 (a) and 11 (b) indicate that as the number of robots starts to increase from two to ten robots, the number of steps required is decreased even when the robot state bolt parallelism increases from one to three. Therefore, the obtained results show that as robot team size increases, the required steps to accomplish the task will be decreased.

The number of the parallelism commands is changed from 2 to 4 as shown in Figs. 12 and 13 to observe its effect on the task performance. As shown from Figs. 12 and 13, robots do not collide with each other during the coordination. This means that the delay decreases even when the robot state bolt parallelism increases from one to three. So that, an increase of robot states instances does not improve the task performance in this experimental test; because its computational load is very small if it is compared to the bolts computational load. Finally, we can say that increasing the parallelism of computational bolts decreases the delay for new commands and maintains good performance when the number of robots increases.

Figs. 14 (a) and 14 (b) plots a relation between the numbers of turns taken by each robot in the team to ac-

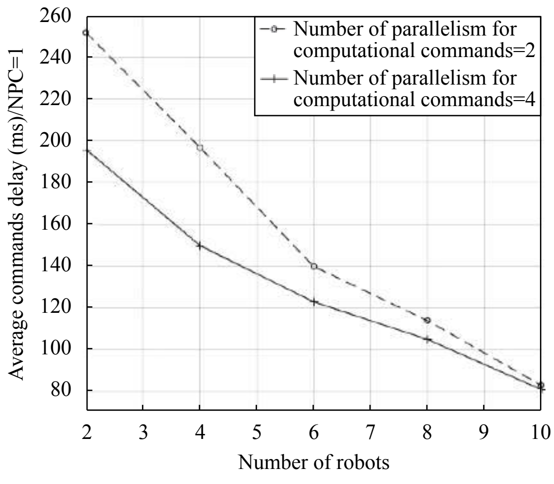


(a) NPC = 1

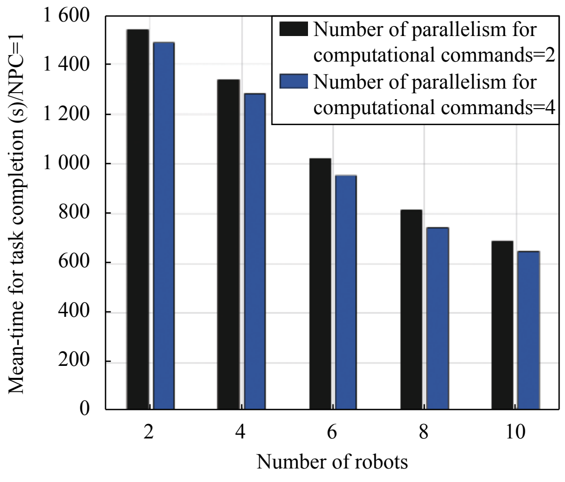


(b) NPC = 3

Fig. 11 Number of steps for different number of parallelism commands (NPC)

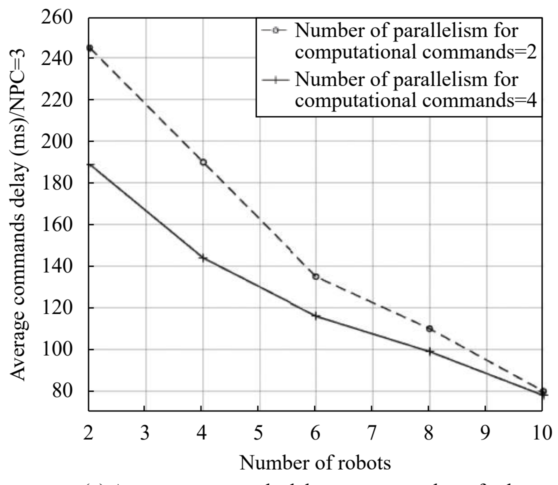


(a) Average commands delay versus number of robots

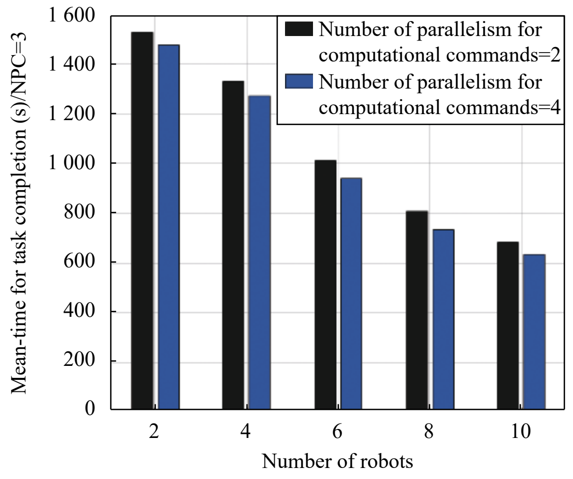


(b) Meantime for task completion versus number of robots

Fig. 12 Meantime task completion time and average commands delay when NPC = 1

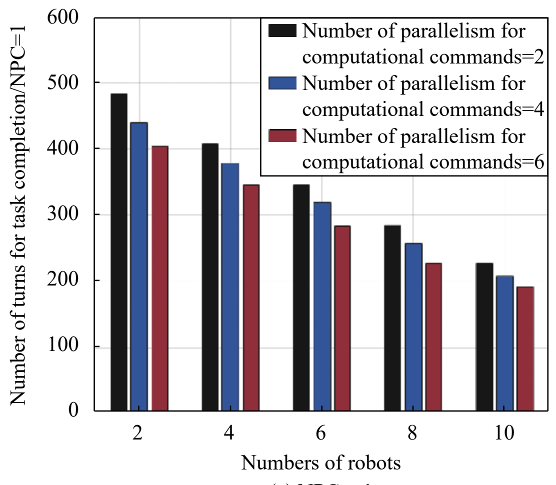


(a) Average commands delay versus number of robots

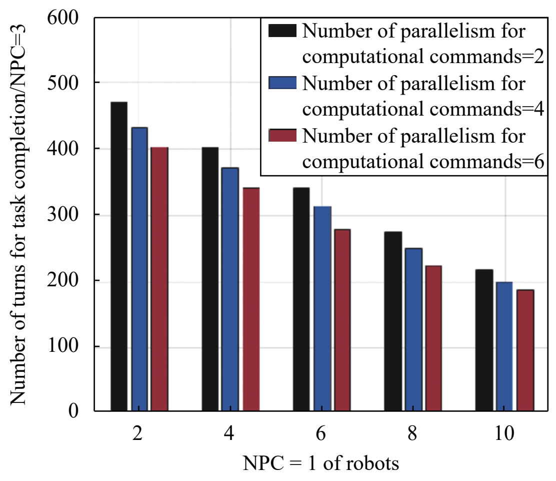


(b) Mean-time for task completion versus number of robots

Fig. 13 Meantime task completion time and average commands delay when NPC = 3



(a) NPC = 1



(b) NPC = 3

Fig. 14 Number of turns for different number of parallelism commands (NPC)

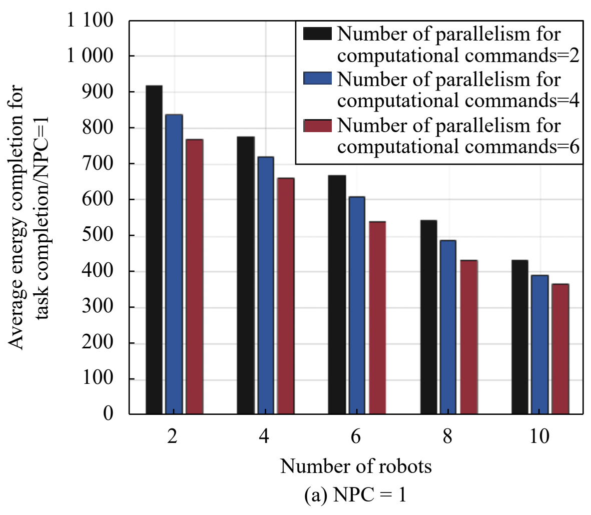
complish the task versus the number of robots when using different sizes of robot teams and the number of parallelism for each computation bolt varies from two to six and the number of parallelism for robot state bolt is changed one time to one and another to three. The obtained results show that the number of turns needed to accomplish the task decreases when the size of the robot team increases even when the robot state bolt parallelism increases from one to three.

The average energy consumption is studied for different numbers of mobile robots; because of the acceleration and deceleration caused by stopping and turning the robot, the robot may consume a large amount of energy because the path may have short distance but consumes more energy since the robot states have different directions. Therefore, it is always preferable to have an efficient energy path with a moderate loss of distance.

Figs. 15 (a) and 15 (b) shows the energy consumed by different team sizes of robots where the number of parallelism for each computation bolts varies from two to six and the number of parallelism for robot state bolt is changed from one to three. It shows that the energy consumed decreases when the size of the robot team increases even when the robot state bolt parallelism increases from one to three.

The communication overlapping between robots in an MRS can be minimized by establishing a communication mechanism to coordinate between robots in an MRS. Robots can share their local information to each other at every step of the movement in order to collect their local maps. The link bridges that are established to connect a pair of robots may have multiple jumps or hops, each of them may raise some delay in the communication network between the robot team.

The integration of local maps is done in the cloud framework to build the global map. Therefore, as a small number of communication paths is constructed, which decreases the total number of hop counts[65]. Fig. 16 shows the total number of hops needed when using a team of 2 robots; it remains within a range of (19 to 36), for 4 ro-



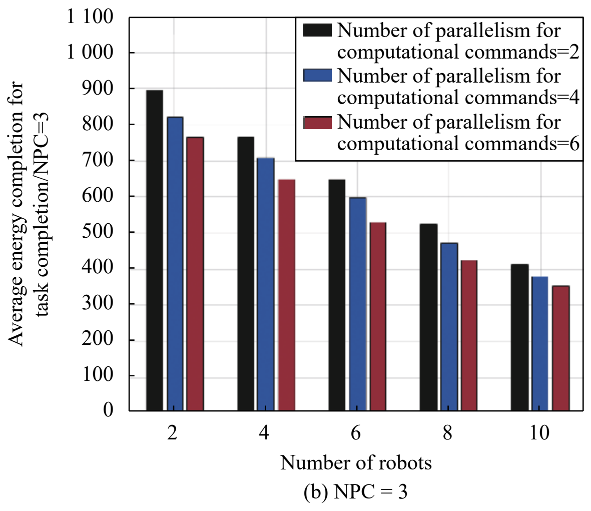


Fig. 15 Energy consumption for different number of parallelism commands (NPC)

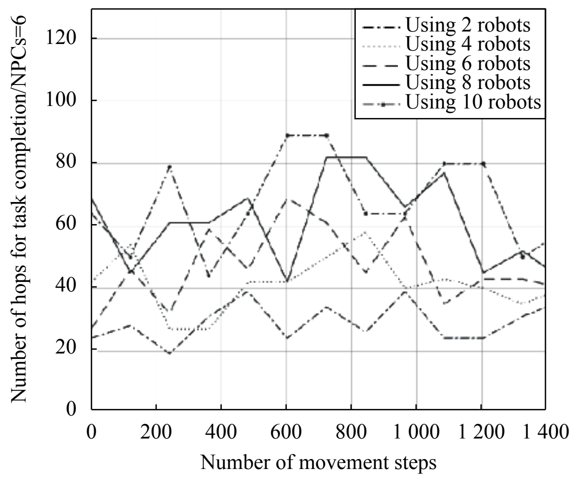


Fig. 16 Number of hops versus number of steps for different robot team sizes

bots it remains within a range of (27 to 58), for a six robot team it remains within a range of (27 to 69), for an eight robot team it remains within a range of (42 to 82), and for a ten robot team it remains within a range of (50 to 89) which means that a small number of hops to complete the task is required when using IoT cloud robotics framework; so, it improves the performance of the task.

5 Conclusions and future works

The HDec-POSMDPs MRS exploration and fire searching algorithm is implemented based on the IoT cloud frame work. The result of the implementation indicates that expensive and tedious computation can be offloaded to the cloud robotics servers. The proposed algorithm achieves significant improvements in the computation burden of the whole task relative to a traditional MRS performing the same task. The preliminary evaluation of this implementation demonstrates that increasing the parallelism of computational instances decreases the delay for new actuation commands which leads to decreasing the mean-time of task completion, minimizing the number of turns in the path from the start pose cells to the target cells, and reducing the energy consumed by each robot.

In the future, this approach must be studied for real-time robots and the QoS must be considered. Another robot application need to be implemented and its performance must be studied to increase the MRS capabilities.

References

[1] S. M. Wen, B. Ding, H. M. Wang, B. Hu, H. Liu, P. C. Shi. Towards migrating resource-consuming robotic software packages to cloud. In *Proceedings of IEEE International Conference on Real-time Computing and Robotics*, IEEE, Angkor Wat, Cambodia, pp. 283–288, 2016. DOI: 10.1109/RCAR.2016.7784040.

[2] R. Janssen, R. Van De Molengraft, H. Bruyninckx, M. Steinbuch. Cloud based centralized task control for human domain multi-robot operations. *Intelligent Service Robotics*, vol. 9, no. 1, pp. 63–77, 2016. DOI: 10.1007/s11370-015-0185-y.

[3] J. Salmerón-García, P. Íñigo-Blasco, F. Díaz-del-Río, D. Cagigas-Muñiz. A tradeoff analysis of a cloud-based robot navigation assistant using stereo image processing. *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 444–454, 2015. DOI: 10.1109/TASE.2015.2403593.

[4] H. J. Li, A. G. Song. Architectural design of a cloud robotic system for upper-limb rehabilitation with multimodal interaction. *Journal of Computer Science and Technology*, vol. 32, no. 2, pp. 258–268, 2017. DOI: 10.1007/s11390-017-1720-4.

[5] A. Manzi, L. Fiorini, R. Esposito, M. Bonaccorsi, I. Mannari, P. Dario, F. Cavallo. Design of a cloud robotic system to support senior citizens: The KuBo experience. *Autonomous Robots*, vol. 41, no. 3, pp. 699–709, 2017. DOI: 10.1007/s10514-016-9569-x.

[6] C. Y. Li, I. H. Li, Y. H. Chien, W. Y. Wang, C. C. Hsu. Im-

proved Monte Carlo localization with robust orientation estimation based on cloud computing. In *Proceedings of IEEE Congress on Evolutionary Computation*, IEEE, Vancouver, Canada, pp. 4522–4527, 2016. DOI: 10.1109/CEC.2016.7744365.

[7] E. Tosello, Z. J. Fan, A. G. Castro, E. Pagello. Cloud-based task planning for smart robots. In *Proceedings of the 14th International Conference on Intelligent Autonomous Systems*, Springer, Shanghai, China, pp. 285–300, 2017. DOI: 10.1007/978-3-319-48036-7_21.

[8] R. Limosani, A. Manzi, L. Fiorini, F. Cavallo, P. Dario. Enabling global robot navigation based on a cloud robotics approach. *International Journal of Social Robotics*, vol. 8, no. 3, pp. 371–380, 2016. DOI: 10.1007/s12369-016-0349-8.

[9] A. Rahman, J. Jin, A. Cricenti, A. Rahman, M. Palaniswami, T. Luo. Cloud-enhanced robotic system for smart city crowd control. *Journal of Sensor and Actuator Networks*, vol. 5, pp. 20–36, 2016. DOI: 10.3390/jsan5040020.

[10] L. J. Wang, M. Liu, M. Q. H. Meng. A pricing mechanism for task oriented resource allocation in cloud robotics. *Robots and Sensor Clouds*, Koubaa A., Shakshuki E., Eds., Cham, Germany: Springer, vol. 36, pp. 3–31, 2016. DOI: 10.1007/978-3-319-22168-7_1.

[11] A. Manzi, L. Fiorini, R. Limosani, P. Sinčák, P. Dario, F. Cavallo. Use case evaluation of a cloud robotics teleoperation system. In *Proceedings of the 5th IEEE International Conference on Cloud Networking*, IEEE, Pisa, Italy, pp. 208–211, 2016. DOI: 10.1109/CloudNet.2016.49.

[12] A. Rodić, M. Jovanović, M. Vujović, D. Urukalo. Application-driven cloud-based control of smart multi-robot store scenario. In *Proceedings of the 25th Conference on Robotics in Alpe-Adria-Danube Region*, Springer, Belgrade, Serbia, pp. 347–357, 2016. DOI: 10.1007/978-3-319-49058-8_38.

[13] A. G. Thallas, K. Panayiotou, E. Tsardoulias, A. L. Symeonidis, P. A. Mitkas, G. G. Karagiannis. Relieving robots from their burdens: The cloud agent concept. In *Proceedings of the 5th IEEE International Conference on Cloud Networking*, IEEE, Pisa, Italy, pp. 188–191, 2016. DOI: 10.1109/CloudNet.2016.38.

[14] A. Rahman, J. Jin, Y. W. Wong, K. S. Lam. Development of a cloud-enhanced investigative mobile robot. In *Proceedings of International Conference on Advanced Mechatronic Systems*, IEEE, Melbourne, Australia, pp. 104–109, 2016. DOI: 10.1109/ICAMechS.2016.7813429.

[15] L. J. Wang, M. Liu, M. Q. H. Meng. Real-time multisensor data retrieval for cloud robotic systems. *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 507–518, 2015. DOI: 10.1109/TASE.2015.2408634.

[16] G. Ermacora, A. Toma, R. Antonini, S. Rosa. Leveraging open data for supporting a cloud robotics service in a smart city environment. In *Proceedings of the 13th International Conference IAS-13*, Springer, Padua, Italy, pp. 527–538, 2016. DOI: 10.1007/978-3-319-08338-4_39.

[17] D. Lorencik, J. Ondo, P. Sincak, H. Wagatsuma. Cloud-based image recognition for robots. In *Proceedings of the 3rd International Conference on Robot Intelligence Technology and Applications*, Springer, Beijing, China, pp. 785–796, 2015. DOI: 10.1007/978-3-319-16841-8_71.

[18] T. Nam Khoon, P. Sebastian, A. B. S. Saman. Autonomous fire fighting mobile platform. *Procedia Engineering*, vol. 41, pp. 1145–1153, 2012. DOI: 10.1016/j.proeng.2012.07.294.

[19] Y. D. Kim, Y. G. Kim, S. H. Lee, J. H. Kang, J. An. Portable fire evacuation guide robot system. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, St. Louis, USA, pp. 2789–2794, 2009. DOI: 10.1109/IROS.2009.5353970.

[20] P. H. Chang, Y. H. Kang, G. R. Cho, J. H. Kim, M. L. Jin, J. Lee, J. W. Jeong, D. K. Han, J. H. Jung, W. J. Lee, Y. B. Kim. Control architecture design for a fire searching robot using task oriented design methodology. In *Proceedings of SICE-ICASE International Joint Conference*, IEEE, Busan, South Korea, pp. 3126–3131, 2006. DOI: 10.1109/SICE.2006.314817.

[21] H. S. Sucuoglu, I. Bogrekci, P. Demircioglu. Development of mobile robot with sensor fusion fire detection unit. *IFAC-PapersOnLine*, vol. 51, no. 30, pp. 430–435, 2018. DOI: 10.1016/j.ifacol.2018.11.324.

[22] K. L. Su. Automatic fire detection system using adaptive fusion algorithm for fire fighting robot. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, IEEE, Taipei, China, pp. 966–971, 2006. DOI: 10.1109/ICSMC.2006.384525.

[23] B. Hu, H. M. Wang, P. F. Zhang, B. Ding, H. M. Che. Cloudroid: A cloud framework for transparent and QoS-aware robotic computation outsourcing. In *Proceedings of the 10th IEEE International Conference on Cloud Computing*, IEEE, Honolulu, USA, pp. 114–121, 2017. DOI: 10.1109/CLOUD.2017.23.

[24] Y. Y. Li, H. M. Wang, B. Ding, P. C. Shi, X. Liu. Toward QoS-aware cloud robotic applications: A hybrid architecture and its implementation. In *Proceedings of International IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, IEEE, Toulouse, France, pp. 33–40, 2016. DOI: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0028.

[25] L. Zhang, H. X. Zhang, Z. Fang, X. B. Xiang, M. Huchard, R. Zapata. Towards an architecture-centric approach to manage variability of cloud robotics. In *Proceedings of DSLRob: Domain-specific Languages and Models for ROBotic Systems*, HAL, Hamburg, Germany, 2015.

[26] H. Z. Chen, G. H. Tian, F. Lu, G. L. Liu. A hybrid cloud robot framework based on intelligent space. In *Proceedings of the 12th World Congress on Intelligent Control and Automation*, IEEE, Guilin, China, pp. 2996–3001, 2016. DOI: 10.1109/WCICA.2016.7578487.

[27] C. Razafimandimby, V. Loscri, A. M. Vegni. Towards efficient deployment in internet of robotic things. *Integration, Interconnection, and Interoperability of IoT Systems*, R. Gravina, C. E. Palau, M. Manso, A. Liotta, G. Fortino, Eds., Cham, Germany: Springer, pp. 21–37, 2018. DOI: 10.1007/978-3-319-61300-0_2.

[28] P. P. Ray. Internet of robotic things: Concept, technologies, and challenges. *IEEE Access*, vol. 4, pp. 9489–9500, 2016. DOI: 10.1109/ACCESS.2017.2647747.

[29] H. H. Yan, Q. S. Hua, Y. Y. Wang, W. G. Wei, M. Imran. Cloud robotics in smart manufacturing environments: Challenges and countermeasures. *Computers & Electrical Engineering*, vol. 63, pp. 56–65, 2017. DOI: 10.1016/j.compeleceng.2017.05.024.

[30] R. Doriya, P. Sao, V. Payal, V. Anand, P. Chakraborty. A

review on cloud robotics based frameworks to solve simultaneous localization and mapping (SLAM) problem. *International Journal of Advances in Computer Science and Cloud Computing*, vol. 3, no. 1, pp. 40–45, 2015.

[31] L. H. Wang, X. V. Wang. Resource efficiency calculation as a cloud service. *Cloud-based Cyber-physical Systems in Manufacturing*, L. H. Wang, X. V. Wang, Eds., Cham, Germany: Springer, pp. 195–209, 2018. DOI: 10.1007/978-3-319-67693-7_8.

[32] R. Arumugam, V. R. Enti, L. B. B. Liu, X. J. Wu, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, G. W. Kit. DAvinCi: A cloud computing framework for service robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Anchorage, USA, pp. 3084–3089, 2010. DOI: 10.1109/ROBOT.2010.5509469.

[33] M. Garzón, J. Valente, J. J. Roldán, D. Garzón-Ramos, J. De León, A. Barrientos, J. Del Cerro. Using ROS in multi-robot systems: Experiences and lessons learned from real-world field tests. *Robot Operating System (ROS)*, A. Koubaa, Ed., Cham, Germany: Springer, pp. 449–483, 2017. DOI: 10.1007/978-3-319-54927-9_14.

[34] J. H. Xiao, D. Xiong, W. J. Yao, Q. H. Yu, H. M. Lu, Z. Q. Zheng. Building software system and simulation environment for RoboCup MSL soccer robots based on ROS and gazebo. *Robot Operating System (ROS)*, A. Koubaa, Ed., Cham, Germany: Springer, pp. 597–631, 2017. DOI: 10.1007/978-3-319-54927-9_18.

[35] P. Yun, J. H. Jiao, M. Liu. Towards a cloud robotics platform for distributed visual SLAM. In *Proceedings of the 11th International Conference on Computer Vision Systems*, Springer, Shenzhen, China, pp. 3–15, 2017. DOI: 10.1007/978-3-319-68345-4_1.

[36] X. V. Wang, L. H. Wang, A. Mohammed, M. Givehchi. Ubiquitous manufacturing system based on cloud: A robotics application. *Robotics and Computer-Integrated Manufacturing*, vol. 45, pp. 116–125, 2017. DOI: 10.1016/j.rcim.2016.01.007.

[37] W. H. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, K. Nakamura. QoS-aware robotic streaming workflow allocation in cloud robotics systems. *IEEE Transactions on Services Computing*, to be published. DOI: 10.1109/TSC.2018.2803826.

[38] J. F. Wan, S. L. Tang, H. H. Yan, D. Li, S. Y. Wang, A. V. Vasilakos. Cloud robotics: Current status and open issues. *IEEE Access*, vol. 4, pp. 2797–2807, 2016. DOI: 10.1109/ACCESS.2016.2574979.

[39] E. Cardarelli, V. Digani, L. Sabattini, C. Secchi, C. Fantuzzi. Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses. *Mechatronics*, vol. 45, pp. 1–13, 2017. DOI: 10.1016/j.mechatronics.2017.04.005.

[40] G. Q. Hu, W. P. Tay, Y. G. Wen. Cloud robotics: Architecture, challenges and applications. *IEEE Network*, vol. 26, no. 3, pp. 21–28, 2012. DOI: 10.1109/MNET.2012.6201212.

[41] A. Saxena, A. Jain, O. Sener, A. Jami, D. K. Misra, H. S. Koppula. RoboBrain: Large-scale knowledge engine for robots. [Online], Available: https://arxiv.org/abs/1412.0691.

[42] J. F. Wan, S. L. Tang, Q. S. Hua, D. Li, C. L. Liu, J. Lloret. Context-aware cloud robotics for material handling in cognitive industrial internet of things. *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2272–2281, 2018. DOI: 10.1109/JIOT.2017.2728722.

[43] B. W. Liu, Y. Chen, E. Blasch, K. Pham, D. Shen, G. S. Chen. A holistic cloud-enabled robotics system for real-time video tracking application. *Future Information Technology*, J. J. Park, I. Stojmenovic, M. Choi, F. Xhafa, Eds., Berlin Heidelberg, Germany: Springer, pp. 455–468, 2014. DOI: 10.1007/978-3-642-40861-8_64.

[44] M. Bonaccorsi, L. Fiorini, F. Cavallo, A. Saffiotti, P. Dario. A cloud robotics solution to improve social assistive robots for active and healthy aging. *International Journal of Social Robotics*, vol. 8, no. 3, pp. 393–408, 2016. DOI: 10.1007/s12369-016-0351-1.

[45] M. Tenorth, K. Kamei, S. Satake, T. Miyashita, N. Hagita. Building knowledge-enabled cloud robotics applications using the ubiquitous network robot platform. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Tokyo, Japan, pp. 5716–5721, 2013. DOI: 10.1109/IROS.2013.6697184.

[46] G. Mohanarajah, D. Hunziker, R. D′Andrea, M. Waibel. Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 481–493, 2015. DOI: 10.1109/TASE.2014.2329556.

[47] D. Hunziker, M. Gajamohan, M. Waibel, R. D′Andrea. Rapyuta: The RoboEarth cloud engine. In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Karlsruhe, Germany, pp. 438–444, 2013. DOI: 10.1109/ICRA.2013.6630612.

[48] G. Mohanarajah, V. Usenko, M. Singh, R. D′Andrea, M. Waibel. Cloud-based collaborative 3D mapping in real-time with low-cost robots. *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 423–431, 2015. DOI: 10.1109/TASE.2015.2408456.

[49] H. Bao, H. M. Wang, B. Ding, S. N. Shang. Cloud-based knowledge sharing in cooperative robot tracking of multiple targets with deep neural network. In *Proceedings of the 24th International Conference on Neural Information Processing*, Springer, Guangzhou, China, pp. 71–80, 2017. DOI: 10.1007/978-3-319-70136-3_8.

[50] S. Kamburugamuve, H. J. He, G. Fox, D. Crandall. Cloud-based parallel implementation of SLAM for mobile robots. In *Proceedings of the International Conference on Internet of things and Cloud Computing*, ACM, Cambridge, UK, Article number 48, 2016. DOI: 10.1145/2896387.2896433.

[51] H. J. He, S. Kamburugamuve, G. C. Fox, W. Zhao. Cloud based real-time multi-robot collision avoidance for swarm robotics. *International Journal of Grid and Distributed Computing*, vol. 9, no. 6, pp. 339–358, 2016. DOI: 10.14257/ijgdc.

[52] A. Rahman, J. Jin, A. L. Cricenti, A. Rahman, A. Kulkarni. Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 2500–2511, 2019. DOI: 10.1109/TII.2018.2874693.

[53] W. H. Chen, Y. Yaguchi, K. Naruse, Y. Watanobe, K. Nakamura, J. Ogawa. A study of robotic cooperation in cloud robotics: Architecture and challenges. *IEEE Access*, vol. 6, pp. 36662–36682, 2018. DOI: 10.1109/ACCESS.2018.2852295.

[54] S. A. Miratabzadeh, N. Gallardo, N. Gamez, K. Haradi, A. R. Puthussery, P. Rad, M. Jamshidi. Cloud robotics: A software architecture: For heterogeneous large-scale autonomous robots. In *Proceedings of World Automation*

Congress, IEEE, Rio Grande, Puerto Rico, pp. 1–6, 2016. DOI: 10.1109/WAC.2016.7583017.

[55] N. Tian, M. Matl, J. Mahler, Y. X. Zhou, S. Staszak, C. Correa, S. Zheng, Q. Li, R. Zhang, K. Goldberg. A cloud robot system using the dexterity network and Berkeley robotics and automation as a service (Brass). In *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, Singapore, pp. 1615–1622, 2017. DOI: 10.1109/ICRA.2017.7989192.

[56] A. Koubaa, B. Qureshi. DroneTrack: Cloud-based real-time object tracking using unmanned aerial vehicles. *IEEE Access*, vol. 6, pp. 13810–13824, 2018. DOI: 10.1109/ACCESS.2018.2811762.

[57] F. Yan, Y. S. Liu, J. Z. Xiao. Path planning in complex 3D environments using a probabilistic roadmap method. *International Journal of Automation and Computing*, vol. 10, no. 6, pp. 525–533, 2013. DOI: 10.1007/s11633-013-0750-9.

[58] K. Mohamed, A. Elshenawy, H. Harb. Exploration strategies of coordinated multi-robot systems: A comparative study. *International Journal of Robotics and Automation*, vol. 7, no. 1, pp. 48–58, 2018.

[59] K. Mohamed, A. El Shenawy, H. Harb. A hybrid decentralized coordinated approach for multi-robot exploration task. *The Computer Journal*, to be published. DOI: 10.1093/comjnl/bxy107.

[60] A. El Shenawy, K. M. Khalil, H. M. Harb. A task decomposition using (HDec-POSMDPs) approach for multi-robot exploration and fire searching. *International Journal of Imaging and Robotics*, no. 19, pp. 2–2019, 2019.

[61] J. Faigl, M. Kulich, L. Přeučil. Goal assignment using distance cost in multi-robot exploration. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, Vilamoura, Portugal, pp. 3741–3746, 2012. DOI: 10.1109/IROS.2012.6385660.

[62] M. Al-khawaldah, T. M. Younes, I. Al-Adwan, M. Nisirat, M. Alshamasin. Automated multi-robot search for a stationary target. *International Journal of Control Science and Engineering*, vol. 4, no. 1, pp. 9–15, 2014. DOI: 10.5923/j.control.20140401.02.

[63] S. Omidshafiei, A. A. Agha-Mohammadi, C. Amato, S. Y. Liu, J. P. How, J. Vian. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 231–258, 2017. DOI: 10.1177/0278364917692864.

[64] Y. Kantaros, M. M. Zavlanos. Distributed intermittent connectivity control of mobile robot networks. *IEEE Transactions on Automatic Control*, vol. 2, no. 7, pp. 3109–3121, 2017. DOI: 10.1109/TAC.2016.2626400.

[65] A. Pal, R. Tiwari, A. Shukla. Coordinated multi-robot exploration under connectivity constraints. *Journal of Information Science and Engineering*, vol. 29, pp. 711–727, 2013.

Ayman El Shenawy received the Ph. D. degree in systems and computer engineering from Al-Azhar University, Egypt in 2013. He is currently working as a lecturer at Systems and Computers Engineering Department, Faculty of Engineering Al-Azhar University, Egypt. He already developed some breakthrough research in the mentioned areas. He made significant contributions to the stated research fields.

His research interests include artificial intelligent methods, robotics and machine learning.

E-mail: eaymanelshenawy@azhar.edu.eg (Corresponding author)

ORCID iD: 0000-0002-1309-6449

Khalil Mohamed received the M. Sc. degree in control engineering from Al-Azhar University, Egypt in 2015. He is currently a Ph. D. degree candidate in robotic systems at Systems and Computers Engineering Department, Al-Azhar University, Egypt.

His research interests includes task assignment in multi-robot systems, task decomposition, predictive control and optimal control.

E-mail: mel_khalil@yahoo.com

Hany Harb received the B. Sc. degree in computers and control engineering from Faculty of Engineering, Ain Shams University, Egypt in 1978, the M. Sc. degree in computers and systems engineering from Faculty of Engineering, Al-Azhar University, Egypt in 1981. He also received the Ph. D. degree in computer science and the M. Sc. degree in operations research (MSOR) from Institute of Technology (IIT), USA in 1986 and 1987, respectively. He is a professor of software engineering in System Engineering Department, Faculty of Engineering, Al-Azhar University, Egypt.

His research interests include artificial intelligence, cloud computing, and distributed systems.

E-mail: harbhany@yahoo.com