Computer Networks 160 (2019) 11–32

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Energy-efficient optimal task offloading in cloud networked multi-robot systems

Akhlaqur Rahman[a,*], Jiong Jin[a], Ashfaqur Rahman[b], Antonio Cricenti[a], Mahbuba Afrin[a], Yu-ning Dong[c]

[a] School of Software and Electrical Engineering, Swinburne University of Technology, VIC 3122, Australia
[b] Data61, CSIRO, TAS 7005, Australia
[c] College of Telecommunications & Information Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China

ARTICLE INFO

Article history:
Received 14 March 2018
Revised 7 February 2019
Accepted 20 May 2019
Available online 22 May 2019

Keywords:
Cloud networked robotics
Multi-robot systems
Task offloading
Path planning
Genetic algorithm

ABSTRACT

Task offloading plays a critical role in cloud networked multi-robot systems for leveraging computation support from cloud infrastructure and benefiting greatly from the well-developed cloud network facilities. However, considering the delay constraint, the extra costs of data transmission and remote computation, it is not trivial to make optimized offloading decisions. In particular, task offloading for robots is more complex due to their on-demand mobility and network connectivity that significantly influence the robot–cloud communication links. Moreover, for multi-robot systems, a suitable balance of workload between local network (robot–robot) and global cloud (robot–cloud) is also required, so as to attain proper utilization of resources. Therefore, it is essential to establish more comprehensive offloading schemes for modeling systems that can handle these higher level of complications. With that view, this paper aims to develop a novel multi-layer decision-making scheme for task offloading which jointly considers the following four aspects: (i) selection of task for offloading, (ii) selection of robot to offload a task, (iii) selection of location to offload/perform task, (iv) selection of access point for offloaded task. An integrated framework for cloud networked multi-robot systems is presented to enable our task offloading scheme where the primary robot can aid from additional local robots to improve the offloading process. In particular, we consider a warehouse scenario with 36 cell workspace where a 40 node taskflow is motivated from a "parcel sorting and distribution" application, to be completed by the primary robot. The offloading decision for each task is formulated as part of a joint optimization problem and it is solved by developing a multi-layer genetic algorithm scheme that takes into account motion, network connectivity and local sharing for its offloading decisions. We evaluate the results of the scheme via comparison with two validated benchmarks. The outcome highlights a significant improvement in overall system performance due to joint involvement of motion (path planning), connectivity (bandwidth estimation) and robot–robot communication (local offloading) that facilitates energy-efficient offloading to cloud, faster completion of tasks and better utilization of available resources.

1. Introduction

The widespread innovation in robotic technology and subsequent increase in their computing capabilities are enabling their usage in different areas of modern society. Especially, in the industrial realm, the technical advancement in wireless network technology [1], Internet of Things (IoT) [2] and artificial intelligence have given rise to more progressive networked robotic applications that moves beyond their traditional deployment in production lines and deals with new challenges of industrial applications such as negotiation based decision-making, dynamic environmental disruptions, human–machine interaction and more personalized consumption demands etc. To combat all the complexity and diversity that it brings, significant hardware and software advancements are required for the robotic infrastructure. Unfortunately, there are still several limitation in the ways a robot hardware can actually be used or upgraded due to economical and functional constraints. As for the software advancements, this is where "cloud networked

* Corresponding author at: John St, Hawthorn, VIC 3122, Australia.
E-mail addresses: akhlaqurrahman@swin.edu.au (A. Rahman), jiongjin@swin.edu.au (J. Jin), ashfaqur.rahman@data61.csiro.au (A. Rahman), tcricenti@swin.edu.au (A. Cricenti), mafrin@swin.edu.au (M. Afrin), dongyn@njupt.edu.cn (Y.-n. Dong).

https://doi.org/10.1016/j.comnet.2019.05.016
1389-1286/

robotics" has made its mark. The introduction of cloud networked multi-robot system has merged the two ever-progressing domains of networked robotics and cloud computing. The added feature of cloud implies less dependence on human input and more support from ubiquitous resources [3] (i.e., computation, storage, crowd-sourcing etc.). Therefore, most cloud aided multi-robot applications now-a-days generally operate with cooperative control that adopts decentralized approach to avoid a single point of failure and perform tasks with reduced energy consumption with lesser time/delay.

Prime examples of multi-robot in Industry 4.0 applications include: material handling [4], assembly line [5], warehouse maintenance, cooperative navigation [6] etc. All these applications require automated robot coordination as well as distributed computing, which matches the scope of the fourth industrial revolution (i.e., Industry 4.0). The integration of cloud network adds another dimension in this context that has the potential of contributing in task completion and making the performance smoother. But it takes place at the expense of complicating the decision-making process, as a proper balance of workload is required between the cloud infrastructure and the heterogeneous robotic units. More specifically, in industrial sector, most of the industrial applications now-a-days have demand for automated and customized services that can deal with personalized consumption demands. This calls for more dynamic applications with sophisticated capabilities that can deal with computation-heavy task requests and utilize its resources to the fullest. Hence the implementation of cloud networked multi-robot system presents itself as the perfect fit. One of the most common and well-studied industrial robotic applications is "Warehouse Logistics" where customized ordered parcels are traditionally sorted and distributed with the help of human labor and support from heavy machineries. For smart factories though, the inclusion of interactive cloud-aided robots with advanced communication technology produces a shift in the modes of applications from carrying out repetitive tasks towards performing dynamic tasks that requires robots to solve complex multi-objective problems, thus playing a pivotal role in design and management of smart warehouses.

There have been numerous studies for multi-robot platforms on the design of automatic warehouses [7], its multi-robot functions (e.g., task assignment [8], coordination, path planning [6], speed improvement) as well as its various range of applications (such as disaster management [9], automated order processing [10], assembly cell control [11] etc.). As for cloud networked robot systems, some notable studies have focused on manufacturing applications [4], maintenance related tasks [12] and computer vision. In recent past, several studies have emphasized on cloud–robot collaborative aspects of industrial applications including cloud-assisted negotiation technique with industrial robots [13], localization through deep learning with cloud support [14] and development of novel green software evaluation model for energy minimization [15]. However, cloud-aided automated robotic approaches for warehouse logistics and studies related to that are few and far between [4]. Some conceptual work by Bonkenburg has suggested the possible ways where robots can be used in the environment of smart warehouses [16]. Our work is partially motivated by this concept, where we consider a warehouse application for automated parcel sorting and distribution. However, the emphasis on our work is on the integrated framework for task offloading that enables communication between the robotic network and cloud for the offloading based decision-making, proper allocation of tasks and transfer of information among resources. This motivates our formulation of a joint optimization problem where task offloading decisions are presented as an allocation problem, to be solved by a novel genetic algorithm (GA) based multi-layer decision-making scheme.

For Multi-Robot Task Allocation (MRTA) optimization problems, the notable current studies center on various algorithms (e.g., heuristic [17], timed automata model [18], market based approach, swarm intelligence [19], task-grouped improved static allocation algorithm, decentralized approach [20] etc.) that perform successfully in solving optimization problems. However, all these approaches mostly focus on one single variable (task/robot/path planning/ allocation) to tackle the problem. In order to keep up with the rapid increase in technology and handle more complex systems, the progressive approach is to prepare more rigid and comprehensive techniques with interdependent parameters, which leads us to our work where we have considered four variables (task offloading, robot selection for offloading, path selection and access point selection) as part of a multi-layer decision-making set.

Firstly, task offloading is one of the major benefits of cloud computing where computation-heavy and resource hungry tasks are migrated to a remote yet powerful cloud server for execution [21]. Since the ubiquitous resources of cloud can be rapidly provisioned and released with minimal service provider interaction or management effort, therefore cloud computing allows energy-constraint robot to offload a portion of the computation to cloud in order to potentially reduce execution time and energy. The key decision here is to identify the appropriate tasks to offload, as it may depend on the constraints as well as the type of tasks and objective of the application. In addition to that, the task offloading decisions are also heavily influenced by two critical factors that impact the robot's communication with the cloud. They are: mobility and bandwidth.

The mobility of the robot relates to the path it selects for movement. The term "path planning" has numerous meanings depending on which field it is used. In robotics, path planning concerns with the problem as how to move a robot between multiple points [22]. This involves motion-planning, trajectory mapping and planning under uncertain environments. Through path planning, the robot can find a route from the beginning to the destination in order to meet certain evaluation criteria; while avoiding static as well as dynamic obstacles. This creates opportunities for dynamic and complex robotic operations that includes motion-aware movement and target oriented decision-making. The access point (AP) selection on the other hand, directly relates to the concept of bandwidth assessment. The proliferation of wireless access technologies offers users the possibility to choose among multiple networks based on the available bandwidth [23]. Users can gain the best connectivity (bandwidth) based on their selection of AP. However, complexity arises because bandwidth is dependent on several parameters such as location, availability and number of users, which makes the process intricate.

Since task offloading is dependent on communication with the cloud, it requires Internet availability. Depending on the location of robot and the selection of AP, a robot can gain access to the available bandwidth, which in turn may impact the offloading process of the robot. In this way, it is quite clear that task offloading, path planning and AP selection are interdependent that allows robot to plan their path and select the communication link while accommodating the offloading decisions. This leads us to our current work which is motivated from Rahman et al., [24,25], where all three parameters were considered as part of a 3-layer joint optimization for a single robot application. Here the concept of a motion and connectivity aware offloading for a single robot was presented by integrating three key optimal decisions (e.g., offloading, task location and AP selection) for each task that improve system performance by optimally offloading tasks to the cloud. Contrary to that work, our approach in this work is for a multi-robot application. In addition to motion and connectivity, we simultaneously focus on two additional key issues: (i) local resource sharing (robot–robot), (ii) offloading to the cloud

(robot–cloud). The added feature of local sharing among the multi-robots has the potential of further minimizing the system energy by distributing the principle robot's workload to other available robots either for computational support or help with offloading to the cloud. Therefore our method in this paper considers four layers (e.g., task offloading, robot selection for offloading, task location and AP selection) of decision-making for each task. We believe, by utilizing the other available robots from the multi-robot systems, the offloading process can be further fastened which can lead to successful task completion with lesser consumption of energy as well as enhancement of the system performance. A thorough simulation has been run later in the paper and the results between the two above mentioned methods have been compared in order to highlight the benefits of our approach.

The application we consider for simulation in this work is based on a warehouse parcel sorting and distribution process, for which we have presented an application taskflow (DAG) detailing all the steps. We also formulated a four-layer joint optimization problem with the objective of minimizing the overall robot energy. We then implemented an evolutionary GA based scheme with a novel 4-layer chromosome/solution for minimizing the overall system energy by identifying the following key decisions: (i) which task to allocate to robot and which task to offload to the cloud, (ii) which robot would offload the task to the cloud, (iii) which location to complete a task and (iv) which access point (AP) to select for offloading a particular task. The simulation results suggest that the implementation of our GA based scheme helps identify near-optimal solutions and in-turn improves the overall system performance for cloud networked multi-robot applications. The contribution of this paper is listed as follows:

- To present an integrated framework for task offloading in cloud networked multi-robot systems.
- To formulate a joint-optimization problem based on the warehouse parcel sorting and distribution application with a novel 4-layer variable (task offloading, robot selection for offloading, path planning, AP selection) to minimize energy.
- To develop a multi-layer genetic algorithm (GA) based decision-making scheme for task offloading in order to solve the joint optimization problem.
- To compare the results of our approach with two validated benchmarks [24] and show the benefits of cloud based task offloading via multiple robots on overall system performance.

The rest of the paper is prepared as follows. Section 2 describes our integrated framework for task offloading in a cloud networked multi-robot system. This leads to the description of our task offloading scheme in multi-robot systems that considers motion and connectivity as part of its decision-making. Later in Section 3, we present a cast study for an automated warehouse parcel sorting and distribution application. This leads us to our problem formulation for joint optimization in the next section and the development of our 4-layer GA based decision-making scheme for task offloading in Section 5. Finally, thorough simulation is run to analyze and compare the results of the decision-making scheme with respect to two validated benchmarks before finishing the paper in Section 7.

2. An integrated framework for robotic task offloading

In this section, we present the components of an integrated framework for robots task offloading. Now-a-days, the integration of machine learning in IoT enabled sensors and computation support from the cloud infrastructure has empowered multi-robot systems to provide on-demand service for a vast range of automated industrial applications [26] with increased efficiency for environmental monitoring, upgraded supply chains, reduced waste, more safety and improved speed of smart factory environment. This motivates our four-tier task offloading framework for cloud networked multi-robot system (Fig. 1). The components are: (i) physical layer consisting of robots and sensors, (ii) network layer with APs, (iii) cloud infrastructure, and (iv) supervisory control center overviewing the application. Given the autonomous nature of the application, robots independently perform task related actions and are hereby the core component. As part of this framework, robots and sensors can communicate with each other and the cloud by sharing a common set of knowledge/information and negotiating according to a common set of rules. Thus it results in a highly smart and flexible framework that is self- organized and reconfigurable in nature. More details on components of the framework is given as follows:

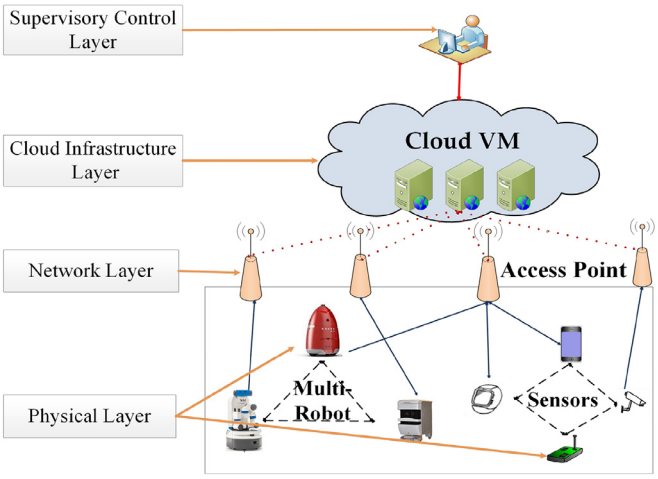


Fig. 1. Framework for task offloading in cloud networked multi-robot systems.

2.1. Physical layer

The industrial physical layer consists of two components: wireless sensor network and robotic agents. The wireless sensor network (WSN) consists of low-cost and limited-energy smart sensor devices embedded in machines that can communicate with each other and collect as well as analyze raw data necessary for application specific tasks. The sensors provide directive information to the robot through cloud by using APs. Based on that, several analytical computation is done and task is assigned to robots for visiting locations and performing tasks that are allocated to them. Here machine learning in sensors help them to identify key situations such as suggest modifications or requirements for actions that can be performed by robots.

The robots are the critical element that can perform actuation and help complete the tasks. Depending on the type of application, the robots possesses the ability to share its workload with the cloud by offloading to the cloud for additional support. Robots may also get guidance provided by the supervisory control center through the cloud. Based on all of that, tasks are shared between robotic and cloud resources. As robots possess the distinctive attribute of mobility, they can plan their route in accordance and hence choose the suitable communication platform to access the cloud while moving. As a result, it gains the best available bandwidth for offloading tasks to cloud. Thus robots offloading with motion and connectivity as part of its decision-making can help improve communication with the cloud. Finally, robots can also form their own local ad-hoc network to communicate and share information among each other as well as assist with sending tasks to cloud.

Thus the framework allows robots to utilize its surrounding local resources (available robots) and cloud based resources (virtual machine) to share workload as part of a fluid communication model with cloud. In the context of our work, there is a primary robot in-charge of the application and may incorporate other available robots to aid with offloading, thus making it multi-robots.

2.2. Network layer

The industrial network layer consists of access points (AP) that enable the robotic and sensor network to communicate with each other. It also bridges the gap between cloud services and the physical layer components for data collection and uploading. In this context the AP is defined as a smart device with Internet capabilities that helps robot access the infrastructure of the cloud. As there are multiple APs, therefore the robots can gain different stream rates for communication depending on its location and choice of AP, in accordance with protocol IEEE 802.11 WLANs. It means, the bandwidth received by a robot may vary depending on the location it offloads tasks from or the robot that is offloading or the AP it is selecting. Thus the tangible network layer enables the in-tangible information to flow freely by integrating the physical components and information entities. More details about the communication bandwidth for task offloading is provided later in the paper.

2.3. Cloud infrastructure layer

Cloud infrastructure refers to hardware and software components such as servers, storage and virtualization software that are needed to support the computing requirements of the application. It includes an abstraction layer that virtualizes resources and logically presents them to users through application program interfaces and API-enabled command-line or graphical interfaces. The organization of cloud typically consists of virtual machines (VM) with shared power that provide functionality needed to execute the entire high-density operating systems and requires massive computational capacity to handle unpredictable and complex user (robot) demands. Some of the notable cloud service providers are: Mendix, Google App Engine, Amazon Web Services etc. In the context of our work, cloud services mainly refer to the VMs that virtualize computing resources as the back-end components and perform on-demand computational support (for offloaded task), data storage (data collected from sensors), analytics (decision-making, verification) to aid local robots.

2.4. Supervisory control layer

The supervisory control layer allows networked robots to be guided/monitored by humans remotely through the cloud infrastructure. Here the information collected from sensors and action reports performed by robots are passed on to cloud and made available for users to monitor through control terminals. As a result, physical layer can communicate with users/engineers in remote locations when required. In addition, possible big data analytics can also provide various statistical results to the users for the purpose of supervisory control and the users can verify/adjust system configuration accordingly. In large industrial operations, this two-way communication allows remote engineers to monitor the robots and maintain the performance of applications.

Based on our framework, we propose a task offloading scheme that collectively considers aspects of motion and connectivity for its offloading decision-making in a cloud networked multi-robot application (as seen in Fig. 2). It exploits the inter-relation between offloading, path planning, network bandwidth and local robot–robot (R–R) sharing. As mobile robots can move on demand, it can go to intended locations and perform action-based tasks. Due to

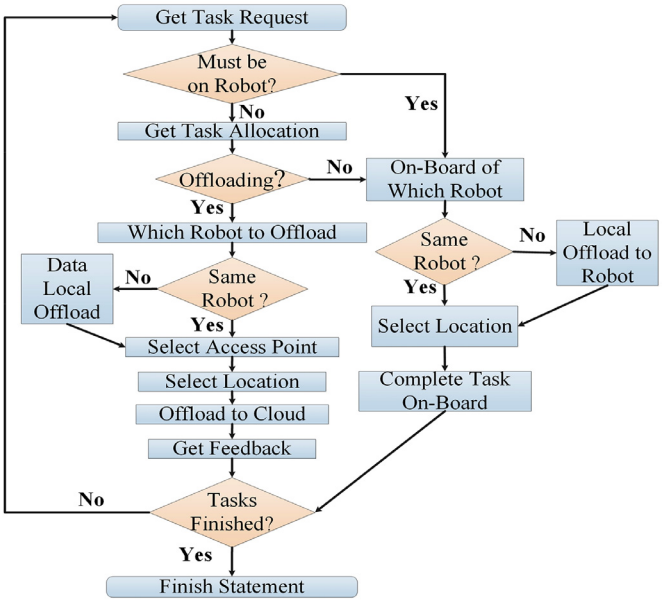


Fig. 2. Task offloading process with collective consideration for motion, connectivity and local sharing.

inclusion of cloud, robot can gain computation support for analytical tasks. In case of offloading, mobile robots also has the opportunity of choosing locations to offload tasks. Since different locations have different stream rate (bit rate) to the APs, the selection of AP and the choice of location interdependently influence the available bandwidth for robot's cloud based communication. In this way, task offloading to the cloud for robotic agents is heavily dependent on its decision-making that considers motion and connectivity issues. Moreover for multi-robot systems, there is an added dimension of local robot–robot communication, which makes the process more complicated, but trades off in terms of better offloading through support from local robots. Hence, offloading scheme for cloud networked multi-robot system is divided into two types depending on the respective decision of offloading for each task.

- **Cloud based task offloading (robot–cloud)**

Depending on the allocation, when a task is selected to be offloaded to the cloud, the robot needs to communicate with cloud infrastructure via network layer APs (Fig. 2). In such cases, the following decisions are considered: *a) which task to offload, b) which robot to offload, c) which location to offload from, d) which access point to select for offloading.*

- **Local task offloading (robot–robot)**

Local task offloading can occur in two scenarios. When a task is offloaded to the cloud, the choice of a separate robot (other than the primary robot) from previous tasks enables local offloading (robot–robot). The secondary robot collects task related information from the primary robot and then transfers that to cloud. This may happen when the primary robot has parallel tasks to complete or the secondary robot is in a better location to offload to the cloud. As for the case when tasks are not offloaded, they take place on-board of a fix robot or are locally offloaded to another robot. This also requires primary robots to share information with the other robots after one task is finished. As robots can form local networks in an ad-hoc manner, therefore the robots use this to transfer information to nearby other available robots (within range) for task completion. Here primary robot is in-charge of the decision-making (Fig. 2), which may include: *a) which task to take*

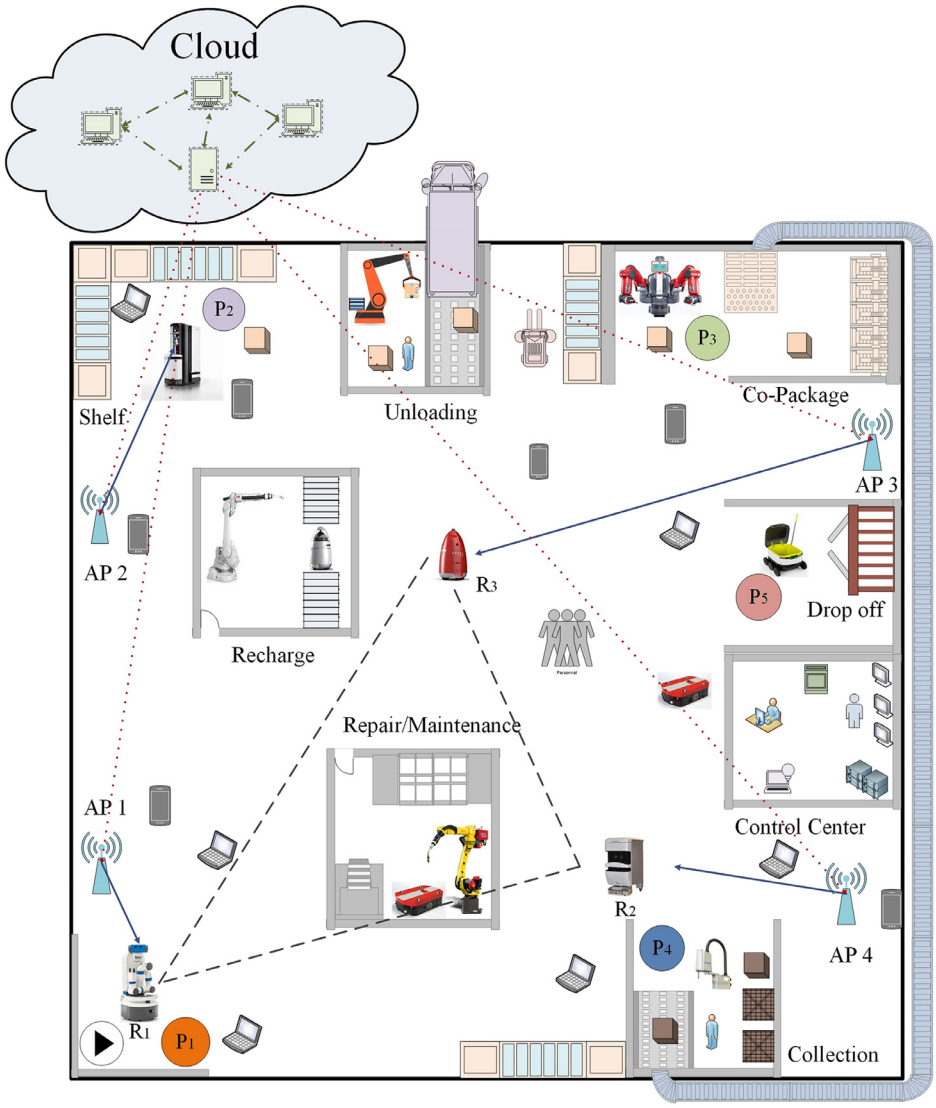


Fig. 3. Floor schematic of a smart warehouse environment.

place on a robot, b) which robot to complete the task, c) weather to locally offload task to another robot.

3. A use case for our framework: warehouse management

Following first 3 revolutions of "Mechanization", "Mass Production" and "Digitization", industrial sector is currently going through a fourth revolution in which emerging autonomous technologies are transforming traditional factories into smart factories for Industry 4.0 [27] applications. The impact of such advancement is being reflected in the warehouse applications as well, where more operations are now moving towards running with automation support. According to recent reports, around 15% of current warehouses are mechanized. Even though 5% of the warehouses are automated, most of them are typically mechanized environment that still employ people in key functions. It suggests that there is room for implementation of automated components i.e., robotic agents. In fact, employment of robots in the warehouse management can be traced back to early 90s, where the approach initially started with teleoperation and later upgraded to automation. Having evolved with time, it has now reached the age of cloud networked robotics, where cloud computing in robotic applications has made significant mark in the industrial realm by enhancing operations of the robots via on-demand computation and storage support.

Through current efforts of IoT in smart warehouses with the inclusion of cloud-enabled robots, opportunity for innovation arises through the integration of networked robotic systems, IoT enabled sensors and the cloud infrastructure for intelligent perception and on-demand shared resources [26]. Such implementation has resulted in several automated warehouse applications for material handling including conveyers, sorters, goods to picker solution and other mechanized equipment that has potential to improve the productivity of existing workforce [16]. In this context of our application, a pool of wireless sensors are deployed in static warehouse machineries (e.g., goods packing, labeling etc.) for data collection and environmental monitoring in order to develop a common operating picture (COP). These sensors are complemented by several dynamic robots that move on-demand to perform object pick up, delivery and drop-off. The integration of these cyber physical systems and wireless sensors enable proper communication over networks for data sharing and automated processing of operations that start from production line all the way to delivery. Since the

design and operation in industrial operations involve numerous varieties of decision-making [28], therefore the inclusion of cloud computing makes an integrated framework of networked robots, sensors and cloud that can reconcile conventional warehouse problems and perform applications in a semi-automated manner with minimal human supervisory oversight, increased efficiency and more safety and speed.

In this paper, we present a "*parcel sorting and distribution application*" in an automated warehouse environment (Fig. 3). Being automated in nature, the application deals with 5 major steps that require the primary mobile robot to complete set of tasks necessary to carry out a parcel for delivery. For our multi-robot system, we incorporate the principle/primary robot to complete the major actuation-based tasks through interactions with different types of agents each with a specific job to help with such as unloading objects from trucks, co-packing, picking orders, checking inventory or shipping goods. As for the supporting robots, they provide analytical and computation support while completing their own set of application related activities. Through this multi-robot communication, the principle robot can transfer tasks locally (robot–robot) to other available robots and complete them on-board. Alternatively, it can also get help from the supporting robots regarding offloading a task to cloud for utilizing its ubiquitous resources. In this way, the other available robots in this shared framework can work as a hub to provide assistance for local or cloud based offloading of tasks (if required), whereas the principle robot carry out fundamental aspects of the warehouse management application as well as necessary decision-making. As seen in Fig. 4, the complete warehouse based application can be divided into four major steps that involves the primary robot visiting five different locations (Fig. 5b).

3.1. Parcel request generation phase (stage 1)

As the warehouse distribution and sorting centers are equipped with sensors, therefore the complete application will be coordinated through advanced warehouse management systems. Each machineries will be equipped with sensors to track inventory movements and progress orders with a high degree of accuracy. As part of it, whenever a new order is set to be sorted and delivered, information regarding its location and target will be sent to the principle robot (R_1), which in this case is a Fetch and Freight robot (Fig. 3), provided by Fetch Robotics [16]. Its primary robot, called Fetch, can extend its torso to reach pickup points while a small secondary robot, called Freight, helpfully holds the tote that Fetch will pick items into. Each Fetch robot can have several of these smaller Freight robots supporting the picking process. Moreover, due to their size they can smoothly move around and collect objects throughout the warehouse and hence been chosen as our principle multi-functioning robot. As a parcel sorting request for a new order is generated, robot gets the location and plans its path from current location (P_1) to go to the given area (P_2) for collecting the parcel.

3.2. Pickup and co-packaging phase (stage 2)

As the robot is reaching the location (P_2) of the shelves, a mobile piece picking robot called Magazino [16] is placed in that area. Magazino, a German startup company, uses 3D cameras for identifying objects and implements a well-defined grasping technique for collecting objects from the shelves. Thus the object is picked and kept in a convenient location. The update is then provided to the principle robot through wireless sensor networks, so that the robot can detect object and pick it up. The next portion of the application involves co-packaging and customization for parcel according to individual needs of the customer. In comparison to more traditional/manual procedures, here the robot carries the parcel to the co-packaging center (P_3) where the well-known robot Baxter from Rethink Robotics [16] can complete the necessary steps of packaging. During these applications, a lot of information and analytical tasks are happening in parallel. That is why local or cloud based offloading may be required for more efficient performance of the system.

3.3. Package collection phase (stage 3)

After the parcel is customized and co-packaged, it is ready for delivery. At that point, the parcels are put on conveyer belt to be sent to the collection center. As updated information is provided to the principle robot (R_1), it moves to the collection point (P_4) to pick up the prepared parcel. While moving the robot needs to plan its path and communicate with the collection center to provide update to the main center. This creates the opportunities to pass on heavy computational tasks to nearby supporting robots for local computation or for assistance with offloading to the cloud.

3.4. Drop-off for delivery phase (stage 4)

As the robot reaches the collection point, it detects the prepared parcel. It uses its own technology to pick up the parcel. Then it updates the main center and additionally creates an order for the delivery robots from Starship technologies [29] to be prepared for incoming parcel. Then the robot delivers the objects in the drop-off point (P_5) to be collected by delivery robot.

As seen from the application details, it involves the principle robot to visit 5 locations (P_1-P_5) and perform computation-heavy tasks to complete the action. Also, due to the nature of the application, it is time constrained, which is why additional support from cloud and other available robots may improve the performance. Therefore other available robots in this cloud networked multi-robot application can help with the communication and local analytical support. For the purpose of simulation later in the paper, we have considered the warehouse environment (from Fig. 3) where the principle robot is Fetch and Freight robot R_1 and the two supporting robots are: Knightscope (R_3) from Knightscope Inc. [30] and Tug Robot (R_2) from Aethon [31]. Through joint collaboration of cloud and multi-robot resources, parcel distribution and sorting process in a smart warehouse is run autonomously to make parcel ready for delivery, starting from distribution to the eventual drop-off.

4. Problem formulation

4.1. System model

Our system modeling for task offloading considers motion and connectivity as part of its decision-making. Therefore, it integrates 4 critical factors in its problem formulation to find the optimal/near optimal decisions.

4.1.1. Task graph and cloud based task offloading

The 40 node task graph (Fig. 5a) in this paper is derived from the proposed parcel sorting and distribution application graph in Fig. 4. The 40 node task graph is defined by a direct acyclic graph (DAG) and presented as a tuple $D = (T, K)$. Here each node is considered as a task and known as $T = \{v_j,\ j = 1 : t\}$ and $t = |T|$. We also assume, K=$\{k_{i,j} = v_i, v_j\}$ and $k = |K|$, where K implies a set of edges and refers to the communication cost from node v_i to v_j. More precisely, the term t_i denotes a task i in the task graph where its execution time is dependent on the computation of v_i^{th} task with input data d_i. All the task nodes are indicated by Tasks t_1.....t_T. We assume that the nodes on the same level of the DAG

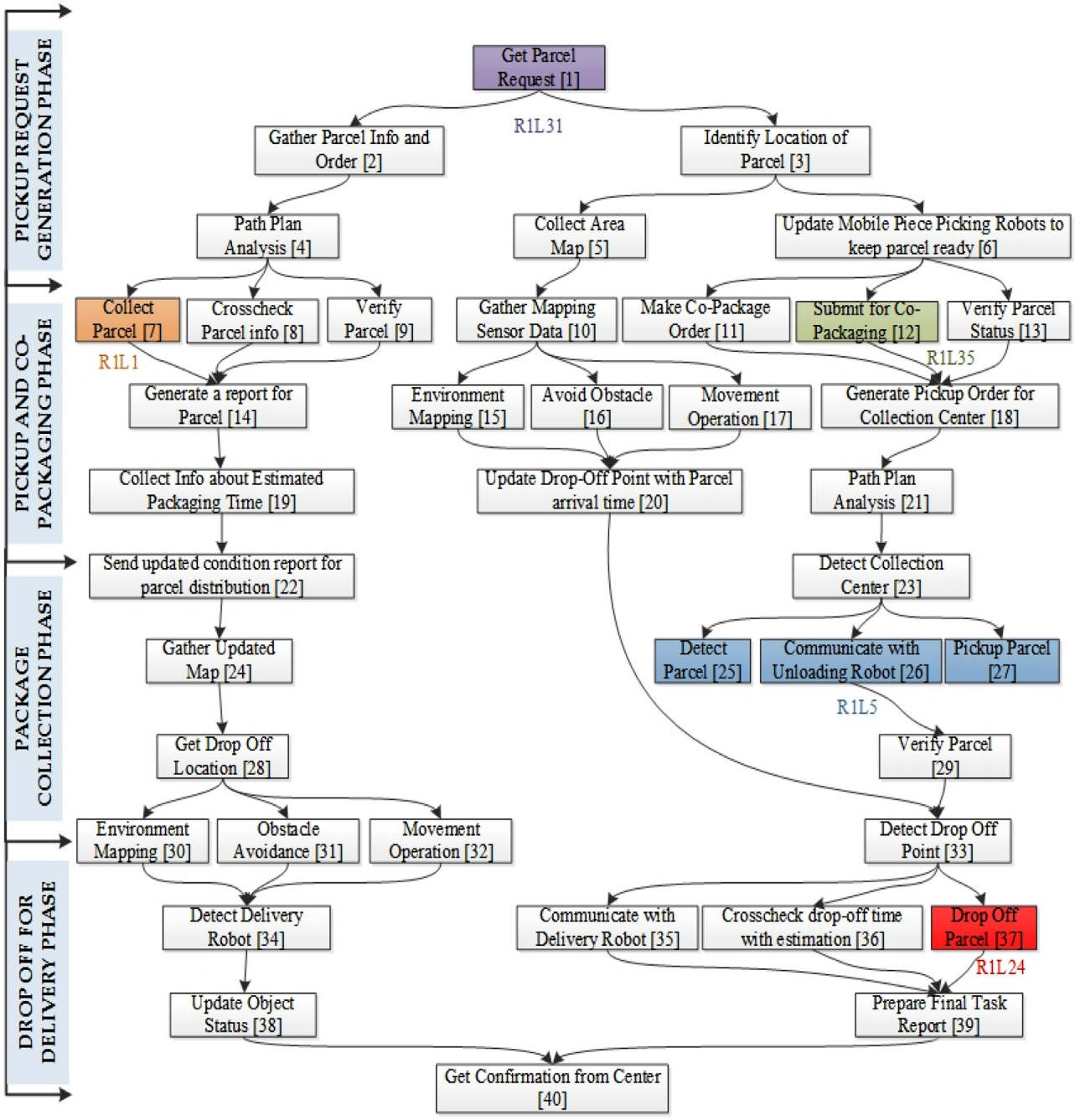


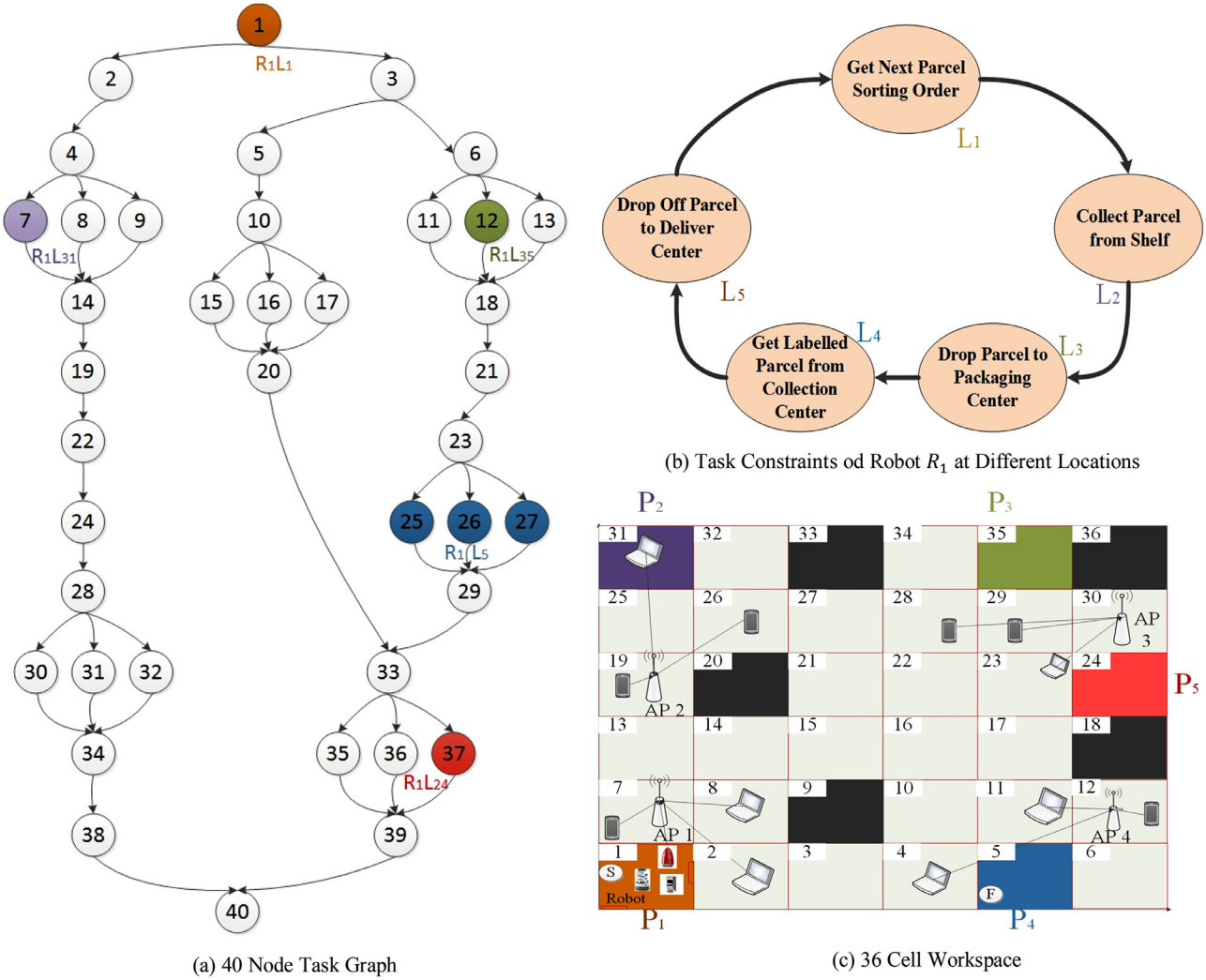
Fig. 4. Details of an automated parcel sorting and distribution application in smart warehouse.

(e.g., Tasks 4, 5 and 6) are independent of each other and limited by the "dependency of precedence". As a result, a task can start only after all its preceding tasks on the previous level have been completed. Finally, highlighted tasks ($P_1 - P_5$) represent the location constraints of certain tasks for robot R_1 and helps identify starting point of different stages of the operation. These tasks are considered as "unoffloadable" and constrained to the primary robot.

In implementing our offloading approach, our goal is to find the optimal set of decisions and perform suitable offloading in order to complete the task flow within the provided constraints. Since the cloud based offloading is dependent on the proper trade-off between robot and cloud VM, therefore the offloading decision in this context points to proper allocation between all the available heterogeneous resources. In this context, these available resources are represented by the robots (R_1, R_2, R_3) and cloud VM.

4.1.2. Robot–robot communication and local offloading

In addition to the cloud based offloading, robot–robot communication also creates gateway for offloading in cloud networked multi-robot systems. As each robot can communicate with the other available robots, therefore they create an ad-hoc cloud, which enables the robots to "locally offload" tasks to other available robotic agents. The set $\mathcal{R}$ indicate a group of robots that are part of the application, where $R_r \in \mathcal{R}$, $\forall \{i = 1 : n\}$. Here R_i denotes the selected robot and n indicates the total number of robots available. Based on that, the available robots can communicate with each other. Depending on the type of decision-making,



(a) 40 Node Task Graph

(b) Task Constraints od Robot R_1 at Different Locations

(c) 36 Cell Workspace

Fig. 5. Visual representation of taskflow, location constraints and workspace.

the robots in this context can locally offload tasks in two scenarios, (i) to offload to another robot for computation support, (ii) to offload to another robot that is used as a hub for further offloading to cloud VM. Either ways the available robots can support the principle robot to reduce workload and improve performance.

For local communication though, we present a popular communication model [32], where the communication parameters (energy/time) are considered based on the distance between the robots. For a threshold distance of l_0, if the distance (l') between two robots is less than l_0, then "free space" channel model is considered. Hence the communication energy and time is derived as:

$$E_{LO}(t_i) = \left(e_{base} + \varepsilon_{fs}.\ l'^2\right) \times d(t_i) :\ l' < l_0 \tag{1}$$

$$T_{LO}(t_i) = \left(e_{base} + \varepsilon_{fs}.\ l'^2\right) \times \frac{d(t_i)}{P_{LO}} :\ l' < l_0 \tag{2}$$

here e_{base} is the baseline energy consumption for operating the transmitter radio for local communication. As mentioned, for $l' < l_0$ the transmission energy consumption is assumed to be "free space" channel model and hence presented by $\varepsilon_{fs}.l'^2$. As for P_{LO}, it is known as the processing power of for robot (R_i) local offloading, whereas $d(t_i)$ is the amount/size of information being offloaded. However, when the distance (l') is greater than the threshold, then it considers the "multipath fading" channel model for communication and the transmission energy consumption is $\varepsilon_{mf}.l'^4$. Therefore, the overall energy and time calculation for local offloading is:

$$E_{LO}(t_i) = \left(e_{base} + \varepsilon_{mf}.\ l'^4\right) \times d(t_i) : l' \geq l_0 \tag{3}$$

$$T_{LO}(t_i) = \left(e_{base} + \varepsilon_{mf}.\ l'^4\right) \times \frac{d(t_i)}{P_{LO}(R_i)} : l' \geq l_0 \tag{4}$$

The communication parameters and the resulting calculation for local offloading is done based on the locations of the robots with respect to each other and therefore is subjected to the decision-making. More details about the performance parameters is provided later in the paper.

4.1.3. Workspace and path planning

Our application workspace in Fig. 5c is derived from Fig. 3, which depicts a warehouse environment. The warehouse workspace is represented by a grid, sized $m \times n$. Each cell in the grid points to a uniform cell location. As the robots can move through each cell, it eventually maps out a path plan for robot movement. In the context of our work, the grid-based model is chosen as our workspace because of its ease with calculation of distances, representation of obstacles and scalability with respect to condition changes.

As mentioned, each cell is represented by, L= (X, Y), where $X = \{x = 1 : m\}, Y = \{y = 1 : n\}$ and $\forall \mathcal{L} \in [1 : l]$. In the context of

our work, this results in a 36 cell warehouse workspace with additional characteristics listed as follows:

(i) Each cell $\mathcal{L}$ is a location the robot can choose for a given task. The whole workspace is characterized by orderly numbered grids, and the size of the grids determines how many cells there are.
(ii) Certain cells are considered as obstacles and therefore are off-limits to all the robots. These obstacles are directly adopted from warehouse environment (Fig. 3) and presented here as O_l.
(iii) The application also considers the task constraints of certain robot, meaning selected tasks are allocated to pre-defined robots as well as fixed locations (e.g., getting parcel from a fixed warehouse shelf). These tasks are presented as $T_{i,\ l} = i, l$; where task i is constrained to location l. The robots are aware of these constraints at the time of operation and hence move accordingly.
(iv) Finally, the application taskflow (Fig. 5a) is integrated with the workspace in the form of starting point P_1 and finish point P_4. This helps robot to identify the task constraints with respect to locations as well as get a clear idea of the task sequence.

One of the key parameters for offloading decision-making is the choice of location for each task. In this context the relative parameter that helps identify these decisions are the distance values between cells that calculates the overall distance the robot covers. As movement also results in added energy, hence lesser movement (unless necessary) is priority for the robot in order to save energy. Therefore the accurate distance between cells in this grid workspace is calculated using a modified A-star method [33]. This process uses the distance between current location and target. Then it moves to the cell with the smallest distance and evaluates cells by combining the distance cost to that cell $h(l)$ and the distance cost to get from that cell to target $g(l)$. For each successor cell we calculate the total cost, $f(l) = g(l) + h(l)$ and the cell with the smallest $f(l)$ is selected as successor. Using a thorough search of the complete workspace, shortest distance is found between starting and destination point. As mentioned, path planning influences offloading decisions. And it is important to get the accurate distance cost and path between any two points in the grid. Hence A-star method is used to calculate the distance $f(l)_{a,\ b}$ between any two points $a\ (x_1,\ y_1)$ & $b\ (x_2,\ y_2)$. Eventually It is used to gain the total distance coverage for robot movement D_{total} where the tasks are part of the movement set $m(t)$.

$$D_{total}(R_r) = \sum_{t_i\ \in m(t)} f(l)_{a,b} \tag{5}$$

Using these cost values, we find movement energy and time, which is then used during calculation of optimal offloading and movement decisions. More details of these calculations are presented during the formulation.

4.1.4. Access point selection and bandwidth estimation

One of the key benefits of cloud networked robotics is the on-demand movement capability of robot that can be utilized to gain better bandwidth. As discussed in [34], bandwidth plays an important role in task offloading and system performance. In this work, we further explore that issue by adding access point selection as a decision that impacts the resulting bandwidth. For AP selection, it can be broadly classified into two categories. First one being "online AP selection" where the choice of AP is made during the online phase, based on relatively stable features and selection criteria bearing physical or statistical meanings [35]. However, it tends to consume high energy usage as well as perform poorly for complex scenarios such as ours. In comparison, the second category of "offline AP selection" method is more suitable for our work. Here all APs are defined by their score function and availability. Based on those, the most suitable solution is selected as the choice of AP. For an unstructured problem set like ours, the offline approach suits the GA based scheme by reducing the level of complexity and compensating for the dynamic nature of the rest of the implementation of our algorithm.

In order to implement the latter approach, the workspace is setup with several APs in different locations for connecting to the cloud via the Internet. However, the robot have to share the WiFi network with other regular users. Depending on the path planning and its eventual location, the robot will have the option of connecting to one of multiple APs. As mentioned, here we implement the offline approach and provide a communication model to setup our workspace with variable APs and number of users. Depending on the choice of AP, the robot can get different bandwidth at different locations. Hence the choice of AP becomes a priority. As the WiFi network is shared with other users, each AP has certain users associated with them. Hence, the robot has to estimate the "fair-share bandwidth" of a given location for communication requirements. Given the size of application workspace, the complete area is covered by at least one AP and hence the robots are always within the coverage of Internet connectivity. Moreover, the number of users associated with each AP for the duration of the application are considered fixed, since the application time period is relatively small and hence less sensitive to dynamic changes.

Let's define an AP by α and total sets of AP as $\mathcal{A}$. The bit rate function is presented by $b\ (\cdot,\ \cdot)$ as $b : (\mathcal{L} \times \mathcal{A}) \rightarrow B$. Here $r(l,\ \alpha) \in B$ is bit rate at which each robot can individually transmit data from location $l \in \mathcal{L}$ with AP $\alpha \in \mathcal{A}$, where $\mathcal{A} = \{1 \ldots\ldots \alpha\}$. We consider B to be the set of bit rates available with the technology being used. Given that set, the individual bit rate r will depend on the location and the AP. The further the robot is from a given AP, the lower the value of r will be. This bit rate value is known to the robot for any given location during the task operation. Since each access point is shared by number of users, the bandwidth at any given location is shared among the number of users as well. Therefore the robot's (R_r) throughput/bandwidth β at location l can be estimated according to protocol IEEE 802.11 WLANs [36] as:

$$\beta(l, \alpha) = \left(b(l,\ \alpha)^{-1} + \sum_{u\ \in U_\alpha} {c_u}^{-1} \right)^{-1} \tag{6}$$

In (6), the term $U_\alpha(t)$ denote the set of users, excluding robot R_r, that are associated with AP α. And c_u signifies the cumulative bit rate for the set of users with respect to a given AP α. Since, each AP is shared by multiple users and all WiFi users use same packet size, therefore the resulting throughput $\beta(l, \alpha)$ is the "fair-share" bandwidth the robot receives at location l if it selects AP α. Throughput equation in (6) can be presented in more details to improve accuracy, but in the current form it already captures the essential features of the packet scheduling for 802.11 MAC in the simplest possible way. Therefore no further information is provided. As all the robots are aware of the total users set N_α and bit rates c_u for each AP α, therefore they can calculate the bandwidth at given locations (with respect to association with suitable APs). Based on that, AP selection is integrated with robot's path plan, local R–R offloading and R–C offloading, so that robots can gain access to better throughput for faster communication with cloud and improved system performance.

4.2. Cost functions

Two types of factors are considered for calculation of energy and latency cost functions, (i) fixed parameters: task input, robot and cloud VM processing power; (ii) variable parameters: local

Table 1
Notation.

Notation	Description
E_{total}	Overall robotic energy consumption for taskflow execution
T_{total}	Overall taskflow completion time/ latency
T_{limit}	Overall time limit for taskflow completion
E_{R_r}	Energy consumption of robot R_r
$E_{limit}(R_r)$	Energy limit of robot R_r
$D_{total}(R_r)$	Total distance covered by the robot
$N(t_i)$	Number of instructions for task t_i
$\beta(l, \alpha)$	Bandwidth (kbps) at location l with AP α
$v(t)$	A task set for WSN communication
$P_i(R_r)$	Robot R_r processing power for sending instruction to cloud
$P_u(R_r)$	Robot R_r processing power for uploading data to cloud
$P_r(R_r)$	Robot R_r processing power for on-board computation
$P_{cc}(R_r)$	Robot R_r processing power during cloud computation
$P_{mov}(R_r)$	Robot R_r processing power during robot movement
$P_d\ (R_r)$	Robot R_r processing power for WSN communication
$P_{LO}(R_r)$	Robot R_r processing power for local offloading
$m(t)$	A task set for robot movement
$v(R_r)$	Robot R_r movement velocity
$\mathcal{H}_r(R_r)$	Robot R_r transfer rate for WSN communication
S_{R_r}	Clock speed of robot R_r processor
S_c	Clock speed of virtual machine on cloud
$d(t_i)$	Uploaded data for the completion of task t_i
$d_d(t_i)$	Transferred data with WSN for task t_i
$BPI\ (R_r)$	Bits per instruction for robot R_r
$CPI\ (R_r)$	Average number of clock cycles per instruction for robot R_r

offload, cloud based offloading, bandwidth and movement. Table 1 lists notations for calculation of energy and latency cost functions, based on task assignments. Our goal is to identify optimal offloading decisions with collective consideration of motion and connectivity in a cloud networked multi-robot application that results in minimum robot energy.

4.2.1. Robotic energy calculation

$$E_{total} = \underbrace{\sum_{i=1}^{|T|} I_{t_i}.L_{t_i}\ .\ E_R(t_i, r, l)}_{On-board} + \underbrace{\sum_{i=1}^{|T|} \neg I_{t_i}.\ R_{\boldsymbol{t_i}}\ .\ L_{t_i}\ .\ \mathcal{A}_{t_i}\ .\ E_C(t_i, r,\ l,\ \alpha)}_{Cloud} \tag{7}$$

The total energy E_{total} consists of energy from all tasks partitioned into on-board, local offload and cloud allocation, as seen in (7). I_{t_i} denotes the offloading decisions and $\neg$ is the NOT operator, signifying the tasks that are offloaded to cloud. L_{t_i} is the unknown variable that indicates the location for each task, whereas $\mathcal{A}_{t_i}$ indicate the selection of AP for offloaded tasks. Finally, the term R_{t_i} means selection of robot for offloaded tasks, which signifies whether the task is offloaded to cloud from on-board of a given robot or the task is transferred to a nearby robot for further offloading to the cloud.

As mentioned, the total energy is summation of all the tasks that are divided into two parts. Tasks performed on-board are denoted by $E_R(t_i, r, l)$, which indicates they are dependent on the selection of task, robot (local offload) and location. As for the tasks that are taking place on cloud VM, they are identified by $E_C(t_i, r, l, \alpha)$, which means it is additionally dependent on the selection of AP as well for cloud based offloading. Eq. (7) can be further elaborated as seen in Eqs. (8) and (9):

$$\underbrace{E_R(t_i, r, l)}_{On-board} = \underbrace{E_{MOV}}_{Movement} + \underbrace{E_{WSN}}_{\substack{Wieless \\ Sensor}} + \underbrace{E_{LO}}_{\substack{Local \\ of\ fload}} + \underbrace{E_{RC}}_{\substack{Local \\ Computation}} \tag{8}$$

$$\underbrace{E_C(t_i, r, l,\ \alpha)}_{Cloud} = \underbrace{E_{MOV}}_{Movement} + \underbrace{E_{WSN}}_{\substack{Wieless \\ Sensor}} + \underbrace{E_U}_{\substack{Data \\ Upload}} + \underbrace{E_I}_{\substack{Sending \\ Instruction}} + \underbrace{E_{CC}}_{\substack{Cloud \\ Computation}} \tag{9}$$

Depending on the selections of I_{t_i}, $R_{\boldsymbol{t_i}}$, L_{t_i} and $\mathcal{A}_{t_i}$, we calculate robots energy and task completion time. For tasks taking place on robot, the parameters include the movement energy $E_{Mov}(t_i,\ l)$, data collection $E_{WSN}(t_i,\ l)$, computation energy $E_{RC}(t_i)$ and local offloading energy $E_{LO}(t_i)$. As seen from the components of the equation, it is dependent on location and the choice of robot.

$$E_{LO}(t_i) = \begin{cases} \left(e_{base} + \varepsilon_{fs}.\ l'^2\right) \times d(t_i) :\ l' < l_0 \\ \left(e_{base} + \varepsilon_{mf}.\ l'^4\right) \times d(t_i) : l' \geq l_0 \end{cases} \tag{10}$$

$$E_{RC}(t_i) = P_r(R_r) \times CPI(R_r) \times \frac{N(t_i)}{S_{R_r}} \tag{11}$$

Here $E_{LO}(t_i)$ is energy for locally offloaded tasks. Depending on robots distance among each other, it either follows "free channel" model (when $l' < l_0$) or "multipath fading" (when $l' > l_0$). As for on-board computation energy, it is denoted by E_{RC}, which is reliant on task and is calculated for the performing robot.

$$E_I(t_i) = P_i(R_r) \times BPI(R_r) \times \frac{N(t_i)}{\beta(l,\ \alpha)} \tag{12}$$

$$E_U(t_i) = P_u(R_r) \times \frac{d(t_i)}{\beta(l,\ \alpha)} \tag{13}$$

$$E_{CC}(t_i) =\ P_{cc}(R_r) \times CPI(R_r) \times \frac{N(t_i)}{S_c} \tag{14}$$

Even for tasks that take place on cloud, there is still energy consumed by robot. This is expressed by $E_C(t_i)$. It generally consists of the energy to upload data $E_U(t_i)$, energy to send instructions to cloud $E_I(t_i)$ and finally energy consumed for running background operations in robot processor during cloud computation $E_{cc}(t_i)$. All these parameters are either dependent on location, AP selections or choice of robot. In some cases, it is reliant on more than one parameters or all three, as seen from equations. Table 1 explains the rest of the parameters.

$$E_{Mov}(t_i,\ l) = \sum_{t_i\ \in m(t)} P_{mov}(R_r) \times \frac{l_{a,b}}{v\ (R_r)} \tag{15}$$

$$E_{WSN}(t_i) = \sum_{t_i\ \in v(t)} P_d(R_r) \times \frac{d_d(t_i)}{\mathcal{H}_r\ (R_r)} \tag{16}$$

Finally $E_{Mov}(t_i,\ l)$ indicates the movement energy to go to particular location for offloading, where a key parameter is robot velocity $v(R_r)$ that is different for each robot. Based on that, total movement energy/time for each robot is calculated in addition to total distance D_{total}. On the other hand, energy $E_{WSN}(t_i,\ l)$ is issued to describe energy consumption of data collection from WSN. Given the scale of the operation, WSN communication power $P_d(R_r)$ and data transfer rate $\mathcal{H}_r(R_r)$ for each robot is equal.

4.2.2. Time calculation

The time calculation is not additive as in the case of robot energy calculation. Since the whole taskflow is divided into a number of levels ($\aleph_{total}$), task completion time is calculated level-wise. For each level of tasks ($\aleph_t$), maximum time is calculated for tasks being completed locally and on-board. This is used to find the total time from that level and eventually for the whole taskflow.As for the rest, time calculation uses the exact same communication model for offloading, as mentioned in the previous section for energy consumption.

$$\mathcal{T}_{total} = \sum_{1}^{\aleph_{total}} max \left\{ \underbrace{\sum_{i=1}^{|\aleph_t|} I_{t_i}.L_{t_i} \ . \ \mathcal{T}_R(t_i, r, l)}_{On-board} + \underbrace{\sum_{i=1}^{|\aleph_t|} \neg I_{t_i}. \ \mathcal{R}_{\boldsymbol{t_i}} \ . \ L_{t_i} \ . \ \mathcal{A}_{t_i} \ . \ \mathcal{T}_C(t_i, r, \ l, \ \alpha)}_{Cloud} \right\} \tag{17}$$

$$\underbrace{\mathcal{T}_R(t_i, r, l)}_{Local} = \underbrace{\sum_{t_i \ \epsilon m(t)} \frac{l_{a,b}}{v(R_r)}}_{Movement} + \underbrace{\sum_{t_i \ \epsilon v(t)} \frac{d_d(t_i)}{T_r(R_r)}}_{WSN} + \underbrace{\begin{cases} \left(e_{base} + \varepsilon_{fs}. \ l'^2\right) \times \frac{d(t_i)}{P_{LO}(R_r)} : \ l' < l_0 \\ \left(e_{base} + \varepsilon_{mf}. \ l'^4\right) \times \frac{d(t_i)}{P_{LO}(R_r)} : l' \geq l_0 \end{cases}}_{Local \ \ offload} + \underbrace{CPI(R_i) \times \frac{N(t_i)}{S_{R_r}}}_{Local \ Computation} \tag{18}$$

$$\underbrace{\mathcal{T}_C(t_i, r, l, \ \alpha)}_{Cloud} = \underbrace{\sum_{t_i \ \epsilon m(t)} \frac{l_{a,b}}{v(R_r)}}_{Movement} + \underbrace{\sum_{t_i \ \epsilon v(t)} \frac{d_d(t_i)}{T_r(R_r)}}_{WSN} + \underbrace{\frac{d(t_i)}{\beta(l, \ \alpha)}}_{Data \ Upload} + \underbrace{BPI(R_r) \times \frac{N(t_i)}{\beta(l, \ \alpha)}}_{Sending \ \ Instruction} + \underbrace{CPI(R_r) \times \frac{N(t_i)}{S_c}}_{Cloud \ Computation} \tag{19}$$

The processor speed for cloud VM is S_c, much larger than processing speed S_{R_r} of each robot. Thus, cost functions for robot and cloud VM task completion time is used to calculate the overall results for $\mathcal{T}_{total}$.

4.3. Joint optimization problem

This paper addresses a four-fold problem. Based on the problem formulation for the proposed application, the objective is for the robot to find the optimal decisions for cloud based offloading, local offloading, path planning and AP selection altogether within the constraints imposed, thus providing task offloading for cloud networked multi-robot system with collective consideration for motion and connectivity in decision-making. Let the following variables indicate their respective decisions for each task.

(i) I_{t_i}= Offloading decision for each task. Here I_{t_i} (= r) indicates task is executed on robot $R_r \in \mathcal{R}$, $\forall \ \{r = 1 : n\}$. And I_{t_i} (= 0) is specifies that task t_i is offloaded to cloud VM. For our formulation, total number of robots n=3. So, possible task allocations decisions on-board of robot are R_1, R_2, R_3.
(ii) R_{t_i}= Selection of robot R_r for offloading a task t_i to the cloud, where $\mathcal{R}_{t_i} \epsilon \ I_{t_i} = 0$. This decision is for identifying which robot will offload the task to cloud.
(iii) L_{t_i}= Location for each task where the set consists of total l values ($L = 1 \ldots l$). For our formulation, $l = 36$.
(iv) $\mathcal{A}_{t_i}$= Selected AP for offloaded task, where AP set has total α values ($\mathcal{A} = 1 \ldots \alpha$). In our problem, $\alpha = 4$.

Based on the application scenario and problem formulation, the objective is to minimize overall energy (E_{total}) consumption in order to meet the time constraint ($\mathcal{T}_{Deadline}$) and individual energy constraint ($E_{R_{limit}}$) of each robot (E_{R_r}).

Find: $\{ I_{t_i} \}$, $\{R_{\boldsymbol{t_i}} \}$, $\{L_{\boldsymbol{t_i}} \}$, $\{\mathcal{A}_{\boldsymbol{t_i}} \}$, $\forall T = \{v_j, \ \ j = 1 : t\}$, and $t = |T|$ to minimize: E_{total}

s.t. : $\mathcal{T}_{total} \leq \mathcal{T}_{Deadline}$ and $E_{R_r} \leq E_{limit}(R_r)$

5. GA based multi-layer decision-making scheme

Genetic algorithms (GAs) derive their name from the fact that they are loosely based on models drawn from the area of population genetics. It is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection that represents an intelligent exploitation of random searches in order to determine optimal solutions. By exploiting intelligently, weak and unfit species are extinct while stronger genes are passed on to the next generation via process of reproduction [37]. Traditionally, GA is used to obtain optimized solutions from a number of candidate solutions. Although randomized, GAs are by no means random; instead they exploit historical information to direct the search into the region of better performance within the search space. Therefore GA is widely used in various field due to its global acceptability, high efficiency as well as its impressive stability [38].

For evolutionary robotics, GA is a machine learning approach that has been traditionally used to optimize the control policy of a robot [39]. It is implemented in applications to rapidly locate "satisficing" solutions when sufficient a *priori* knowledge is unavailable. Previously, path planning, cloud based offloading, multi-robot coordination or AP selection have been individually studied as examples of classical machine learning problems that require adaptive learning without significant domain knowledge for finding solutions. However, with rapid increase in technology, the complexity of handling dynamic and multifunctioning systems are exponentially increasing because of factors such as: dependencies among parameters, difficulty to map, interconnections etc. In order to avoid situations where certain aspects of development may become "intractable" due to constant progress and evolution in response to progressive conditions and demands, it is of utmost importance to prepare more comprehensive techniques for modeling systems in order to deal with dynamic changes and high level of complications. Therefore, more interdependent parameters are integrated to formulate and solve joint optimization problems where the algorithm is trained to be more rigid and driven towards an area of optimal result with high probability in an efficient way.

The heuristic approach of GA works efficiently for large scale multi-objective optimization problems, because it approximates brute force without enumerating all the elements, thereby bypasses performance issues specific to exhaustive search [40] which suits the NP-complete problem set in this work. More specifically, GA has several advantages over the evolutionary techniques. Primarily, meta-heuristic approaches such as GA searches parallel from a population of possible points. Therefore, it can avoid being trapped in local optimal solution unlike many traditional methods. Especially, when the problem size is big and unstructured, traditional algorithms are insufficient in terms of computation latency for finding solution. In comparison to them, GA performs much

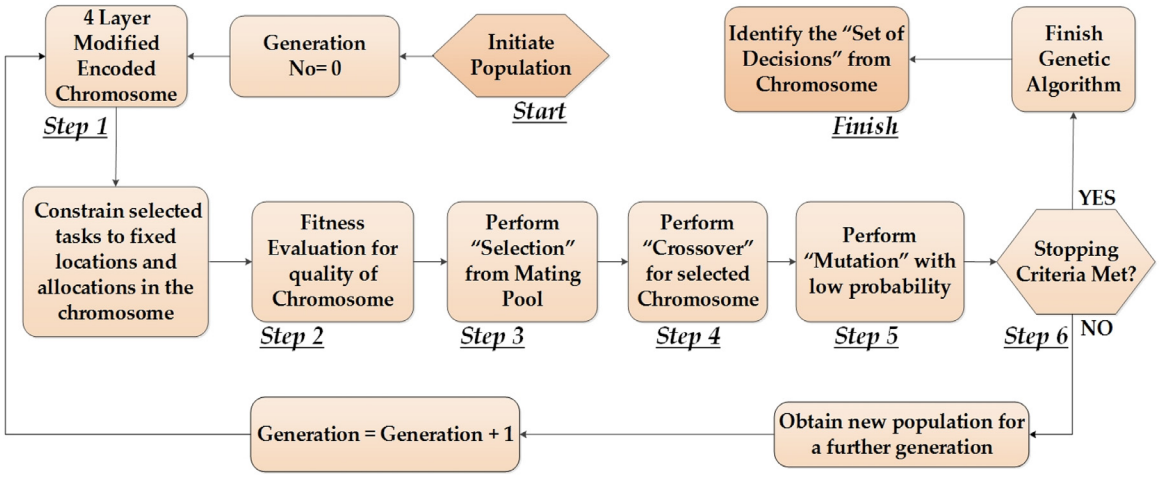


Fig. 6. Steps of GA based multi-layer decision-making scheme.

faster and generally trades-off in terms of solution accuracy (which is near-optimal). GA also works on the chromosome, which is the encoded version of the potential solution's parameter, rather than the parameter themselves. Therefore in case of problem sets with high number of parameters, GA has better chance of finding solution. Moreover, GA uses probabilistic selection rules, rather than deterministic selection. Hence, GA's chromosome gives users the power to inculcate the spirit of the problem into GA solution, thus giving meaningful direction to GA in its search for the optima, without having to worry about feasibility.

A clear distinction with other evolutionary algorithms such as particle swarm optimization (PSO) [41], bee optimization [42], ant colony optimization [43] is found with regards to joint optimization. GA is initially a discrete technique that is suitable for combinatorial problems like ours. In contrast, PSO/bee/ant algorithms tend to be continuous methods that may perform less efficiently for combinatorial problems. GA technique also has a higher variability of increasing probability to find better results, due to implementation of steps such as crossover and mutation (discussed later in the paper), which makes the population more diverse and thus more immune from getting stuck in a local optima. In theory the diversity also aids the algorithm to be faster in reaching close to the global optima since it can allow the algorithm to explore the solution space much faster. Another key advantage of GA can be found in cases when the objective function is not smooth, rather noisy or stochastic. In situations such as this, the derivative methods don't always hold up in performance. Since GA doesn't require any information about the structure of objective function in advance, it has high probability of dealing with uncertainty and producing better results. Finally, GA can effectively deal with dynamic scenarios since it is more adaptable and has fewer built-in assumptions that can potentially constrain the problem set from finding optimal/near optimal solutions. Given the overall benefits of GA and the scope of our application, a GA based scheme has been developed in this work.

The proposed GA-based multi-layer decision-making scheme identifies the following four key decisions: (a) which task to offload to cloud, (b) which robot to offload to cloud, (c) which location to complete the task, (d) which AP to select for offloading. The objective is to solve the joint optimization problem and identify these key decisions in order to minimize E_{total}. This joint optimization exploits the benefits from each of the individual elements and considers all four symbiotic parameters (cloud based offloading, robot–robot sharing, path planning and AP selection) jointly as part of the solution and hence simultaneously explores the search space for optimized and cost-efficient result, which would be reflective of the system performance. In order to implement genetic algorithm based scheme, we need to follow six regulatory steps (as shown in Fig. 6).

5.1. Novel four-layer chromosome encoding

The genetic algorithm (GA) scheme initiates by randomly generating a primary population (P), consisting of individuals whose genetic material represents sample points in the solution space. These individuals are a collective unit of novel 4-layer solution known as chromosomes. Here each layer points to a separate sets of decisions. These decisions are selected from the possible options in the search space and are presented in the following order: (i) offloading decisions, (ii) robot selection for offloaded tasks, (iii) location selection for each task, and *iv)* selected AP for the offloaded tasks. Therefore, $P = [I_{t_1} I_{t_2} .. I_{t_i} .. I_{t_T}]\ [R_{t_1} R_{t_2} .. R_{t_i} .. R_{t_T}]\ [L_{t_1} L_{t_2} .. L_{t_i} .. L_{t_T}]\ [\mathcal{A}_{t_1} \mathcal{A}_{t_2} .. \mathcal{A}_{t_i} .. \mathcal{A}_{t_T}]$. An example of the decision set can be given as: $I_1 = [1]\ [\boldsymbol{\times}]\ [13]\ [\boldsymbol{\times}]$, which suggest task 1 is completed on robot R_1 at cell 13 in the workspace. As a result, no offloading is required by any robot, hence no AP is selected. Another example of decision set is: $I_2 = [0]\ [3]\ [8]\ [2]$. It indicates that the task is completed on cloud VM (0), while it is offloaded to the cloud by robot R_3 from cell 8 in the workspace through AP 3.

In addition to that, tasks are further divided into two groups. The tasks that are constrained to any fixed location/allocation (constrained to robot), are considered as unoffloadable, whereas the rest of the tasks are offloadable. The unoffloadable tasks, their fixed allocations and locations are indicated in Fig. 4. In order to consider the unoffloadable tasks, the encoded chromosome in this section is further modified by fixing the constrained tasks to their fixed location (L_{t_i}) and allocation ($I_{t_i} = 1$ for robot R_1). In terms of GA scheme, this would save unnecessary latency during the later phase by not allowing this particular bits to be changed. Hence these solutions will remain fixed. As for practical reason, this provides real-world context where certain tasks such as data collection, parcel pickup, parcel delivery etc. tend to be in fixed location or fixed to certain robots. Therefore the constraints can be added in this scheme to compensate for those scenarios.

5.2. Fitness (parameter) calculation and evaluation

In this stage, each individual/solution in the population is evaluated by invoking the fitness function f to measure the

Input: Workspace, DAG, I_{t_i}, R_{t_i}, L_{t_i}, $\mathcal{A}_{t_i}$
Output: E_{total}, $\mathcal{T}_{total}$, D_{total}, E_{R_r}, D_{R_r}
1: Initialize E_{total}, $\mathcal{T}_{total}$, D_{total}, E_{R_r}, D_{R_r}
2: for each level $\aleph_{j\,mnv} \in \aleph_{total}$ **do** */* Calculate energy, time, distance*/*
3: **for** each task $t_i \in \aleph_j$ **do**
4: $\aleph_j(I_{t_i}) \in \{0, R_r\}, \forall\, r = \{1 \ldots\ldots n\}$ */*Find Task Allocation */*
$\aleph_j(R_{t_i}) \in R_r, \forall\, r = \{1 \ldots\ldots n\}$ */*Find Offloading Robot */*
$\aleph_j(L_{t_i}) \in L = \{1 \ldots\ldots l\}$ */*Find Task Location */*
$\aleph_j(\mathcal{A}_{t_i}) \in \mathcal{A} = \{1 \ldots\ldots \alpha\}$ */*Find Access Point */*
5: **if** $I_{t_i}(R_r)$ *(task on robot R_r)* **do**
6: **for** $R_r \in R, \forall\, r = \{1 \ldots\ldots n\}$ do
7: **if** $t_i \in m(t)$ *[movement task set]* **do**
8: $E_R(t_i, l) = E_{Mov}(t_i, l, r) + E_{WSN}(t_i, l, r) + E_{LO}(t_i, l, r) + E_{RC}(t_i, l, r)$
9: $E_{R_r}(t_i) = E_R(t_i, l, r)$
10: $E_{total} = E_{total} + E_R(t_i, l, r)$
11: $T_R(t_i) = T_{Mov}(t_i, l) + T_{WSN}(t_i, l, r) + T_{LO}(t_i, l, r) + T_{RC}(t_i, l, r)$
12: $D_{total} = D_{total}(R_r) + f(l)_{a,b}$ */*Total Distance */*
13: $D_{R_r} = D_{R_r} + D_{total}(R_r)$ */*Individual Robot Distance */*
14: **else do**
15: $E_R(t_i, l) = E_{WSN}(t_i, l, r) + E_{LO}(t_i, l, r) + E_{RC}(t_i, l, r)$
16: $E_{R_r}(t_i) = E_R(t_i, l, r)$ */*Calculate individual robot energy **
17: $E_{total} = E_{total} + E_R(t_i, l, r)$
18: $T_R(t_i) = T_{WSN}(t_i, l, r) + T_{LO}(t_i, l, r) + T_{RC}(t_i, l, r)$
19: **end if**
20: $E_{R_r} = E_{R_r} + E_{R_r}(t_i)$ */*Calculate individual robot energy */*
21: **end for**
22: **else** $\neg I_{t_i}(0)$ *(task on cloud)* **do**
23: **if** $t_i \in m(t)$ *[movement task set]* **do**
24: $E_C(t_i, l, r, \alpha) = E_{Mov}(t_i, l, r, \alpha) + E_{WSN}(t_i, l, r, \alpha) + E_U(t_i, l, r, \alpha) + E_I(t_i, l, r, \alpha) + E_{cc}(t_i, l, r, \alpha)$
25: $E_{R_r}(t_i) = E_C(t_i, l, r, \alpha)$ */*Calculate individual robot energy **
26: $E_{total} = E_{total} + E_C(t_i, l, \alpha)$
27: $T_C(t_i, l, r, \alpha) = T_{Mov}(t_i, l, r, \alpha) + T_{WSN}(t_i, l, r, \alpha) + T_U(t_i, l, r, \alpha) + T_I(t_i, l, r, \alpha) + T_{cc}(t_i, l, r, \alpha)$
28: $D_{total} = D_{total}(R_r) + f(l)_{a,b}$ */*Total Distance */*
29: $D_{R_r} = D_{R_r} + D_{total}(R_r)$ */*Individual Robot Distance */*
else do
30: $E(t_i) = E_{WSN}(t_i, l, r, \alpha) + E_U(t_i, l, r, \alpha) + E_I(t_i, l, r, \alpha) + E_{cc}(t_i, l, r, \alpha)$
31: $E_{R_r}(t_i) = E_C(t_i, l, r, \alpha)$
32: $E_{total} = E_{total} + E_C(t_i, l, r, \alpha)$
33: $T_C(t_i, l, r, \alpha) = T_{WSN}(t_i, l, \alpha) + T_U(t_i, l, r, \alpha) + T_I(t_i, l, r, \alpha) + T_{cc}(t_i, l, r, \alpha)$
34: **end if**
35: $E_{R_r} = E_{R_r} + E_{R_r}(t_i)$ */*Calculate individual robot energy */*
36: **end if**
37: **end for**
38: **if** $T_R(t_i, l, r) > T_C(t_i, l, r, \alpha)$ **do**
39: $\mathcal{T}_{total} = \mathcal{T}_{total} + T_R(t_i, l)$
40: **else do**
41: $\mathcal{T}_{total} = \mathcal{T}_{total} + T_C(t_i, l, \alpha)$
42: end for
43: Calculate E_{total}, $\mathcal{T}_{total}$, D_{total}, E_{R_r}, D_{R_r}

Fig. 7. Pseudo-code for robotic energy, time and distance calculation.

quality of the solution in the search space. For the given problem, the fitness is considered as $f = E_{total}$. Since the objective is to minimize energy, hence the term "lower energy" indicates "higher fitness" in this context. Similarly "best fitness" also designates the solution that results in the "lowest value". In order to get fitness value, the calculation of critical performance parameters i.e., energy (E_{total}), time/delay (T_{total}), distance (D_{total}), individual energy (E_{R_r}) and individual robot distance coverage (D_{R_r}) are done using Eqs. (1)–(19). As taskflow is divided into several levels, a breadth-fast search is performed to identify their task dependencies and to divide them into groups for level-wise calculation. A detailed pseudo-code in Fig. 7 explains the step by step calculation for fitness score and other performance parameters.

5.2.1. Calculation of E_{total} (fitness) & E_{R_r}

From the proposed allocation in the encoded chromosome, level-wise calculation is done to get the values of robot energy for tasks offloaded to cloud $E_C(t_i, l)$ and tasks completed on robot $E_R(t_i, l)$. Since energy values are additive, hence the resulting energy E_{total} from the corresponding level is collected and added to the overall values. Thus, energy from all the levels adds up to ultimately get the final updated value of total energy E_{total}.

Another important parameter is the individual energy consumption of each robot E_{R_r}. Based on the selection of robot for local task completion or local offloading, energy is consumed for tasks as seen in the calculation. For each consumption, energy cost corresponding to a task is added to the selected robot R_r. In this way, for each task, energy values may be added to corresponding

robots (based on task allocation) and total energy consumption by each robot E_{R_r} can be calculated.

5.2.2. Calculation of $\mathcal{T}_{\text{total}}$

Since the tasks on the same level happen in parallel, total time for tasks is not directly additive. Therefore, slight modification is required to compensate for parallel tasks. Tasks from each level are further divided into two types based on allocations (0 or R_r) and put them into two different lists. Total time for tasks on cloud and on robots are calculated separately. A simple comparison is then performed between the cumulative time values (for each level) to see which takes longer. Here the higher value between the two is considered as the actual latency/time cost from that DAG level. Similar to the previous process, values from each level is then added to overall results to get the total task completion time $\mathcal{T}_{total}$.

5.2.3. Calculation of D_{total} & $\mathrm{D_{R_r}}$

Distance is calculated according to the fourth layer of the chromosome, which is location decisions (L_{t_i}). For every task, that is part of set *m(t)*, distance cost between the corresponding points for each robot is calculated (via A-star method). Based on the allocation (robot or cloud), the distance value is calculated sequentially for each level and distance values are added either for either cloud or robot based allocation $D_{total}(R_r)$. These values are then added to overall values in order to ultimately gain the total distance covered by the robots D_{total}. In addition to that, for each distance cost, the values are also added to the corresponding individual robots (R_r). In this way, individual distance coverage D_{R_r} for each robot is also calculated.

5.3. Selection phase

After the fitness calculation stage, the mating pool is filled in iteratively from the current generation. From then on, two chromosomes are randomly selected in each pass and the individual with higher fitness (i.e. low energy) is finalized to fill the mating pool. The process is repeated until the mating pool is completely filled in. We also adopted elitism by keeping the historical best solution in the mating pool. Thus the next generation is produced by selecting individuals with higher fitness via the selection probabilities to produce offspring via genetic operators.

In addition to that, infeasible solutions are also dealt with during this stage. Here these solutions are given very low fitness score, which results in really high energy values. Therefore these solutions can never be picked up for the mating pool (since their fitness will be always inferior to feasible answers) and the solution will remain accurate.

5.4. Crossover phase

The strategy employed by crossover is to construct new individuals from existing high-performance individuals by recombining subcomponents. Here two selected chromosomes from the previous phase "reproduce" in crossover section and produce "offsprings". For this phase, the "uniform crossover" process is considered as it uses a fixed mixing ratio between two parents. This process allows parent chromosomes to contribute in the gene level rather than the segment level. During this stage, individual bits in the string are compared between their two parents. And, all the bits are swapped with a fixed probability.

As mentioned, infeasible solutions are removed by giving them low fitness scores in order to avoid degradation of GA performance. However, if any such solution is still produced at crossover, it would be eliminated at fitness calculation or mating pool generation stage of the following pass of GA.

5.5. Mutation phase

At the end of selection and crossover, there is now a new population full of possible solutions (decision set). However, the chromosomes may become too similar to each other in some cases. At this point, the mutation operator updates these individuals by independently modifying one or more of the gene values of an existing individual. More specifically, a portion of new individuals have some of their bits flipped with low probability (0.5) by the operator. This is done intentionally, so as to ensure proper diversity and possibility of finding global optimum in search-space. Further justification for this choice is provided later in the manuscript.

Similar to "crossover" section, the infeasible results are given low fitness scores so as to avoid the quality of results becoming poorer. As for the constrained elements (fixed allocation and location of task) in the chromosome, they are also dealt with in this section. For that purpose, the chromosome is further modified at the end of this stage to compensate for the constrained tasks by forcefully changing the elements to their fixed allocation and location. After that, we obtain a new population of individuals and same process is continued.

5.6. Self-stopping criteria

A self-stopping criteria is embedded so that the process doesn't evoke unnecessary latency or processing power. Since our GA scheme provides the "lowest energy" (best fitness score) after each generation, it will only stop when there is no change in best fitness score for a prefixed (determined by user) number of generations. At that point, GA is terminated immediately and result is considered as near-optimal. Further details are later provided to explain our reasoning behind the choice.

6. Simulation results and analysis

We ran extensive simulations and analyze the different aspects of our GA scheme for multi-robot with cloud (GAMRC) approach, which are: decision-making, fitness score, offloading, AP usage, robot's usage and path planning. Based on that, we assess the quality of the algorithm and its effectiveness in the context of the application. We further determine the performance of our scheme by comparing our findings with GA scheme for a single cloud-aided robot (GASRC) from Rahman *et.al.*, in [24] and [25] where mobility-driven and communication-aware task offloading was performed in similar applications for a 30 node and 25 node taskflow. For a single robot application, all tasks are allocated between the robot and cloud VM. The mobility indicates the location choice for each task, whereas AP selection points to the gateway towards cloud communication for allocated tasks. The comparison between single and multi-robot cloud based approach in this section would help identify/verify the benefits of the multi-robot aspect/dimension in task offloading. More precisely, this would point out how additional robots may help out in local computation sharing of tasks as well as in offloading to cloud.

In addition to that, we also compare the results with a GA scheme for multi-robot (GAMRB) on-board approach where only R–R communication has been used for task completion. This approach doesn't consider cloud infrastructure as a possible source of allocation. Through this comparison of results, the impact of cloud in such applications may be recognized. Both benchmark approaches (GASRC and GAMRB) are calculated via genetic algorithm based method as well. Hence the results from these methods are near-optimal. As for authentication of these benchmarks, these results have been previously evaluated properly for different DAGs via comparison with exhaustive search [20], All-on-Robot (AoR) approach [24], greedy algorithm [34] as well as a single robot

Table 2
Simulation parameter setup.

Parameter	Value (Min: E_{total})				
Task nodes	40				
Deadline ($\mathcal{T}_{Deadline}$)	Time (150 s)				
Population	500				
Stopping	1500 generations without change in fitness score				
Obstacle cells	Cell 9,18,20,33,36				
AP & users	4 APs and each AP is associated with 3 users				
Performance parameters of each robot	**Robot No.**	R_1	R_2	R_3	
	Processor	Core i5-4460	Core i5-7600	Core i5-7600K	
	S_{R_r}	3.2 GHz	3.5 GHz	3.8 GHz	
	$P_r(R_r)$	55 W	40 W	25 W	
	$P_u(R_r)$	80 W	60 W	40 W	
	$P_{cc}(R_r)$	20 W	15 W	10 W	
	$P_{LO}(R_r)$	11 W	12 W	13 W	
	$P_m(R_r)$	50 W	35 W	20 W	
	$CPI(R_r)$	10	8	6	
	$BPI(R_r)$	4	3	2	
Allocation and location constraint	**Task**	**Zone**	**Task**	**Zone**	**Allocation**
	1	1	25,26	5	Robot R_1
	7	31	27	5	
	12	35	37	24	

offloading method with fixed movement and bandwidth [20]. Therefore, the comparison of our GAMRC scheme in this paper with such credible reference methods would help validate our findings.

6.1. Simulation setup and parameter selection

In this section, we provide a detailed explanation of two types of parameters that are related to this problem. One is application-oriented and the other is GA-oriented.

(i) At first, Table 2 describes the application parameters for simulation purposes. One such key parameters is workspace model and the setup design. The 36 cell workspace considered in this work is motivated from industrial warehouse presented in Fig. 3. The obstacle cells are marked black (Fig. 5c) and points to cells that are off limits for movement selection. The design of the workspace and the corresponding obstacles resemble the proposed application in Fig. 3. The details of the obstacles are presented in Table 2. Another important parameter is the taskflow, that is motivated from the application task graph in Fig. 4. The 40 node taskflow defines tasks that need to completed in the constrained scenario, where selected tasks ($P_1 - P_5$) must be completed by robot R_1. Hence the location constraints for these tasks are also presented in the table. The constraints here complement the requirements of our proposed application (warehouse management). For the multi-robot approach, total 3 robots are considered and the performance parameters (e.g., power rating, constraints, processor details etc.) for each robot is presented in Table 2. The detailed description and objective of each robot is thoroughly presented to highlight their abilities that match the type of tasks required to be completed by them. Furthermore, details of their hardware components are also presented to explain the collaboration of these heterogeneous components in the context of this application. Another important relationship is between the primary robot processor and cloud VM processor. In comparison to the robot processor, the cloud VM is considered to be minimum M times faster than the fastest robot processor ($S_c = M \times S_{R_r}$). This is motivated from the knowledge that cloud VMs are generally much faster than the on-board robots. Finally, for communication modeling, an infrastructure using the standardized design of IEEE 802.11 WLANs is presented with a maximum bit rate of 54 Mbit/s and 8 available stream bit rate (6,9,12,18,24,36,48,54 Mbit/s). Based on these selections, the whole workspace is designed with 4 APs where each AP has 3 users. The reason to choose multiple APs and users is to provide context of a real-world application, where multiple users may use the network and hence bandwidth may need to be shared. Following that model, each of the locations in the workspace correspond to bandwidth value for a selected robot based on AP selection, as seen in Eq. (6). All this information is available to robots at the time of operation and are taken into consideration for parameter (i.e., energy, time, distance) calculation and overall decision-making.

(ii) Another important part of the simulation is the selection of parameters for the development of GA based scheme. There are three key parameters considered that significantly influence the performance of our genetic algorithm method. They are: population size, mutation rate and no. of generation selected as the self-stopping criteria. In order to highlight the reasoning for our selection, Fig. 8 provides the performance of GA (value of minimum energy) with respect to these parameters, individually.

As seen in Fig. 8a, the minimum energy drops with the increase in population size, however it plateaus after a certain point (population size 400–700). Therefore, all these values within the range is a reasonable choice as the population size. As we have selected population size of 500, it results in minimum energy of 2836.50 J, which is close to the point where the saturation occurs. Hence population size is chosen as 500 (as indicated by Δ in the figure).

The same thing can be seen in case of varying mutation rate (Fig. 8b) where we present the performance of GA (Min. Energy) with mutation rate being changed from 0.1 (too low) to 0.7 (too high). Similar to previous figure, the performance suggests that GA results become saturated after a certain point, even with variance in mutation rate. Since the results don't vary too much for the latter values of mutation rate, we have considered the traditionally used value of 0.5 as the mutation rate (as indicated by Δ in the figure).

Finally, we have run simulation by varying the generation no. (as stopping criteria) from 300 to 2100. As seen in the figure, the smaller generation numbers provide a higher value of minimum energy (poor results). At around 1500 the minimum energy drops significantly and then saturates. Hence, we have chosen our stopping criteria as 1500 generation for our problem (Table 2), which

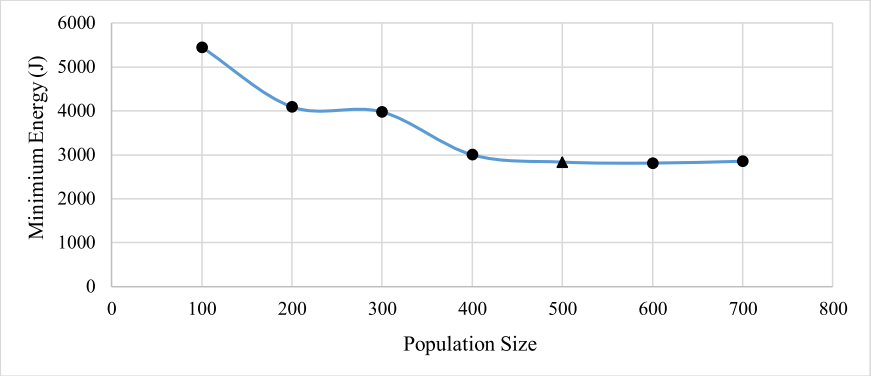


Figure 8a. Performance of GA (Min. Energy) with Respect to Varying Population Size

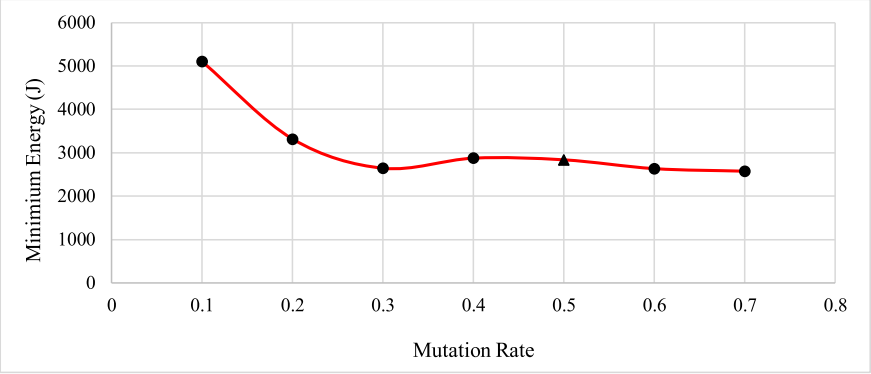


Figure 8b. Performance of GA (Min. Energy) with Respect to Varying Mutation Rate

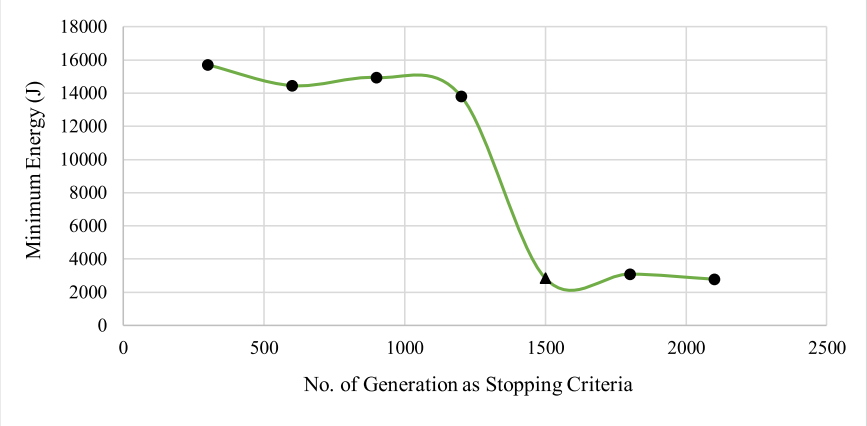


Figure 8c. Performance of GA (Min. Energy) with Respect to Varying Generation No. as Stopping Criteria

Fig. 8. (a). Performance of GA (Min. Energy) with respect to varying population size. (b). Performance of GA (Min. Energy) with respect to varying mutation rate. (c). Performance of GA (Min. Energy) with respect to varying generation no. as stopping criteria.

means GA is self-maintained and stops running when no changes in results can be found for 1500 generations.

6.2. Simulation results

6.2.1. Performance analysis of GA scheme

The results from GA based scheme is presented in Table 3. As seen in the table, the four-layers present the 4 sets of decisions. Layer-1 presents the task allocation decisions where '1′, '2′, '3′ indicates the robot assigned to the task, whereas '0′ indicates the task has been offloaded to cloud. Layer-2 points to the selection of robot for offloading the tasks to the cloud. Therefore this is a subset of the allocation decisions, where tasks have only been offloaded (I_{t_i} = '0′). Tasks that are completed on robot cannot be offloaded and hence are defined by in layer 2 as '✗'. Layer-3 points to the location decision. For tasks completing on robot, the location points to the cell in the workspace where the corresponding robot has completed the task. For instance, [3 ✗ 27 ✗] for task 4 means that the task is completed by robot R_3 at cell 27. In contrast, for tasks that are offloaded, location points to the cells the task was offloaded from by the corresponding robot. For example, results for task 30 is presented as: [0 2 17 4]. It means task 30 is offloaded to cloud VM by robot $R_2$from cell 17. Finally, layer-4 of results indicate AP selection, which along with location,

Table 3
Analysis of GA based decision-making scheme.

Task No.	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*
Allocation	1	0	0	3	0	2	1	0	3	3	3	1	0	3	0	3	3	3	0	2
Offloading Robot	×	3	3	×	1	×	×	3	×	×	×	×	3	×	3	×	×	×	3	×
Location	1	7	21	27	27	31	31	27	30	30	35	35	21	16	8	15	21	21	21	21
Access Point	×	1	4	×	3	×	×	3	×	×	×	×	2	×	1	×	×	×	3	×

Task No.	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*
Allocation	2	0	3	0	1	1	1	0	3	0	3	0	0	3	3	3	1	3	0	3
Offloading Robot	×	3	×	3	×	×	×	3	×	2	×	2	3	×	×	×	×	×	3	×
Location	21	21	10	4	5	5	5	11	17	17	11	4	4	16	17	24	24	17	15	15
Access Point	×	2	×	1	×	×	×	1	×	4	×	4	4	×	×	×	×	×	2	×

Table 4
Performance of each robot for GAMRC approach.

	Robot R_1	Robot R_2	Robot R_3
Energy constraint (limit)	5000 J	1000 J	3000 J
Energy consumption	936.18 J	410.78 J	1489.53 J
Path planning results	1–*27*–31–35–5–24	1–31–21–*17*–*4*	1–*7*–*21*–*27*–30–35–*21*–16–*8*–15–*21*–10–*4*–*11*–*4*–16–17–24–17–*15*
Total distance covered	204.84 m	132.42 m	335.54 m

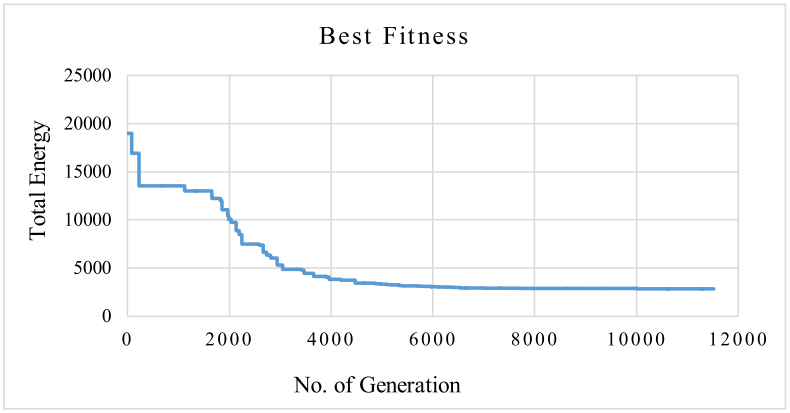
corresponds to available bandwidth utilized by robot. Similar to layer-2, this is a subset of allocation decisions where tasks have been offloaded. For tasks that are completed on robot, they do not require any association with AP, hence denoted by '✗' in the results.

6.2.1.1. Analysis of fitness score and offloading performance. Fig. 9 shows performance of our proposed GA scheme (GAMRC) where key indicator is fitness. According to Fig. 9a, average fitness graph shows a declining trend, which signifies that our GA scheme is working properly. Since the objective is to find the minimum energy, the falling graph suggests that overtime results go from initial findings towards lower and more precise values of fitness/ energy. Similar findings can be seen for best fitness graph (Fig. 9b). As mentioned before, in the context of the application, the term "best fitness" indicates lowest value of robot energy. From the initial generation, GA results are evaluated based on their fitness scores. Over the course of the complete algorithm run, each time a lower fitness is found, it replaces the previous value and becomes the new best fitness score. When the best fitness score doesn't change for a pre-defined number of generation, then the result is assumed to be near-optimal. Hence the best fitness score finishes as a constant line in the graph. Fig. 8c is a representation of the allocation histogram that indicates the well-distributed nature of the allocation results. Even though, around 14 tasks were completed by the primary robot R_1, however robot R_2 and R_3provided aid through local offloading along with the support from cloud infrastructure. More specifically, R_3 was the biggest contributor in terms of task allocation. But, it is reasonable since $R_3$is the most powerful robot in this scenario, therefore takes the majority of the workload.

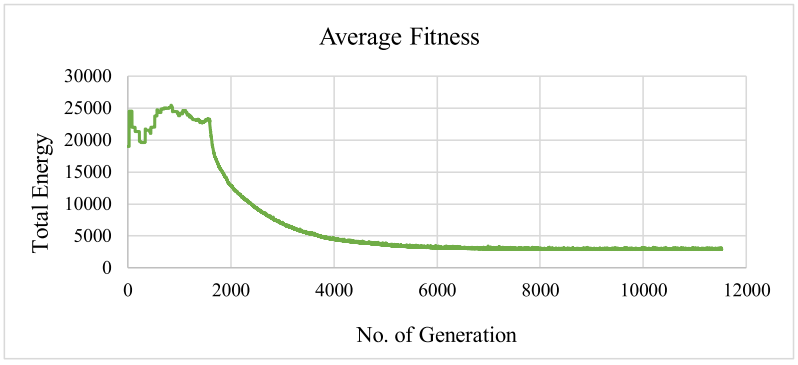
6.2.1.2. Analysis of individual robot performance. We present the performance of individual robots in Table 4. As seen, each robot has an energy limit, which also influences the offloading decisions. The near-optimal decision-set requires each robot to be within their energy constraint. As seen from the results, energy consumption of robot R_1(936.18 J), R_2(410.78 J) and R_3(1489.53 J) are all within their respective energy constraints of 5000 J, 1000 J and 3000 J. This proves that the algorithm can identify the near-optimal decision-set while meeting energy constraint condition of individual robot. It also highlights that R_3carries most of the workload, which results in highest energy consumption among them, whereas R_2 is the least utilized robot. All these results are clear indication that robot R_1offloaded tasks to the cloud VM as well to local robots (R_2 and R_3) for computation support as well as easier communication with cloud.

6.2.1.3. Analysis of path planning performance. As for the path planning results from Table 4, it shows the cells each robot has visited as well as the order in which it has visited the cells. The underlined cells in the results mean that offloading took place in these cells/locations in workspace. These results help prepare the path plan for individual robot. Additionally it also highlights the total distance covered by each robot. According to the results, robot R_3 covers more area through movement and hence has a higher distance coverage (335.54 m) than R_1(204.84 m) as well as R_2 (132.42 m). The underlined cells in the path planning results point to the cells that the robots have visited for the purpose of task offloading. It is further evident from Fig. 10, which shows the path plan of each robot for our proposed GAMRC scheme. In these graphs, the continuous directed lines helps trace the path of each robot. The dotted lines mean that the robot communicated with the cloud for offloading during its stay in these cells. Finally the non-directed continuous lines means an intermediate path while it was moving towards a selected cell.

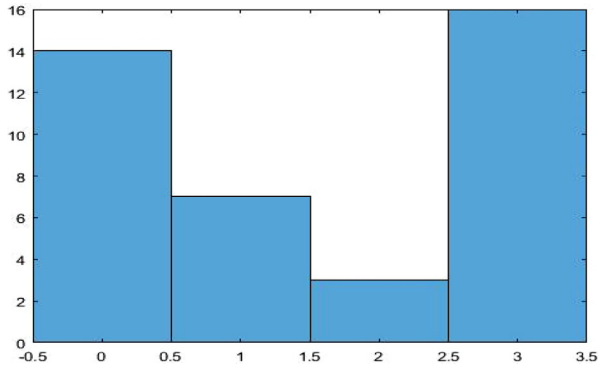
The overall path planning results highlights a clear view of each of the robot's movement for the duration of the application. These results also help relate to the access point (AP) selection



(a) Best Fitness (i.e., Lowest Energy)



(b) Average Fitness



(c) Task Allocation Histogram

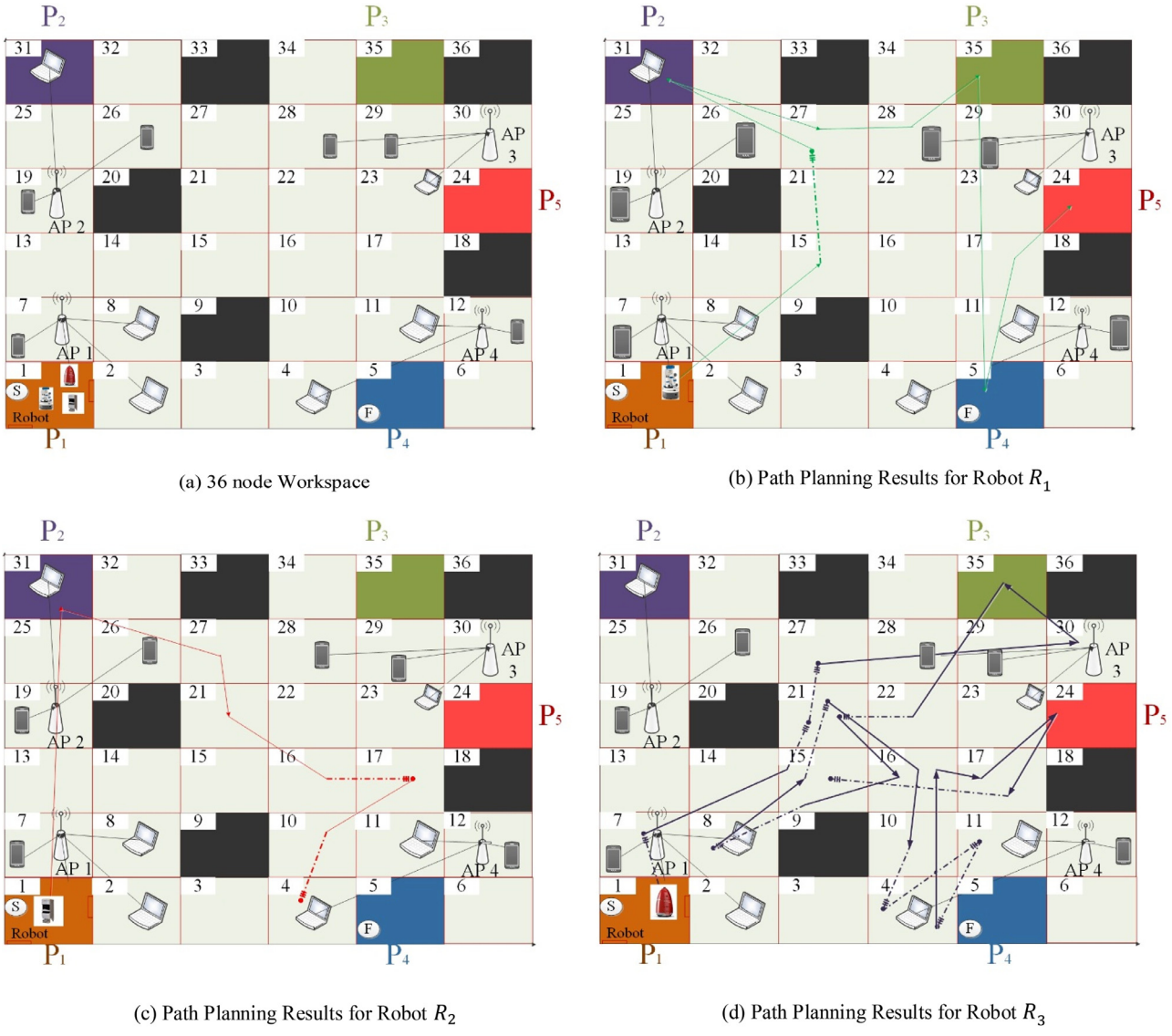
Fig. 9. Fitness performance and offloading decisions of GA scheme.

and therefore the overall task offloading performance, since movement has impact on the available bandwidth each robot gains at different locations for cloud-based communication.

6.2.1.4. Analysis of AP usage. Depending on the task allocation and path plan results, the robots are allotted to different locations for task completion as well as for offloading during the course of the operation. Based on those decisions, each robot has different availability of bandwidth, which also influences the AP selection decisions. Fig. 11 depicts the AP selection for each task. The results show that total 4 tasks are offloaded using AP 4 and total 3 tasks are offloaded using AP 3, while 3 tasks are also offloaded via selection of AP 2. Finally total 4 tasks are offloaded via selection of AP 1. This attains to total 14 usage of APs, meaning total 14 tasks have been offloaded to the cloud. We can further realize from Table 3 that total 11 of these tasks have been offloaded by robot R_3, which makes it the dominant robot in the case of offloading to cloud. In comparison, the performance of R_1 and R_2 are mere as the offload only 1 and 2 tasks respectfully. Therefore the findings from this section suggest that the ability of robot R_3 to cover more distance resulted in better access to available bandwidth (through AP selection), which resulted in robot R_1 getting aid from R_3 for majority of task offloading during application, which ultimately increased the potential of improving system performance.

6.2.2. Performance evaluation of GA scheme

In Table 5, we evaluate performance of our GA scheme for multi-robot and cloud (GAMRC) with results from GA scheme for single robot and cloud (GASRC) and GA scheme for multi-robot onboard (GAMRB). The results clearly highlight that GAMRC entails much lower energy (2836.50 J) than GASRC (4778.80 J) and GAMRB (7057.19 J). In terms of time/delay, GAMRB doesn't finish within the time constraint. As GASRC, even though the tasks are finished



(a) 36 node Workspace

(b) Path Planning Results for Robot R_1

(c) Path Planning Results for Robot R_2

(d) Path Planning Results for Robot R_3

Fig. 10. Path planning performance of each robot for GAMRC scheme.

Table 5
Task offloading performance comparison (Min:E_{total}).

Result parameters	GA (multi-robot with cloud), GAMRC	GA (single robot with cloud), GASRC	GA (Multi-robot on-board), GAMRB
Generation No	11,524	8399	N/A
Offloaded Task (cloud)	14	21	0
Minimal Energy	2836.50 J	4778.80 J	7057.19 J
Total Time	80.31 s	109.45 s	413.16 s
Total Distance	672.8 m	297.98 m	1161.96 m

within the delay constraint, it still is higher than the GAMRC process, which is the fastest to complete the applications. The reason for better performance by GASRC and GAMRC can be identified via deeper analysis.

6.2.2.1. Offloading performance comparison. According to Table 5, GASRC and GAMRC entail lower energy because of their ability to offload tasks to the cloud. Here the benefit of GASRC is evident, since the single robot R_1 allows for most offloading of tasks (21) to the cloud. Despite the fact that 14 tasks are offloaded to cloud for GAMRC, the added trait of local R–R communication means that robot R_1 can utilize nearby robots R_2 and R_3 for local completion of tasks or help with offloading of tasks to the cloud. Thus, results in better access to resources (cloud VM and local robot), more execution of parallel tasks and faster completion of tasks for cloud networked multi-robot systems.

6.2.2.2. Path planning comparison. According to Table 5, the total distance covered by GAMRC (672.8 m) is higher than GASRC (297.98 m). Even though, more movement causes higher energy consumption, but for GAMRC there are three active robots that are moving in tandem to get access of better bandwidth for easier offloading. Moreover, the robots are also utilizing their local communication to cover more area effectively. Hence the primary robot R_1 can get assistance from robot R_2 and robot R_3 for completing local tasks or to help with communication for cloud-based offloading.

Table 6
Comparison of robots performance among the three methods (Min:E_{total}).

	GA (Multi-robot with cloud), GAMRC			GA (Single robot with cloud),GASRC	GA (Multi-robot on-board), GAMRB		
Robots	R_1	R_2	R_3	R_1	R_1	R_2	R_3
Energy constraint	5000 J	1000 J	3000 J	5000 J	5000 J	1000 J	3000 J
Energy consumption	936.18 J	410.78 J	1489.53 J	4778.80 J	2532.2 J	1610.78 J	2914.21 J

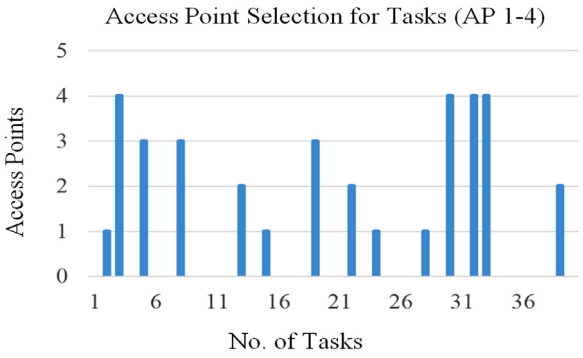
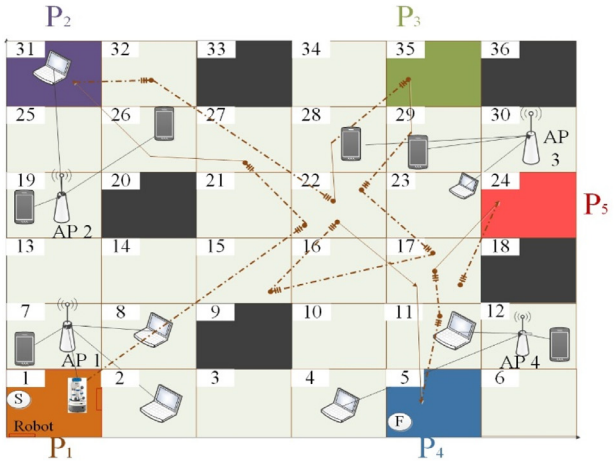


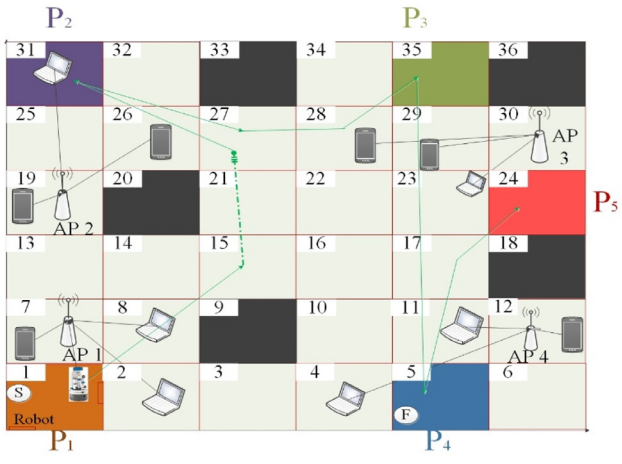
Fig. 11. Visual representation of access point (AP) selection for the offloaded tasks.

Unfortunately for GAMRB, the lack of cloud availability hampers the system performance, even though the robots cover the most distance (1161.96 m), but it doesn't provide much benefit as some tasks are too latent and consumes high energy in the local processor. Here the additional movement may utilize the robots more, but robot's local processors can't compensate for the high computational requirements of these tasks. In comparison to that, for GAMRC, the robots can offload such tasks to cloud and even move accordingly to help with offloading. This saves valuable time and energy, as reflected in the performance. Finally, analysis of movement performance for robot R_1 (Fig. 12) also help reach the same conclusion, where additional robots and cloud assistance means R_1 moves much lesser (204.84 m) in GAMRC than other two validated methods of GASRC (297.98 m) and GAMRB (582.7 m).

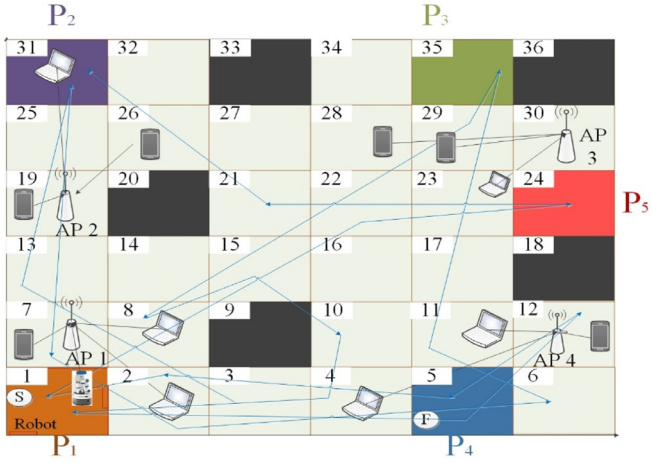
6.2.2.3. Performance comparison of robot. Given that each robot has its own energy constraint, proper utilization of energy is a top priority in such applications. Due to the involvement of cloud, the energy used for each robot was within the energy limitation (as seen in Table 6) for our GAMRC approach. In comparison to that, the GASRC process that runs the operation with robot R_1, entails an



(a) Robot (R_1) Path Plan for GASRC [297.98 m]



(b) Robot (R_1) Path Plan for GAMRC [204.84 m]



(c) Robot (R_1) Path Plan for GAMRB [582.7 m]

Fig. 12. Path planning performance comparison of robot R_1 among all three methods.

energy of 4778.80 J, which is significantly higher for a single robot, even though it is slightly under the energy bounds of 5000 J. The same can be said about GAMRB process, that results in a higher value of overall energy as well as higher energy consumption for each of the robot (2532.2 J, 1610.78 J and 2914.21 J for R_1, R_2 and R_3 respectively). Among all three methods, the lowest energy consumption for the principle robot R_1 is for the GAMRC method (936.18 J). It indicates that even though all the methods manage to meet individual robot's energy constraint, the performance of the GA based scheme for cloud networked multi-robot system results in the lowest energy consumption for each of the robots (including the primary robot R_1).

7. Conclusion

Task offloading based decision-making is a critical issue for cloud computing in networked robotic applications. Contrary to single robot applications, the offloading for cloud networked multi-robot system is further complicated by the addition of local R–R communication along with the cloud based R–C offloading. Therefore a proper balance of workload between local and cloud based offloading is required. Moreover, the offloading performance is greatly influenced by the ability of robot to move on-demand and gain access to better gateways (communication links) for connecting to the cloud. Therefore, in this paper we have merged all this aspects and proposed a novel 4-layer decision-making scheme to identify all the near-optimal solutions by leveraging the complementary strength of network connectivity, motion planning and local robot–robot interaction. Initially, we present a framework for task offloading in a cloud networked multi-robot system so that the robots can communicate with each other and offload tasks by utilizing its motion and connectivity features. We formulate a joint optimization problem for offloading where the 36 cell workspace and 40 node taskflow is derived from the motivational application of parcel sorting and distribution in an automated warehouse. Based on the scenario, the optimization problem for offloading is tackled by a GA based scheme that proposes 4-layers of decision-making: (i) task allocation, (ii) robot selection for offloading, (iii) movement decision and (iv) AP selection. After running the simulation, results are acquired and verified through comparison with two validated benchmarks, one that considers GA based offloading for a single robot [24] and the other that implements a method for multi-robot systems without any inclusion of cloud. The outcome implies that the performance of our approach is superior in terms of energy usage as well as completion time/latency. We also conclude from further investigation that the addition of cloud helps complete the computation-heavy tasks much faster. Moreover, the local R–R sharing utilizes the available resources to offload tasks in a more efficient way. Overall, all these factors combine together to attain a system performance that is more enhanced in every way than the other implemented procedures considered as benchmarks. For future work, we hope to modify our scheme by implementing a real-time formation of networked robots, perform simulations on our developing hardware [44] and further validate our work by tackling more complicated applications in different smart city and smart factory environment.

Declaration of Competing Interest

None.

References

[1] X. Li, D. Li, J. Wan, A.V. Vasilakos, C.-F. Lai, S. Wang, A review of industrial wireless networks in the context of Industry 4.0, Wirel. Netw. 23 (1) (November 2017) 23–41.

[2] D. Yuan, J. Jin, J. Grundy, Y. Yang, A framework for convergence of cloud services and Internet of Things, in: IEEE 19th International Conference on Computer Supported Cooperative Work in Design *(CSCWD)*, Calabria, Italy, May 2015, pp. 349–354.

[3] P. Mell, T. Grance, The NIST Definition of Cloud Computing, 2011.

[4] J. Wan, S. Tang, Q. Hua, D. Li, C. Liu, J. Lloret, Context-aware cloud robotics for material handling in cognitive industrial internet of things, IEEE Internet Things J. 5 (July (4)) (2017).

[5] M. Stenmark, J. Malec, K. Nilsson, A. Robertsson, On distributed knowledge bases for robotized small-batch assembly, IEEE Trans. Autom. Sci. Eng. 12 (March (2)) (2015) 519–528.

[6] C. Mühlbacher, G. Steinbauer, S. Gspandl, M. Reip, Model-based testing of an industrial multi-robot navigation system, in: 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), São Paulo, Brazil, May 2017, pp. 1652–1654.

[7] D. Claes, F. Oliehoek, H. Baier, K. Tuyls, Decentralised online planning for multi-robot warehouse commissioning, in: 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), São Paulo, Brazil, May 2017, pp. 492–500.

[8] L. Luo, N. Chakraborty, K. Sycara, Distributed algorithm design for multi-robot task assignment with deadlines for tasks, in: IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, October 2013, pp. 3007–3013.

[9] M. Erdelj, M. Król, E. Natalizio, Wireless Sensor Networks and Multi-UAV systems for natural disaster management, Comput. Netw. 124 (2017) 72–86 September 2017.

[10] N. Boysen, D. Briskorn, S. Emde, Parts-to-picker based order processing in a rack-moving mobile robots environment, Eur. J. Oper. Res. 262 (October) (2017) 550–562.

[11] J. Jarvis, R. Rönnquist, D. McFarlane, L. Jain, A team-based holonic approach to robotic assembly cell control, J. Netw. Comput. Appl. 29 (August) (2005) 160–176.

[12] J. Wan, S. Tang, D. Li, S. Wang, C. Liu, H. Abbas, et al., A manufacturing big data solution for active preventive maintenance, IEEE Trans. Ind. Inf. 13 (February) (2017) 2039–2047.

[13] S. Wang, C. Zhang, C. Liu, D. Li, H. Tang, Cloud-assisted interaction and negotiation of industrial robots for the smart factory, Comput. Electr. Eng. 63 (October) (2017) 66–78.

[14] J. Cao, B. Zeng, J. Liu, Z. Zhao, Y. Su, A novel relocation method for simultaneous localization and mapping based on deep learning algorithm, Comput. Electr. Eng. 63 (October) (2017) 79–90.

[15] G. Hou, K. Zhou, T. Qiu, X. Cao, M. Li, J. Wang, A novel green software evaluation model for cloud robotics, Comput. Electr. Eng. 63 (October) (2017) 139–156.

[16] T. Bonkenburg, Robotics in Logistics: a DPDHL perspective on implications and use cases for the logistics industry, DHL Trend Research, 2016.

[17] L. Zhou, Y. Shi, J. Wang, P. Yang, A Balanced Heuristic Mechanism for Multirobot Task Allocation of Intelligent Warehouses, Math. Probl. Eng. 2014 (November) (2014).

[18] S. Subbiah, C. Schoppmeyer, J.M.D.L.F. Valdès, C. Sonntag, S. Engell, Optimal Management of Shuttle Robots in a High-Rise Warehouse Using Timed Automata Models, IFAC Proc. Vol. 46 (9) (2013) 1358–1363.

[19] W. Li, W. Shen, Swarm behavior control of mobile multi-robots with wireless sensor networks, J. Netw. Comput. Appl. 34 (July (4)) (2011) 1398–1407.

[20] A. Rahman, J. Jin, A. Cricenti, A. Rahman, M. Palaniswami, T. Luo, Cloud-enhanced robotic system for smart city crowd control, J. Sensor Actuator Netw. 5 (December) (2016).

[21] C.M. Sarathchandra Magurawalage, K. Yang, L. Hu, J. Zhang, Energy-efficient and network-aware offloading algorithm for mobile cloud computing, Comput. Netw. 74 (December) (2014) 22–33.

[22] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, et al., Path planning with modified a star algorithm for a mobile robot, Proc. Eng. 96 (2014) 59–69.

[23] L. Chen, A distributed access point selection algorithm based on no-regret learning for wireless access networks, in: Proceedings of IEEE 71st Vehicular Technology Conference, Taipei, Taiwan, May 2010, pp. 1–5.

[24] A. Rahman, J. Jin, A. Cricenti, A. Rahman, M. Panda, Motion and connectivity aware offloading in cloud robotics via genetic algorithm, in: IEEE Global Communications Conference (GLOBECOM), Singapore, December 2017, pp. 1–6.

[25] A. Rahman, J. Jin, A. Cricenti, A. Rahman, A. Kulkarni, Communication-aware cloud robotic task offloading with on-demand mobility for smart factory maintenance, IEEE Trans. Ind. Inf. 15 (5) (2019) early access.

[26] F. Tao, Y. Cheng, L.D. Xu, L. Zhang, B.H. Li, CCIoT-CMfg: cloud computing and Internet of Things-based cloud manufacturing service system, IEEE Trans. Ind. Inf. 10 (February (2)) (2014) 1435–1442.

[27] S. Wang, J. Wan, D. Zhang, D. Li, C. Zhang, Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination, Compu. Netw. 101 (June) (2016) 158–168.

[28] Z. Bi, L.D. Xu, C. Wang, Internet of Things for enterprise systems of modern manufacturing, IEEE Trans. Ind. Inf. 10 (January) (2014) 1537–1546.

[29] Starship Technologies Parcel Delivery Robots; [Online]. Available: https://aws.amazon.com/ec2/.

[30] Knightscope Robots [Online]. Available: https://www.knightscope.com/.

[31] The TUG® Smart Autonomous Mobile Robot [Online]. Available: http://www.aethon.com/.

[32] Y. Jin, J. Jin, A. Gluhak, K. Moessner, M. Palaniswami, An intelligent task allocation scheme for multihop wireless networks, IEEE Trans. Parallel Distrib. Syst. 23 (June (3)) (2011) 444–451.

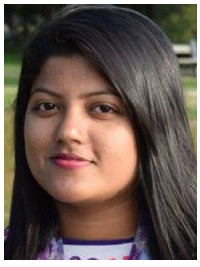
[33] S.-G. Cui, H. Wang, L. Yang, A simulation study of A-star algorithm for robot path planning, in: 16th International Conference on Mechatronics Technology, 2012, pp. 506–510.
[34] A. Rahman, J. Jin, A. Cricenti, A. Rahman, D. Yuan, A cloud robotics framework of optimal task offloading for smart city applications, IEEE Global Communications Conference (GLOBECOM), December 2016.
[35] T.N. Lin, S.H. Fang, W.H. Tseng, C.W. Lee, J.W. Hsieh, A group-discrimination-based access point selection for WLAN fingerprinting localization, IEEE Trans. Veh. Technol. 63 (2014) 3967–3976.
[36] A. Kumar, E. Altman, D. Miorandi, M. Goyal, New insights from a fixed point analysis of single cell IEEE 802.11 WLANs, in: IEEE/ACM Transactions on Networking, 15, 2005, pp. 1550–1561.
[37] A. Konak, D.W. Coit, A.E. Smith, Multi-objective optimization using genetic algorithms: a tutorial, Reliab. Eng. Syst. Saf. 91 (9)) (September 2006) 992–1007.
[38] W. Zhang, S. Tan, Q. Lu, X. Liu, W. Gong, A Genetic-algorithm-based approach for task migration in pervasive clouds, Int. J. Distrib. Sens. Netw. 2015 (January) (2015).
[39] J.C. Bongard, Evolutionary robotics, Commun. ACM 56 (2013) 74–83.
[40] A. Rahman, B. Verma, Ensemble classifier generation using non-uniform layered clustering and Genetic Algorithm, Knowl.-Based Syst. 43 (May 2013) 30–42.
[41] M.A. Hossain, I. Ferdous, Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique, in: Proceedings of 2013 International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, February 2014, pp. 137–141.
[42] P. Bhattacharjee, P. Rakshit, I. Goswami, A. Konar, A.K. Nagar, Multi-robot path–planning using artificial bee colony optimization algorithm, in: Proceedings of Third World Congress on Nature and Biologically Inspired Computing, Salamanca, Spain, October 2011, pp. 219–224.
[43] R. Rashid, N. Perumal, I. Elamvazuthi, M.K. Tageldeen, M.K.A.A. Khan, S. Parasuraman, Mobile robot path planning using Ant Colony Optimization, in: Proceedings of 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA), September 2016, pp. 1–6.
[44] A. Rahman, J. Jin, Y. Wee Wong, K. Shan Lam, Development of a Cloud-enhanced Investigative Mobile Robot, International Conference on Advanced Mechatronic Systems (ICAMechS'), November 2016.

Akhlaqur Rahman received his PhD degree in Electrical and Electronic Engineering from Swinburne University of Technology in 2019. He received his B.Sc in Electrical and Electronic Engineering from American International University-Bangladesh in 2012. Before joining Swinburne, Akhlaqur was a Lecturer and Coordinator for the department of Electrical and Electronic Engineering (from 2013 to 2014) at Uttara University in Bangladesh. Akhlaqur's research focus is on cloud networked robotics, network optimization, multi-robot systems and Internet of Things (IoT). Akhlaqur is also the first ever MATLAB ambassador from Australia and is employed by Mathworks Inc.

Jiong Jin (IEEE M'11) received the B.E. degree with First Class Honours in Computer Engineering from Nanyang Technological University, Singapore, in 2006, and the Ph.D. degree in Electrical and Electronic Engineering from the University of Melbourne, Australia, in 2011. He is currently a Senior Lecturer in the School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology, Melbourne, Australia. Prior to it, he was a Research Fellow in the Department of Electrical and Electronic Engineering at the University of Melbourne from 2011 to 2013. His research interests include network design and optimization, fog and edge computing, robotics and automation, Internet of Things and cyber-physical systems as well as their applications in smart manufacturing, smart transportation and smart cities.

Ashfaqur Rahman is the Principal Research Scientist and Team Leader at Data61 division of CSIRO. He is a data scientist and working as a researcher for nearly 15 years. His key research areas are machine learning and Data mining. More specifically Ensemble learning and fusion, Genetic Algorithm based network optimization, Distributed Machine Learning for Big Data, Feature selection/weighting methods, Image segmentation and classification. Dr. Rahman received his Ph.D. degree in Information Technology from Monash University, Australia in 2008. He has published around 80 peer-reviewed journal articles, book chapters and conference papers.

Antonio Cricenti is the current Department Chair of the Department of Telecommunications, Electrical, Robotics and Biomedical Engineering in School of Software and Electrical Engineering, Faculty of Science, Engineering and Technology, Swinburne University of Technology. He received the Bachelor Engineering (Elec) (Hons) from the University of Melbourne, a Graduate Diploma in Education from the Melbourne College of Advanced Education (University of Melbourne) and was awarded the Master of Engineering and PhD from Swinburne University of Technology. Before joining Swinburne, he worked at Bechtel Pacific Limited as a consulting electrical engineer, and at General Motors Holden Ltd as an automotive design engineer. His research interests include Software Defined Networking, Intelligent Transport Systems, Internet of Things, Teletraffic Engineering and Engineering Education.



Mahbuba Afrin is currently pursuing her PhD degree at Swinburne University of Technology, Australia, holding Tuition Fee Scholarship (TFS). She has also been awarded Data61 PhD Scholarship to work in collaboration with Commonwealth Scientific and Industrial Research Organisation (CSIRO). She received her M.Sc. (2017) and B.Sc. (2015) in Computer Science and Engineering from University of Dhaka, Bangladesh. Before starting PhD programme, she worked as a Lecturer at Department of Computer Science and Engineering in United International University, Bangladesh (2015–2017) and as a Research Assistant in Green Networking Research Group, University of Dhaka (2013–2015). Her research interests include Cloud Networked Robotics, Multi-Robot Systems, Cloud Computing, Cloud Radio Access Network, Mobile Cloud Computing and Internet of Things (IoT).

Yu-ning Dong received his Ph.D. in Radio Engineering from Southeast University in 1988, and his M.Phil degree from the Department of Computer Science, Queens University of Belfast, UK, in 1998. He is now a professor and doctoral tutor at the School of Communication and Information Engineering, Nanjing University of Posts and Telecommunications. IEEE Member, Senior Member of China Electronics Association, Senior Member of China Institute of Communications. His research interest are: Cross-layer optimization, studying technologies and applications of multimedia transmission, QoE/QoS modeling, green communication, content analysis, image and video information processing in heterogeneous wireless IP networks.