A new cloud autonomous system as a service for multi-mobile robots

Aida A. Nasr

Abstract

Today, mobile robot is used in most industrial and commercial fields. It can improve and carry out work complex tasks quickly and efficiently. However, using swarm robots to execute some tasks requires a complex system for assigning robots to these tasks. The main issue in the robot control systems is the limited facilities of robot embedded system components. Although, some researchers used cloud computing to develop robot services. They didn’t use the cloud for solving robot control issues. In this paper, we have used cloud computing for controlling robots to solve the problem of limited robot processing components. The main advantage of using cloud computing is its intensive computing power. This advantage motivates us to propose a new autonomous system for multi-mobile robots as a services-based cloud computing. The proposed system consists of three phases: clustering phase, allocation phase, and path planning phase. It groups all tasks/duties into clusters using the k-means algorithm. After that, it finds the optimal path for each robot to execute its duties in the cluster based on the Nearest neighbor and Harris Hawks Optimizer (HHO). The proposed system is compared with systems that use a genetic algorithm, simulated annealing algorithm, and HHO algorithm. From the finding, we find that the proposed system is more efficient than the other systems in terms of decision time, throughput, and the total distance of each robot.

Keywords Multi-robot · Cloud computing · HHO · Autonomous system · Path planning · Scheduling

1 Introduction

Today mobile robot is used in many locations: streets, factories, government organizations, hospitals, and universities. They can move and explore, transport things and goods, and complete complex tasks. Moreover, scientists use mobile robots in space and ocean exploration [1–3]. One example of space exploration is NASA tried sending robots for exploration in regions called analogs (i.e. analog is a location where the environment is hassling to locations like Mars or the moon, where a robot may be used). One NASA analog exists in the Arizona desert. NASA robotics specialists perform experiments in the desert to estimate new ideas for rovers, spacewalks, and ground support. Some of these experiments are performed by a group called Desert RATS, which is referred to by Desert Research and Technology Studies [4].

The multi-mobile robot system (MMRS) is a group of robots that are cooperating in performing one task/ duty or performing some tasks/duties related to each other [5]. This kind of robot system needs an efficient plan for executing duties and communicating with each other. To implement the MMRS management system, robot engineers need some powerful robot components to be fast and operate in real-time. Outfitting each robot in a swarm with high-powered components results in a massive and expensive robot. While the use of cloud computing may be an ideal solution to solve this problem, cloud computing is characterized by high-performance capabilities and great speed in performing various data processing resulting in reduced response time. In addition, using cloud computing services is cheaper than building robot components and implementing a system for each robot [6].

Last years, researchers and cloud engineers began to use cloud computing for developing robotic systems as a service. Researchers in [7, 8] showed the idea of using cloud computing with robotic systems for executing tasks. The
main idea of using cloud computing services for coordinating the work between multi-robot systems is not new, where before 2005 appeared what is called service-oriented architecture SOA [9], which is used to develop different robots such as Robotic Swarm robotics applications.

Cloud robotics is a field of robotics that aims to use cloud technologies, services, and various cloud resources for robotics [10]. In this paper, we propose a new cloud system for managing swarm robot systems to perform multi-duty in real-time. The system makes robots benefit from cloud power via internet connection automatically without human interactions for controlling executed duties. The new system can be used to manage organizations that depend on robots, in a completely autonomous way. The contributions of the paper are as follows:

- Proposing a new cloud multi-robot model.
- Applying k-means to cluster all duties into groups.
- Developing a new hybrid path planning algorithm based on the HHO algorithm and Nearest Neighbor algorithm to determine the best path.
- Comparing the new algorithm with Genetic algorithms, simulated annealing algorithm, and traditional HHO algorithm.

2 Cloud multi-robot model and allocation problem

Cloud computing platform is a common platform used for executing tasks and supplying various users with different kinds of computing services such as Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS), Robot applications as a Service (RaaS) and many new services are shown every day [11]. In this paper, we propose a new model called Cloud Multi-Robot Model CMRM (see Fig. 1).CSRMM is used for managing the work between multi-robots remotely and over the internet using cloud computing. The main advantage of this model is that the processing takes place in the cloud platform instead of the robot controller. CMRM consists of seven components: robots, duties, data, the system, manager, cloud services, and virtual machines (VMS). We explain the function of each component below in detail.

Multi-robot is a group of robots that work with each other to perform some duties. Robots and users send data over the internet to cloud management systems. The data sent contains robot location, robot specifications, number of duties, duty location, and duties description. A Cloud management system consists of an information system that is used for saving data. Furthermore, there is a pool of cloud services such as RaaS, SaaS, IaaS, and PaaS. The manager is the brain of the cloud management system. It is responsible for sending data to VMs for processing and taking actions according to results. The main sim of the manager is to manage the robots. The manager can use some VMs for performing its tasks in parallel (Fig. 1).

The multi-robot duty allocation (MRDA) problem is a problem of how multi-robots can be allocated to many duties for achieving a specific goal, where $m$ robots are working together to carry out $n$ duties. We can represent this problem as follows:

$$A : D \rightarrow R,$$

where $A$ refers to the allocation problem, $D$ is the duties set $D = \{d_1, d_2, d_3, \ldots, d_n\}$ and $R$ is the robot set $R = \{r_1, r_2, r_3, \ldots, r_m\}$. An allocation algorithm is used for assigning robots to perform duties. The input to the algorithm is the robots list and duties list, while the output is the allocation solution.

3 Previous work

Multi-robots are used in many industrial and commercial fields. However, they face some issues such as communication security, path planning, negotiation, allocation, and control system. Many researchers proposed cloud computing for developing robot services but without looking at the development of a robot management system. Researchers that developed management systems for robots didn’t use cloud computing platforms. They also enhanced the management system from the side of allocating robots to execute duties instead of the overall system. In this section, we show some of these works. In [12], the authors presented a new taxonomy for allocating multi-agent robotics. It classifies the aspects of resource allocation in robotics into four classes: resource type, performance metrics, application structure, service model, and application mechanism.

Authors in [13] developed a resource allocation method for the swarm robot’s algorithm. Their main contribution is minimizing task time execution and resource consumption. The method consists of four phases: bid generation, winner update, task trade, and list update. In the bid generation phase, all possible paths to the station are generated. The selected path of the robot is computed by considering the task order. In the winner update phase, robots send and receive the bid messages with neighbors. They allocate tasks using the decentralized idea. The third phase is responsible for selecting the best task and specifying the winner for it while in the fourth phase the task list is updated. The main disadvantage of this algorithm is that it calculates all possible paths and uses the allocation and reallocation method for distributing tasks to the robots this takes more start time.
In [14], the authors proposed two algorithms: a self-organizing dynamic task allocation algorithm that depends on the optimal mass transport theory (OMT) and a dynamic task allocation method based on adaptive grouping (OMT(a)) to solve swarm robot allocation tasks. The algorithms handle the problem of an unbalanced distribution of robots to perform all tasks using homogeneous robots with the same facilities. This makes the algorithm not adaptive to any environment. In addition, the algorithm has a high running time.

Authors in [15] developed an algorithm called improved particle swarm optimization and greedy (IPSO-G), which is a combination between particle swarm optimization and greedy algorithms. The algorithm tries to solve the problems of robot resource allocation, finding near-optimal solutions for multi-robot collaborative planning and unbalanced robot allocation problems. The algorithm has good results for a small number of robots and tasks.

Authors in [16] proposed a new path planning algorithm for finding the optimal path in low-time running. The algorithm is built on the rule which says that “the shortest distance between two points in the straight line between them”. The authors used this rule to get the best path between the start point and the endpoint. They focused on the critical region to start the searching and move the path until they find the best. The straight method gives good results for path planning problems, but it cannot be used for robot allocation problems.

In [17], the authors proposed an allocation algorithm called the water wave optimization algorithm with the genetic algorithm (WWO_GA). The algorithm consists of two phases: the water wave optimization phase and the genetic phase. Mainly, it applied the WWO but it used the GA to increase the number of solutions for each iteration. The algorithm succeeded to achieve a high performance than using each phase individually, but it has a low execution speed. In addition, the algorithm cannot be used for solving path planning algorithms.

Authors in [18] used the Newtonian law of gravity for proposing a new multi-robot task allocation (MRTA) algorithm. Authors assumed that tasks and robots are fixed objects and movable objects, respectively. A specific mass is assigned for each task, which refers to the task quality. According to the gravity low, tasks apply a gravitational force to the robots to change their locations and find the best allocation solution. The algorithm depends on the distributed masses of tasks to get the solution. This assumption sometimes gives poor solutions.

Robot allocation algorithms are developed not only for services robots but also they are developed for distributing tasks between humans and robots. Authors in [19] proposed a new algorithm called the disassembly sequence planning (DSP) algorithm. The DSP algorithm is developed for the human–robot collaboration (HRC) setting. It distributes the workers between humans and robots to save the total time.

Authors in [20] developed a new allocation method based on a reinforcement learning algorithm for allocating some tasks into the free robots. The paper showed that the new method found results better than the heuristic methods.

4 Background

In this paper, three free algorithms for managing bots to perform tasks are used in the proposed system. These algorithms are the k-means algorithm, the nearest-neighbor (NN) algorithm, and the Harris Hawks Optimizer (HHO). Each algorithm has a specific role in the management plan. The k-means algorithm is used for dividing all duties into clusters. The NN algorithm is used to generate the first
solution for the HHO population. The HHO algorithm is used to find the path planning solution. In this section, we explain each algorithm in detail.

4.1 The k-means algorithm

The k-means algorithm \[21\] is one of the popular cluster algorithms. It tries to partition the dataset into \(k\) clusters where every data point in the dataset belongs to one cluster \[22\]. Then, it tries to group the nearest data points into one cluster according to the mean of these points. The k-means algorithm works as follows:

1. Set \(k\) as the number of clusters.
2. Initialize the number of means equal to \(k\).
3. Set an initial value for each mean.
4. Calculate the distance between each point in the data set with each mean to define the nearest mean.
5. Assign the selected point to the cluster with the nearest mean.
6. Calculate the new mean for each cluster.
7. Keep iterating until there is no change to the means’ values.

4.2 The Nearest neighbor for traveling salesman problem

The nearest neighbor (NN) algorithm is used for selecting a traveling salesman tour with the smallest distance \[23\]. The main idea of the algorithm is that the salesman begins from home and then travels to the next closest location. After that, he travels to the nearest unexplored location and repeats this operation until all the locations have been visited. Then, he returns home. As shown in Fig. 2, we assume that the home is location A. The NN algorithm selects the closest location to the home (e.g. location F) then it selects the nearest location F (e.g. location E). The next steps show the steps of the NN algorithm:

1. Define the home node.
2. Find the route with the smallest distance of the nearest unvisited nodes.
3. Update the current node.
4. Repeat steps 2 and 3 until back to the home node.
5. Return the complete path.

4.3 Harris Hawks optimizer (HHO)

In 2019, the Harris Hawks Optimizer (HHO) was introduced as a new heuristic population-based approach for optimization problems \[24, 25\]. The main notion of this technique is emulated by the cooperative actions of smart Harris’ Hawks birds while hunting a prey.

Harris’s hawk uses a smart tactic named “surprise pounce” or “seven kills” strategy. The seven kills tactic is implemented in practice by having team members launch active attacks from various locations to increase their chances of success in catching the rabbit.

The traditional algorithm consists of three stages: exploration stage, transformation stage, and exploitation stage. In the exploration stage, the Harris hawk observes and waits until a pray appears. They change their locations randomly to find a prey. Equation 2 can be used to generate the new locations of the Harris hawks.

\[
h_k(t + 1) = \begin{cases} L(t) - r_1[L_r(t) - 2r_2L(t)] & Q \geq 0.5 \\ (L_{rabbit}(t) - L_k(t)) - r_3(LB + r_4(UB - LB)) & Q < 0.5 \end{cases}
\]

where \(L_r(t)\) is a hawk (i.e. solution) that is selected from the current population, \(L(t)\) is the current position vector of the solution while \(L_{rabbit}(t)\) is the rabbit position (i.e. best solution) and \(L_k(t + 1)\) is the position vector of hawk k in the next iteration t. while \(r_1, r_2, r_3, r_4\) and \(Q\) are random numbers \(\in [0\, 1]\), \(LB\) and \(UB\) values show the upper and lower bounds of variables and \(L_k(t)\) is the average position of the current population. We can compute \(L_\alpha(t)\) using Eq. (3)

\[
L_\alpha(t) = \frac{1}{h} \sum_{i=1}^{h} L(t)
\]

where \(h\) is the total number of hawks in the iteration t.

In the second stage, the HHO algorithm applies a transformation stage as an intermediate stage before applying the exploitation stage. In the transformation stage, the algorithm computes the rabbit escaping energy \(E\) that decreases when the rabbit runs away from the hawks. It is modeled as follows in Eq. (4).
\[ E = 2E_0 \left(1 - \frac{t}{\text{Max}_T} \right) \]  

(4)

where \( E_0 \) is the initial escaping energy in \([-1,1] \) and \( \text{Max}_T \) is the maximum number of iterations energy. When \(|E| \geq 1\), this means rabbit has the enough energy to escape. In the other case \(|E| < 1\), this means that the rabbit has low energy for escaping and the exploitation stage will be started.

The exploitation stage in this stage, hawks will attack the rabbit where \( E < 1 \). However, the rabbit often tries to escape from the hawks according to its escape energy. Indeed, there are four different mechanisms for chasing patterns namely: (i) soft besiege, (ii) soft besiege with progressive rapid dives, (iii) hard besiege, and (iv) hard besiege with progressive rapid dives. A random number \( r \) is generated to determine the probability of a prey successfully escaping \((r<0.5)\) or unsuccessfully escaping \((r \geq 0.5)\) before a surprise pounce.

The soft besiege occurs when \( r \geq 0.5 \) and the rabbit has enough escaping energy when \(|E| \geq 0.5 \). In this case, the algorithm creates a new solution using the next equations:

\[
L(t+1) = \Delta L(t) - E|J_L_{rab}(t) - L(t)| \quad (5)
\]

\[
J = 2(1 - r_S) \quad (6)
\]

\[
\Delta L(t) = L_{rab}(t) - L(t) \quad (7)
\]

where \( J \) is a random rabbit jump which represents the strength of the escaping. In addition, \( \Delta L(t) \) refers to the difference between the rabbit position and the Current position of the hawk in the current iteration \( t \).

The algorithm applies soft besiege with progressive rapid dives operation when \( r < 0.5 \) and \( E \geq 0.5 \) and the new solutions are generated by eqs. 8, 9 and 10.

\[
L(t+1) = \begin{cases} 
Y & \text{if } F(Y) < F(L(t)) \\
W & \text{if } F(W) < F(L(t)) 
\end{cases} \quad (8)
\]

\[
Y = L_{rab}(t) - E|J_L_{rab}(t) - L(t)| \quad (9)
\]

\[
W = Y + S \times LF(d) \quad (10)
\]

where \( Y \) and \( W \) are the new solutions to the problem with \( d \) dimension. \( S \) is a random vector with size \( 1 \times D \) and \( LF \) is the Levy Fight function which is given by:

\[
LF(l) = 0.01 \times \frac{u \times \sigma}{\|v\|^3} \quad (11)
\]

\[
\sigma = \left( \frac{\Gamma(1+\beta) \times \sin \left( \frac{\pi \beta}{2} \right)}{\Gamma \left( \frac{\beta}{2} \right)} \right)^{1/\beta} \quad (12)
\]

where \( \beta \) equals 1.5 and \( u \) and \( v \) are selected randomly from 0 to 1.

Hard besiege operation is applied when \( r \geq 0.5 \) and the rabbit has low energy \(|E| < 0.5 \). In this case, the algorithm updates the population using Eq. (13).

\[
L(t+1) = L_{rab}(t) - E|J_L_{rab}(t) - L(t)| \quad (13)
\]

Finally, the algorithm performs Hard besiege with progressive rapid dives when \(|E| < 0.5 \) and \( r < 0.5 \). This occurs when the rabbit cannot escape, and hawks can near more to catch the rabbit. New solutions are generated using Eqs. (14).

\[
L(t+1) = \begin{cases} 
Y & \text{if } F(Y) < F(X(t)) \\
W & \text{if } F(W) < F(X(t)) 
\end{cases} \quad (14)
\]

\[
Y = L_{rab}(t) - E|J_L_{rab}(t) - L(t)| \quad (15)
\]

\[
W = Y + S \times LF(D) \quad (16)
\]

Authors in [17] summarize all the stages of the HHO algorithm in Fig. 3.

Although the HHO algorithm gives good results for solving some problems, it has high complexity and falls in the local optimum solution [13]. As a result, it gives poor solutions. In addition, the HHO algorithm uses a weak fitness function. It often points to the wrong direction of future solutions and far away from the right direction. Figure 4 shows the solutions of a new generation. From the figure, we find that some good solutions are rejected because the algorithm measures the acceptance according to the rabbit location (best solution) in the previous population. Moreover, the HHO generates the first population randomly. This makes the success of the algorithm depend on luck. And the energy calculation is based on the random value \( w \), which is used to determine a method for producing solutions. This leads to bad solutions. In this paper, we solve all these problems by proposing a new fitting function for generating more efficient solutions.
5 Cloud manager system for multi-robots (CMSR)

In Sect. 2, we propose a new cloud model for multi-robot called CMRM for solving robot allocation based on cloud computing platforms. In the model, users send data about duties and robots over the internet to an information system. The data is sent to the cloud manager to take the allocation decision. In this section, we show the design of the CMSR system. The main idea of CMSR is to manage multi-robots for performing submitted duties in different locations with low time and small distances. The proposed system is considered the first system for managing multi-robots remotely over the internet. In addition, the CMSR system is adaptive. It changes the allocation plan after the environment is updated and new robots are added. In addition, the proposed system is smart because it depends on machine learning techniques for distributing robots for executing tasks.

The proposed system consists of three phases: clustering phase, allocation phase, and path planning phase. The outline steps of the CMSR system are described in algorithm 1.

![Image](HHO_population_generation_method.png)

**Algorithm 1: Cloud Manager System for Multi-robot**

**Input**: requests, number of robots, deadline for each duty, coordination of locations  
**Output**: distribution solution (which robot will perform these duties and how)  
1- Each robot sends its data to the cloud (e.g., location, specifications)  
2- The user sends all duties data.  
3- The user sends the location of each duty.  
4- Apply a k-means algorithm to cluster the duties according to the location.  
5- Save the final centroids in C_List that result from applying k-means  
6- Allocating each cluster to the nearest robot using C_List  
7- **For each robot**  
8- Apply path planning algorithm  
9- **End for**  
10- If there are new robots added to the environment  
11- Go to step 6  
12- Return the duties of each robot and send the path planning solution for all robot
5.1 Clustering phase

The first step of the CMSR system is dividing all duties into clusters according to the distance between them, where the group of duties that are close to each other are considered as one cluster. We use the k-means algorithm for clustering all duties. The system sets the number of clusters equal to the number of robots. The system applies the steps in Sect. 4.A after applying k-means and generating the clusters, the system saves the final means (i.e., centroids) into C_List to use them in the next phase.

5.2 Allocation phase

After all, clusters are generated, the system allocates each cluster to one robot or multi-robots according to the available robots. Firstly, it measures the distance between each centroid in C_List and the robot location. The robot is assigned to the duties cluster that has the closest centroid to the robot. Figure 5 shows an example of the allocating phase.

There are two scenarios for allocating a robot to a cluster. The first scenario is allocating each robot to one cluster. This scenario is applied when the number of robots is constant and equals the number of clusters. The second scenario is applied when the number of robots is updated and some robots are added to the environment. In this case, the system repeats the steps of k-means until all robots are allocated.

5.3 Path planning phase

In this phase, the new system uses the modified HHO algorithm to create the best path that the robot follows to execute all duties according to the deadline of each duty. The robot converts the scheduling problem into a TSP problem. Then, it applies the NN algorithm to create the initial solution. By using the right shift method, the system generates the first population. The system represents a solution as an array, for example:

Solution = [1 3 4 6]

This solution means that the robot will start to execute the duty with number 1 then the duty with number 3 and so on. The length of the solution of one robot is different from the length of another robot according to the number of duties in each cluster. Because we consider the deadline condition, each robot starts from the duty with the smallest deadline.

The new system uses a new fitness function (17) to accept and reject each solution in the new population. Where \( F(s) = 1 \) if the solution is accepted when \( \frac{L(T)_{\text{max}} - L(T + 1)}{L(T)_{\text{max}} - L(T)_{\text{mean}}} < 1 \) otherwise the solution is rejected. This fitness function leads to finding good solutions in the right direction.

\[
F(S) = \begin{cases} 
1 & \text{if } \frac{L(T)_{\text{max}} - L(T + 1)}{L(T)_{\text{max}} - L(T)_{\text{mean}}} < 1 \\
0 & \text{Otherwise}
\end{cases} \quad (17)
\]

\[
E(t + 1) = 1 - 2E_t \left( \frac{t}{MaxT} \right) \quad (18)
\]
The steps of the modified HHO algorithm are shown in Fig. 6. From the figure, we see that the new system applies two exploration strategies in parallel by dividing the hawks into two teams and finding more solutions. The new system also applies Eq. (18) to update escape energy. The path planning algorithm shows the modified-HHO steps.

6 Performance evaluation

In experiments, we have used java language to implement the simulator of the system. The experiments were done using laptop with 12 GB RAM, core i5, and Windows 10. To evaluate the proposed system, the following metrics are used: total distance, throughput, and decision time.

**Path planning algorithm**

**Input:** Locations of duties and location of the selected robot

**Output:** The robot path

1- Select the duty with the smallest deadline to be the start point.
2- Apply the NN algorithm to get the $l_{rabbit}$ solution
3- Apply the right shift method to generate the first population of HHO
4- For each solution in pop_1
5- Find the best solution with the smallest distance
6- Set $L_{rabbit}$ with the best solution
7- End for
8- For $t = 1$ to $t_{max}$
9- Calculate fitness value for each hawk using equation (17)
10- For each solution i in the current population
11- perform exploration process using Equation (2)
12- Update escaping energy $E$ using Equation (18) and jumping strength using Equation 11
13- If ($E >= 1$)
14- Find each hawk solution using Equation (2)
15- Set random value $r$
16- if ($r >= 0.5$ and $|E| >= 0.5$) then
17- Apply Soft besiege
18- else if ($r >= 0.5$ and $|E| < 0.5$) then
19- Apply Hard besiege
20- else if ($r < 0.5$ and $|E| >= 0.5$) then
21- Apply Soft besiege with progressive rapid dives
22- else if ($r < 0.5$ and $|E| < 0.5$) then
23- Apply Hard besiege with progressive rapid dives
24- End for
25- End for
26- Return $L_{rabbit}$

Fig. 5 An example of robot allocation
6.1 Total distance

The total distance is the distance traveled by the robot until it performs all assigned duties. Figures 7, 8 and 9 show the results of the total distance of robots with CMSR, HHO, GA, and SA algorithms in centimeters. From Fig. 7, we note that each robot is allocated to one cluster with a different number of duties. According to the distances between the duties, the k-means algorithm divides all duties into clusters. This makes the execution of the duties faster and takes low time. Figure 8 shows the results after adding two robots to the environment. The results are still better than the other algorithms. Because the system is adaptive, it can handle the allocation problem when the environment changes. This is shown in Fig. 9, where the number of robots and duties increases, and still the results of the system are better than the other algorithms. This means that the proposed CMSR system is more scalable and flexible than the GA, traditional HHO, and SA algorithms.

6.2 Throughput

We calculate the throughput of the system in a minute under some assumptions. Assume that each robot has a speed of 2 m/sec. So, we can calculate the time it takes to finish all duties in the allocated cluster. Where the time that is taken by $R_1$ to execute 50 duties with speed 2 m/sec equals $\text{total distance} / 2 = 1000/2 = 500$ s. and throughput = number of executed duties / times = 50/500 = 0.1 in second and 6 duties in one minute. This result came thanks to the use of cloud computing in making decisions and directing robots on the best path.

Figures 10, 11 and 12 show the results of throughput for a different number of robots and a different number of duties. From the figures, we find that the CMSR system throughput increases with increasing the number of duties or the number of robots. Because the proposed system considers all updates in the environment, adding new robots to the work environment will increase the throughput. This is shown in Fig. 11 where two robots are added to the environment for helping in duties execution and decreasing the execution time (Fig. 12).

6.3 Decision time

Decision time is the time taken by the CMSR system to send the allocation decision to the robots. Figure 13 shows the decision time of each algorithm versus the CMSR system. From the figure, we observe that the new system takes 5 s only to manage the execution of 600 duties with 5 robots while the HHO, GA, and SA take 20, 25, and 15, respectively. This means that the CMSR system can save time and is suitable for real-time applications.

Cloud computing platform can take decisions in low time because it has high computing power. In addition, the CMSR system is scalable and reliable, where it considers adding new robots to the environment. This feature gives the system a large ability to handle any error and replace one robot with another. The main problem with the new system is the use of the internet. The system requires the use of the internet to find the allocation solution (Fig 13).
**Fig. 7** Total distance results of 5 robots performing 600 duties

| TOTAL DISTANCE | R1 (50 Duties) | R2 (100 Duties) | R3 (145 Duties) | R4 (150 Duties) | R5 (155 Duties) |
|----------------|----------------|-----------------|----------------|----------------|----------------|
| CMSR 1000      | 1000           | 1224            | 1345           | 1454           | 1788           |
| HHO 1678       | 124           | 3145            | 3445           | 4457           | 5551           |
| GA 2951        | 3000           | 3887            | 3941           | 5100           | 7489           |
| SA 3000        |                | 4102            | 4454           | 6114           | 8410           |

**Fig. 8** Total distance results of 7 robots performing 600 Duties

| TOTAL DISTANCE | R1 (50 Duties) | R2 (70 Duties) | R3 (90 Duties) | R4 (90 Duties) | R5 (100 Duties) | R6 (100 Duties) | R7 (100 Duties) |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| CMSR 1000      | 1000           | 1224           | 1345           | 1454           | 1788           | 1700           | 1750           |
| HHO 1678       | 124           | 3145           | 3445           | 4457           | 5551           | 5410           | 6145           |
| GA 2951        | 3000           | 3887           | 3941           | 5100           | 7489           | 7894           | 7314           |
| SA 3000        |                | 4102           | 4454           | 6114           | 8410           | 9145           | 9000           |

**Fig. 9** Total distance robots with 750 Duties

| TOTAL DISTANCE | R1 (90 Duties) | R2 (120 Duties) | R3 (175 Duties) | R4 (180 Duties) | R5 (185 Duties) |
|----------------|----------------|----------------|-----------------|----------------|----------------|
| CMSR 1210      | 1210           | 1451           | 2100            | 2201           | 2197           |
| HHO 2014       | 2014           | 3314           | 5112            | 5421           | 5500           |
| GA 3112        | 3112           | 4578           | 6100            | 6444           | 7441           |
| SA 3245        | 3245           | 4978           | 6400            | 6947           | 9441           |
Fig. 10  Throughput results of 5 robots and 600 duties

![Graph showing throughput results for 5 robots and 600 duties.]

Fig. 11  Throughput results of 7 robots and 600 Duties

![Graph showing throughput results for 7 robots and 600 duties.]

Fig. 12  Throughput results of 5 robots and 750 Duties

![Graph showing throughput results for 5 robots and 750 duties.]

These graphs illustrate the throughput performance of different robots performing various duties, with each robot having a different number of duties assigned. The x-axis represents the number of duties for each robot, while the y-axis shows the throughput. The results are compared for 5 robots under 600 duties, 7 robots under 600 duties, and 5 robots under 750 duties, with the throughput measured across different algorithms: CMS, HHO, GA, and SA.
7 Conclusions

Multi-robot allocation problem still is opened and needed new solutions. One of the issues in multi-robot allocation algorithms is high time complexity. In this paper, we use cloud computing platform to solve this issue. We proposed an autonomous system based on cloud computing to control several robots for executing the tasks. The proposed system called Cloud Manager System for multi-robot (CMSR) is hybrid of the k-means, NN, and the modified HHO algorithms. The new system is dynamic system that is developed to find the allocation solutions of multi-robots for executing multi-tasks. In addition, it uses a new cloud model manager for sending the right decisions to robots in real-time. The main benefit of using cloud computing is related to data sharing between the robots in real time. From the experiment results, we find that the CMSR system gives better solutions than the traditional HHO, SA, and GA algorithms. The new system can take the decision in low time. In addition, it has high throughput. Moreover, the new system gives smaller distances than the other systems. In the future, we will modify the system to work in a hybrid secure cloud computing.

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

Declarations

Conflict of interests The authors have no conflicts of interest to declare.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Alatise MB, Hancke GP (2020) A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access 8:39830–39846
2. Hu B, Guo K, Wang X, Zhang J, Zhou D (2021) RRL-GAT: Graph attention network-driven multi-label image robust representation learning. IEEE Internet of Things Journal
3. Tingxiang F, Long P, Liu W, Pan J (2020) Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. Int J Robot Res 39(7):856–892
4. https://www.nasa.gov/audience/foreducators/robotics/home/index.html. Accessed 24 Jun 2022
5. Schillinger P, Garcia S, Makris A, Roditakis K, Logothetis M, Alevizos K, Ren W et al (2021) Adaptive heterogeneous multi-robot collaboration from formal task specifications. Robot Auton Syst 145:103866
6. Lei Y, Stouraitis T, Vijayakumar S (2021) Decentralized ability-aware adaptive control for multi-robot collaborative manipulation. IEEE Robot Autom Lett 6(2):2311–2318
7. Kapitonov A, Lonschakov S, Bulatov V, Kia B, and White J (2021) Robot-as-a-service: from cloud to peering technologies. In: 2021 The 4th international conference on information science and systems, pp 126–131
8. Mouradian C, Yangui S, Glitho RH (2018) Robots as-a-service in cloud computing: search and rescue in large-scale disasters case study. In 2018 15th IEEE annual consumer communications and networking conference (CCNC), pp 1–7. IEEE, 2018
9. Naghmeh N, Ismail W, Ghani I, Nazari B, Bahari M (2020) Understanding Service-Oriented Architecture (SOA): a systematic literature review and directions for further investigation. Inf Syst 91:101491
10. Pignaton de Freitas E, Olszewska JI, Carbonera JL, Fiorini SR, Khamis A, Ragavan SY, Barreto ME, Prestes E, Habib MK, Redfield S and Chibani A (2020) Ontological concepts for
information sharing in cloud robotics. J Ambient Intell Humaniz Comput, pp.1–12
11. Alarifi A, Dubey K, Amoon M, Altameem A, Abd El-Samie FE, Altameem A, Sharma SC, Nasr AA (2020) Energy-efficient hybrid framework for green cloud computing. IEEE Access 8:115356–115369
12. Mahbuba A, Jin J, Rahman A, Rahman A, Wan J, Hossain E (2021) Resource allocation and service provisioning in multi-agent cloud robotics: a comprehensive survey. IEEE Communications Surveys and Tutorials
13. Dong-Hyun L (2018) Resource-based task allocation for multirobot systems. Robot Auton Syst 103:151–161
14. Qiuzhen W, Mao X (2020) Dynamic task allocation method of swarm robots based on optimal mass transport theory. Symmetry 12(10):1682
15. Kong X, Gao Y, Wang T, Liu J, Xu W (2019) Multi-robot task allocation strategy based on particle swarm optimization and greedy algorithm. In 2019 IEEE 8th joint international information technology and artificial intelligence conference (ITAIC), pp. 1643–1646. IEEE, 2019
16. Nasr AA, El-Bahnasawy NA, El-Sayed A (2021) Straight-line: a new global path planning algorithm for Mobile Robot. In: 2021 international conference on electronic Engineering (ICEEM), pp. 1–5. IEEE, 2021
17. Soleimanpour-moghadam M, Nezamabadi-pour H (2021) A multi-robot task allocation algorithm based on universal gravity rules. Int J Intell Robot Appl 5(1):49–64
18. Amer DA, Attiya G, Zeidan I, Nasr AA (2021) Employment of Task Scheduling based on water wave optimization in multi robot system. In: 2021 International conference on electronic engineering (ICEEM), pp. 1–6. IEEE, 2021
19. Lee ML, Behdad S, Liang X, Zheng M (2022) Task allocation and planning for product disassembly with human–robot collaboration. Robot Comput Integr Manuf 17(6):102306
20. Park B, Kang C, Choi J (2022) Cooperative multi-robot task allocation with reinforcement learning. Appl Sci 12(1):272
21. Ghassemi P, Chowdhury S (2018) Decentralized task allocation in multi-robot systems via bipartite graph matching augmented with fuzzy clustering. In: International design engineering technical conferences and computers and information in engineering conference, vol. 51753, p. V02AT03A014. American Society of Mechanical Engineers
22. Sinaga KP, Yang M-S (2020) Unsupervised K-means clustering algorithm. IEEE Access 8:80716–80727
23. Raya L, Saud SN, Shariff SH, Abu Bakar KN (2020) Exploring the performance of the improved nearest-neighbor algorithms for solving the euclidean travelling salesman problem. Adv Nat Appl Sci 14:10–19
24. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
25. Amer DA, Attiya G, Zeidan I, Nasr AA (2021) Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. J Supercomput 78:2793–2818

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.