IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 4, 2019, accepted April 13, 2019, date of publication April 22, 2019, date of current version May 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2912306

LSAR: Multi-UAV Collaboration for Search and Rescue Missions

EBTEHAL TURKI ALOTAIBI[1], SHAHAD SALEH ALQEFARI[1], AND ANIS KOUBAA[2,3]

[1]Computer Science Department, Al-Imam Mohammad Ibn Saud Islamic University, Riyadh 11432, Saudi Arabia
[2]Prince Sultan University, Riyadh 12435, Saudi Arabia
[3]CISTER Research Center, 4200-135 Porto, Portugal

Corresponding author: Ebtehal Turki Alotaibi (etalotaibi@imamu.edu.sa)

This work was supported in part by the Robotics and Internet of Things Lab, Prince Sultan University.

ABSTRACT In this paper, we consider the use of a team of multiple unmanned aerial vehicles (UAVs) to accomplish a search and rescue (SAR) mission in the minimum time possible while saving the maximum number of people. A novel technique for the SAR problem is proposed and referred to as the layered search and rescue (LSAR) algorithm. The novelty of LSAR involves simulating real disasters to distribute SAR tasks among UAVs. The performance of LSAR is compared, in terms of percentage of rescued survivors and rescue and execution times, with the max-sum, auction-based, and locust-inspired approaches for multi UAV task allocation (LIAM) and opportunistic task allocation (OTA) schemes. The simulation results show that the UAVs running the LSAR algorithm on average rescue approximately 74% of the survivors, which is 8% higher than the next best algorithm (LIAM). Moreover, this percentage increases with the number of UAVs, almost linearly with the least slope, which means more scalability and coverage is obtained in comparison to other algorithms. In addition, the empirical cumulative distribution function of LSAR results shows that the percentages of rescued survivors clustered around the [78%–100%] range under an exponential curve, meaning most results are above 50%. In comparison, all the other algorithms have almost equal distributions of their percentage of rescued survivor results. Furthermore, because the LSAR algorithm focuses on the center of the disaster, it finds more survivors and rescues them faster than the other algorithms, with an average of 55%∼77%. Moreover, most registered times to rescue survivors by LSAR are bounded by a time of 04:50:02 with 95% confidence for a one-month mission time.

INDEX TERMS Autonomous agents, drones, search and rescue, unmanned aerial vehicles.

I. INTRODUCTION

Recently, unmanned aerial vehicles (UAVs), known as drones, have been shown to be quite effective in several applications such as smart agriculture [1], surveillance [2], [3], survey and mapping [4], delivery [5], and search and rescue (SAR) [5]–[8] (Figure 1).

In this paper, we focus on the SAR problem by employing a team of multiple UAVs. In natural disasters (such as earthquakes, floods, and fires), it becomes crucially important to retrieve survivors in the minimum amount of time possible. Using UAVs with multi-model sensors (such as high quality cameras and gas detectors) helps to reduce the search time, because UAVs can provide aerial images that allow people needing assistance to be identified in an unprecedented and efficient manner.

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Tang.

FIGURE 1. UAVs' real life applications.

The control of UAVs can be accomplished manually by an expert pilot. However, this approach may not achieve the highest efficiency, as the coordination between pilots during a disaster is not straightforward, considering such environments are subject to dynamic critical events.

A more efficient approach is to deploy a team of autonomous UAVs that coordinate between each other to accomplish SAR tasks in the minimum amount of time while saving the maximum number of survivors. This problem is rather challenging and is mapped to the multi-robot task allocation (MRTA) problem [9], known as an NP-hard problem [10]–[12]. As the robots employed in this case are UAV robots, Kurdi *et al.* [12] refer to this problem as multi-UAV task allocation (MUTA). The SAR application MUTA consists of finding an unknown number of survivors in a planar search area. It should be noted that the location of the survivors is also not known in advance. Further, because the robots are flying, this complicates MUTA compared to regular MRTA; hence, a highly dynamic search in different dimensions is required. Due to such complexity, several heuristic-based approaches have been proposed.

Max-sum [13]–[16] is a centralized optimization approach applied to several UAV applications, including MUTA in SAR applications [15]. Max-sum uses message passing that can be configured to work in an approximate mode. The main drawback of max-sum algorithms is the need to re-plan the whole assignment for each time period to optimize the assignment. Thus, it may not be effective for real-time applications with high dynamicity; in addition, it may not scale well with a large number of UAVs due to the increase in communication overheads.

On the other hand, auction-based approaches are decentralized and are based on a bidding-auctioning process [17]–[24]. The key point is to apply an auctioneer to announce tasks for bids, and a robot with the best bid will win the task. In the context of the multi-UAV SAR problem, the auction-based approach assigns the most suitable UAV to a survivor according to the bidding value calculated as the distance between a UAV and a survivor. However, the bidding negotiation overhead consumes more time and computational resources compared to other approaches [25], [26].

The opportunistic task allocation (OTA) strategy was proposed in [27]. In OTA, a UAV selects a random block in the search area that has not yet been explored. If a survivor is found, the UAV will rescue them immediately. OTA is based on a random search strategy, which may produce good results; however, there is no guarantee that this will occur.

The fourth closely related algorithm is the locust-inspired approach (LIAM) [12], [27], [28], which is a problem-dependent heuristic, tailored to the task allocation problem in multi-UAV SAR missions. In LIAM, the UAVs switch between three operational modes according to the mission time. In each mode, the UAV flies with a different speed and different battery consumption rates. LIAM is a decentralized approach that requires heavy communication between individual UAVs, which mostly have limited computational and energy capabilities. In addition, LIAM considers all search regions equally without prioritizing particular regions.

In natural disasters, there is a center in which most of the survivors are located, which is a key factor in this work. Therefore, SAR missions should be planned in a way that focuses more on the center itself, with decreasing importance given with increasing distance from this location (Figure 2). We refer to this as the layered SAR (LSAR) algorithm.

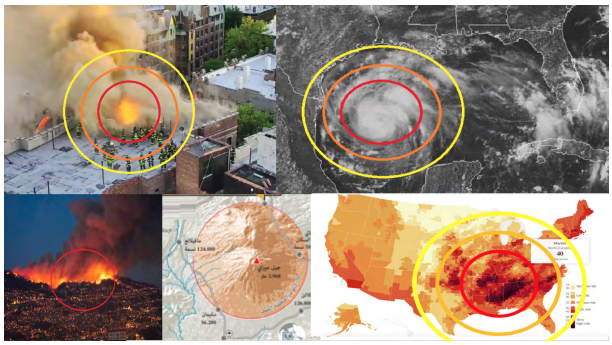


FIGURE 2. Disasters' forms.

In addition, with the emergence of cloud robotics and the possible connectivity of drones over the Internet [29]–[32], the LSAR algorithm is centralized in the sense that it assumes that UAVs communicate with a cloud server that coordinates the SAR missions among them. This assumption is realistic, as a cloud-based management system for the Internet of Drones (Dronemap) has already been developed and implemented in [29].

This paper is organized as follows: after this brief introduction, the details of closely related work are described in Section II. The LSAR algorithm is discussed in Section III, followed by a theoretical comparison of candidate algorithms in Section IV. The experimental results and discussions on different performance measurements are contained in Section V. Finally, the concluding remarks and future works are presented in Section VI.

II. LITERATURE REVIEW

There are four types of SAR problem, according to their application: maritime, combat, urban, and wilderness [33].

Maritime SAR refers to incidents where people are lost at sea. In [34], Lee *et al.*, provided a mixed integer linear program for application to maritime SAR problems. In such problems, the probability of locating survivors changes due to wind and currents. They summarized maritime SAR challenges as follows:

- limited fuel capacity of commercial UAVs,
- uncertainty and dynamicity of survivor locations,
- autonomous control of fuel service stations.

Their mixed integer linear program (MILP) model addressed these three challenges and worked efficiently on different numerical examples. Ghazali *et al.* [33], employed UAVs with rotary wings for maritime SAR operations, hovering around disaster areas to locate survivor locations. Moreover, the authors provided an algorithm for this purpose whereby photographs were taken of areas that may contain survivors, then the photographs were divided into four quarters, and process was repeated until a survivor was found.

Authors in [35] addressed the maritime SAR problem using cognitive and automated UAVs. They applied cooperative game theory to enable cognitive multi-UAVs, and tried to achieve their goal based on UAVs with on-board computers and with pre-knowledge of victims last seen locations. These two factors were obviously the main weakness of this work for the following reasons: First, UAVs have limited capacity for an onboard computer, and second, the paper violated the main constraints of an SAR problem-having unknown victim locations. They assumed the last location of survivors would be known.

When there is a disaster, such as an earthquake or terrorist attack, human teams start looking for survivors manually using dogs, which is termed urban SAR (USAR). More recently, human teams have begun to use robots, because they can perform more successfully than humans in such situations. Mainly, this is because they are able to squeeze into spaces too small for people and reach very dangerous areas. The attack on the World Trade Center (September 11) provided an unfortunate opportunity to analyze data collected during human-robot interactions [36]. This study reported on most of aspects related to USAR missions and human-robot interactions. Therefore, it provided a set of recommendations in the form of more research studies, a call for organization, construction of models for robot states, and statements and feedback on received observations.

In [37], the paper presented autonomous USAR robots from the perspective of achievements in research and technical aspects. The proposed solution attempted to handle two aspects of the SAR problem: First, for survivor search and detection, the system needs to be as close as possible to the survivors so they can be detected, even if the sensor only covers a very small portion of survivor′s body. This will lead to difficulties in detecting most survivor locations. Second, for navigation towards goal poses, they rely on the fact that the robot can only observe the position of the survivor when they are directly in front, which requires the robot to turn around each side for detection.

In [38], the authors developed different algorithms that deployed multiple cooperative MAVs for SAR missions after disasters. Their solution included real-time image stitching, indoor navigation, digit-detection, and vision-based pose estimation. The implemented method has two phases: First, aerial photography and map stitching are used by MAVs to explore the search area. Second, MAVs search and identify each house and its street in the city. However, this proposed solution is not fully autonomous, because initially (before MAVs start the rescue mission) the human operator must manually use the stitched map to identify routes with no obstacles for MAVs to enter the city. Further, in the second phase, the operator also needs to manually identify the number of victims in each house, based on the received feedback from the MAVs built-in camera.

Wilderness SAR (WiSAR) is the search process for people who are lost (or in distress) in the wilderness. Because the rescue team may also be infected when they reach the disaster area, medical UAV helicopters may help in wilderness SAR operations by reaching distant disaster areas quickly and safely, by providing urgent medical care, and by transferring injured patients as quickly as possible. The actual rescue can be carried out either by landing the helicopter on uncontrolled terrain, or by evacuating the patient and delivering them to the helicopter staff at the nearest safe landing area. Moreover, UAVs can offer detailed information about the area, which can help in future explorations [39]. In [40], the authors developed a camera-based position-detection system for SAR operations and integrated these into UAV plans. This system has been proven to identify real-time targets and post-targets, and to collect photographs of disaster areas for subsequent applications.

In [41], the WiSAR problem was handled using UAVs, which protect human life from risks during work in complex and unsafe environments. They presented human body detection and tracking algorithm, by using an onboard sensor on the UAV that can capture color and depth data. The paper validated the system with real and simulated environments, and proved its ability to detect multiple survivors.

The authors of [42] discussed SAR problems after an earthquake, and what kind of multi-robot coverage algorithms could handle the problem. The authors surveyed and compared the performance of a set of real time multicopter algorithms, which developed autonomous multi-drone strategies for SAR after earthquakes. The paper classified algorithms based on consumed energy and the time required for accomplishing the mission. The paper presented different coverage algorithms: Edge Counting [43], Node Count [44], Learning RealTime A* (LRTA*) [45], and PatrolGRAPH [46]. All of these can be used during search missions to cover the disaster area. Their experimental results demonstrated that the Node Count algorithm is the most efficient solution in multi-robot searches. For example, LRTA* suffers from drawbacks due to the increased complexity of the heuristics implemented to choose the next vertex to be visited. Further, Edge Counting and PatrolGRAPH* are less efficient, both in single- and multi-robot cases.

Combat SAR is an operation carried out during war. In [47], the authors presented an auction-based approach and a novel prediction approach to address the SAR problem for dynamic allocation. The auction-based algorithm assigned tasks for each robot by using techniques for determining winner tasks. Further, when the robot was inoperative, the prediction approach only has to allocate a task for the idle robot to perform. They measured the completion time and the required steps, indicating the level of consumed energy. The drawback of this approach is that they assumed to have previous knowledge of possible tasks and the initial locations of survivors; hence, the robots would only search these areas. However, this is opposed to the main condition in SAR-survivor locations are unknown.

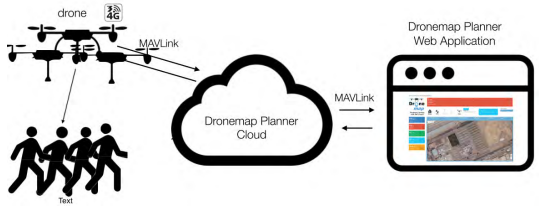


FIGURE 3. System architecture.

III. THE LSAR ALGORITHM

The key idea of the LSAR algorithm is that in natural disasters there is a center where most of the survivors are located. Assuming that these centers can be detected [48]–[50], the SAR mission should be planned in a way that focuses more attention on the center, and gradually less attention with increasing distance from the center [51].

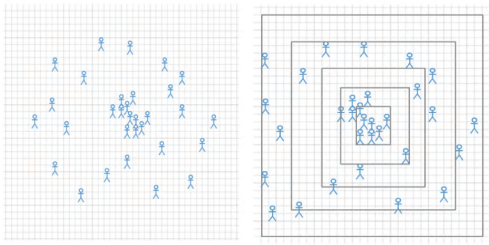
The LSAR algorithm is proposed as a centralized algorithm based on using a Cloud server for the Internet-based drone mission Controller. Accordingly, dronemap planner cloud (Cloud Server) is a possible structure that can be used for this task [29]–[32], which can control UAVs through the Internet (Figure 3). The limited capabilities of UAVs (such as battery power, processing unit, and memory) makes offloading all computation from UAVs (and transferring them to the cloud server) beneficial for improving the overall performance. The dronemap planner cloud controls UAVs through the cloud, can schedule their mission, and manage communications between UAVs remotely. Further, dronemap planner cloud supports the MAVLink protocol that manages communication between the dronemap planner cloud, the user, and the UAVs. This protocol is supported by most commercial UAVs.

A. SYSTEM MODEL

The system architecture is presented in Figure 3.

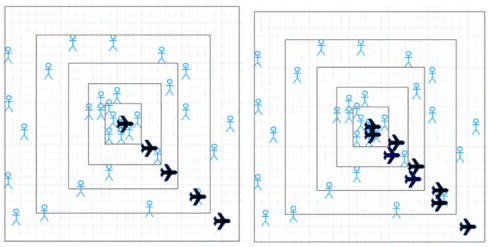
We consider a team of drones connected through a cloud server and collaborating to accomplish an SAR mission. Dronemap planner [29]–[32] is a cloud-based management system that was recently proposed, and can be used to ensure communication and collaboration between a team of drones.

The proposed LSAR algorithm is a centralized approach. The cloud server receives information on the disaster center [48]–[50], as an emergency call from the area that has a number of survivors in unknown locations (Figure 4 (a)). In addition, the cloud server running the LSAR algorithm divides the disaster area into a set of incremental, numbered, square shaped layers `L` (Figure 4 (b)). Layer number `x` has a survivor list denoted as `Lx[SurvivorList]`, which records the locations of found survivors as pair of latitude and longitude co-ordinates in that layer during a search mission. These layer lists are located and updated on the cloud server. Moreover, the cloud server running the LSAR algorithm employs a set of UAV aircraft (denoted as `P`) to search for missing survivors and rescue them. Therefore, each



(a) (b)

FIGURE 4. Partitioning to layers.



(a) (b)

FIGURE 5. UAVs planes assignment.

UAV aircraft running the LSAR algorithm can switch between two different modes: searcher and rescuer. As shown in Figure 5 (a), generally the cloud server only assigns one UAV for each layer. However, there are some cases where the cloud server assigns multiple UAVs to one layer (Figure 5 (b)).

B. ALGORITHMS

The LSAR system has two phases: (*i.*) the environment partitioning phase, which is a pre-process that samples the disaster area, and (*ii.*), the SAR phase, which is the SAR process coordinated by the cloud server.

1) PARTITIONING ALGORITHM

Given the disaster area specified by its coordinates in Figure 6 (a), in addition to the center of the disaster specified by its latitude and longitude coordinates, the cloud server running the LSAR algorithm divides the disaster area into a set of incremental, numbered, square shape layers `L`. Initially, the cloud server unifies a standard unit for a region; the smallest location unit specified by its coordinates. For example, these regions sizes could be 1 *m*, 1 *km*, or 1 *ft*. (Algorithm.1: line.1). Then, the cloud server determines how many regions are contained in the disaster area (Algorithm.1: line.2, Figure 6 (b)), According to this information, a disaster matrix is created whereby the real environment is sampled as a set of regions (Algorithm.1: line.5).

The layer thickness describes how thick a layer will be in terms of the number of adjacent regions. Because the layers have an incremental square shape, each layer should have two rows (upper and lower rows) and two columns (right and left columns). Hence, the cloud server calculates the

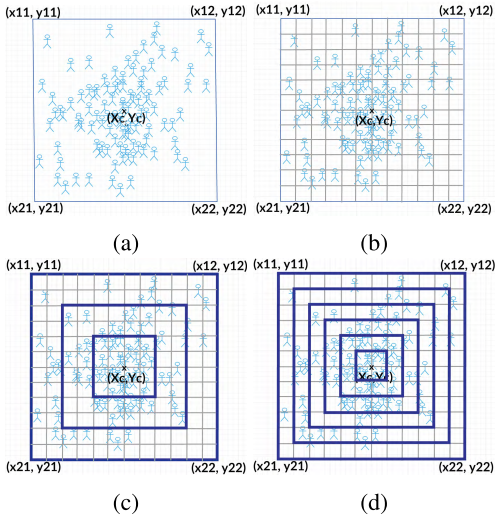


FIGURE 6. Partitioning disaster's area.

thickness of layers according to the width/2 or height/2 and the number of available UAV aircraft (Algorithm.1: line.6). In Figure 6, the height and width are 12 units and it is assumed that the system has 7 UAV aircraft; hence, the thickness is $\lceil (12/2)/7 \rceil$ = one unit, as shown in Figure 6 (c). Moreover, number of layers is also calculated according to the number of UAV aircraft (Algorithm.1: line.7-8). In this case, number of layers is 6 and the cloud server will subsequently assign many UAV aircraft to one layer (Figure 5 (b)).

However, if the system has a limited number of UAV aircraft (such as 3), compared to the minimum thickness in Algorithm.1: line.7, the thickness of a layer is $\lceil (12/2)/3 \rceil$ = two units, as shown in Figure 6 (d). Further, the number of layers is equal to the number of UAV aircraft. The cloud server will subsequently assign one UAV aircraft to each layer, as shown in Figure 5 (a).

Once the environment is sampled as a matrix of regions, the cloud server fills each layer list with the region indexes with which it is assigned. For example, Algorithm.1 line.17 fills the matrix with regions in the upper border of a layer, line.18 fills it regions in the right border, line.19 fills it with regions in the left border, and line.20 fills the layer with regions in the lower border of a layer. This process repeats many times equal to the calculated thickness (Algorithm.1: line.23). Subsequently, the algorithm moves to fill the next layer (Algorithm.1: lines.24-26). Therefore, the output is a list of layers, and each one records a set of region coordinates to which it is assigned.

2) LSAR ALGORITHM

Given the disaster area and center coordinates, the cloud server calls the partitioning algorithm to create set of layers (Algorithm1: line.1). According to this strategy, survivors closer to the center have a higher rescue priority than survivors in the outer layers, because most of them are located closer to the disaster center. Moreover, the cloud server defines a special way to distribute the aircraft over layers

Algorithm 1 Partitioning

Input : The disaster area is described by its latitude and longitudes (x_{11}, y_{11}) (x_{12}, y_{12}) (x_{21}, y_{21}) (x_{22}, y_{22}), disaster's center latitude and longitude $(x_c.y_c)$

Output: Set of layers L

```
1  region ← 1 x 1 unit
2  regions ← area / region
3  width ← number of adjacent regions horizontally
4  height ← number of adjacent regions vertically
5  Matrix[width][height] ← regions
6  Layer thickness ← ⌈[]⌋(width/2)/|P|
7  if |P| > width / 2 then
8  |  layer number ← width/2
9  else
10 |  layer number ← |P|
11 end
12 t ← 0
13 x ← 0
14 while x < layer number do
15 |  for i ← 0, height do
16 |  |  for j ← i, width − i do
17 |  |  |  L_x ← Matrix[i][j]
18 |  |  |  L_x ← Matrix[j][i]
19 |  |  |  L_x ← Matrix[j][width − i]
20 |  |  |  L_x ← Matrix[width − i][j]
21 |  |  end
22 |  end
23 |  t ← t + 1
24 |  if t = thickness then
25 |  |  x ← x + 1
26 |  |  t ← 0
27 |  end
28 end
```

to reflect the LSAR argument (Algorithm.1: line.2-9). Here, assuming the number of aircraft $|P|$ equals the number of layers, it sets every UAV aircraft as a searcher and assigns it to each layer in a one-to-one manner (e.g. it assigns plane x to layer x). This starts from layer number zero (center layer) towards the outer layers. When the number of aircraft $|P|$ is larger than the number of defined layers $|L|$ the cloud server reassigns the remaining aircraft to layers (starting with layer number zero) towards the outer layers, until they are assigned completely. In this way, any extra UAV planes are assigned to layers with higher probability to locate survivors. According to the UAV aircraft distribution in Figure 5 (a) and (b), the LSAR algorithm has two executions (a and b) described below.

a: ONE UAV PLANE FOR EACH LAYER

When there is a small number of UAV aircraft, the LSAR algorithm assigns only one for each layer, as shown in Figure 5 (a). Initially, the UAV aircraft executes its

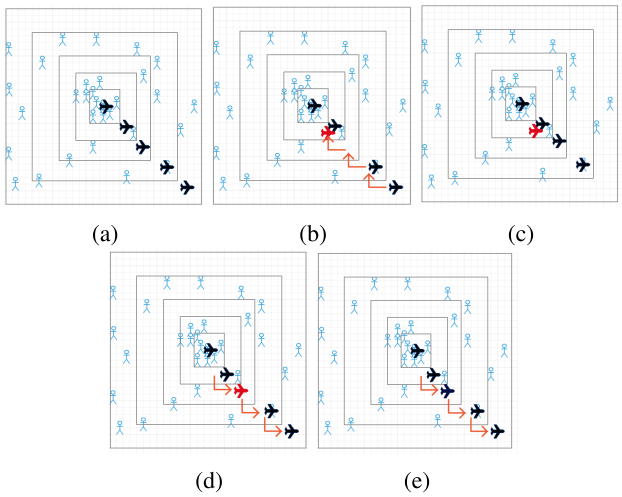


FIGURE 7. LSAR execution: One UAV plane for each layer.

searcher mode; it searches for survivors in its assigned layer (Algorithm.3: line.3). If a survivor is found, the searcher UAV aircraft adds its location (defined by its latitude and longitude) to survivor list `Lx[SurvivorList]` of layer number `x` (Algorithm.3: line.4-5). At the same time, the searcher UAV aircraft checks the survivors list `Lx[SurvivorList]` periodically whether or not it reaches a predefined threshold. If so, the searcher UAV aircraft sends a helping call to the cloud server (Algorithm.3: line.7-8). Then, the server makes an inward shift starting from the caller layer, because the probability of locating survivors on the inner layers higher than in outer layers, i.e., the second layer shown in Figure 7 (b) sends a helping call. The server interrupts UAV aircraft missions and makes an inward shift starting from the third layer and outer layers, one step toward layer two. The searcher UAV aircraft that was in layer three becomes a rescuer UAV aircraft in layer two (Algorithm.2: line.13-14), while all other shifted UAV aircraft continue as searchers in their new hosting layers (Figure 7 (c), (Algorithm.2: line.15-17). The rescuer UAV aircraft then starts rescuing all survivors, according to their locations recorded in L_2`[SurvivorList]` (Algorithm.3: line 11-12). Then, the rescuer UAV aircraft sends a notification call to the server reporting that the rescue mission is complete (Algorithm.3: Line.13-14). When the server receives a notification call coming from layer two (Algorithm.2: line.23), it recovers the rescuer UAV aircraft mode in layer two to the searcher mode (Figure 7 (c) and Algorithm.2: line.24). Then, it moves all UAV aircraft (starting from layer two) to their original layer to continue their interrupted search mission (Figure 7 (d-e)). Note that the interrupted missions could be recovered using the list of survivors attached to each layer; `Lx[SurvivorList]` (Algorithm.2: line.26-28).

b: MULTIPLE UAV PLANES FOR EACH LAYER

When there is a large number of UAV aircraft, the LSAR algorithm assigns many for each layer, as shown in Figure 8 (a). The UAV aircraft initially work in the same way

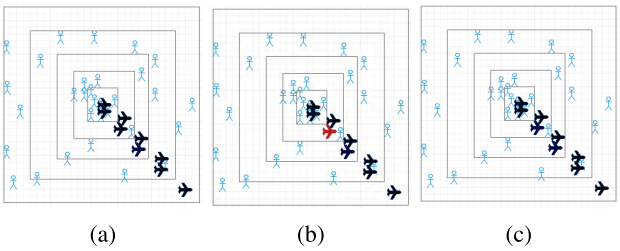


FIGURE 8. LSAR execution: Multiple UAV planes.

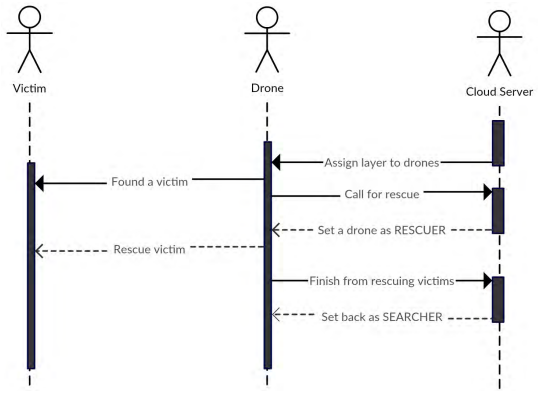


FIGURE 9. The LSAR sequence diagram.

described earlier, and a different interaction only happens when the cloud server receives a helping call. In this case, the server changes one of the UAV aircraft from searcher to rescuer, i.e., the second layer shown in Figure 8 (b) sends a helping call. The server interrupts a searcher UAV aircraft mission in that layer; hence, a searcher UAV aircraft becomes a rescuer in the same layer (Algorithm 2: lines.19-20). The rescuer UAV aircraft then starts rescuing all survivors according to their locations recorded in L_2`[SurvivorList]` (Algorithm.3: lines.11-12). When the rescuer UAV aircraft sends a notification call to the server reporting that the rescue mission is complete, and the server receives a notification call coming from layer two, it recovers the rescuer UAV aircraft mode in layer two to the searcher mode (Figure 8 (c) and Algorithm.2: line.24). When the layer has been fully explored and all survivors on this layer have been rescued, the cloud server reassigns the UAV aircraft of that layer to the outer layer (Algorithm.2: line.17-19). The interaction between the cloud server and UAV aircraft is summarized in the sequence diagram shown in Figure 9.

IV. THEORETICAL COMPARISON OF CANDIDATE ALGORITHMS

The research problem addressed in this paper is to find an efficient optimization algorithm for the SAR problem. In this section, we theoretically compare the LSAR algorithm with other candidate algorithms by considering the differences between their techniques. Table 1 presents the main characteristics affecting the performance of LSAR, LIAM, auction-based, and OTA algorithms.

Algorithm 2 Cloud Server

Input : Planes set P,disaster area described by its latitudes and longitudes (x_{11}, y_{11}) (x_{12}, y_{12}) (x_{21}, y_{21}) (x_{22}, y_{22}), disaster's center latitude and longitude $(x_c.y_c)$
Output: Online schedule

Layers set (L)← Partitioning Algorithm (x_{11}, y_{11}),(x_{12}, y_{12}),(x_{21}, y_{21}),(x_{22}, y_{22}),$(x_c.y_c)$
for $i \leftarrow 0$, $|P|$ *and* $x \leftarrow 0$, $|L|$ **do**
  $P_i[type] = SEARCHER$
  if $i > |L|$ **then**
    t=i mod $|L|$ Assign P_i to layer L_t
  else
    Assign P_i to layer L_x
  end
end
if *call(x) is received* **then**
  if *call*[*type*] *is helpCall at layer* L_x **then**
    if $|P|$ *at layer* $L_x = 1$ **then**
      P_{x+1} [*type*] $= RESCUER$
      Assign P_{x+1} to layer L_x
      for $i \leftarrow x{+}1, |P|$ **do**
        $P_i[type] = SEARCHER$
        Assign P_i to layer L_{i-1}
      end
    else
      P_x [*type*] $= RESCUER$
    end
  end
  if *call*[*type*] *is notificationCall at layer* L_x **then**
    P_{x+1} [*type*] $= SEARCHER$
    if $|P|$ *at layer* $L_x = 1$ **then**
      for $i \leftarrow x, |P|$ **do**
        $P_i[type] = SEARCHER$
        Assign P_i to layer L_{i+1}
      end
    end
  end
end

Algorithm 3 Plane

Input : Assigned Layer L_x
Output: Online search and rescue missions

while *time* $\leq$ *mission time* **do**
  if P_i[*type*] *is SEARCHER for layer* L_x **then**
    Search for a survivor in the layer L_x
    if *a survivor s is found* **then**
      L_x[SurvivorList] $\leftarrow s$
    end
    if $|L_x[SurvivorList]| \geq survivorThreshold$ **then**
      Call (*helpCall*, cloud server, x)
    end
  end
  if P_x[*type*] *is RESCUER for layer* L_y **then**
    Rescue all survivors $\in L_y$[SurvivorList]
    if $L_y[SurvivorList] = \Phi$ **then**
      Call (*notificationCall*, cloud server, y)
    end
  end
  if L_y *is fully explored* $\wedge$ L_y *is not last layer* **then**
    Assign P_y to layer $L_y + 1$
  end
end

The theoretical comparison between the LSAR and other algorithms involves to test a hypothesis which may be proven or contradicted by the empirical results.

The differences (D1, D4, D7, and D9 in the table) should be reflected by the increased number of rescued survivors by LSAR compared with the number of survivors rescued by LIAM, auction-based, Max-sum and OTA algorithms. This is because UAV aircraft in the LSAR algorithm search for survivors in the area where there is a higher probability of locating them. In terms of battery consumption, the difference (D2) may reduce the time needed to rescue a survivor by the LIAM algorithm more than the LSAR algorithm. In addition, according to the differences (D3, D6, and D11), the centralized LSAR is expected to produce better coverage than the decentralized LIAM, auction-based, and OTA algorithms. However, it requires more coordination time.

The auction-based algorithm is similar to the LSAR, according to similarity (S1). However, the difference in the responding technique described in difference (D5) concludes that the bidding negotiation in the auction-based algorithm may consume more time than the LSAR algorithm. This may contradict the results concluded earlier based on (D6). Hence, the empirical results will decide whether the coordination in the centralized approach consumes more time than the negotiation in the decentralized approaches. Difference (D8) assumes that max-sum produces an optimal assignment, which should improve the overall performance metrics. However, due to their computation overhead, such algorithms may be unsuitable for dynamic SAR environments; these may produce lower throughput and higher rescue times.

The random SAR strategy in the OTA algorithm may achieve some good results; however, this is not guaranteed. The searcher UAV aircraft in areas around the center of disasters should continue their roles and other rescuer UAV aircraft should help to balance the two missions, to benefit the greatest number of survivors. Hence, according to difference (D10), the LSAR may produce better throughput, time, and coverage than the OTA algorithm when the center of the disaster is detected.

V. EXPERIMENTAL EVALUATION

A. SIMULATION MODEL

We used the MASPlanes++ [52] simulator to conduct an extensive comparative simulation study among the LSAR

TABLE 1. Theoretical comparison between LIAM and benchmark algorithms.

LIAM algorithm	
Difference (D1)	The LIAM divides the searching area into equal priority regions while LSAR divides it into varying priority regions.
Difference (D2)	LIAM UAV aircraft switch between three modes according to the mission time. In each mode, the UAV aircraft fly with different speeds and different battery consumptions. The LSAR does not deploy any battery consumption strategies.
Difference (D3)	The LIAM is decentralized but LSAR is a centralized algorithm.
Auction-based algorithm	
Difference (D4)	The auction-based divides the searching area into equal priority regions, while LSAR divides it into varying priority regions. Similarity (S1) They divide the SAR missions into two different types and create a decision between them. Every UAV aircraft is a searcher at the beginning.
Difference (D5)	When a survivor is found by a searcher UAV aircraft running auction-based algorithm, it should send the survivor's location to each plane, they calculate the bidding value and send it back to the searcher plane who will take a decision to assign the survivor to the winner plane. Under the same situation, an LSAR searcher UAV aircraft will send a helping message to the central operator. The operator will then take a direct decision in the form of shifting-in all UAV aircraft. Auction-based is decentralized, but LSAR is a centralized algorithm.
Difference (D6)	The auction-based divides the searching area into equal priority regions while LSAR divides it into varying priority regions. Similarity (S1) They divide SAR missions into two different types and create a decision between them. Every UAV aircraft is a searcher at the beginning.
Max-sum algorithm	
Difference (D7)	The max-sum algorithm divides the searching area into equal priority regions while LSAR divides it into varying priority regions.
Difference (D8)	The max-sum searches for the optimal assignment while LSAR does not. The max-sum re-plans the assignment for each period to optimize the assignment while LSAR does not re-plan the assignment.
OTA	
Difference (D9)	The OTA algorithm divides the searching area into equal priority regions, while the LSAR divides it into varying priority regions. OTA regions have the same probability of having SAR UAV searcher and rescuer aircraft. However, LSAR regions have a different probability of having SAR UAV searcher and rescuer aircraft nearer to the center.
Difference (D10)	The OTA algorithm divides the searching area into equal priority regions, while LSAR divides it into varying priority regions. OTA regions have the same probability of having UAV searcher and rescuer aircraft. However, LSAR regions have different probabilities of having UAV searcher and rescuer aircraft nearer to the center.
Difference (D11)	The OTA is decentralized, but LSAR is a centralized algorithm.

TABLE 2. Parameters and scenarios.

Performance Measurements	Number of UAVs	Number of survivors	Survivors lifetime ranges
Salability and Sustainability	2^k, k=1…7	2^k, for k=1…12,	one fixed range [10-72 hours]
Responsiveness	Fixed to 4 UAVs	2^k, for k=1…12,	[<10, 10-20, 20-30, 30-40, 40-50, 50-60] minutes.

approach and the other candidate algorithms. MASPlanes++ is a simulation environment geared towards testing the dynamic coordination and task allocation methods in SAR problems. MASPlanes++ implements four well-established benchmark algorithms: theLIAM, auction-based, max-sum, and OTA schemes. Three main performance measurements are evaluated, as follows:

1) **Scalability:** We evaluate the impact of increasing the number of UAV aircraft.
2) **Sustainability:** We aim to evaluate how the algorithms perform in more complicated environments.
3) **Responsiveness:** We evaluate the time taken to rescue all the survivors.

The parameters used to control each scenario are listed in Table 2. The total number of created scenarios was 492 [(i.e. $5 \times 7 \times 12 + 6 \times 12$)]. In other words, number of algorithms to be tested $\times$ number of values for the number of UAV aircraft $\times$ number of values for the number of survivors $+$ number of values for the number of survivors $\times$ number of values for survivor life expectancy ranges. Each scenario was run several times to ensure the neutrality of the results [51].

MASplanes++ calculated three performance metrics that are related to our performance measurements as net throughput, mean time to find and rescue a survivor, and total running time. Survivor locations were generated in a way to simulate real-life disasters, where most survivors were located around the center of the disaster, decreasing gradually further away from the center. The parameter settings of the evaluation environment are detailed in Table 3. In the following section, we will discuss the results for the three scenarios: (1) scalability, (2) sustainability, and (3) responsiveness.

B. SCALABILITY AND SUSTAINABILITY

In this section, we evaluate both system scalability and sustainability when changing the number of UAV aircraft and survivors. According to [12], each result was averaged and presented in graphs defined by the following: a logarithmic base 2 scale for the number of UAV aircraft on the x-axis, and by a linear scale for the performance measurement on the y-axis. The value of the critical variables that control this evaluation are listed in Table 2.

1) PERCENTAGE OF RESCUED SURVIVORS (NET THROUGHPUT)

To measure the net throughput of the system, we collected the total number of rescued survivors in each scenario.

TABLE 3. Parameter settings of evaluation environment.

Parameter	Settings	
Plane Max Speed [1]	40 miles/hour	
Search Area Size	999m × 999m, 111 Regions × 111 Regions, 3m × 3m blocks	
Hotspots	10 (Radius: 200m, DOF: 2.5)	
Block Search	Power Consumption Penalty	5 units of power
	Time Penalty	10 seconds
Survivor Rescue	Power Consumption Penalty	10 units of power
	Time Penalty	60 seconds
Power Consumption	Idle	1 unit of power per 300 milliseconds
	Standard	1 unit of power per 100 milliseconds

[1] Most of the settings in [12] are followed here with some changes

The percentages of rescued survivors were calculated and plotted against the number of UAV aircraft in the line-log graphs (Figure 10). As the graphs level from top-left to bottom-right (in Figure 10), the number of survivors increases, and the problem becomes more difficult. However, as the number of UAV aircraft on the x-axes in each graph increases, the problem becomes less difficult.

In the earlier results described in Figure 10 (a-c), the LSAR algorithm found most of the survivors using a smaller number of UAV aircraft than other algorithms. For example, Figure 10 (a) shows that the LSAR only needs 4 UAV aircraft to find 100% of the survivors, while the LIAM needs 8 UAV aircraft to achieve the same percentage. Moreover, the other algorithms need even more UAV aircraft to find the same percentage of survivors. However, in the same results set, it is notable that the general behavior of LSAR is closer to the behavior of the LIAM algorithm.

Moreover, there are some differences in the general behavior of the LSAR graph between Figure 10 (a), (b) and (c). For example, LSAR found approximately 50% of the survivors when there were 2 UAV aircraft, approximately 25% when there were 4 UAV aircraft, and 50% (again) when there were 8 UAV aircraft. This is because when the number of survivors is small, they would not be concentrated around a center. Hence, the power of LSAR could not be shown clearly when the problem contained a small number of survivors. Due to the same reason, when the number of survivors increases in Figure 10 (g-l), the percentage of rescued survivors by the LSAR algorithm significantly outperforms the other algorithms.

Moreover, the performance shown in the LSAR graph is almost linear with an increasing number of UAV aircraft and has the least slope, which means more scalability and coverage than the other algorithms. To sum up, the empirical results confirm the hypothesis. They show that the LSAR algorithm found more survivors than the other algorithms in highly constrained problems, with better scalability and coverage factors.

2) MEAN RESCUE TIME

The rescue time of a survivor is calculated as the difference between the simulation start-time to the time taken for a survivor to be found by any UAV aircraft. This time is calculated for each survivor and averaged in Figure 11. The behavior of the LSAR algorithm is not clear in Figure 11 (a) due to the same reason described earlier; in problems with a low number of survivors, the LSAR displays random behavior. However, when the number of survivors is large, the behavior of LSAR should improve as the number of UAV aircraft increases. This is shown clearly in Figure 11 (b-l), where the average rescue time decreases with increasing UAV aircraft numbers.

Because the LSAR algorithm focuses on the center of the disaster more than other algorithms, it finds more survivors and rescues them quicker (Figure 11(b-l)), which was expected in the hypothesis. Finally, most LSAR graphs in Figure 11 (b-l) exhibit a linear behavior. In addition, they show too low a slope line (almost constant line) which means high scalability and coverage.

3) EXECUTION TIME PERFORMANCE

The execution of each simulation scenario was recorded and displayed in Figure 12. The earlier results (Figure 12 (a-f)) show that the execution times of LSAR was higher than other algorithms. This is because when the number of survivors is small, LSAR selects one survivor as a center randomly, then distributes the probability among other layers according to their distance away from the center. However, this center may be the furthest survivor from other survivors.

Even though the behavior of the LSAR graph in Figure 12 (a-f) is too high, it displays a more constant linear behavior than the other algorithms, which means high scalability and coverage.

In the later results in Figure 12 (g-l), LSAR takes less time, almost the same as the LIAM and OTA algorithms, because most of the survivors concentrate around disaster centers in such highly constrained problems, which makes, an earlier decision.

The average results for the percentage of rescued survivors, the mean time to rescue a survivor, and the execution times are summarized in Figs. 13-15, respectively.

Table 4 presents the statistical results of all the above experiments in terms of the mean and standard deviation for each performance measurement. The UAV aircraft running the LSAR algorithm consume the least mean time to rescue a survivor; they require 68 min (on average) to rescue a survivor. Moreover, most of the recorded results are very close to that average as the standard deviation is too low. The UAV aircraft running the OTA algorithm consume between

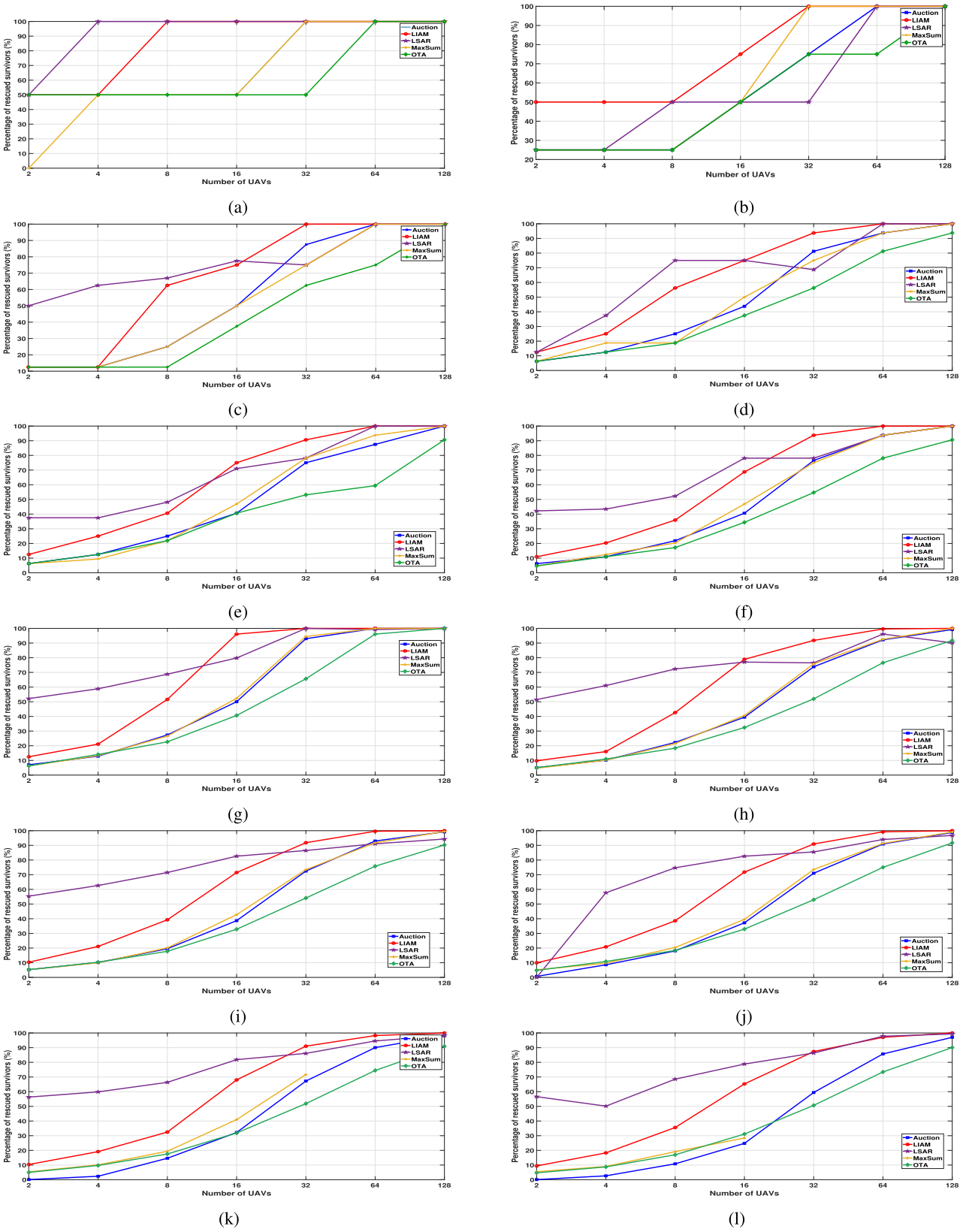


FIGURE 10. Percentage of rescued survivors. (a) 2 survivors. (b) 4 survivors. (c) 8 survivors. (d) 16 survivors. (e) 32 survivors. (f) 64 survivors. (g) 128 survivors. (h) 256 survivors. (i) 512 survivors. (j) 1024 survivors. (k) 2048 survivors. (l) 4096 survivors.

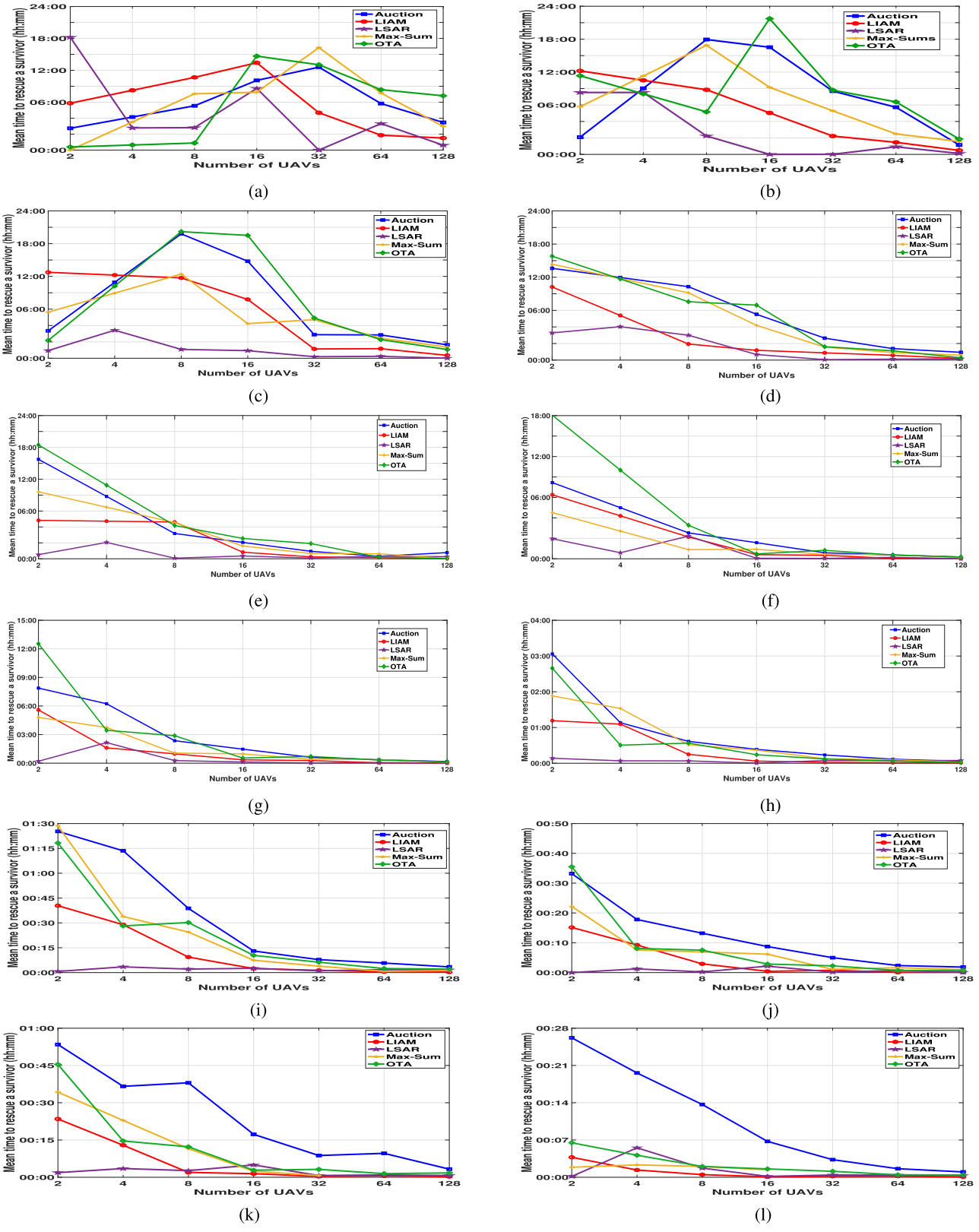


FIGURE 11. Mean time to rescue a survivor. (a) 2 survivors. (b) 4 survivors. (c) 8 survivors. (d) 16 survivors. (e) 32 survivors. (f) 64 survivors. (g) 128 survivors. (h) 256 survivors. (i) 512 survivors. (j) 1024 survivors. (k) 2048 survivors. (l) 4096 survivors.

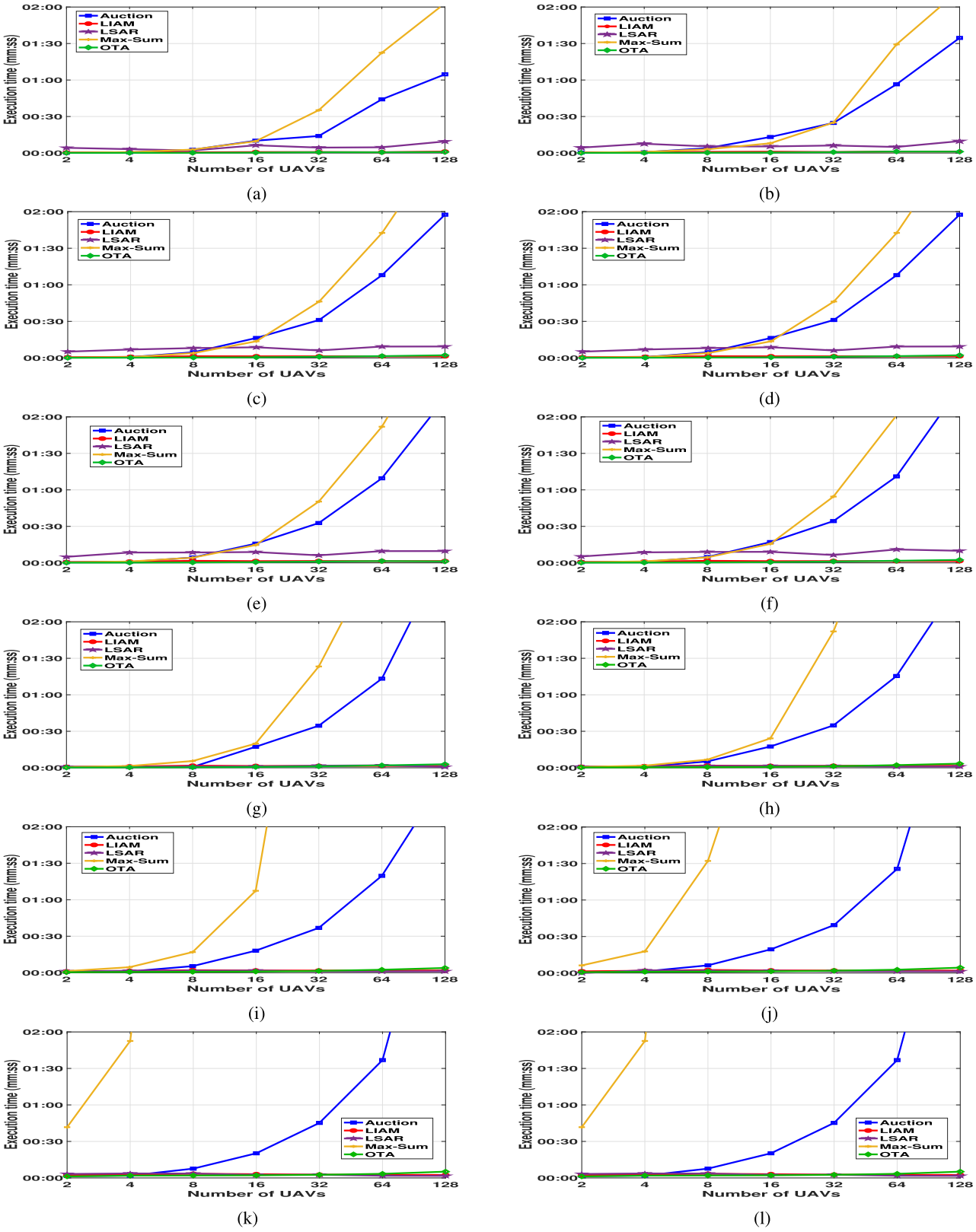


FIGURE 12. Execution time. (a) 2 survivors. (b) 4 survivors. (c) 8 survivors. (d) 16 survivors. (e) 32 survivors. (f) 64 survivors. (g) 128 survivors. (h) 256 survivors. (i) 512 survivors. (j) 1024 survivors. (k) 2048 survivors. (l) 4096 survivors.

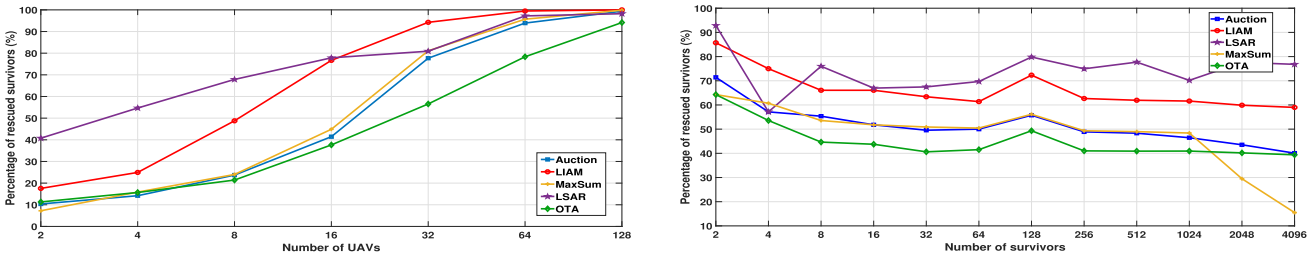


FIGURE 13. Average of results for the percentage of rescued survivors. (a) Different UAVs number. (b) Different survivors number.

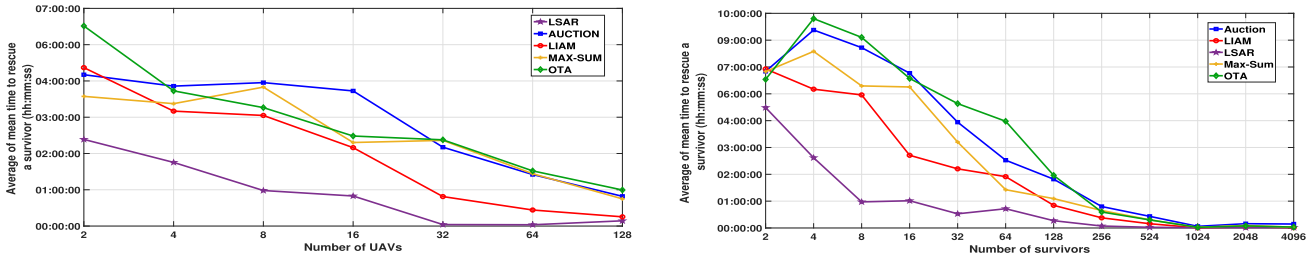


FIGURE 14. Average of results for the mean time to rescue a survivor. (a) Different UAVs number. (b) Different survivors number.

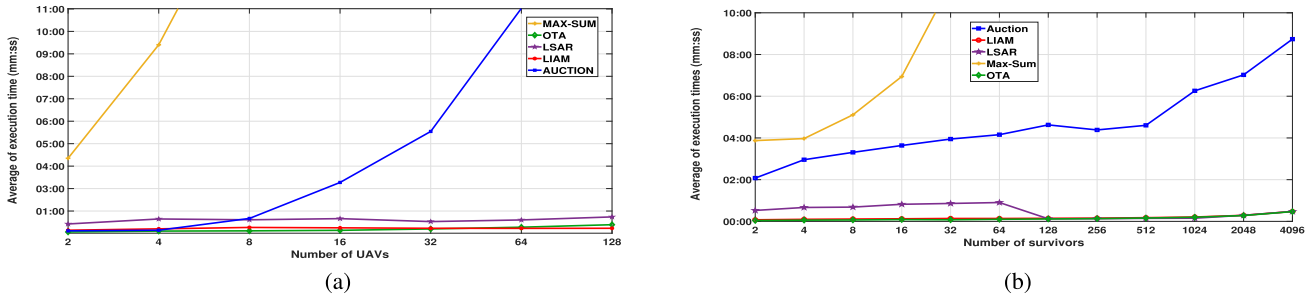


FIGURE 15. Average of results for the execution time. (a) Different UAVs number. (b) Different survivors number.

15-240 min (on average) to rescue a survivor, which is the highest average among other algorithms, with most of the recorded results also being close to that average. Regarding the metric of percentage of rescued survivors, the UAV aircraft running the LSAR algorithm were able to rescue approximately 74% of survivors, and most of the results were around this percentage. However, UAV aircraft running the OTA algorithms only rescued 45% of survivors, whilst requiring the lowest execution time to finish the SAR missions (16 s on average).

The UAV aircraft running the max-sum algorithm required too much time, as the max-sum algorithm is not applicable to dynamic environments. This is clearly shown, as they required an average execution time of approximately 7.5 h.

Figures 16 and 17 present the distribution of empirical results for the LSAR, LIAM, OTA, auction-based, and max-sum algorithms. As shown in Figure 16, the range of the mean time for a UAV aircraft to rescue a survivor was 0:00:00.0-16:48:00.0 running the LSAR algorithm, 0:00:00.0-13:08:00.0 running the LIAM algorithm, 0:00:00.0-19:49:00.0 running the OTA algorithm, 0:00:00.0-18:14:00.0 running the auction-based algorithm, and 0:00:00.0-15:45:00.0 running the max-sum algorithm. Although the range of mean LSAR times to rescue a survivor was longer than some algorithms (such as LIAM and max-sum), most of the LSAR mean times were bounded by 04:50:02.00 with 95% confidence. Figure 17 shows that the percentage of rescued survivors by UAV aircraft running the LSAR and its comparative algorithms ranged between 0% and 100%, but with different distributions. The area under the LSAR curve shows that the results clustered around the 78%-100% range under an exponential behavior curve, which means a small number of results were less than 50% and most of the results were above 50%. All the other curves exhibit almost linear behavior, which means equal distribution of their results.

TABLE 4. Descriptive statistics.

metrics	statistics	LSAR	LIAM	OTA	Auction	Max-Sum
mean-time-to-rescue-a-survivor (hh:mm:ss.ms)	Mean	**1:08:06.8**	2:31:12.2	3:46:35.3	3:34:37.3	2:58:51.9
	STEDEV	**0.1095**	0.1604	0.2259	0.1964	0.17214
Percentage-of-rescued-survivors (%)	Mean	**74%**	66.25%	45.01%	51.52%	47.04%
	STEDEV	**0.2352**	0.3505	0.3147	0.3675	0.37565
Execution Time (hh:mm:ss.ms)	Mean	0:00:52.0	0:00:19.3	**0:00:16.1**	0:08:21.5	7:26:35.1
	STEDEV	0.00041	**0.00014**	0.00018	0.00887	0.91813

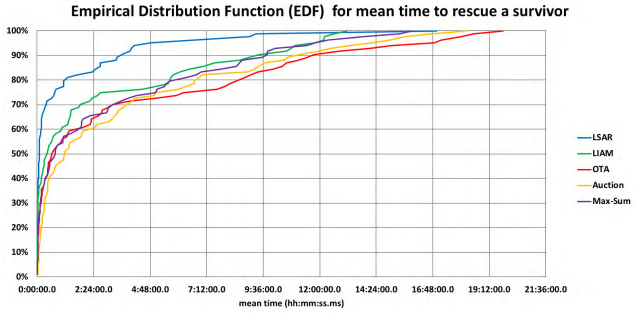


FIGURE 16. ECDF for mean time to rescue a survivor.

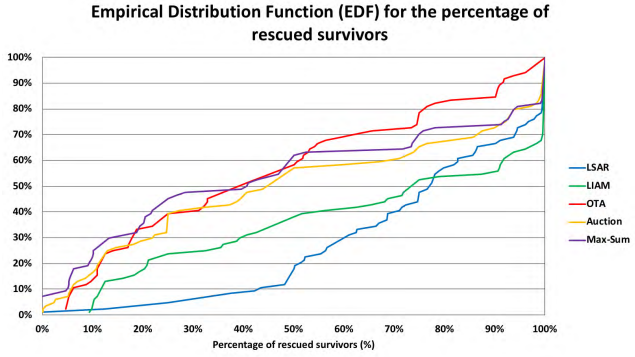


FIGURE 17. ECDF for percentage of rescued survivors.

C. RESPONSIVENESS

In this section, we evaluate how the system responds to different survivor life expectancies. Three-dimensional (3D) performance models that illustrate the responsiveness of the LSAR system (under different running conditions) were generated and presented. Here, the 3D space is defined by a logarithmic base 2 scale for the number of survivors on the x-axis, a linear scale for survivor life expectancies on the y-axis, and by the responsiveness measure in terms of net throughput, mean time to rescue a survivor, or the algorithm execution time on the z-axis. The value of the critical variables that control this evaluation are listed in Table 2.

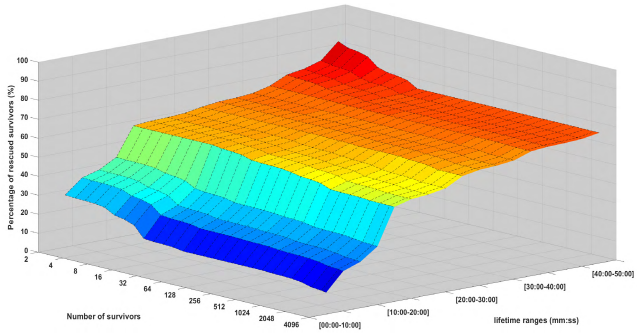


FIGURE 18. Percentage of rescued survivors by LSAR.

1) PERCENTAGE OF RESCUED SURVIVORS (NET THROUGHPUT)

Figure 18 shows that UAV aircraft running the LSAR algorithm rescued the largest percentage of survivors when their number is too low, and they have highest life expectancy; for example, more than 80% of survivors were rescued when there were 8 and their life expectancies were selected randomly between 50 to 60 min. In addition, stable behavior is apparent when the number of survivors increases, and they live between 40 to 50 min. Further, approximately 60%-80% are rescued regardless of the number of survivors. Moreover, another stable behavior is exhibited when the number of survivors increases, and they live between 30 to 40 min; here, approximately 40%-60% are rescued. Therefore, the number of survivors has less impact on the response of the LSAR than their life expectancy ranges. However, a gradual decrease is shown only when the number of survivors increases and they live for very short time (between 10 to 20 min). It should be remembered that the mission time is set to be finished in one month. Further, the lowest percentage of rescued survivors is 0%-20%, which occurs when the number of survivors is 4096 and they only live between 10 to 20 min.

2) MEAN RESCUE TIME

Figure 19 shows that UAV aircraft running the LSAR algorithm rescued survivors faster when they live longer and their numbers are large. Here, UAV aircraft need less than 1 min to rescue survivors. Even though this result seems to be unexpectedly natural, it is acceptable according to the LSAR technique, meaning that when the number of survivors is too low there will be no center to be detected, which makes the LSAR algorithm achieve inferior result. The critical factor affecting LSAR time is the range of survivor life expectancies; as the life expectancy range increases, the surface descends sharply. When the number of survivors is too low, and they live for very short time, the UAV aircraft need approximately 5.75 h to find them.

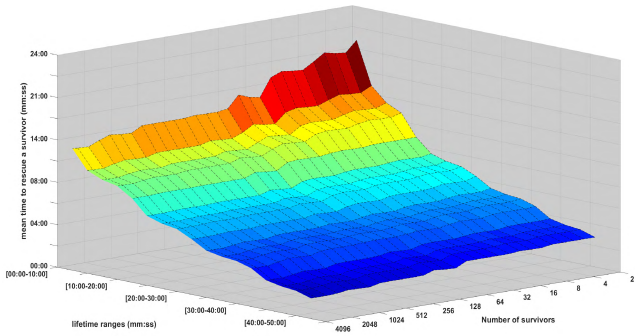


FIGURE 19. Mean time to rescue a survivor by LSAR.

When the number of survivors is too low, and they are in the highest life expectancy window, more than 80% of survivors were rescued when there were 8 survivors and their life expectancies were between 50 and 60 min. In addition, the surface shows stable behavior when the number of survivors increases, and they live between 40 and 50 min; then approximately 60%-80% are rescued. Moreover, another stable behavior is shown when the number of survivors increases, and they live between 30 and 40 min.

VI. CONCLUSION

This paper has presented a novel task-distribution technique for SAR scenarios involving multiple autonomous UAV aircraft. The impact of the proposed techniques has already been discussed through exhaustive empirical experiments and statistics. In summary, the following interesting observations can be deduced from the experiments:

- UAV aircraft running the LSAR algorithm rescue on average approximately 74% of survivors, which is 8% higher than next best algorithm (LIAM), followed by the auction-based and max-sum algorithms, and finally the OTA algorithm, which is worse than LSAR by 28.9%.
- UAV aircraft running the LSAR algorithm require the least amount of time to rescue a survivor, which was 55% less than the next fastest algorithm (LIAM), followed by the max- sum and auction-based algorithms, and finally by the OTA algorithm, which is slower than the LSAR by 77%.
- The empirical cumulative distribution function of the LSAR results show that the percentages of rescued survivors clustered around the 78%-100% range under an exponential curve, which means that most of the results are above 50%. All other algorithms have almost equal percentage distributions for rescued survivors.
- Most of the registered times to rescue survivors by the LSAR algorithm are bounded by 04:50:02 with 95% confidence for a one-month mission time.
- The main factor to be optimized in the SAR problem concerns how early the rescue mission commences.
- The SAR problem is not affected by the number of survivors to be rescued; to be accurate, it is affected by the way they are clustered.

Although this work proposed a novel technique for the SAR problem, there are some limitations and open research problems that still need to be investigated and solved in future studies. One of the major factors that is clearly limiting the performance of the LSAR algorithm is the density of the survivors; in other words, detecting the disaster center in which most of the survivors are located. Hence, using image processing techniques for this purpose offers the best practical results in real-life applications. Moreover, the messages passed between the server and UAV aircraft (such as survivor and disaster locations) are not encrypted. This security issue needs to be addressed. Additionally, deploying the LSAR algorithm on real UAV aircraft for SAR purposes will be conducted in our future work.

REFERENCES

[1] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards smart farming and sustainable agriculture with drones," in *Proc. Int. Conf. Intell. Environ.*, Jul. 2015, pp. 140–143.

[2] R. L. Finn and D. Wright, "Unmanned aircraft systems: Surveillance, ethics and privacy in civil applications," *Comput. Law Secur. Rev.*, vol. 28, no. 2, pp. 184–194, 2012.

[3] T. Wall and T. Monahan, "Surveillance and violence from afar: The politics of drones and liminal security-scapes," *Theor. Criminol.*, vol. 15, no. 3, pp. 239–254, 2011.

[4] T. C. Luciani, B. A. Distasio, J. Bungert, M. Sumner, and T. L. Bozzo, "Use of drones to assist with insurance, financial and underwriting related activities," U.S. Patent 2016 0 063 642 A1, Mar. 3, 2016.

[5] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello, "Mixed-initiative planning and execution for multiple drones in search and rescue missions," in *Proc. ICAPS*, Jun. 2015, pp. 315–323.

[6] D. Câmara, "Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios," in *Proc. IEEE Conf. Antenna Meas. Appl. (CAMA)*, Nov. 2014, pp. 1–4.

[7] L. Apvrille, T. Tanzi, and J.-L. Dugelay, "Autonomous drones for assisting rescue services within the context of natural disasters," in *Proc. 31st Gen. Assem. Sci. Symp. (URSI GASS)*, Aug. 2014, pp. 1–4.

[8] P. Molina *et al.*, "Drones to the rescue!" *Inside GNSS*, Jul./Aug. 2012. [Online]. Available: http://infoscience.epfl.ch/record/180464

[9] B. P. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *Proc. ICRA*, vol. 3, Sep. 2003, pp. 3862–3868.

[10] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004.

[11] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 12, p. 399, 2013.

[12] H. A. Kurdi *et al.*, "Autonomous task allocation for multi-UAV systems based on the locust elastic behavior," *Appl. Soft Comput.*, vol. 71, pp. 110–126, Oct. 2018.

[13] A. Corrêa, "Binary max-sum for clustering-based task allocation in the RMASBench platform," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2016, pp. 1046–1053.

[14] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, and J. A. Rodriguez-Aguilar, "Binary max-sum for multi-team task allocation in RoboCup rescue," in *Optimisation in Multi-Agent Systems and Distributed Constraint Reasoning (OptMAS-DCR)*. Paris, France, 2014.

[15] F. M. D. Fave, A. Rogers, Z. Xu, S. Sukkarieh, and N. R. Jennings, "Deploying the max-sum algorithm for decentralised coordination and task allocation of unmanned aerial vehicles for live aerial imagery collection," in *Proc. IEEE Int. Conf. Robot. Automat.*, May 2012, pp. 469–476.

[16] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings, "Decentralized coordination in robocup rescue," *Comput. J.*, vol. 53, no. 9, pp. 1447–1461, 2010.

[17] E. J. M. Casado, D. Scarlatti, D. Esteban-Campillo, I. Maza, and F. Caballero, "Network of unmanned vehicles," U.S. Patent 8 914 182 B2, Dec. 16, 2014.

[18] H.-L. Choi, L. Brunet, and J. P. How, "Consensus-based decentralized auctions for robust task allocation," *IEEE Trans. Robot.*, vol. 25, no. 4, pp. 912–926, Aug. 2009.

[19] L. Johnson, H.-L. Choi, and J. P. How, "The hybrid information and plan consensus algorithm with imperfect situational awareness," in *Distributed Autonomous Robotic Systems*. Tokyo, Japan: Springer, 2016, pp. 221–233.
[20] P. Segui-Gasco, H.-S. Shin, A. Tsourdos, and V. Segui, "A combinatorial auction framework for decentralised task allocation," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2014, pp. 1445–1450.
[21] S. Trigui *et al.*, "A distributed market-based algorithm for the multi-robot assignment problem," *Procedia Comput. Sci.*, vol. 32, pp. 1108–1114, 2014.
[22] O. Cheikhrouhou, A. Koubâa, and H. Bennaceur, "Move and improve: A distributed multi-robot coordination approach for multiple depots multiple travelling salesmen problem," in *Proc. IEEE Int. Conf. Auton. Robot Syst. Competitions (ICARSC)*, May 2014, pp. 28–35.
[23] A. Koubâa, O. Cheikhrouhou, H. Bennaceur, M.-F. Sriti, Y. Javed, and A. Ammar, "Move and improve: A market-based mechanism for the multiple depot multiple travelling salesmen problem," *J. Intell. Robot. Syst.*, vol. 85, no. 2, pp. 307–330, 2017.
[24] A. Koubâa *et al.*, "Coros: A multi-agent software architecture for cooperative and autonomous service robots," in *Cooperative Robots and Sensor Networks*. Cham, Switzerland: Springer, 2015, pp. 3–30.
[25] M. Alighanbari, "Robust decentralized task assignment algorithms for UAVS," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, MA, USA, 2007.
[26] K. Macarthur, "Multi-agent coordination for dynamic decentralised task allocation," Ph.D. dissertation, Univ. Southampton, Southampton, U.K., 2011.
[27] H. Kurdi, J. How, and G. Bautista, "Bio-inspired algorithm for task allocation in multi-UAV search and rescue missions," in *Proc. AIAA Guid., Navigat., Control Conf.*, 2016, p. 1377.
[28] H. Kurdi, "Patient: Dynamic task allocation in an autonomous multi-UAV mission," US Patent Office, May 2017.
[29] A. Koubâa *et al.*, "Dronemap planner: A service-oriented cloud-based management system for the Internet-of-Drones," *Ad Hoc Netw.*, vol. 86, pp. 46–62, Apr. 2019.
[30] A. Koubaa, M. Alajlan, and B. Qureshi, "ROSLink: Bridging ROS with the Internet-of-Things for cloud robotics," in *Robot Operating System (ROS)*. Cham, Switzerland: Springer, 2017, pp. 265–283.
[31] R. Chaâri *et al.*, "Cyber-physical systems clouds: A survey," *Comput. Netw.*, vol. 108, pp. 260–278, Oct. 2016.
[32] A. Koubaa and B. Quershi, "DroneTrack: Cloud-based real-time object tracking using unmanned aerial vehicles over the Internet," *IEEE Access*, vol. 6, pp. 13810–13824, Mar. 2018.
[33] S. N. A. M. Ghazali, H. A. Anuar, S. N. A. S. Zakaria, and Z. Yusoff, "Determining position of target subjects in maritime search and rescue (MSAR) operations using rotary wing unmanned aerial vehicles (UAVs)," in *Proc. Int. Conf. Inf. Commun. Technol. (ICICTM)*, May 2016, pp. 1–4.
[34] S. Lee and J. R. Morrison, "Decision support scheduling for maritime search and rescue planning with a system of uavs and fuel service stations," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2015, pp. 1168–1177.
[35] M. Rahmes, D. Chester, J. Hunt, and B. Chiasson, "Optimizing cooperative cognitive search and rescue UAVs," *Proc. SPIE*, vol. 10643, 2018, Art. no. 106430T.
[36] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 33, no. 3, pp. 367–385, Jun. 2003.
[37] S. Kohlbrecher *et al.*, "Towards highly reliable autonomy for urban search and rescue robots," in *Robot Soccer World Cup*. Cham, Switzerland: Springer, 2014, pp. 118–129.
[38] J. Q. Cui *et al.*, "Drones for cooperative search and rescue in post-disaster situation," in *Proc. IEEE 7th Int. Conf. Cybern. Intell. Syst. (CIS) IEEE Conf. Robot., Automat. Mechatronics (RAM)*, Jul. 2015, pp. 167–174.
[39] C. K. Grissom, F. Thomas, and B. James, "Medical helicopters in wilderness search and rescue operations," *Air Med. J.*, vol. 25, no. 1, pp. 18–25, 2006.
[40] J. Sun, B. Li, Y. Jiang, and C.-Y. Wen, "A camera-based target detection and positioning UAV system for search and rescue (SAR) purposes," *Sensors*, vol. 16, no. 11, p. 1778, 2016.
[41] A. Al-Kaff, M. J. Gómez-Silva, F. M. Moreno, A. de la Escalera, and J. M. Armingol, "An appearance-based tracking algorithm for aerial search and rescue purposes," *Sensors*, vol. 19, no. 3, p. 652, 2019.
[42] C. Nattero, C. Recchiuto, A. Sgorbissa, and F. Wanderlingh, "Coverage algorithms for search and rescue with uav drones," in *Proc. 13th Workshop AIIA Symp. Artif. Intell.*, 2014.
[43] S. Koenig and R. G. Simmons, "Easy and hard testbeds for real-time search algorithms," in *Proc. AAAI/IAAI*, vol. 1, Aug. 1996, pp. 279–285.
[44] R. E. Korf, "Real-time heuristic search," *Artif. Intell.*, vol. 42, nos. 2–3, pp. 189–211, 1990.
[45] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Ann. Math. Artif. Intell.*, vol. 31, nos. 1–4, pp. 41–76, 2001.
[46] M. Baglietto, G. Cannata, F. Capezio, A. Grosso, A. Sgorbissa, and R. Zaccaria, "Patrolgraph: A distributed algorithm for multi-robot patrolling," in *Proc. 10th Int. Conf. Intell. Auton. Syst.*, Baden, Germany, 2008, pp. 415–424.
[47] C. Wei, K. V. Hindriks, and C. M. Jonker, "Dynamic task allocation for multi-robot search and retrieval tasks," *Appl. Intell.*, vol. 45, no. 2, pp. 383–401, 2016.
[48] M.-C. Chen, C.-H. Chen, M.-S. Huang, J.-Y. Ciou, and G.-T. Zhang, "Design of unmanned vehicle system for disaster detection," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 4, 2015, Art. no. 784298.
[49] C. Baker, G. Ramchurn, L. Teacy, and N. Jennings, "Planning search and rescue missions for uav teams," *Univ. Southampton Institutional Repository*, vol. 285, pp. 1777–17782, Jun. 2016.
[50] H. Unabor, "Geospatial response with remote sensing, GIS, OpenStreetMap and Ushahidi: The Haiti earthquake of 12th January, 2010," *Int. J. Sci. Eng. Res.*, vol. 5, no. 2, pp. 250–257, 2014.
[51] *Shahad Illustration Demo for LSAR: Multi-Drone Collaboration for Search and Rescue Missions*. Accessed: Apr. 4, 2019. [Online]. Available: https://www.youtube.com/watch?v=3uJcn2hzkuU
[52] G. Bautista. (2016). *Masplanes*. [Online]. Available: https://github.com/guille-byte/MASPlanes

EBTEHAL TURKI ALOTAIBI received the B.S. degree in computer science from Al-Imam Mohammad Ibn Saud Islamic University, Riyadh, Saudi Arabia, in 2014, and the M.S. degree in computer science (Artificial intelligence field) from Mohammad Ibn Saud Islamic University, in 2016. From 2012 to 2015, she was a Research Assistant with Non-clairvoyant Scheduler for Hybrid IaaS Clouds (Novel-HIC) Project, Science and Technology Unit, Riyadh. Since 2014, she has been a Lecturer with the Computer Science Department, Al-Imam Mohammad Ibn Saud Islamic University. She is the author of over eight articles in highly rated journals and conferences. Her research interests include NP-Hard problem, optimization algorithms, and robot/multi-robot applications. She received the Excellent Research Award from Al-Imam Mohammad Ibn Saud Islamic University for five publications, in 2017, and for two publications, in 2018.

SHAHAD SALEH ALQEFARI received the M.Sc. degree from King Saud University, Saudi Arabia, in 2015. She is currently a Lecturer of computer science. Her current research interests include integrating intelligence algorithm for drones and robots solution, in the context of multi-task allocation, security, and robot operating systems (ROS).

ANIS KOUBAA received the M.Sc. degree from University Henri Poincar'e, France, in 2001, and the Ph.D. degree from INPL, France, in 2004. He is currently a Professor of computer science, an Aide to Rector of Research Governance, and the Director of the Robotics and Internet of Things Research Lab, Prince Sultan University. He is also a Senior Researcher with CISTER/INESC and ISEP-IPP, Porto, Portugal, and a Research and Development Consultant with Gaitech Robotics, China. His current research interests include providing solutions towards the integration of robots and drones into the Internet of Things (IoT) and clouds, in the context of cloud robotics, robot operating systems, robotic software engineering, wireless communication for the IoT, real-time communication, safety and security for cloud robotics, intelligent algorithms design for mobile robots, and multi-robot task allocation. He is also a Senior Fellow of the Higher Education Academy, U.K. He has been the Chair of the ACM Chapter, Saudi Arabia, since 2014.