Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 678210, 20 pages
http://dx.doi.org/10.1155/2014/678210

Research Article

Cooperative Behaviours with Swarm Intelligence in Multirobot Systems for Safety Inspections in Underground Terrains

Chika Yinka-Banjo,[1] Isaac O. Osunmakinde,[2] and Antoine Bagula[3]

[1] *Department of Computer Science, Faculty of Science, University of Cape Town, Private Bag X3, Rondebosch, Cape Town 7701, South Africa*
[2] *School of Computing, College of Science, Engineering and Technology, University of South Africa (UNISA), P.O. Box 392, Pretoria 0003, South Africa*
[3] *Department of Computer Science, Faculty of Science, University of Western Cape, Private Bag X17, Bellville 7535, South Africa*

Correspondence should be addressed to Chika Yinka-Banjo; chikagog@gmail.com

Received 6 February 2014; Revised 16 May 2014; Accepted 26 May 2014; Published 20 July 2014

Academic Editor: Leo Chen

Underground mining operations are carried out in hazardous environments. To prevent disasters from occurring, as often as they do in underground mines, and to prevent safety routine checkers from disasters during safety inspection checks, multirobots are suggested to do the job of safety inspection rather than human beings and single robots. Multirobots are preferred because the inspection task will be done in the minimum amount of time. This paper proposes a cooperative behaviour for a multirobot system (MRS) to achieve a preentry safety inspection in underground terrains. A hybrid QLACS swarm intelligent model based on Q-Learning (QL) and the Ant Colony System (ACS) was proposed to achieve this cooperative behaviour in MRS. The intelligent model was developed by harnessing the strengths of both QL and ACS algorithms. The ACS optimizes the routes used for each robot while the QL algorithm enhances the cooperation between the autonomous robots. A description of a communicating variation within the QLACS model for cooperative behavioural purposes is presented. The performance of the algorithms in terms of without communication, with communication, computation time, path costs, and the number of robots used was evaluated by using a simulation approach. Simulation results show achieved cooperative behaviour between robots.

1. Introduction

Multirobot systems share the need to cooperate, creating the problem of modelling behaviour. When dealing with multiple robots, with randomness involved, the dynamic and unpredicted nature of the environment has to be considered. Hence, the behavioural modelling system has to cope with the random (dynamic and unpredictable) nature of the system. Researchers, on the other hand, have been captivated by this cooperative and coordinated problem of multirobot systems (MRS) in recent times. A list of literature on multiple robots' cooperation implemented in space was reviewed in [1]. Using multiple robots to achieve tasks has been more effective than using a single robot. See for instance [2, 3] (and all references therein) for some specific robotic tasks.

For a full discussion of underground mines, benefits, and disaster rates, see [4]. Safety is a major element in the underground mine; despite a significant reduction through safety research measures, the number of disasters in underground mines in South Africa and the world at large remains high [5]. To contribute to underground mine safety, that is, to prevent disasters from occurring, interacting autonomous multirobots are sent before mine operations resume to inspect how safe the underground mine is. This is achieved by assessing the status of the rocks, roofs, and level of toxic gases.

A multirobot planning algorithm for tunnel and corridor environments is implemented in [6]. The overall problem formulation is implemented using a topological graph and spanning representation. Activities, such as a single robot

drive and differential robots drive that can rotate in place, are carried out at different positions of the graph. Different methods have been used to tackle coordination of MRS in different domains [7]. Complete task coordination by multirobots was handled [3, 8]. An extension of a market-based approach was used. This was achieved by generalizing task descriptions into tasks trees, thereby allowing tasks to be traded in a market setting at a variable level of abstraction. Figure 1 shows an overview of the inspection environment.

We consider a scenario in which an MRS cooperates to achieve a common goal of safety inspection in a partially observable environment such as the underground terrain. Each robot requires to be guided using the proposed cooperative behavioural model. Finding a plan to achieve this cooperative model often involves solving an NP-hard problem. This is so because it involves multiple interacting robots. The interaction which comes as a result of minimizing time involved in achieving underground inspection requires reasoning among the robots. In this case, we use the QL technique to cleverly achieve the reasoning aspect of this work and combine it with optimal route finding using ACS. The two major questions answered in this problem are the following: (1) which state/room should I inspect now without repetition and a collision with another robot? (2) How do I get there using the closest link? However, all robots have partial knowledge of the environment and they are all equipped with the necessary sensors required for the inspection. The major contributions of this paper are as follows:

(i) development of a hybrid QLACS swarm intelligent model based on Q-Learning and the ant colony system for addressing cooperative behaviours in MRS achieving underground terrain safety inspections;

(ii) detailed experimental evaluations of the proposed QLACS model conducted on a simulated publicly available underground tunnel. Also, the proposed model is benchmarked with related methods;

(iii) systematic description of worked scenarios on an optimal route finder and MRS cooperative inspection using QLACS for ease of implementation by system engineers, robotics researchers, and practitioners.

We do not know of any study that analytically describes how a swarm intelligent model can easily be implemented and applied to a multirobot system in a heterogeneous environment. In the next section, we discuss related work in the area of multirobot systems for underground terrains' safety, reinforcement learning (RL), and ant colony optimization (ACO). We then detail our proposed cooperative behaviour framework and approach. The experimental results and evaluations of the safety inspections follow. Finally, we conclude by summarizing and discussing other future extensions.

2. Theoretical Background

In this section, we review some of the related work to MRS for underground terrains safety, which is the foundational domain for our research. A look at RL algorithms and its paradigms follows in Section 2.1. ACO and its paradigm applied in MRS conclude Section 2.

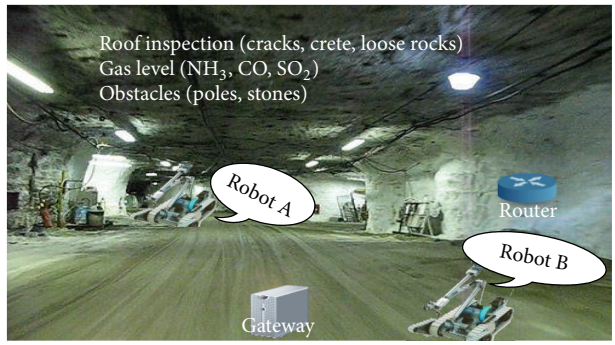


Figure 1: Overview of MRS inspection in an underground mine.

2.1. Related Multirobot Systems for Underground Terrains Safety. Effective, exhaustive, quick, and safe navigation of an MRS is subject to its ability to perceive, interpret, and interact (cooperate) within their environment. MRS can be used to achieve different tasks autonomously in structured and unstructured environments. In these environments, the automation of the operations is effected in different areas of MRS research: biologically inspired robot teams, communication, architectures and task planning, localization and mapping, object transportation and manipulation, learning, and so forth [9]. However, an environment such as the underground terrain has been a grey application domain in MRS research.

As indicated in Section 1, Peasgood et al. [6] implemented a multiphase algorithm for multirobot planning in tunnel environments. The algorithm as presented assumed a centralized planning architecture. They further compared the multirobot planning with a sequential planner. Their future work was to consider a decentralized planner architecture and might explore hybridizing the two planning algorithms.

Thorp and Durrant-Whyte discussed a starter on field robots [10]. From their discussion, field robotics involves the automation of platforms such as air, sea, and land in harsh unstructured environments such as underwater exploration, mining, agriculture, and highways. Field robots are made up of three parts: navigation and sensing, planning and control, and safety. Their work also discussed the challenges and progress of field robots. One of the major challenges of field robots in harsh environments such as an underground mine is the problem of position determination (localization). This is so because the global positioning system (GPS) can only help in an environment where the sky views are guaranteed. However, progress has been made in automating some of the environments that are cooperatively constrained.

The proposed model presented in [4] promised to handle the safety part of field robots presented by Thorp and Durrant-Whyte [10] in underground mines. Their model architecture had three layers; the first layer handles the cooperative behaviour of the model, the second layer deals with the scalability degree of the model, and the last layer

Steps:
initialize $Q(s, a)$ *arbitrarily*
repeat (for each episode):
initialize s
Repeat (for each step of episode):
Choose a *from* s *using policy derived from* Q
Take action a*, observe* s, s'

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s,a)\right]$$

$s \leftarrow s'$
until s *is terminal*

Algorithm 1: Q-learning algorithm.

handles the applicability of the model. This paper is building on what has already been proposed in [4].

An investigation into automating the underground mine environment after blasting, called "making safe," was carried out in [11] to ensure the safety of the environment after blasting. Blasting in an underground mine is the controlled use of explosives to excavate, break down, or remove rock. The need to investigate the stability of the environment after blasting before any mining operation takes place is of the highest priority, hence, the reason for automation. The automation was centred on a persistent area of concern in South African underground mine operation called hanging walls which is caused as a result of rock burst and fall of ground. There are also other persistence areas such as the levels of toxic gases which pose great disaster threats to the lives of miners, for instance, heat sicknesses, explosions, pneumoconiosis (occupational safety and health-fall of ground management in South Africa, SAMRASS-code book for mines), and so forth. Some of these disasters might result in fatalities and/or disabilities. Again, when an accident happens during mining operations, rescuers find it difficult to respond immediately to accidents. Looking at the aforementioned concerns, there will be a need to create models for safety inspection of underground mine operations. For instance, monitoring the underground mine environment for detecting hazardous gases and/or smoke should be one of the important safety measures. Continuous monitoring of workers and equipment is another crucial safety measure [5]. Picking up the sound from roof cracking to monitor when a roof is about to fall is also a safety item.

2.2. Reinforcement Learning. For a robot to operate in a harsh unstructured environment, considering every possible event in defining its behaviour is intricate [12]. It is, however, essential to develop robots that can conform to changes in their environment. RL is one of the artificial intelligence (AI) algorithms that can achieve learning by experience. This enhances robots' interaction with the environment. We investigate the use of RL to assist the behaviours of an MRS in safety inspection of underground mines. RL is a sequence of actions and state transitions with some associated rewards. What is being learned in RL is an optimal policy (what is the right thing to do in any of these states (s)) [13].

At any time step each robot is in a specific state in relation to the environment and can take one of the following actions: inspect, ignore, or shutdown. Each robot receives feedback after performing an action, which explains the impact of the action in relation to achieving the goal. The effect of the action can be either a good or bad reward. This reward is measured in terms of values. Therefore, the value of taking an action a in any state s of the underground terrain is measured using the *Action-Value function* called Q-value $Q^{\pi}(s, a)$. When a robot is starting from state s, taking action a, and using a policy pi(π), an expected return which is defined as the sum of the discounted rewards is achieved.

In this research, Q-learning, a method of RL, is explored. The purpose of RL methods is to study $Q^{\pi}(s, a)$ values so as to achieve optimal actions in the states. QL is an online RL method that requires no model for its application and stores the reinforcement values outcome in a look-up table. The QL architecture used in this work consists of learning threads, which amount to the number of robots involved in the inspection behavioural task. Each robot in the learning thread carries out Q-learning in the environment. Algorithm 1 explains the QL algorithm used in the behavioural model.

α is the learning rate set between 0 and 1. At 0, Q-values are never updated, hence nothing is learned; learning can occur quickly at 1. γ is the discount rate set between 0 and 1 as well. This models the fact that the future rewards are worth less than the immediate rewards. $\max_{a'}$ is the maximum reward that is attainable in the state following the current state. That means the reward for taking the optimal action thereafter.

QL is a competitive and search-based algorithm inspired by computational theory. It is not necessarily a multiagent algorithm but can be adapted to a multiagent or multigoal scenario. The success of this algorithm relies on the value and policy iterations, which can be adjusted by some unfairness (heuristics) to fit the current problem scenario. The most competitive action is selected by its value and action leads to another state or condition. Both value and policy are updated after each decision. Harnessing QL for an MRS scenario increases the cost exponentially and the overall performance drops in the same direction. As the robots increase cost, such as completion time, memory usage, and awareness factor

Table 1: Variant of ACO.

S/N	Year	Algorithm
1	1991	Ant system (AS)
2	1992	Elitist A.S
3	1995	Ant-Q
4	1996	Ant colony system
5	1996	Max-Min A.S (MMAS)
6	1997	Ranked based A.S
7	1999	ANTS
8	2000	BWAS
9	2001	Hypercube A.S

(other robots in the environment), search time increases. However, following our heuristic model of QL which was mainly determined by the policy we set to achieve our goal, our QL performs well above traditional QL.

2.3. Ant Colony Optimization. ACO is a type of swarm intelligence (SI). Bonabeau et al. [14] defined SI as any attempt to design algorithms or distributed problem-solving devices inspired by collective behaviour of social insect colonies and other animal societies. This implies that anytime something is inspired by swarms, it is called swarm intelligence. Researchers have been thrilled by swarms because of some of their fascinating features. For instance, the coordinated manner in which insect colonies work, notwithstanding having no single member of the swarm in control, the coordinated ways in which termites build giant structures and how the flocks move as one body, and the harmonized ways in which ants quickly and efficiently search for food can only be attributed to an emergent phenomenon [15, 16].

Ants, an example of a social insect colony, achieve their self-organizing, robust, and flexible nature not by central control but by stigmergy. Stigmergy, also known as a pheromone trail, describes the indirect communication that exists within the ant's colony. The indirect communication which is triggered by pheromone trails helps in recruitment of more ants to the system. Ants also use pheromones to find the shortest paths to food sources. Pheromone evaporation prevents stagnation, which also helps to avoid premature convergence on a less than optimal path [17].

ACO is necessarily a multiagent based algorithm. The first implementation of this optimization algorithm is in a classical search based (combinatory) problem, the travelling salesman problem, giving the shortest path to a specified destination. After this feat, many researchers have used it or its variants to model different problems. In this work, a variant of ACO is used to find the optimal path for MRS. Table 1 describes variants of ACO and their meaning [18]; see also [19] and all references therein.

In AS, the pheromones are updated by all the ants that complete a tour. ACS is the modified version of AS, which introduces the pseudorandom proportional rule. In elitist AS, ants that belong to the edges of the global best tour get an additional amount of pheromone during the pheromone update. MMAS introduces an upper and lower bound to the values of the pheromones trails. All the solutions are ranked to conform to the ants' length in ranked-based AS [20].

Our interest is in implementing ACS to find the best possible round trip inspection path in our model environment. This best possible inspection path will be supplied as input or made available for the robots using QL for inspection decisions. In most of the route finding scenarios, the nodes are linearly joined and are not largely distributed. This is a scenario where ants can forage along at least two paths (multigoal path environment, though in our case there is a path exposed to four different goals; see Section 3).

3. Proposed Cooperative MRS Behaviour Framework

The following are some of the factors we are considering in the course of building the model. Each robot has to learn to adjust to the unknown underground mine environment. In this case, each robot requires intelligence and a suitable machine learning algorithm is adopted. No robot has global information of the environment because of its limited sensing capabilities. Each robot has limited communication capabilities; therefore, each robot has to keep track of the information of others to remain in a team. Figure 2 describes the proposed framework as an approach for building the distributed, coordinated inspecting model for MRS.

The framework indicates three layers of the model: the bottom layer, the middle layer, and the topmost layer. The learning capability of the MRS is achieved in the bottom layer, with the reinforcement learning algorithm and swarm intelligence technique. This intelligence enables robot A to take action knowing the action of robot B and vice versa. At the middle layer, scalability in terms of the number of robots the system can accommodate is achieved. This is expedient because a team of robots tends to achieve tasks more quickly and effectively than single robots. This scalability is handled with some memory management techniques, as indicated in Figure 2. The real life implementation is achieved by using the information acquired from the topmost layer. Figure 3 is the breakdown of the framework in Figure 2.

There is a base-station or server that serves as a backup for the information captured and analysed from individual robots. The model proposed in this research deals with the way in which robots need to find their way within communication range and uses a broadcast approach for effective communication of navigation status. A team of robots cooperatively inspecting an area in the underground mine will need to know where they are and where to go next, so it is obviously a continuing problem and our contribution in this case is that before robot R1 takes an action, it broadcasts its location and inspection status to other robots, R2, R3, and so forth, and vice versa. An unreachable robot receives packets of information based on the destination address through rerouting from the nearer robots. The reliability of this broadcast method is the ability to determine the extent of the task executed already by looking at the memory of any leading robot in the team.

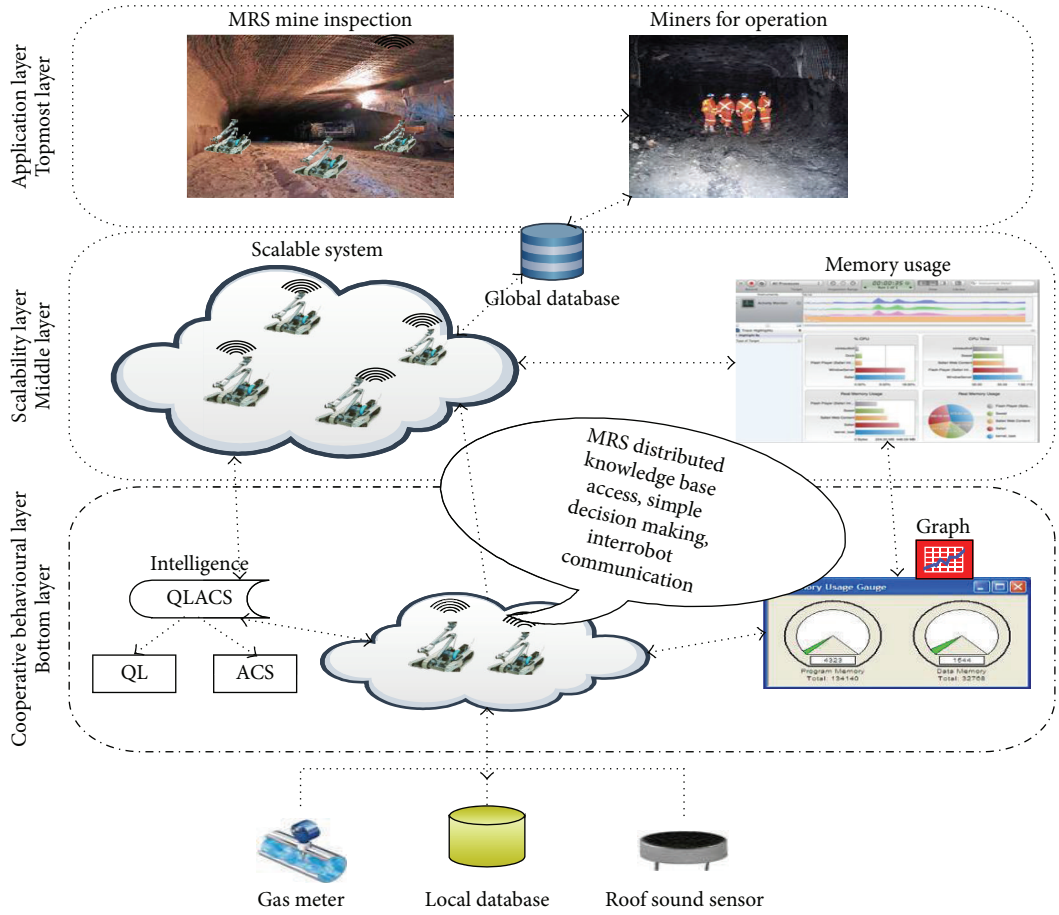


Figure 2: Framework of the proposed QLACS model for cooperative behaviours.

Table 2: State and possible actions of the environment.

	State	Possible Actions
1	*A* (lower left part (LLP))	Inspect, ignore
2	*B* (lower middle part (LMP))	Inspect, ignore
3	*C* (lower right part (LRP))	Inspect, ignore
4	*D* (middle left part (MLP))	Inspect, ignore
5	*E* (central part (MCP))	Inspect, ignore
6	*F* (upper left part (ULP))	Inspect, ignore
7	*G* (upper right part (URP))	Inspect, ignore
8	*H* (outside mine part (OMP))	Shutdown

3.1. Problem Formulations. Suppose we have seven rooms/states connected by doors/links representing underground mine regions as shown in Figure 4 and labeled as shown in Table 2. We label each room *A* through *F*. The outside of the mine can be thought of as one big room (*H*). Notice that doors *F* and *C* lead outside the mine *H*. We put two robots in rooms *F* and *C* as their starting states, respectively. The robots inspect one state at a time, considering the obstacles encountered in the process. The robots can change direction freely, having links/doors to other rooms/states. In each of the states in Figure 4 two actions are possible: inspection of roof cracks (RC) and level of toxic gases (TG), or ignoring the state, as it has been inspected earlier. According to our proposed model, a robot can inspect at least half of the given underground mine region to attain its maximum performance, which in turn attracts a good reward. When the current state of a robot is the end point of the inspection task, the robots exit the mine using the exit points *C* and *F*, respectively. The possible transition states of the robots are displayed using the state diagram in Figure 5. The state diagram and the transition matrix are formed using the QL algorithm.

The global map is assumed to be known by the robots but there is no prior knowledge of the local map. Robots only know what they have sensed themselves and what their teammates communicate to them. Not only does our improved QL depend on the map of the environment but also each robot learns through experience about local changes. They explore and inspect from state to state until they get to their goal states. Our proposed model QLACS achieves an offline mode for the route finding algorithm (ACS). This means that the global view of the map would have been provided before the learning robots start.

The model is explained using simulations; the results are presented in Section 4. Some of the results also have graphical interpretations. Analysis by a state transition diagram is

Table 3: Initial reward matrix.

		Robot's action							
		A	*B*	*C*	*D*	*E*	*F*	*G*	*H*
Robot's state	*A*	—	50, 100	—	50, 100	—	—	—	—
	B	50, 100	—	50, 100	—	50, 100	—	—	—
	C	—	50, 100	—	—	—	—	50, 100	150
	D	50, 100	—	—	—	50, 100	50, 100	—	—
	E	—	50, 100	—	50, 100	—	50, 100	50, 100	—
	F	—	—	—	50, 100	50, 100	—	50, 100	150
	G	—	—	50, 100	—	50, 100	50, 100	—	—
	H	—	—	50, 100	—	—	50, 100	—	—

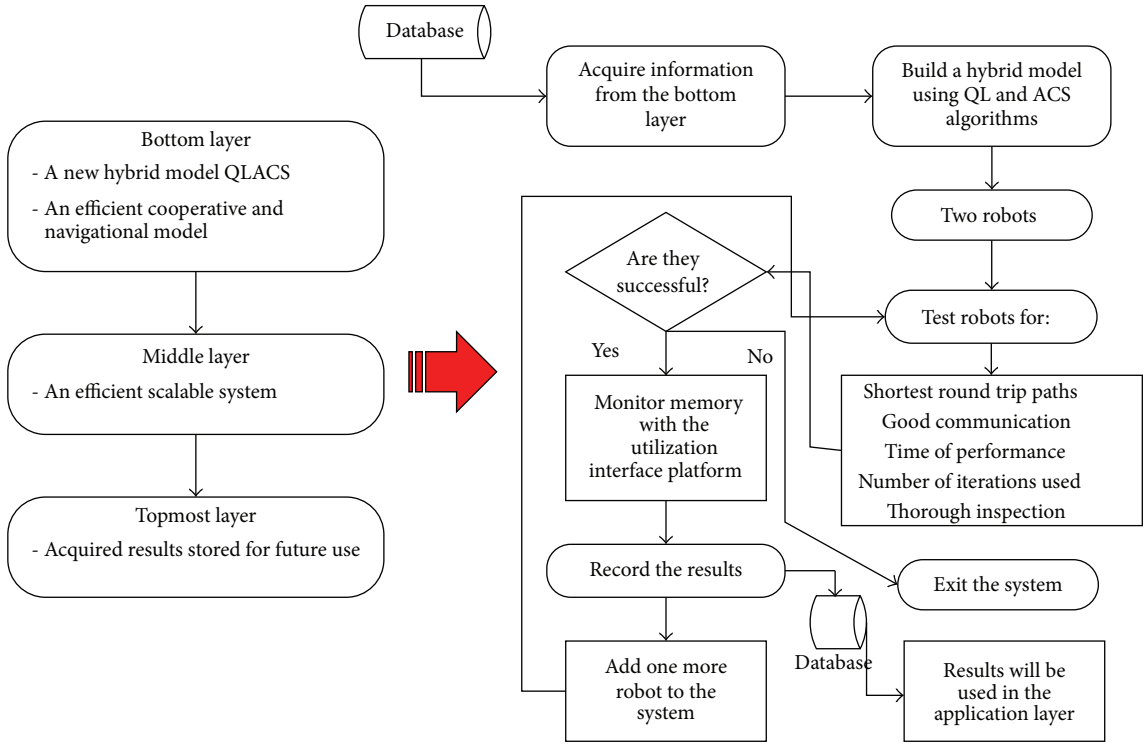


Figure 3: Breakdown of the framework. (a) Contributions of the framework, Layer by layer, and (b) processes of the multirobot behavioural system.

presented in Figure 5. The possible state actions of the robots are presented in Table 3. The states of the robots are reduced as follows: searching or inspecting for roof cracks and toxic gas levels in each state or room and recording the outcome of the inspection in the robots' memories. The processes involved in achieving good communication while the robots inspect the states are broadcasting inspected states to the other robots and ignoring the inspected/broadcasted state, avoiding collision with obstacles and other robots and finally moving to the next available state that has not been inspected.

The Q-learning algorithm is used to determine what action is to be selected in any particular state. Let the action (a) = inspect, ignore, shutdown, state space (s) = dimensions in the topological map, sensor readings (s_r), hazardous conditions (H_c) = roof crack (RC), toxic gas level (TG). Each robot is configured with an aerial scanner and chemical sensitive sensor that will provide readings (s_r), to determine if there is a hazardous condition, H_c, in the environment. The selection of an action by a robot through the Q-learning algorithm is based on the information from the broadcast. The whole framework uses a decentralized Q-learning scheme such that each robot has its own thread with its Q-learning and table. All Q-values on the tables of the robots in the team are initialized to zero; that is, states corresponding to such positions have not been inspected. When a robot enters a new state it broadcast its information for the current state to all other robots in the environment. The broadcast indicates whether the current state has been

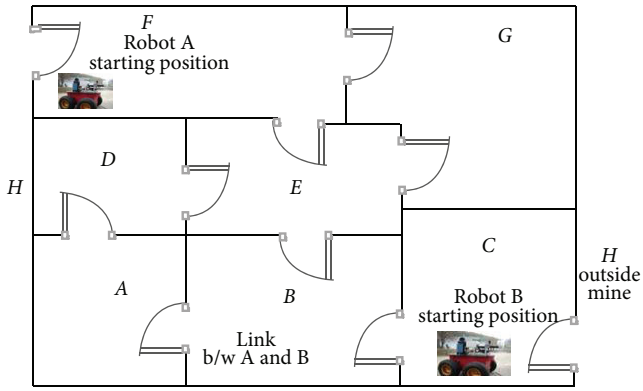


Figure 4: Model of the environment with two entrances and exits (2EE).

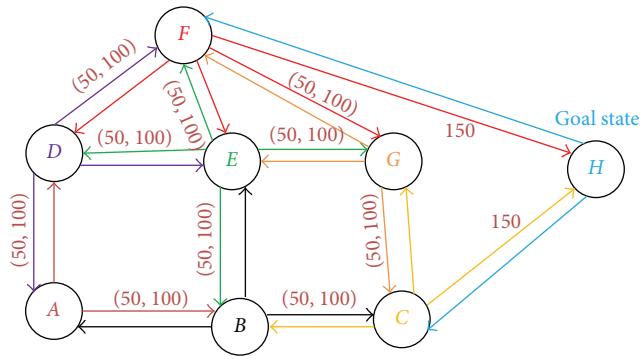


Figure 5: Events and transition diagram of the modeled environment with H as goal state.

inspected or not. The broadcast is a search of corresponding positions in the memories of all visible robots. If the resultant search returns a zero; the broadcast indicates that the state has not been inspected and if it returns a value greater than zero it indicates that the state is not available for inspection. All robots must receive a broadcast before they act in any particular state. When a robot gets to a state, it receives a broadcast and passes its value to the QL algorithm to make a decision. The robot carries out the decision of the QL algorithm. The policy of the Q-learning algorithm makes a robot carry out an inspection if the resultant search of the broadcast returns zero and ignores it if the resultant search is greater than zero. A robot only shuts down if all states have been visited and inspected. As the broadcast information is passed to the Q-learning algorithm, the policy is iterated towards an optimal value and condition.

The broadcast avoids the cost of multiple inspections and the QL ensures that robots take appropriate actions. The only state in which inspection is carried out would have sensor readings (s_r), indicating H_c. For taking an action (a) in state (s), the robot gets a reward (R), which is used in (10) to compute the Q-value for the current state and can send a broadcast to other robots. Figure 5 and Table 3 show the restrictions and possible transitions among node points, indicating the possible rewards for any particular action (a) in any state (s). Every other transition besides the goal state could result in a reward of 50 or 100 and at completion the reward is the maximum, 150. We consider it wasteful to make the robots scan first before sharing intelligence because such action would slow them down and make them expend more energy. Robots are to provide the inspection results, showing actions taken in different states and the nature of conditions detected in any particular state. The introduction of the broadcast approach to determine the team's exploration reduces the execution time and energy cost of the whole team and makes the collaboration effective. So in a multirobot scenario the task can be effectively executed if the robots are made to share intelligence as they progress. Robots do not have to waste time and energy in doing the same task already carried out by other robots in the team.

GIVEN. On a mathematical justification of the above, suppose a safety preinspection of toxic gases or rock fall or some combination of the two is being carried out on a piece of complex underground terrain in Figure 1, say $L\,\mathrm{Km}^2$; there is a limited number of MRS with different capacities, R, and precious inspection time, T minutes. Every region/state x_1 in the terrain requires a capacity of robot R_1 of MRS and limited time T_1 while every state x_n requires robot R_n of MRS and inspection time, T_n. Let P_1 be the positive reward of QL for correct behaviour on state x_1 and let P_n be the reward on state x_n. This research aims to maximise positive rewards by choosing optimal values for states $x_1, \ldots, x_n$ as follows:

Maximise:

$$P_1 \cdot x_1 + P_2 \cdot x_2 + \cdots + P_n \cdot x_n$$

(objective function)

Subject to constraints:

$$x_1 + x_2 + \cdots + x_n \le L$$

(limited total states)

$$R_1 \cdot x_1 + R_2 \cdot x_2 + \cdots + R_n \cdot x_n \le R \tag{1}$$

(limited MRS capacity)

$$T_1 \cdot x_1 + T_2 \cdot x_2 + \cdots + T_n \cdot x_n \le T$$

(limited inspection time)

Non-negativity constraints:

$$x_1 \ge 0, \quad x_2 \ge 0, \ldots, x_n \ge 0$$

(MRS cannot inspect negative states).

In terms of solving this optimization problem, we use the proposed QLACS model to compare the time and number of states inspected. The number of robots used is also compared. The graphs results in Section 4 also give more insight into the solution of the problem.

3.2. Basic Navigation and Cooperative Behaviours Using QLACS. QLACS has two components. The first component is formed by an improved ACS and the second component

is formed by an improved QL. The improvement occurs because some heuristics were added to the ordinary QL and ACS to achieve the hybrid QLACS. However, the second component of QLACS, which is an improved QL, was initially used to solve the proposed problem. After much analysis, we realized that the system needs to be optimized for effective cooperation and communication.

Using the second component of QLACS to solve the basic navigation and cooperative behaviour, the possible actions were set for each robot as inspect, ignore, and shutdown (after reaching the goal state H). Also, a reward system that will reflect the possible actions of the robots was chosen. In other words, a robot gets 150 points only when the goal is achieved (shutdown), 100 points for ignoring an already inspected area (ignore), and 50 points for inspecting an uninspected area (inspect). Figure 5 shows the transition events of the model and Table 3 displays the possible state action for each robot. The way the QLACS second component works here is based on both navigation and communication behaviours.

Achieving navigational behaviour with the second component of QLACS has some cost associated to it. In our scenario, because we want the robots to share intelligence by broadcasting results (robots search through other robots' memory), our problem is not solely navigational but also cooperative. Our behavioural actions are inspect, ignore, and shutdown. We noted that these actions of interest are not navigation oriented; there is no way we could use them in making decisions on transition. The functions for decision can be integrated to assist the other. Therefore, our behavioural model is an integration of two behaviours: (1) navigational behaviour and (2) cooperating behaviour through decision making. The integration works with a route finding method called *RandomStateSelector,* which we introduced in the second component of QLACS to help determine where the robot goes from the starting point to the exit point. Two parts of the *RandomStateSelector* method are introduced in this work. The first one is the *RandomStateSelector_C_H,* which is used to transit from state C to H and the second one, *RandomStateSelector_F_H,* transits from state F to H. This method works but not effectively because some of the states are repeated several times because of the random selection method. However, the decision part of this second component of QLACS, which is handled by a method called *CheckInspection,* worked efficiently. *CheckInspection* is responsible for sharing the broadcast among the agents. The broadcast looks at all the stored Q-values on all the robots and returns a signal for the action that needs to be taken. Therefore, we concluded that the heuristically accelerated component of QLACS has proven to be effective by showing evidence of effective communication in making inspection decisions using our model. It did not guarantee shortest possible time for inspection because of repeated states decisions. In this light, we only harnessed the communication strength of the second component of QLACS for communication and cooperation. Figure 5 and Table 3 form the basis for the QLACS second component.

To take care of the random state selector problem encountered in implementing the algorithm used for the second part of QLACS, we introduced an optimized route finder algorithm. This route finder algorithm, which forms the first component of QLACS, is a swarm intelligence technique. Figure 6 and Table 4 form the basis of the exploration space of the agents (ants) which achieved the optimum route finding. The weighted graph in Figure 6 is based on the number of obstacles the ants will encounter from the points of entry F and C. The combined table in Table 4 contains the weights of the obstacles and evidence of an edge between any two vertices (states). It shows that there is a connection between any two vertices in a graph. Generally, a "1" or "2" depicts the existence of an edge while a "0" represents the absence of an edge, that is, no transition between such vertices. The constructed graph is an undirected multigraph, providing evidence of some agents coming from F or C of the mine (logical space). It is unidirectional because agents can move in any particular direction (multigraph). This means that the same weights on the edges apply to both directions. The graph does not show H; we assume that once the agents reach F or C, they exit if all inspections have been done.

Table 4: Combined adjacency and weight matrix.

	F	G	E	D	A	B	C	H
F	0	1	1	1	0	0	0	1
G	1	0	2	0	0	0	1	0
E	1	2	0	2	0	2	0	0
D	1	0	2	0	2	0	0	0
A	0	0	0	2	0	2	0	0
B	0	0	2	0	2	0	1	0
C	0	1	0	0	0	1	0	1
H	1	0	0	0	0	0	1	0

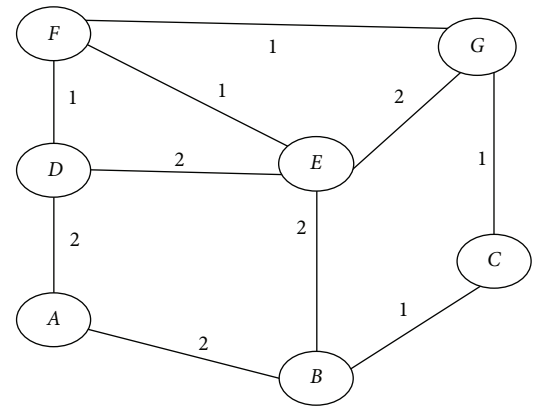


Figure 6: Weighted map/graph of the model environment.

3.3. Hybrid Model Development . The parameters for building the analytical hybrid QLACS (equations (2) through (11)) model are presented in Tables 5 and 6.

3.3.1. Algorithmic and Mathematical Analysis

ACS Starts. Computation of edge attractiveness

$$\eta_{i,j} = \frac{1}{D_{i,j}}. \tag{2}$$

Table 5: A list of parameters for the ACS model.

ACO parameters	Meaning
α	Pheromone influence factor
β	Influence of adjacent node distance
ρ	Pheromone evaporation coefficient
Q	Attractiveness constant
e	Visited edge
e'	Edge not visited
L_k	Length tour of ant k
τ	Pheromone concentration (amount)
η	Specific visibility function (attractiveness)
$\Delta\tau^k$	Pheromone concentration by Kth ant
$P_r(i, j)$	Probability of moving from i to j
$D_{i,j}$	Visibility or distance between i and j
f_i	Fitness of individual in a population
P_i	Probability of being selected among f_i
N	Number of individuals in the population
i, j	Denotes any two adjacent nodes in the graph
M_k	Set of unvisited nodes

Table 6: A list of parameters for the QL model.

QL Parameters	Meaning
Q	Q-value update
s	State
a	Action
R	Reward
γ	Learning rate

Computation of instantaneous pheromone by ant k

$$\Delta\tau^k = \frac{Q}{L_k}. \tag{3}$$

Update of pheromone

$$\tau_{i,j} = (1-\rho) * \tau_{i,j} + \Delta\tau^k_{i,j}. \tag{4}$$

Computation of edge probability

$$P_r(i, j) = \frac{[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}{\Sigma_{e'=(i,j)} [\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}. \tag{5}$$

Adoption of roulette wheel

$$\text{Cumulative}\left(P_r(i, j)\right) = \sum_{i=1}^{N+1} P_r(i, j), \tag{6}$$

$$f_i = \frac{\sum_{j=1}^{N} f_j}{N}, \tag{7}$$

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j}. \tag{8}$$

Equations (2) through (8) build the complete route finding model. Equations (2) through (4) are prerequisite to (5). Equation (5) is prerequisite to roulette wheel selection. At the end of (8), new states are selected and the trail is updated. The best path from both directions is selected and used as input in Q-learning. Note that $D_{i,j}$ = weight on edges, $P_r(i, j)$ = chance of moving to a node $E(I, j)$, L_k = sum of visited nodes by ant k.

QL Starts. Each robot in its QL thread computes its learning rate:

$$\gamma = \frac{0.5}{\left[1 + \text{Frequency}(s, a)\right]}. \tag{9}$$

Q-values are updated:

$$Q(s, a) = R(s, a) + \gamma \tag{10}$$

making a broadcast (Decision = Inspect/Ignore/Shutdown)

$$Q(s, a) = \begin{cases} Q = 0 & \text{Inspect if } s_j \neq \text{goalstate} \\ Q > 0 & \text{Ignore if } s_j \neq \text{goalstate} \\ Q \geq 0 & \text{Shutdown if } s_j = \text{goalstate}. \end{cases} \tag{11}$$

Equation (9) is Gamma, the learning rate, which is always between zero and 1. This equation is calculated based on the frequency of action of each robot in inspecting states. Equations (9) to (11) are state-dependent. The states are kept in a buffer and then accessed at run time. ACS and QL do not work simultaneously. ACS works to completion and QL takes the final output as its input. ACS is not repeatedly called while QL is working.

Our behavioural model is an integration of two algorithms: (1) route finding algorithm (2) communication and cooperative algorithm. The integration works the following way. The optimal route finder (ACS) determines where the robot goes from the starting point to the destination, while QL determines what action it takes when it gets to any of the states. This collaboration works effectively because the optimal route finder has been proven to give the best possible transitions and the heuristically accelerated QL has proven to be effective by showing evidence of effective communication in making inspection decisions. Consequently, it guarantees the shortest possible time for inspection in the absence of wasteful inspection decisions. This framework forms the basis for our cooperative behaviour model for MRS (QLACS). The pseudocode for the implementation of QLACS is outlined in Algorithm 2.

3.3.2. QLACS Hybrid Approach Evolution. Figure 7 is the architecture of hybrid QLACS explaining the handshake between the two components.

(i) Graph Construct Module. This is the interconnection of states in our model environment (see Figure 4). It encompasses all possible transitions from F to C and vice versa. Thus from the model we construct an adjacency/weight graph matrix that can be traversed by any graph-oriented algorithm.

INPUT: Edge distance(obstacles), pheromones, ants' trail, associated probabilities, starting and terminating indexes that is, from *F* or *C*
OUTPUT: Effective cooperation, inspection and navigation

(1) ***Boolean*** CompletedFlag = ***False*** //Boolean variable indicates completion for all the threads
(2) ***Declare*** CompletedThreadBuffer //Data structure stores Info about completed thread
(3) Initialize all Qvalues to ***Zero*** //All Qvalues positions are initialized to zero
(4) Initialize Best Paths From ACO algorithm //starting from *F* and *C*
(5) ***While (CompletedFlag*** <> ***True)*** //Checks for when all robots have finished and flag is true
Begin
Create *N* number of Threads in Parallel
Threads get Current States in Parallel from ACO algorithm
Threads get Next States and Indexes in Parallel from ACO algorithm
Threads compare Qvalues of all corresponding positions of Current States (Each Robot Broadcast Qvalue info)
IF ((Q == 0) & (ThreadNextState <> ***GoalState))*** //Checks if a particulate state is available
Begin
State is Available, Robot with the CurrentThreadID Inspects
Compute and Update Qvalue
End
IF (Q > 0) //checks if a state is not available, because an already inspected state has $Q > 0$
Begin
State is already inspected, Robot with the CurrentThreadID Ignore
Compute and Update Qvalue
End
IF ((Q == 0) & (ThreadNextState == GoalState)) //Checks for goal state and shuts down.
Begin
Compute and Update Qvalue
Goal state is reached and Inspection Completed
Thread with the CurrentThreadID Shuts down
Store CurrentThreadID ***in CompletedThreadBuffer***
End
IF (Count [CompletedThreadBuffer] == NumberOfRobot) //Learning stops when this happens
Begin
CompletedFlag = True
End
End of While Loop.

Algorithm 2: Pseudocode for QLACS.

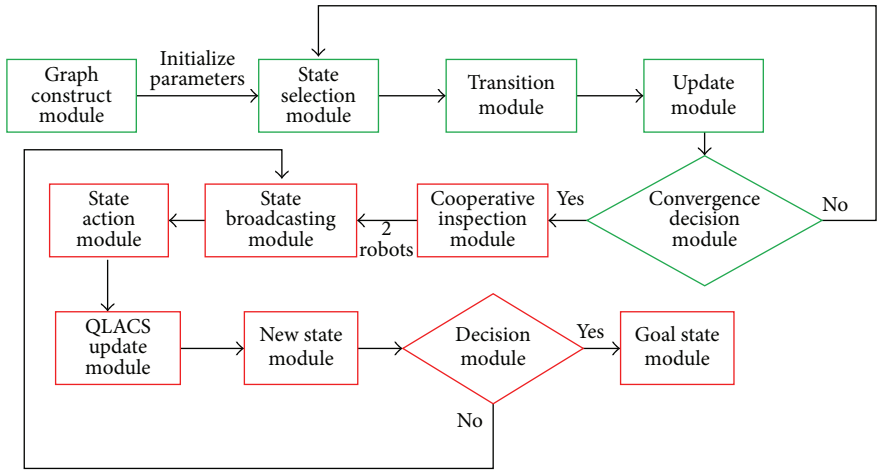


Figure 7: Hybrid architecture.

Example I of QLACS (Using Figure 5 for optimized route finding)
Parameters used in the first component of QLACS
$Q = 2.0, \alpha = 3, \beta = 2, \rho = 0.01$
Starting State: F
Terminating State: C/F
Terminating condition: When all states have been visited at least once and it is at terminating state
State Space = $\{F, G, E, D, A, B, C\}$
Initialize pheromones positions to 0.01
Rand = (0, 1) returns a random value between 0 and 1
Equations
(i) Computation of attractiveness $\eta_{i,j}$ using (2)
(ii) Computation of instantaneous pheromones by ant k for all potential states using (3)
(iii) Pheromone update using (4)
(iv) Computation of probabilities for the potential states using (5)
(v) Adaptation of roulette Wheel using (6)
Equation (5) can be reduced to Let $w(i, j) = [\tau_{i,j}]^{\alpha}[\eta_{i,j}]^{\beta}$

$$\text{Sum} = \sum_{i=1}^{N+!} w(i, j)$$

$$\text{So } P_r(i, j) = \frac{w}{\text{sum}}$$

Algorithm 3: Parameters for QLACS Example I.

In our case, there are eight states: primarily seven internal states and a common terminating state. Since the states are not just linear, the environment allows for multiple options; that is, an agent/ant can choose from any present state. This type of scenario is also called a multigoal scenario.

(ii) State Selection Module. Here, the agents select the start and end states, which according to our model in Figure 4 are F and C. These states are selected based on the cumulative probability of two adjacent nodes in (6).

(iii) Transition Module. This module takes care of the transition rules of the agents by calculating the pheromone concentrations, distances, and probabilities using (2) through (4).

(iv) Update Module. After transition from one state to another, an update of the pheromone is computed, after which multipath planning with shortest path is achieved.

(v) Convergence Decision Module. This is where the best trail/path decision is taken. This is the end product of the first component of QLACS, which is then moved to the second component of QLACS for cooperative behaviour.

(vi) Cooperative Inspection Module. This is where the robots are deployed to start inspection. The robots are to use the acquired best paths starting from F and C, respectively, as input for the second component of QLACS. The two robots are to use the learning rate from (9) to learn the environment and use (10) for cooperation.

(vii) State Broadcasting Module. This module handles the broadcasting behaviours of the two robots, which are achieved by using (10). Each robot checks its memory represented by Q-values before taking any decision.

(viii) State Action Module. State broadcasting by each robot is immediately followed by action selection. In other words, the state to inspect or ignore is achieved here using (11).

(ix) QLACS Update Module. After each action selection, the Q-values of each robot are updated using (10).

(x) New State Module. This module takes care of the robot's new state after updating the Q-values. Each robot runs in its own threads, managing its memory, yet sharing information.

(xi) Final Decision Module. This decision module determines if the robot should exit the environment or still do more inspections. It is also controlled by (11).

(xii) Goal State Module. The completion of the second component of QLACS is getting to the goal state after successful inspection of states. This goal state according to our model in Figure 4 is called H.

3.3.3. Analytical Scenario. For the benefit of robotic and system engineers, practitioners, and researchers, this paper uniquely presents scenarios on the ease of implementation of the proposed QLACS as described in Algorithms 3 and 4 and Tables 7 and 10. The first component of QLACS is explained in Example I, shown in Algorithm 3 and Table 7. The first component that achieved optimal route paths for the robots has the parameters listed in Algorithm 3. The calculation for different states transition for the first component of QLACS is analysed in Table 7.

Repeat steps 1–6 for subsequent current states until the termination condition and state are reached. At the end of seven updates we have Tables 8 and 9. The total length shown in Table 8 represents the total number of obstacles encountered by each agent while trailing to achieve the optimal route for the robots. The number of obstacle shown

Table 7: QLACS Example I navigational analytical solution.

Starting state: F
Potential states: D, E, G
(1) Use (2), $\eta_{F,D} = 1$, $\eta_{F,E} = 1$, $\eta_{F,G} = 1$
(2) Use (3), $L_k = 1$, $\Delta\tau^k(F,D) = 2/1 = 2$, $\Delta\tau^k(F,E) = 2$, $\Delta\tau^k(F,G) = 2$
(3) Use (4), $\tau_{F,D} = (1 - 0.01) * 0.01 + 2 = 2.0099 = 2.01$, $\tau_{F,E} = 2.01$, $\tau_{F,G} = 2.01$
First pheromone update

F	G	E	D	A	B	C
0.01	2.01	2.01	2.01	0.01	0.01	0.01

(4) Use (5) $w(F,D) = [\tau_{F,D}]^{\alpha}[\eta_{F,D}]^{\beta} = (2.01)^3 = 8.12$, $w(F,E) = 8.12$, $w(F,G) = 8.12$
Sum $= w(F,D) + w(F,E) + w(F,G) = 24.36$
$P_r(F,D) = \dfrac{w}{\text{sum}} = \dfrac{8.12}{24.36} = 0.33$, $P_r(F,E) = 0.33$, $P_r(F,G) = 0.33$
Probabilities

F	G	E	D	A	B	C
0	0.33	0.33	0.33	0	0	0

(5) Use (6)

H	F	G	E	D	A	B	C
0	0	0.33	0.33	0.33	0	0	0

Call Rand
Rand = 0.07, Rand falls in state E. Roulette wheel selects E as the next state, end of roulette wheel.
(6) Update trail: F, E

Current state: E
Potential states: B, D, G
(1) Use (2), $\eta_{E,B} = 1/2$, $\eta_{E,D} = 1/2$, $\eta_{E,G} = 1$
(2) Use (3), $L_k = (F - E - D) = 1 + 2 = 3$, $L_k = (F - E - B) = 1 + 2 = 3$, $L_k = (F - E - G) = 1 + 2 = 3$
$\Delta\tau^k = \dfrac{2}{3} = 0.67$
(3) Use (4), $\tau_{E,B} = (1 - 0.01) * 2.01 + 0.67 = 2.66$, $\tau_{E,D} = 2.66$, $\tau_{E,G} = 2.66$
Second pheromone update

F	G	E	D	A	B	C
0.01	2.66	2.01	2.66	0.01	2.66	0.01

(4) Use (5) $w(E,B) = (2.66)^3 * (1/2)^2 = 4.71$, $w(E,D) = 4.71$, $w(E,G) = (2.66)^3 * (1)^2 = 18.82$
Sum $= 4.71 + 4.71 + 18.82 = 28.24$
$P_r(E,B) = \dfrac{w}{\text{sum}} = \dfrac{4.71}{28.24} = 0.17$, $P_r(E,D) = \dfrac{4.71}{28.24} = 0.17$, $P_r(E,G) = \dfrac{18.82}{28.24} = 0.67$
Probabilities

F	G	E	D	A	B	C
0	0.67	0	0.17	0	0.17	0

(5) Use (6)

H	F	G	E	D	A	B	C
0	0	0.67	0	0.17	0	0.17	0

Call Rand
Rand = 0.9, Rand falls in state D. Roulette wheel selects D as the next state, end of roulette wheel.
(6) Update trail: F, E, D

Example II of QLACS (For good cooperation and communication between robots)
Parameters used in the second component of QLACS (Using output from the first component of QLACS)
Reward Scheme: Inspect = 50, Ignore = 100, Shutdown = 150
State space: Optimized path from QLACS_R1 = {*F*, *E*, *D*, *A*, *B*, *C*, *G*, *C*} and QLACS_R2 = {*C*, *B*, *A*, *D*, *F*, *E*, *G*, *C*}
Starting State: *F*/*C*
S_j = Terminating State: *C* then *H*
Terminating condition: When all states have been visited.
Initialize *Q*value positions to zeros
Equations
(i) Compute learning rate using (9)
(ii) Compute update on $Q(s, a)$ using (10)

Algorithm 4: Parameters for QLACS Example II.

Table 8: Pheromone update of a full cycle.

	Pheromone update							
Current states	*F*	*G*	*E*	*D*	*A*	*B*	*C*	
F	0.01	2.01	2.01	2.01	0.01	0.01	0.01	1st update
E	0.01	2.66	2.01	2.66	0.01	2.66	0.01	2nd update
D	3.13	0.01	3.03	0.01	3.03	0.01	0.01	3rd update
A	0.01	0.01	0.01	0.45	0.01	0.45	0.01	4th update
B	0.01	0.01	0.67	0.01	0.67	0.01	0.7	5th update
C	0.01	0.91	0.01	0.01	0.01	0.89	0.01	6th update
G	0.71	0.01	0.69	0.01	0.01	0.01	0.71	7th update

C = terminating state and ant *k* has moved through all states at least once.
Trail: *FEDABCGC*.
Number of obstacles = 1 + 2 + 2 + 2 + 1 + 1 + 1 + 1 = 10.

Table 9: Probability update of a full cycle.

	Probability table						
Current states	*F*	*G*	*E*	*D*	*A*	*B*	*C*
F	0	0.33	0.33	0.33	0	0	0
E	0	0.67	0	0.17	0	0.17	0
D	0.69	0	0.16	0	0.16	0	0
A	0	0	0	0.5	0	0.5	0
B	0	0	0.16	0	0.16	0	0.68
C	0	0.52	0	0	0	0.49	0
G	0.45	0	0	0	0	0	0.45

C = terminating state.

as 10 is calculated using the trail result in Table 8. Table 9 displays the probability update table for a full cycle of route finding. In the case of this first example, the first robot will terminate its inspection through *C* and then *H*.

Algorithm 4 displays the parameters of the second example of QLACS. The second component of QLACS handles this aspect of the model, which is the communication and cooperative part. Once the first component hands the output to the second component, it becomes the second component input and it runs with it. From Algorithm 4, QLACS_R1 represents the input for robot 1 and QLACS_R2 represents the input for robot 2 received from the first component of QLACS. The terminating condition is when all states have been visited.

The rest of Example II displayed in Table 10 explains the cooperative behavioural scenario from one state to another for two robots. The tables in the last row of Table 10 show the communication and cooperative output achieved using the second component of QLACS.

4. Experimental Evaluations: Safety Inspections for MRS Behaviours

The experimental results of implementing QLACS in an environment that consists of obstacles and links are tabulated in this section. The experimental setup is explained in Section 4.1. Different performance categories are shown in this section: without communication category and with communication category. In the without communication category, as displayed in Section 4.2, we found that robots can inspect all states individually without knowing that another robot exists. Robots can also inspect some of the states, thereby leaving some states not inspected. The communication category is explained in Section 4.3 while the performance of the QLACS measured with other existing methods is tabulated in Section 4.4.

4.1. Experimental Setup. Figure 8 displays different sets of experiments conducted in the environment using two robots. Figure 8(a) shows how two robots resume inspection from two different entrances. In each set of experiments, the robots take the readings of the roof cracks and level of toxic gases using their sensors. The same behaviours happen in Figures 8(b) and 8(c), respectively, at different inspection locations. The inspection locations vary in the four experimental setups shown in Figure 8.

4.2. Experiment 1: Performance of QLACS without Cooperation. The result of implementing the QLACS without communication in the proposed environment (Figures 1 and 4) is shown in Tables 11 and 12. In the case of Table 11, robot 1 (R1), enters the mine through state *F* while robot 2 (R2) enters the mine through state *C*. However, each robot learns by inspecting some of the states and ignoring some of the

Table 10: QLACS Example II cooperative behaviour.

Starting simulation Robot 1 **Starting state: *F*** (1) Use (9) $\gamma(F) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Check the Q-value for state F (use (11)) $Q(F, a) = 0$ Selected action, a = inspect (3) use (10) $Q(F, a) = 50 + 0.25(0) = 50$ End of value iteration	**Current state: *E*, Robot 1** (1) Use (9) $\gamma(E) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Check the Q-value for state E (Use (11)) $Q(E, a) = 0$ Selected action, a = inspect (3) Use (10) $Q(E, a) = 50 + 0.25\,(\max(0, 0, 0)) = 50$ End of value iteration	**Current state: *C*, Robot 2** (1) Use (9) $\gamma(C) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Check the Q-value for state C (use (11)) $Q(C, a) = 0$ Selected action, a = inspect (3) Use (10) $Q(C, a) = 50 + 0.25\,(\max(0, 0, 0)) = 50$ End of value iteration
Current state: *B*, Robot 2 (1) Use (9) $\gamma(B) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Check the Q-value for state B (use (11)) $Q(B, a) = 0$ Selected action, a = inspect (3) Use (10) $Q(B, a) = 50 + 0.25\,(\max(0, 0, 0)) = 50$ End of value iteration	**Current state: *D*, Robot 1** (1) Use (9) $\gamma(D) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Broadcast (use (11)) $Q(D, a) = 0$ Selected action, a = inspect (3) Use (10) $Q(D, a) = 50 + 0.25\,(\max(0, 0, 0)) = 50$ End of value iteration	**Current State: *A*, Robot 2** (1) Use (9) $\gamma(A) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Broadcast (use (11)) $Q(A, a) = 0$ Selected action, a = inspect (3) Use (10) $Q(A, a) = 50 + 0.25\,(\max(0, 0, 0)) = 50$ End of value iteration
Current state: *A*, Robot 1 (1) Use (9) $\gamma(A) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(A, a) = 50$, that is, $Q > 0$ Selected action, a = Ignore (3) Use (10) $Q(A, a) = 100 + 0.4\,(\max(0, 0, 0)) = 100$ End of value iteration	**Current state: *D*, Robot 2** (1) Use (9) $\gamma(D) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(D, a) = 50$, that is, $Q > 0$ Selected action, a = ignore (3) Use (10) $Q(D, a) = 100 + 0.4\,(\max(0, 0, 0)) = 100$ End of value iteration	**Current state: *B*, Robot 1** (1) Use (9) $\gamma(B) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(B, a) = 50$, that is, $Q > 0$ Selected action, a = Ignore (3) Use (10) $Q(B, a) = 100 + 0.4\,(\max(0, 0, 0)) = 100$ End of value iteration
Current state: *F*, Robot 2 (1) Use (9) $\gamma(D) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(F, a) = 50$, that is, $Q > 0$ Selected action, a = ignore (3) Use (10) $Q(F, a) = 100 + 0.4\,(\max(0, 0, 0)) = 100$ End of value iteration	**Current state: *C*, Robot 1** (1) Use (9) $\gamma(B) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(B, a) = 50$, that is, $Q > 0$ Selected action, a = Ignore (3) Use (10) $Q(B, a) = 100 + 0.4\,(\max(0, 0, 0)) = 100$ End of value iteration	**Current state: *E*, Robot 2** (1) Use (9) $\gamma(E) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(E, a) = 50$, that is, $Q > 0$ Selected action, a = ignore (3) Use (10) $Q(E, a) = 100 + 0.4\,(\max(0, 0, 0)) = 100$ End of value iteration
Current state: *G*, Robot 1 (1) Use (9) $\gamma(G) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Broadcast (Use (11)) $Q(G, a) = 0$ Selected action, a = Inspect (3) Use (10) $Q(G, a) = 50 + 0.25\,(\max(0, 0, 0)) = 50$ End of value iteration	**Current state: *G*, Robot 2** (1) Use (9) $\gamma(G) = \frac{0.5}{[1+0.25]} = \frac{0.5}{1.25} = 0.4$ (2) Broadcast (use (11)) $Q(G, a) = 50$, that is, $Q > 0$ Selected action, a = ignore (3) Use (10) $Q(G, a) = 100 + 0.4\,(\max(50, 50, 50))$ $= 100 + 20 = 120$ End of value iteration	**Current state: *C*, Robot 1** (1) Use (9) $\gamma(C) = \frac{0.5}{[1+0.4]} = \frac{0.5}{1.4} = 0.36$ (2) Broadcast (Use (11)) $Q(C, a) = 50$ Selected action, a = Ignore (3) Use (10) $Q(C, a) = 100 + 0.36\,(\max(50, 50))$ $= 100 + 18 = 118$ End of value iteration
Current state: *C*, Robot 2 (1) Use (9) $\gamma(C) = \frac{0.5}{[1+0.36]} = \frac{0.5}{1.36} = 0.37$ (2) Broadcast (Use (11)) $Q(C, a) = 50$, that is, $Q > 0$ Selected action, a = ignore (3) Use (10) $Q(C, a) = 100 + 0.37\,(\max(50, 50))$ $= 100 + 0.37 * 50 = 118.5$ End of value iteration	**All QLACS 1 states exhausted** **Goal state: *H*, Robot 1** (1) Use (9) $\gamma(H) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (2) Broadcast (use (11)) $Q(H, a) = 0$ Selected action, a = Shutdown (3) Use (10) $Q(H, a) = 150 + 0.25\,(\max(0, 0)) = 150$ End of value iteration	**All QLACS 2 states exhausted** **Goal state: *H*, Robot 2** (4) Use (9) $\gamma(H) = \frac{0.5}{[1+1]} = \frac{0.5}{2} = 0.25$ (5) Broadcast (use (11)) $Q(H, a) = 0$ Selected action, a = Shutdown (6) Use (10) $Q(H, a) = 150 + 0.25\,(\max(0, 0)) = 150$ End of value iteration

Table 10: Continued.

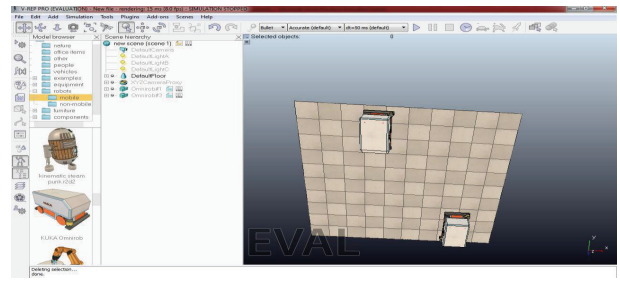
	Robot 1				Robot 2			
		Inspect	Ignore	Shutdown		Inspect	Ignore	Shutdown
End of policy iteration	*F*	Yes	No	No	*C*	Yes	No	No
	E	Yes	No	No	*B*	Yes	No	no
	D	Yes	No	No	*A*	Yes	No	No
	A	No	Yes	No	*D*	No	Yes	No
	B	No	Yes	No	*F*	No	Yes	No
	C	No	Yes	No	*E*	No	Yes	No
	G	Yes	No	No	*G*	No	Yes	no
	C	No	Yes	Yes through *H*	*C*	No	Yes	Yes through *H*

states. Since there is no communication, they exit the mine after learning, consequently not inspecting all the states. The same routine is reflected in Table 12, but in this case, each robot ends up inspecting all the states before exiting the mine. Analysis of Tables 11 and 12 shows that resources are wasted and the inspection job is not effective and efficient. Comparing the two tables, the time and distance cost are high, though higher in Table 12 because of many repetitions. For instance, the time and distance cost in Table 12 column 2 are 48.0028 and ((*F, G, E, D, A, B, C*), (*C, B, A, D, E, G, F*)), respectively. It also shows that the states are repeated in Tables 11 and 12. The memory usage is equally high. This led us to create a more efficient QLACS with communication by introducing some heuristics. The processes involved in Tables 11 and 12 are explained in Table 13.

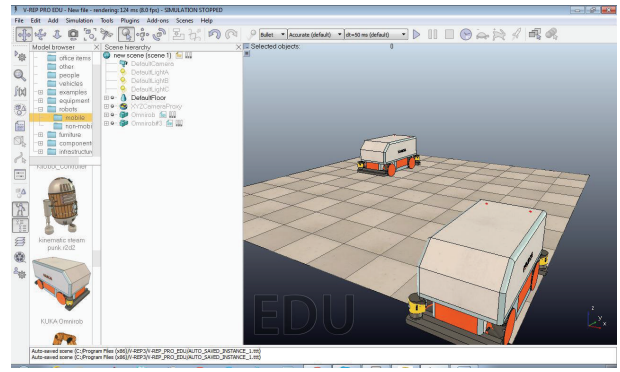
One can see from Tables 11 and 12 that there is no good communication among the robots, hence the evolvement of Table 14.

4.3. Experiment 2: Performance of QLACS with Good Cooperation. The heuristic added to the known QL made this experiment show good communication. This is where our contribution to communication is shown. As a robot inspects and learns the environment, it broadcast its Q-values to the other robot, which is in the form of a lookup table. In this case, each robot checks for Q-values; when a Q-value is equal to zero; the robot randomly selects a state for inspection. If a Q-value is greater than zero, the robot checks if the state is available for inspection or for ignoring. When a robot encounters a state with a Q-value equal to zero and the thread next to the state is equal to the goal state (H), then it shuts down. It must have checked the lookup table to see that all states have been inspected. The result in Table 14 shows good communication between two robots. No states were inspected more than once. The iterations for every run with their times, memory usage, and effective communication are also displayed in Table 14.

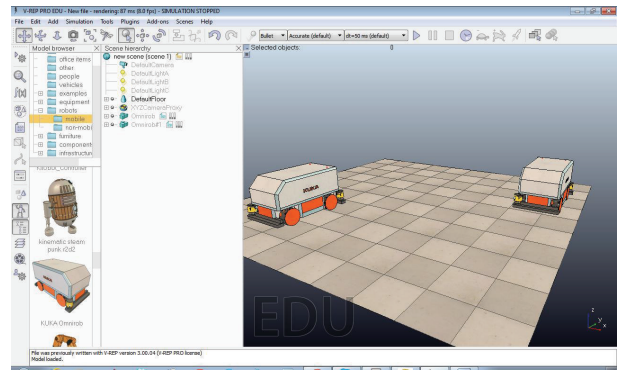
Comparing Table 14 with Tables 11 and 12, one cannot but notice the huge differences in the memory, time, and distance costs. The communication between the robots in Table 14 resulted in achieving good inspection; however, the random method of choosing next inspection states did not give an optimized route, thereby increasing time cost in passing and checking through inspected states.



(a) Two robots starting inspection from two different entrances



(b) Two robots inspecting as they navigate in the environment



(c) Two robots inspecting different locations with different positions

Figure 8: Different experimental behaviours for two robots.

4.4. Experiment 3: Performance of QLACS for the Navigation Behaviour of the Proposed Model. The result of implementing the first component of QLACS for effective navigation of the proposed model is tabulated in Table 16. Table 15 shows

Table 11: QLACS without communication (inspecting some states).

Number of runs	1	2	3	4	5	6	7
Iterations	9	10	10	13	13	9	13
Time (sec)	43.0025	30.0017	34.002	30.0017	31.0017	31.0018	27.0016
Memory usage (bytes)	18872	18160	19204	18208	17896	18164	18308
Inspected states (R1)	*G*	*F*, *G*, *C*	*C*	*F*, *D*, *B*	*E*, *A*	*F*, *E*, *D*, *A*, *C*	*F*, *G*, *E*, *A*
Inspected states (R2)	*B*, *A*, *G*	*C*	*C*, *B*, *A*, *E*, *F*	*B*, *A*	*C*, *B*, *E*	*B*, *A*, *E*	*A*, *G*

Table 12: QLACS without communication (inspecting all states).

Number of runs	1	2	3	4	5	6	7
Iterations	114	10	70	10	9	10	10
Time (sec)	43.0024	48.0028	41.0023	34.0019	34.0019	34.0019	35.002
Memory usage (bytes)	28440	17960	27216	17000	18456	17672	17968
Inspected states (R1)	*F, G, E, D, A, B, C*	*F, G, E, D, A, B, C*	*F, G, E, D, A, B, C*	*F, G, E, D, A, B, C*	*F, G, E, D, A, B, C*	*F, G, E, D, A, B, C*	*F, G, E, D, A, B, C*
Inspected states (R2)	*C, B, A, D, E, G, F*	*C, B, A, D, E, G, F*	*C, B, A, D, E, G, F*	*C, B, A, D, E, G, F*	*C, B, A, D, E, G, F*	*C, B, A, D, E, G, F*	*C, B, A, D, E, G, F*

Table 13: Processes used in achieving Tables 11 and 12.

(a) For Table 11

Inspecting some States
(1) Initialize the starting and goal states
(2) Initialize all Qvalues to zeroes
(3) Select a random action If Qvalues of all possible actions at current state are zeroes
(4) Select the action with the highest Qvalue If it is the Max Qvalue
(5) Compute and update Qvalue of the selected action
(6) Get new state among possible states
(7) If **new state = goal state** then go to **Steps 5 and 9**
(8) Repeat **Steps 3 to 6** until **Step 7**
(9) Shutdown

(b) For Table 12

Inspecting all States
(1) Initialize the starting and goal states
(2) Initialize all Qvalues to zeroes
(3) Select a random action If Qvalues of all possible actions at current state are zeroes
(4) Select the action with the highest Qvalue If it is the Max Qvalue
(5) Compute and update Qvalue of the selected action
(6) Get new state among possible states
(7) If **new state = goal** state then go to **Steps 5 and 10**
(8) Repeat **Steps 3 to 7** until **Step 9**
(9) All states except the goal state has taken **Action = Inspect**
(10) Shutdown

the selected parameters used in achieving the optimal path. The optimal path found after nine test runs is the path with a distance cost of 10 for both entries to the environment, displayed in Table 16. Therefore, columns 4, 5, 7, 8, and 9 can be used as the optimized path input for the first component of QLACS. Then QLACS will use any of the optimized paths to navigate to the specified states and take decisions accordingly. For instance, the test run result for nine ants gives 16 iterations under 60.0035 seconds and giving the shortest paths to both robots coming from entries *F* and *C* of the environment. The path that gives us cost as 10 is obtained from FEDABCGC (1.2.2.2.1.1.1) and CBADFGEGC (1.2.2.1.1.2.2.1), respectively.

Alpha (α), Beta (β), and Rho (ρ) represent the heuristic properties of the ACS algorithm. The Alpha factor is the measure of the influence of pheromone concentration that can influence the selection of the path with the highest pheromone concentration. Beta is the measure of the influence which is related to the distance between any two adjacent nodes. It is also a heuristic factor that can measure the influence distance in selecting the next state. It is not limited to pheromones. Rho has to do with the rate at which the pheromones evaporate. Rho shows how often new paths are explored by the agents/ants rather than reinforcing old paths. Each agent cooperates by having access to other agents' pheromone values. The pheromone value is initialized to 0.01 and it is continually updated until learning stops. It is similar to the first component of QLACS, where we initialize all QLACS positions to zero and update for every new step.

Table 16 gave the optimized route to be used by the robots to achieve inspection tasks. This resulted in combining Tables 14 and 16 to achieve Table 17. Table 17 shows optimized time cost, memory usage, route cost, and good communication.

4.5. Experiment 4: Benchmarking the New Hybrid Model (QLACS) with Popular Methods. Choosing the optimized paths, QLACS performs cooperative inspection behaviour through communication among robots. Looking at the sample result on Table 17, QLACS chooses the shortest and complete possible inspection routes from different runs. In this case, the best paths were obtained from Table 16 by using 9 agents, 10 agents, and 12 agents. All the test runs gave best trail paths from both entrances listing the complete states

Table 14: QLACS with communication.

Number of runs	1	2	3	4	5	6	7
Iterations	9	10	9	13	10	10	9
Time (sec)	33.0019	31.0017	31.0018	31.0018	29.0023	30.0017	31.0018
Memory usage (bytes)	15748	15948	15748	18232	16576	15948	15748
Inspected states (R1)	*F*, *G*, *E*	*F*, *G*, *E*, *D*	*F*, *G*, *E*	*F*, *E*	*F*, *G*, *E*, *D*, *B*	*F*, *G*, *E*, *D*	*F*, *G*, *E*
Inspected states (R2)	*C*, *B*, *A*, *D*	*C*, *B*, *A*	*C*, *B*, *A*, *D*	*F*, *G*, *E*, *D*, *A*	*C*, *A*	*C*, *B*, *A*	*C*, *B*, *A*, *D*

Table 15: Navigation behaviour parameter specification.

ACO properties	Type of ACO	Population	Length of path	Pheromone coefficient β	Heuristic coefficient α	Evaporation rate ρ
Properties	ACS	9 or 12	8	2	3	0.01

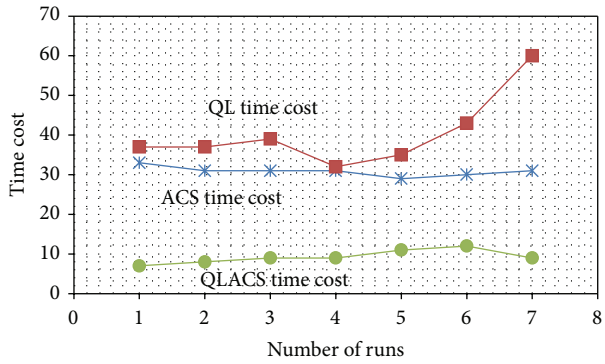


Figure 9: Comparison of time costs for QL, ACS, and QLACS.

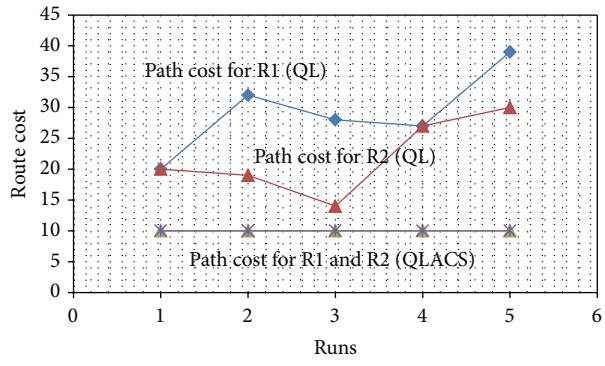


Figure 10: Comparison of route costs for QL and QLACS.

with length 10 as shown in Table 16; that is, they have states from *F* to *C* and from *C* to *F*. The length is the addition of weights, along the line (trail edges). Then the QLACS uses the optimized paths to make decisions on inspections. The result from Table 17 shows that no state is inspected twice or visited more than required. The QLACS model concentrates on making the best inspection decisions for MRS. The first run on Table 17 shows that nine agents/ants were used for the route finding, the optimized route was achieved after nine iterations under 7.0004 sec, and the states where all inspected effectively without redundancies.

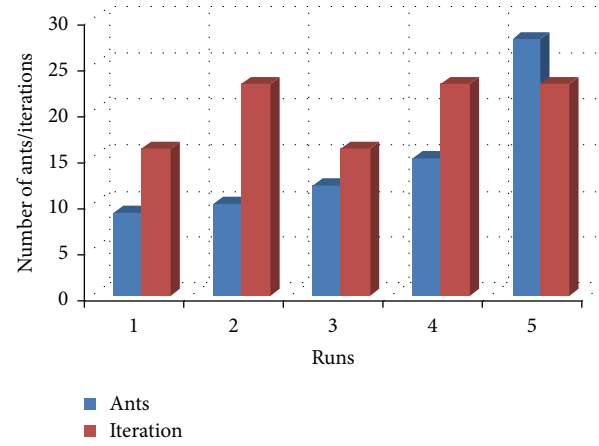


Figure 11: Number of ants/iteration required for QLACS to find optimal path.

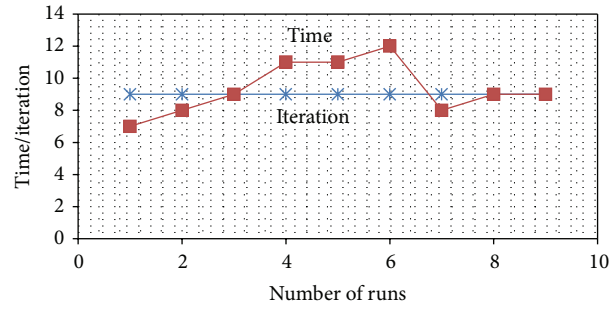


Figure 12: Time and iteration required for QLACS to achieve optimal behavior.

4.5.1. Comparative Study of the Proposed Model (QLACS) with QL Only and ACS Only. Based on the results tabulated in Table 18 and Figure 9, the average time costs of achieving the MRS behaviours for QL, ACS, and QLACS were compared. Two robots will use an average of 30.8590 sec to achieve thorough inspection behaviour using QL, an average of 40.4309 sec to achieve optimal navigation behaviour using ACS, and an average of 9.2868 sec to achieve both navigation and cooperative behaviour using QLACS. The result shows

Table 16: Navigation behaviour computations.

Number of runs	1	2	3	4	5	6	7	8	9
Number of Ants	5	7	8	9	10	11	12	15	28
Iterations	12	26	14	16	23	14	16	23	22
Time (sec)	37.0021	37.0021	39.0022	60.0035	32.0019	33.0019	35.002	43.0025	65.0037
Best train distance (R1)	12	12	10	10	10	10	10	10	10
Best trail distance (R2)	10	12	12	10	10	12	10	10	10

Table 17: New hybrid model QLACS computations.

Number of runs	1	2	3	4	5	6	7	8	9
Iterations	9	9	9	9	9	9	9	9	9
Number of ants	9	9	9	10	10	10	12	12	12
Time (sec)	7.0004	8.0005	9.0005	11.0006	11.0006	12.0007	8.0005	9.0005	9.0006
Memory usage (bytes)	10288	10280	10248	10288	10288	10288	10164	10712	10164
Inspected states (R1)	F, G, E, D	F, G, E, D	F, G, E, D	F, G, E, A, B	F, G, E, A, B	F, G, E, A, B	F, G, D, A	F, G, D, A	F, G, D, A
Inspected states (R2)	C, B, A	C, B, A	C, B, A	C, D	C, D	C, D	C, B, E	C, B, E	C, B, E

Table 18: Time cost comparison for QL, ACS, and QLACS.

Runs	QL time cost (sec)	ACS time cost (sec)	QLACS time cost (sec)
1	33.0019	37.0021	7.0004
2	31.0019	37.0021	8.004
3	31.0018	39.0022	9.0005
4	31.0018	32.0019	9.0005
5	29.0023	35.002	11.0006
6	30.0017	43.0025	12.0007
7	31.0018	60.0035	9.0006
Average	30.8590	40.4309	9.2868

that our proposed integrated algorithm performs better with reduced time cost. On the same note, the route costs for the QL and QLACS were also compared. The results in Table 19 and Figure 10 show that the proposed model QLACS gave a much lower route cost than the QL.

The number of ants and iterations used to achieve this is displayed in Figure 11. The more the ants, the more the iteration. The best routes created in the five test runs shown in Figure 11 are run numbers 1 and 3. They used fewer ants and iterations to achieve the optimal routes. The optimal result for the proposed model is achieved under nine iterations for every run. The iteration remains constant for any number of agents and robots. The blue line with star markers at Figure 12 is the iteration value for 9 runs. The red line shows the different amounts of time required for each run. The time is also almost stable for the QLACS, though it fluctuated a little in the middle. This signifies that the proposed model is more robust and effective in finding the optimal route and coordinating MRS. The reduced route cost and shorter computation time achieved with the QLACS satisfied the criteria for cooperative behavioural purposes.

4.6. Experiment 4: Scalability of QLACS Using Two, Three, and Four Robots. In this section, we present some results obtained by experimenting with the cooperative behavioural action of two, three, and four robots using the QLACS model. The performance of the two, three, and four robots was evaluated by running the simulation three times, using the same number of agents. The performance of the proposed QLACS model shows good communication between the two, three, and four robots under the states inspected, in the last four columns of Table 20. An almost stable time was used in achieving the inspection for all the robots. The detail of the simulation result is laid out in Table 20.

A notable observation emanating from Table 20 is that there is a need for a larger mine area of inspection because robot R1 in rows 4 to 6 could inspect only one state, while robots R1 and R2 in rows 7 to 9 could inspect only one state each. This implies that the size of the field of inspection is proportional to the number of robots to be deployed.

5. Concluding Remarks

This paper has shown how MRS behave cooperatively in underground terrains. The QL algorithm has been investigated for its potential quality of good communication, while the ACS algorithm was explored for good navigation properties. The strengths of the two algorithms have been harnessed and optimized to form a hybrid model QLACS which addressed the behavioural situation in MRS effectively.

The analytical solution of the new model was explained as shown in Algorithms 3 and 4 and Tables 7 and 10. The hybrid architecture for the model was also described in Figure 7. The experimental setup describing different navigation locations for two robots was shown in Figure 8.

The new hybrid model QLACS for cooperative behaviour of MRS in an underground terrain was proven able to find the optimal route and handle cooperation between two robots

Table 19: Route cost comparison for QL and QLACS.

Runs	QL (sec)		QLACS (sec)	
	Path cost for R1	Path cost for R2	Path cost for R1	Path cost for R2
1	20	20	10	10
2	32	19	10	10
3	28	14	10	10
4	27	27	10	10
5	39	30	10	10
Average	29.5	22	10	10

Table 20: Summary of scalability performance on QLACS.

Row numbers	Number of robots	Time (sec)	Number of states inspected			
			Robot 1 (R1)	Robot 2 (R2)	Robot 3 (R3)	Robot 4 (R4)
1	2	10.0006	4	3		
2	2	11.0007	4	3		
3	2	8.0005	4	3		
4	3	16.0009	1	3	3	
5	3	17.001	1	3	3	
6	3	12.0006	1	3	3	
7	4	11.0006	1	1	3	2
8	4	14.0006	1	1	3	2
9	4	10.006	1	1	3	2

effectively. The cooperative behavioural problem was narrowed down to two situations: navigational and cooperative behaviours. ACS was used to establish the shortest optimal route for two robots while the QL was used to achieve effective cooperation decisions. The results achieved after integrating the two algorithms showed a reduction in route costs and computation times, as shown in Figures 9, 10, 11, and 12. The comparative analysis between QL, ACS, and QLACS proved that the last-named is more robust and effective in achieving cooperative behaviour for MRS in the proposed domain.

The result of this work will be used for future simulation of MRS applications in hazardous environments. An indication of expansion of this research area has conspicuously surfaced in both real life implementations and model increase. The model used in this work can also be improved upon by increasing the state space.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors gratefully acknowledge the financial support of the Organisation for Women in Science for the Developing World (OWSDW), L'oreal-Unesco for Women in Science, and the resources made available by the University of Cape Town (UCT) and University of South Africa (UNISA), South Africa.

References

[1] J. Leitner, "Multi-robot cooperation in space: a survey," in *Proceedings of the Advanced Technologies for Enhanced Quality of Life (AT-EQUAL '09)*, vol. 37, pp. 144–151, IEEE Computer Society, Iaşi, Romania, July 2009.

[2] G. Dudek, M. R. M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multi-agent robotics," *Autonomous Robots*, vol. 3, no. 4, pp. 375–397, 1996.

[3] A. Chella, G. Lo Re, I. Macaluso, M. Ortolani, and D. Peri, "Chapter 1. A networking framework for multi-robot coordination," in *Recent Advances in Multi Robot Systems*, A. Lazinica, Ed., pp. 1–14, 2008.

[4] Yinka-Banjo, C. O. ; Osunmakinde, and I. O. ; Bagula A, "Autonomous multi-robot behaviours for safety inspection under the constraints of underground mine Terrains," *Ubiquitous Computing and Communication Journal*, vol. 7, no. 5, pp. 1316–1328, 2012.

[5] S. Yarkan, S. Güzelgöz, H. Arslan, and R. Murphy, "Underground mine communications: a survey," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 3, pp. 125–142, 2009.

[6] M. Peasgood, J. McPhee, and C. Clark, *Complete and Scalable Multi-Robot Planning in Tunnel Environments*, Digitalcommons, 2006.

[7] F. E. Schneider, D. Wildermuth, and M. Moors, "Methods and experiments for hazardous area activities using multi-robot system," in *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3559–3564, May 2004.

[8] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks," *International Journal of Robotics Research*, vol. 25, no. 1, pp. 73–101, 2006.

[9] L. E. Parker, "Current research in multi-robot systems," in *Proceedings of the 7th International Symposium on Artificial Life and Robotics (ISAROB '03)*, vol. 7, pp. 1–15, 2003.

[10] C. Thorp and H. Durrant-Whyte, "Field robots," in *Proceedings of the 10th International Symposium on Robotics Research*, vol. 6, pp. 329–3240, 2003.

[11] S. R. Teleka, J. J. Green, S. Brink, J. Sheer, and K. Hlophe, "The automation of the "Making Safe" process in South African Hard-Rock underground mines," in *International Journal of Engineering and Advanced Technology (IJEAT '12)*, vol. 1, pp. 1–7, April 2012.

[12] L. Cragg and H. Hu, "Application of reinforcement learning to a mobile robot in reaching recharging station operation," in *Intelligent Production Machines and Systems*, pp. 357–363, 2005.

[13] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa, "On the relation between ant colony optimization and heuristically accelerated reinforcement learning," in *Proceedings of the 1st International Workshop on Hybrid Control of Autonomous System*, pp. 49–55, 2009.

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence, from Natural to Artificial Systems*, Oxford University Press, Oxford, UK, 1999.

[15] Y. Liu and K. M. Passino, *Swarm Intelligence: Literature Overview*, Department of Electrical Engineering, University of Ohio, 2000.

[16] L. Li, A. Martinoli, and Y. S. Abu-Mostafa, "Emergent specialization in swarm systems," in *Intelligent Data Engineering and Automated Learning—IDEAL 2002*, vol. 2412 of *Lecture Notes in Computer Science*, pp. 261–266, 2002.

[17] B. Englot and F. Hover, "Multi-goal feasible path planning using ant colony optimization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, pp. 2255–2260, May 2011.

[18] Z. N. Naqiv, H. Matheru, and K. Chadha, "Review of ant colony optimization on vehice routing problems and introduction to estimated-based ACO," in *International Conference on Environment Science and Engineering (IPCBEE '11)*, vol. 8, pp. 161–166, 2011.

[19] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: algorithms, applications and advances," Tech. Rep. IRIDIA/2000-32, 2000.

[20] R. Claes and T. Holvoe, "Cooperative ant colony optimization in traffic route calculations," in *Advances on Practical Applications of Agents and Multi-Agent Systems*, vol. 155 of *Advances in Intelligent and Soft Computing*, pp. 23–34, 2012.