Area-Optimized UAV Swarm Network for Search and Rescue Operations

Laik Ruetten
Dept. of Computer Science
Univ. of Wisconsin-La Crosse
La Crosse, Wisconsin, USA
laik.ruetten@gmail.com

Paulo Alexandre Regis
Dept. of Computer Science
Southeastern Louisiana Univ.
Hammond, Louisiana, USA
pregis@southeastern.edu

David Feil-Seifer
Computer Science & Eng.
Univ. of Nevada, Reno
Reno, Nevada, USA
dave@cse.unr.edu

Shamik Sengupta
Computer Science & Eng.
Univ. of Nevada, Reno
Reno, Nevada, USA
ssengupta@unr.edu

***Abstract*—Intelligent robot swarms are increasingly being explored as tools for search and rescue missions. Efficient path planning and robust communication networks are critical elements of completing missions. The focus of this research is to give unmanned aerial vehicles (UAVs) the ability to self-organize a mesh network that is optimized for area coverage. The UAVs will be able to read the communication strength between themselves and all the UAVs it is connected to using RSSI. The UAVs should be able to adjust their positioning closer to other UAVs if RSSI is below a threshold, and they should also maintain communication as a group if they move together along a search path. Our approach was to use Genetic Algorithms in a simulated environment to achieve multi-node exploration with emphasis on connectivity and swarm spread.**

***Index Terms*—UAV networks, UAV swarm, genetic algorithm, search and rescue, autonomous robotics**

I. Introduction

UAVS are used today for search and rescue (SAR) missions, almost always requiring a human operator. SAR missions aim to search for victims, survey terrain, or perform supply drops. In 2006, in the aftermath of Hurricane Katrina, two UAVs were used to survey the damaged area in the search for trapped survivors [1]. UAVs are preferable to use rather than human rescuers for many reasons. Victims are usually stuck in dangerous areas, and human rescuers should not have to be in those areas too for longer than they have to (if at all). UAVs can also move significantly quicker than humans, and therefore this speeds up missions significantly.

Theoretically, the more UAVs deployed to a search area, the faster an area could be searched, because each additional UAV adds more coverage. However, it would be very difficult for a person to operate more than one UAV at once. It would also be ideal to not be limited by the number of human operators available. If UAVs could operate and coordinate themselves, more UAVs could be used without the limitations of using human operators [2].

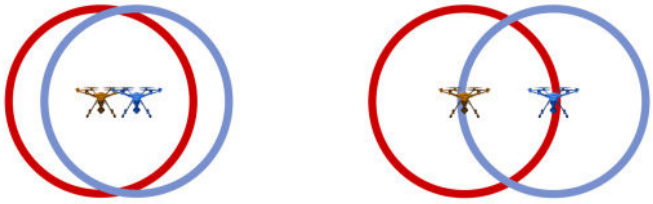
In order for intelligent UAV swarms to be successful, they must be able to organize and maintain their own communication networks. Intelligent UAV swarms can have many components working together to make up the full system. Those components could be flight control, anti-collision systems, sensor interpretation, navigation algorithms, or wireless communication. Wireless communication is important because many other components rely on it. Sensor data is useless unless it can be retrieved from the UAV. Navigation and search algorithms cannot work efficiently if UAVs do not know where each other are [3]. UAV swarms heavily relies on wireless communication for success.

Our research focus is to give each UAV within a swarm the ability to self-organize itself with focus on:

- maximizing search area
- maintaining wireless intercommunication

What this research defines as area optimization is shown in Fig. 1. UAVs need to ensure that they are within communication range of each other, but not so close that they are losing area coverage. There must be a balance between ensuring robust communication networks and optimizing the area coverage of the swarm.

This research was inspired directly off of previous work for mesh topology robot networks [4]. It is a system for UAVs to be able to rely on other UAVs for positioning, as if there is an elastic band between them. One robot drives forward, and the other corrects its path once the RSSI (Received Signal Strength Indicator) between the two drops below a threshold. However, each robot can only rely on one other robot for positioning data, and each robot sends positioning data to only one other robot. This manifests into only supporting straight line formations.



(a) Not Optimal Coverage (b) Optimal Area Coverage

Fig. 1: The rings around the UAVs represent the range of communication the UAVs have. If UAVs are too close together, they are not optimizing the area coverage that they could achieve.

978-1-7281-3783-4/20/$31.00 ©2020 IEEE

Authorized licensed use limited to: NORTHWESTERN POLYTECHNICAL UNIVERSITY. Downloaded on October 15,2021 at 06:50:40 UTC from IEEE Xplore. Restrictions apply.

Our research differs from the previous work because we created a system for UAVs to rely on any amount of UAVs for positioning data, rather than only sending to one UAV and retrieving from another UAV. This also allows UAVs to be connected to each other in any arrangement possible, not just a straight line.

This new system lays the groundwork for how more than two UAVs would behave. If a UAV is connected to multiple other UAVs, it will try to stay in range of all of them. This system also works to prevent UAVs of the same swarm from colliding into each other. There is not a leading coordinator UAV, but instead all of the UAVs adjust relative to every other UAV that it is relying on.

The remainder of this paper is organized as follows. Section II provides a literature review of related work. In Section III we propose the new system model and assumptions for this paper. Section IV describes an area-optimized search and rescue method using Genetic Algorithms. Section V provides validation results and discussion. Finally, Section VI concludes the paper.

II. Background

In researching background for this project, we looked at what researchers have done previously in the field of search and rescue UAV swarms. We are building directly off of the work of [4], so we must spend some time explaining what that research accomplished and where it fell short. After identifying what that research fell short on, we searched for other similar research problems that we could pull from to form our solution.

A. RSSI Mesh Network Foundations

The program architecture in [4] lays the foundation for a self-sustaining mesh network using RSSI. This paper calls its method a *Left-Hand Search* or *Rubber-Band* method. The *Left-Hand* search method is a common method implemented by fire departments. A lead fireman follows the left wall of a building, while the rest of the firefighters are interlocked via elastic bands to increase the search area and simultaneously ensure the safety of the firemen. The technique in [4] works metaphorically the same way. The program creates a mesh topology wireless network of UAVs, and then each UAV is assigned another UAV to rely on for positioning (appropriately named *Rely_On*). The UAVs use RSSI as a virtual elastic rubber-band.

However, [4] falls short of providing the full software architecture of a multi-UAV system. Each individual UAV only knows about the positioning of one other UAV. This is fine when there are only two UAVs, but problems arise when more than two UAVs are in the swarm. There nothing preventing a UAV from crashing into a UAV that it is not aware of. We improve the functionality in [4] by adding support for functionality of a multi-UAV system.

B. Other Autonomous Swarm Implementations

Virtual Spring Mesh systems have been used to ensure the distance between UAVs, but it uses distance rather than wireless signal strength [2]. This method uses Hooke's law of $F = -k * x$. This equation describes a spring force using displacement x as input and k as a constant. Therefore, the *distance* between UAVs determines UAV positioning. Using this law to describe spring behavior, the UAVs are coded to believe that they are connected to each other with virtual springs, which then affects their movements to ensure they stay spread out from each other [5]. This is a clever method, but it is based on distance between UAVs instead of communication strength. The problem with this system is that even if UAVs have solid signal strength, Hooke's law deters UAVs from travelling farther away to increase range. Similarly, if UAVs have too weak of a signal strength, the UAVs are deterred from getting closer together, even if they need to. Spring systems based on distance deter the UAVs from changing their positioning when they need to for the sake of communication.

Our system is different because it uses RSSI rather than using only distance. External motion capture to measure distance is not going to be available in real world disasters, so we must be able to use the communication strength metric for UAVs to change their positioning on their own. Strictly using distance is not a good method to use because communication strength will drop when there are obstacles in between UAVs, so the UAVs should move closer together when that happens.

Our algorithm must be able to support many different methods of search. [6] describes different methods of search, including fully teleoperated or fully autonomous. There are different strategies of search, such as the swarm expanding from a point, or the swarm traveling together along a narrow path. No matter what the strategy, the algorithm we develop must be able to support it.

In a catastrophe scenario, it is crucial to act within the first couple of hours. It is when we have the highest probability to rescue people alive. That is why it is important for our algorithm to work quickly and to adapt to the environment. Mudslides, explosions, forest fires, or any other disasters will cause the environment to be unpredictable. The UAV swarm must have the ability to search an area without knowledge of the exact features beforehand. UAVs must also be able to help autonomously, otherwise they will be more of a burden than they are worth once a disaster blows through [7].

One possible method of doing this search that can adapt to environments is using an application of genetic algorithms (GA). GAs are a strategy used in the past to plan the paths of UAVs. In previous work, GAs have successfully planned the route for one UAV in a known environment with obstacles [8]. GAs have also been successful in multi-UAV mission planning [9]. We hope to use a similar method using GAs for giving UAV swarms the ability to travel in unknown environments. Previous swarming algorithms for exploring uncertain environments work by modeling behaviors of animals in the wild, such as ant colonies [10]. We hope to make the GAs run on board the UAVs, but that is for future work [11]. It might be better to first use a remote server to run iterations of the GA and then give commands to the UAVs remotely.

Authorized licensed use limited to: NORTHWESTERN POLYTECHNICAL UNIVERSITY. Downloaded on October 15,2021 at 06:50:40 UTC from IEEE Xplore. Restrictions apply.

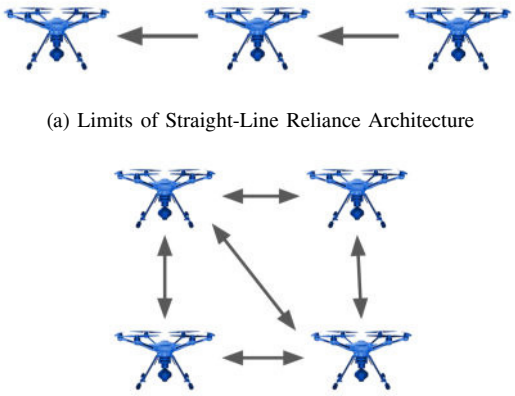
III. Proposed method

The previous system [4] we will refer to as the Straight-Line Reliance Architecture. The Straight-Line Reliance Architecture works by assigning one UAV as the coordinator UAV. Then the RSSI value is measured from the coordinator to every other UAV in the swarm. The IDs of the UAVs are then placed into a one-dimensional array, which is sorted by the measured RSSI value to the coordinator.

After the array is sorted, the reliance architecture is assigned. For every UAV at index i, the UAV will rely on the UAV at index $i-1$ (one index to the left). The UAV will store the ID of the UAV it relies on in a variable appropriately named *rely_on*. Similarly, the UAV will be assigned to send its positioning data to the UAV at index $i+1$ (one index to the right). The UAV will store the ID of the UAV it is assigned to send positioning data to in a variable appropriately named *send_to*. The coordinator will always have an RSSI value of 0 to itself, so it will be first in the matrix and will not have a *rely_on* UAV. Similarly, the UAV with the weakest signal to the coordinator will be at the end of the array and not have a *send_to* UAV.

Algorithm 1 Mesh Reliance Architecture Assignment

Input : A 2-D array M of all zeros of size $num\_UAVs \times Num\_UAVs$
Output: M as a valid connectivity matrix
for $i \leftarrow 0$ **to** num_UAVs **do**
  for $j \leftarrow 0$ **to** i **do**
    $M[j][i] \leftarrow 1$
  end
end
for $i \leftarrow 0$ **to** num_UAVs **do**
  // Create a set of integers such that $\forall j,\ M[i][j] == 1$
  $possible\_values \leftarrow \{j | (0 \leq j < Num\_UAVs)$ **and** $(M[i][j] \neq 0)$ **and** $(j \in \mathbb{Z})\}$
  if $possible\_values \neq \varnothing$ **then**
    $choice \leftarrow$ random.choice($possible\_values$);
  else $choice \leftarrow -1$;
  if $choice \neq -1$ **then**
    // Index choice is guaranteed to remain a 1
    $possible\_values$.remove($choice$)
  end
  // Remaining choices have a possibility of not being connections (probability related to the number of UAVs)
  for $value \in possible\_values$ **do**
    if *random.random() > 1.0/Num_UAVs* **then**
      $M[i][value] \leftarrow 0$;
    else
      $M[i][value] \leftarrow 1$;
  end
end
// reflect over diagonal axis of matrix to make symmetric **for** $i \leftarrow 0$ **to** num_UAVs **do**
  for $j \leftarrow 0$ **to** i **do**
    $M[i][j] \leftarrow M[j][i]$
  end
end



(a) Limits of Straight-Line Reliance Architecture

(b) Possibilities with Mesh Reliance Architecture

Fig. 2: A previous system [4] enables UAVs to rely on another UAV for positioning. Our work adds support for UAVs to rely on more than one UAV, which means support for more swarm configurations. The arrows point from a UAV to the UAV that it relies on.

The system that we developed we will refer to as the Mesh Reliance Architecture. The Mesh Reliance Architecture works by assigning the swarm a connectivity matrix (or two-dimensional array), with each UAV's connections being represented by row of the matrix. Instead of sending positioning data in just one direction (from a UAV to the UAV's *send_to*), each reliance connection in this system sends information both ways (UAVs send positioning data to each other).

The concept is that instead of one coordinator UAV instructing an entire swarm, this system instead makes each UAV act independently and adjust its positioning relative to its neighbors. Each row-column location in the matrix corresponds to a reliance connection in the swarm. For example, if there is a reliance between UAV 0 and UAV 3, then there will be a 1 at *connectivity_matrix*[0][3]. If there was no connection, there would be a 0. Fig. 3 is an example of a connectivity matrix for an 8 UAV swarm.

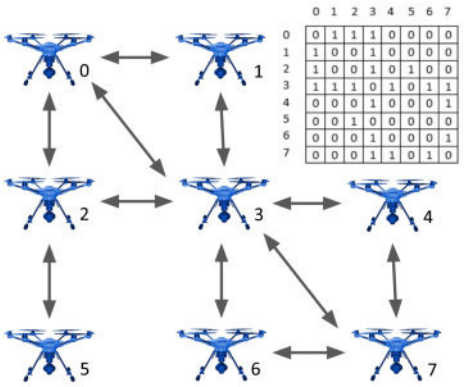


Fig. 3: Connectivity matrix of a particular 8 UAV swarm. Each row-column combination represents if there is a connection between UAVs. Each UAV stores its corresponding row as an array.

Authorized licensed use limited to: NORTHWESTERN POLYTECHNICAL UNIVERSITY. Downloaded on October 15,2021 at 06:50:40 UTC from IEEE Xplore. Restrictions apply.

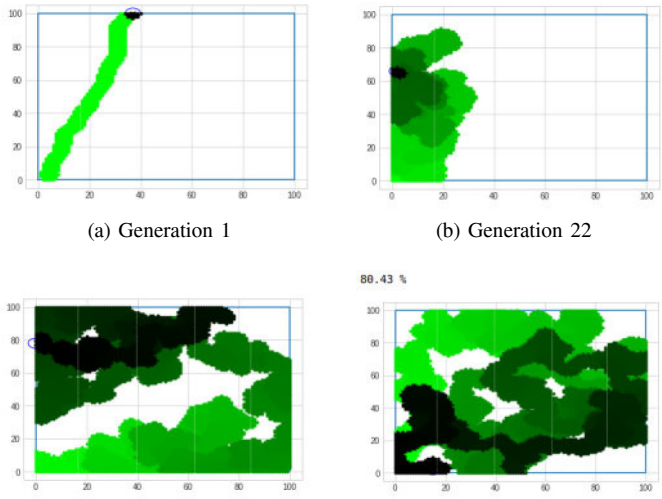
Algorithm 1 shows a method of assigning a reliance structure to a swarm. The reliance is assigned between UAVs randomly. The algorithm requires input of an empty (i.e. containing only zeros) two-dimensional array, or matrix. The output will be a valid connectivity matrix. A connectivity matrix is valid if there is at least one '1' in every row and every column. This ensures that each UAV is connected to at least one other UAV, and it also ensures that all UAVs are connected in one swarm (there are not multiple unconnected swarms). Additionally, the diagonal of the matrix must contain only '0's (a UAV cannot rely on itself for positioning). There are alternative ways this algorithm could be implemented. This algorithm could run on one UAV and then that UAV would send the output data to the other UAVs (each UAV storing its corresponding row of the matrix). Alternatively, a remote server that controls all the UAVs could perform the algorithm and contain all the data. This is just one way a valid connectivity matrix could be created. A different algorithm could be used as long as the algorithm outputs a valid connectivity matrix.

IV. Swarm Control Simulation

Once the new Mesh Reliance Architecture system was created, we validated the method through simulation. As an example of a method of how to control the movement of this type of swarm, we developed a method that is a combination of neural networks and genetic algorithms (GA) [12]. GAs have been successfully used to control UAV swarms before, but our exact method is different [13]. The neural network has inputs such as its (x, y) coordinate positioning and signal strength to its reliance neighbors. Its two outputs are x and y directions to move. The simulation uses a genetic algorithm involving mutation and asexual reproduction to make slight adjustments to the neural network in an attempts to make the UAV behave better. We tested this control system with area travel and with spreading out a swarm. Future work will combine travel and spread so that it can control a swarm of UAVs as it moves.

For an accurate simulation to be implemented, the characteristics of real-world equipment needs to be modeled. In the simulation, we used the equation of $RSSI = |10{*}2{*}log(dist){+}1|$ to translate the UAVs' distances from each other into an RSSI connection value. This equation is derived from the distance-RSSI relation equation $dist = 10^{((TxPower-RSSI)/(10*n))}$. n was given the value of 2, which is the value of n in free space. $TxPower$ is arbitrarily assumed to be 1 (since this is just a test simulation it does not matter what power level the communication transmits at, it just has to transmit). We also only looked at the magnitude of the RSSI; we did not care so much about the negative or positive value, which refers to the direction the signal is traveling.

After the characteristics and model of the equipment are decided upon, we are able to create a simulation of the UAV swarm. The characteristics of a single link will be the basis for all connections in this simulated network. The virtual rubber-band system still works just like the work done in [4]. If a simulated UAV attempts to move a direction that would put the UAV's RSSI value between its reliance neighbors below a threshold, the UAV will not move in that direction.



(a) Generation 1 (b) Generation 22

(c) Generation 92 (d) Generation 343

Fig. 4: Shown above is one run of a simulated UAV over many generations. The max area searched was 80.43%, which is typical for this algorithm. The UAV is simulated for the same amount of time each generation, and survives if it covers more area than the previously fittest UAV. The lighter green represents earlier in the path, and the darker is later on. This correlates to the battery life of the UAV over time.

A. Searching An Area

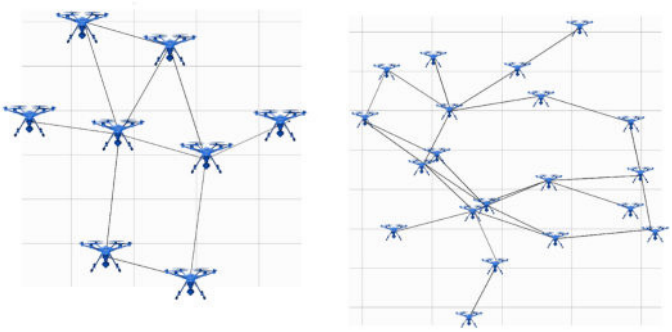
The first focus of the project is to give UAVs the ability to search an uncertain area. In emergency situations, there is no time to map out an area for UAVs to search, and if an explosion or storm did decent damage, their environment will be unpredictable. For this reason, UAVs must be able to be able to search environments without prior knowledge of the landscape.

The method attempted was giving each UAV within the simulation its own neural network, with inputs such as the UAV's coordinate positioning and the communication strength with its neighbors. The neural network outputs two values to tell which way the UAV should move (an x direction and a y direction). The UAV swarms are assigned a *fitness* value, which is based on the percentage of the search area that was searched in a certain time limit. This limit is implemented to simulate the UAV's battery life and the urgency of disaster situations. A GA is used to select for swarms with good *fitness* values, and to make adjustments to the good neural networks in hopes of making them better. Fig. 4 shows the results of one UAV attempting to search as much of an area on its own as it can within a time limit.

B. Optimal Area Coverage

The next focus of the project is to make the swarm spread out as far as possible without UAVs losing connectivity to each other. First, the system only consisted of two UAVs. The UAVs move outward from a point as far as possible, and then stop when they read that their RSSI is too low. This was able to be done in practice with two Erle-brain unmanned ground vehicles driving away from each other.

Authorized licensed use limited to: NORTHWESTERN POLYTECHNICAL UNIVERSITY. Downloaded on October 15,2021 at 06:50:40 UTC from IEEE Xplore. Restrictions apply.



(a) Results with 8 UAVs (b) Results with 20 UAVs

Fig. 5: The lines show how the UAVs are connected (which UAVs rely on each other). Some UAVs still end up close together due to not seeing UAVs that they do not rely on. Future work will make the UAVs be able to change their reliance structure mid-flight.

Afterwards, support for any amount of UAVs is implemented in the simulation environment, and then tested for different amounts of UAVs. The simulation in this step uses the same neural networks and GA as the Search Area implementation, except with an altered *fitness* function. This time, the *fitness* value is based on the sum of the distances between all the UAVs [14]. Fig. 5 shows results of the algorithm spreading out the positioning of 8 drones and 20 drones.

V. Results and Discussion

The existing technology from [4] uses a one-dimensional array to assign the architecture of the swarm. The "architecture of a swarm" refers to which UAVs rely on other UAVs for positioning. Every UAV relies on the UAV one index to the left, and sends positioning data to the UAV one index to the right. The UAV at the beginning of the list is the coordinator and does not rely on another UAV. The UAV at the end of the list does not have anyone to send to. However, the problem with this system is that even if the initial start of the network is a random mesh, the UAV network will eventually assume that straight line instead of a mesh network formation. This is not good because we want the swarm to be able to assume and remain in different formations.

Instead of the Straight-Line Reliance Architecture, we improved the program by implementing a Mesh Reliance Architecture instead. This architecture is better than the Straight-Line Reliance Architecture because it supports more arrangements of UAVs, and therefore more search formations.

In terms of optimal area coverage, each solution the algorithm comes up with just needs to be a good solution for the mesh; it does not have to be the best. Later solutions from the algorithm should be better than earlier solutions (in Fig. 6, we can see this progress slow in later generations).

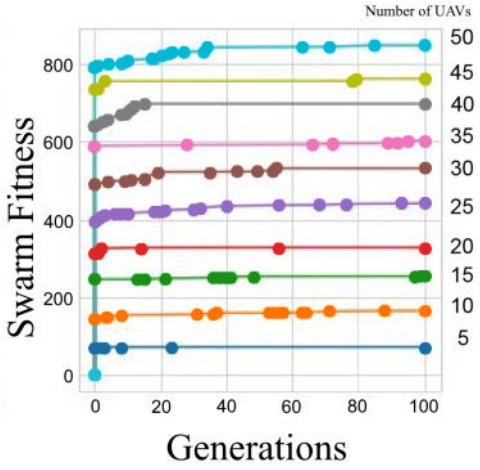


Fig. 6: Fitness vs Generations; Each different colored line is the graph of a different number of UAVs. Each line is labeled with the number of UAVs it represents along the right side. From this graph, we see there is not a correlation between number of UAVs and fitness progress in the algorithm.

For a set amount of generations, time complexity of the area spread algorithm ended up being linear. The algorithm was ran for different amounts of UAVs between 5 UAVs and 100 UAVs, incrementing by 5. Time in seconds was measured between the start of the algorithm and when the algorithm finished. These experiments were run on an Ubuntu machine with an Intel Core i7-4770 3.4GHz processor and 16 GB of RAM. No other programs were running on the machine while the code was running.

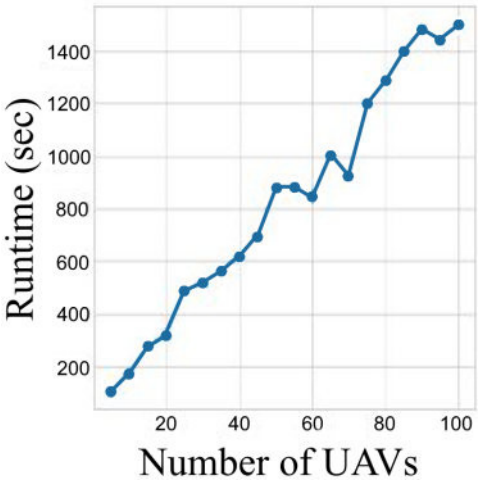


Fig. 7: Runtime vs Number of UAVs; There is a linear correlation between runtime and number of UAVs for the same amount of generations. This specific graph used 300 generations.

Authorized licensed use limited to: NORTHWESTERN POLYTECHNICAL UNIVERSITY. Downloaded on October 15,2021 at 06:50:40 UTC from IEEE Xplore. Restrictions apply.

VI. Conclusions and Future Work

The Straight-Line Architecture created in [4] was a great foundation to build off of, but it had its shortcomings. We successfully improved upon the system with the creation of the Mesh Reliance Architecture. This Mesh Reliance Architecture gives support for more swarm formations than straight lines while still ensuring that every UAV is connected to the swarm.

This architecture was tested in simulations using genetic algorithms and neural networks. Neural networks were used because of their use in other control systems, such as autonomous cars [15]. GA methods of exploring search areas and spreading out swarms were created and evaluated. GAs were used because of their skill at solving optimization problems. We found that our methods are a valid method of organizing a swarm, but not as good for swarm travel. We also found that the addition of each drone only increased computation time linearly (we considered it a success that it was not exponential). The theoretical work performed here will guide the future practical design and implementation. One candidate is the use of Crazyswarm technology, which was a huge inspiration for this work [16]. That system will undoubtedly introduce more obstacles for us to overcome.

Emergency situations need code to run as quickly as possible. The GA method runs very slowly, so as is, it is not very good solution. However, the GA itself has not been tuned in anyway. It was made to simply work as a proof of concept. Future work will improve upon this GA by experimenting with different mutation rates, methods of reproduction, and other strategies to get the swarms to evolve to better solutions more quickly.

More possible future fixes to run time include:

- Each swarm gets one shared neural network instead of each UAV getting their own
- Switch to a method less computationally intensive than neural networks
- Use other optimization techniques other than Genetic Algorithms (GAs are slow; other swarm control systems may be faster [17][18])
- Run simulation in parallel across multiple computers. The main factor that makes run-time longer is the addition of more UAVs to the simulation (Fig. 7). In the field, there will be multiple UAVs with individual computers, so running computations in parallel will be closer to a real-world implementation. Theoretically, run-time should not increase with each new UAV addition because it is also the addition of another computer to do computational work.

The main shortcoming that we hope to address is that the Mesh Reliance Architecture stays static after its initial configuration by Algorithm 1. The problem with this is in situations such as the one shown in Fig. 5b. All the UAVs that rely on each other are appropriately distanced apart, but UAVs that do not rely on each other have nothing preventing them from colliding. In future work, we hope to make our Mesh Reliance Architecture dynamic, meaning that UAVs can add and remove UAVs from its list of UAVs it relies on.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1757929. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *2010 International Conference on Emerging Security Technologies*. IEEE, 2010, pp. 142–147.

[2] V. Lomonaco, A. Trotta, M. Ziosi, J. d. D. Y. Ávila, and N. Díaz-Rodríguez, "Intelligent drone swarm for search and rescue operations at sea," *arXiv preprint arXiv:1811.05291*, 2018.

[3] S. Waharte, N. Trigoni, and S. Julier, "Coordinated search with a swarm of uavs," in *2009 6th ieee annual communications society conference on sensor, mesh and ad hoc communications and networks workshops*. IEEE, 2009, pp. 1–3.

[4] T. Brodeur, P. Regis, D. Feil-Seifer, and S. Sengupta, "Search and rescue operations with mesh networked robots," in *Proceedings-International Conference on Computer Communications and Networks*, 2018.

[5] M. Di Felice, A. Trotta, L. Bedogni, K. R. Chowdhury, and L. Bononi, "Self-organizing aerial mesh networks for emergency communication," in *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. IEEE, 2014, pp. 1631–1636.

[6] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello, "Mixed-initiative planning and execution for multiple drones in search and rescue missions," in *Twenty-Fifth International Conference on Automated Planning and Scheduling*, 2015.

[7] D. Câmara, "Cavalry to the rescue: Drones fleet to help rescuers operations over disasters scenarios," in *2014 IEEE Conference on Antenna Measurements & Applications (CAMA)*. IEEE, 2014, pp. 1–4.

[8] V. R. Ragusa, H. D. Mathias, V. A. Kazakova, and A. S. Wu, "Enhanced genetic path planning for autonomous flight," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 1208–1215.

[9] J. Tian, L. Shen, and Y. Zheng, "Genetic algorithm based approach for multi-uav cooperative reconnaissance mission planning problem," in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2006, pp. 101–110.

[10] F. Yang, X. Ji, C. Yang, J. Li, and B. Li, "Cooperative search of uav swarm based on improved ant colony algorithm in uncertain environment," in *2017 IEEE International Conference on Unmanned Systems (ICUS)*. IEEE, 2017, pp. 231–236.

[11] H. D. Mathias and V. R. Ragusa, "A multi-objective genetic algorithm for path planning with micro aerial vehicle test bed," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2016, pp. 59–82.

[12] F. H.-F. Leung, H.-K. Lam, S.-H. Ling, and P. K.-S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Transactions on Neural networks*, vol. 14, no. 1, pp. 79–88, 2003.

[13] A. Majd, A. Ashraf, E. Troubitsyna, and M. Daneshtalab, "Integrating learning, optimization, and prediction for efficient navigation of swarms of drones," in *2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 2018, pp. 101–108.

[14] K. Daniel, S. Rohde, N. Goddemeier, and C. Wietfeld, "Cognitive agent mobility for aerial sensor networks," *IEEE Sensors Journal*, vol. 11, no. 11, pp. 2671–2682, 2011.

[15] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[16] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304.

[17] P.-Y. Yin, F. Glover, M. Laguna, and J.-X. Zhu, "Cyber swarm algorithms–improving particle swarm optimization using adaptive memory strategies," *European Journal of Operational Research*, vol. 201, no. 2, pp. 377–389, 2010.

[18] K. O. Jones, "Comparison of genetic algorithm and particle swarm optimization," in *Proceedings of the International Conference on Computer Systems and Technologies*, 2005, pp. 1–6.

Authorized licensed use limited to: NORTHWESTERN POLYTECHNICAL UNIVERSITY. Downloaded on October 15,2021 at 06:50:40 UTC from IEEE Xplore. Restrictions apply.