SOSUAS: Stability Optimized Swarming for Unmanned Aerial Systems

Troy Neubauer and Mustafa İlhan Akbaş
Department of Electrical Engineering and Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, Florida
neubaut1@my.erau.edu, akbasm@erau.edu

***Abstract*—Unmanned aerial systems (UASs) have unprecedented potential in various fields, from surveillance to packet delivery. The swarms formed by multiple UAS increase the number of potential UAS applications as they can be utilized for complicated tasks. Therefore, various swarming methodologies are proposed to coordinate the UAS positions in swarms. Virtual forces and molecular geometry based approaches have been instrumental in UAS swarming applications. These approaches apply forces between UAS nodes similar to electron pairs for near-ideal volume coverage. However, the ratio of these forces and the stability of the node positions in the network have been vexing challenges. In this paper, we propose an approach to optimize the virtual forces for a stable 3D swarming behavior in a realistic communications environment. The results of extensive simulations show that the approach can identify the relationship among the attractive and repulsive virtual forces for given positional properties of a UAS swarm.**

***Index Terms*—Positioning; Flying Ad Hoc Networks; Unmanned Aerial Systems; Optimization; Network Modeling**

I. Introduction

The improved communication, computation and mobility capabilities of unmanned aerial systems (UASs) in recent years have enabled the utilization of systems composed of multiple UASs in various applications [1]. Several approaches have been proposed to position multiple UASs in different formations during their operation. The positioning of the UASs in the swarm becomes even more critical in three dimensional (3D) environment since in recent applications UAS swarms not only offer monitoring for the ground operations but also observe or make measurements in the 3D space. Accordingly, the coordination and communication methods used in these applications have challenges associated with the location identification, formation, and signaling [2].

Virtual forces and molecular geometry based approaches have been instrumental in UAS swarming applications. Molecular geometry based approaches apply forces between UAS nodes similar to electron pairs around a central atom. This leads to near ideal volume coverage around a central node [3], [4]. Other virtual forces based approaches use these forces for collision avoidance along with swarm formations. The UAS swarming protocols based on virtual forces generally have scalability and applicability challenges due to the static 3D space constraint assumptions for the position convergence of network nodes and the unrealistic evaluation of the conditions in network communications.

In this paper, we propose an approach to optimize the virtual forces for a stable 3D swarming behavior of a UAS swarm in a realistic communications environment. We use a Tree-structured Parzen Estimator (TPE) [5] based optimization to explore the challenge of position convergence. We also integrate TPE with NS-3 discrete-event network simulator [6] for using a realistic communications environment while optimizing the swarming protocol parameters. TPE drives the process forward by providing new swarm parameters to try via "*ask*" and "*tell*" functions in each simulation run. The *ask* function polls the optimizer for a value to try given its state and a random variance. We present an error function to assess the error at each simulation. Then the *tell* function tells the optimizer how fit a particular input value is based on the error function. Based on the results of extensive simulations, our analysis and optimization, we provide a function which determines ideal virtual force ratios based on the desired distances in the swarm.

The main contributions of this paper can be summarized as follows:

- We present a methodology to optimize virtual forces algorithms for finding ideal swarming parameters using an error function, which prioritizes swarm stability.
- We analyze the virtual forces based algorithms for UAS positioning in a realistic, packet-based, network simulation.
- We use the analysis and optimization results to propose our positioning approach, a multi-agent protocol that improves the positioning of UAS swarms using virtual forces.

The rest of the paper is organized as follows. The related work is summarized in Section II. We provide a detailed description of our approach in Section III and the performance evaluation supporting our approach is given in Section IV. We conclude and present potential future research directions in Section V.

II. Related Work

UAS Swarms have the potential to be used in numerous applications. The organization of multiple UASs as an aerial network provides the ability to perform operations and complete missions in an effective manner [7]. Hence, various clustering and routing protocols for the network formation

have been proposed for UAS swarms [8], [9] and many protocols such as WiFi, LoRa, and LTE are used to facilitate the communication in UAS swarms [10]. Positioning of the UASs, communication conditions, and power constraints have been identified as critical factors for the communications and the mission completion by the UAS swarms [11], [12]. In this paper, we focus on the positioning of the UASs within the aerial network and the formation of the UAS swarm in the 3D environment.

The utilization of UAS swarms for multiple applications is expected to be common in various environments including cities. The positioning of the UASs in the swarm and the overall topology of the swarm will be important for these applications. For instance, Chen et al. demonstrate the importance of swarm positioning for a specific application, packet delivery, and explore the advantages and disadvantages for four communication architectures [13]. Even though the applications where the UASs align themselves on a plane in air, such as ground surveillance, gained momentum and popularity, 3D applications of UAS swarms have been also becoming increasingly important [14]–[16].

There are several important methodologies used for UAS positioning in swarms such as virtual forces, molecular geometry based approaches, potential field based methods, virtual spring models and swarm formation control [17]. In this paper, we focus on approaches that use virtual forces to shape up the UAS swarm topology. APAWSAN [3] is such an example, which uses a 3D positioning strategy for UAS swarms inspired by molecular geometries and virtual forces within the molecules. The approach is also extended for autonomous clustering and adaptation of forces based on obstacles in the 3D environment [4], [18]. The Regular Tetrahedron Formation (RTF) strategy [19] is another geometrical positioning protocol for the swarming of robots in 3D environment. Each robot in the swarm using RTF utilizes a virtual spring mechanism to move so that the team forms a tetrahedron after the initial random positioning. Bhandari et al. [20] use a location-based distributed clustering algorithm to group UASs into clusters for resource and stability constraints. The usage of clustering reduces the average distance and improves network overhead.

The optimization algorithms have been instrumental in various ways for UAS swarms and UAS positioning. Devens [21] used Mixed Integer Linear Programming to optimize the flight execution time for the UAS swarms in crowded airspace. Toyoshima et al. [22] use Deep Q-Network (DQN) for actor node mobility control in 3D wireless sensor and actor networks (WSAN). The system is also evaluated for various positioning of events the network aims to observe. Another related work in WSAN [23] use a fuzzy approach to decide on the selection of the actor node, which is akin to the central node in our approach. The fuzzy approach uses the data supplied by sensors to make the decision for required jobs. While all of these approaches present invaluable contributions, it is critical to analyze of the position stability for the UAS swarm and different values of the attracting and repelling forces.

III. Approach

A. Swarming Model Overview

In our approach, given a set of UAS nodes with wireless communication among them, a single node is designated as the central node. Virtual forces are applied in a manner similar to the forces which keep atomic nuclei intact. The central node applies an attraction force to all remaining (peripheral) nodes, proportional to the distance between the nodes with a constant weight a. To prevent the swarm from collapsing in on itself, each peripheral node applies a repulsive force to every other peripheral node, inversely proportional to the distance between the pair and a constant weight r. The goal of the approach is to position UASs symmetrically around the central node at equilibrium, with no gaps or disturbances.

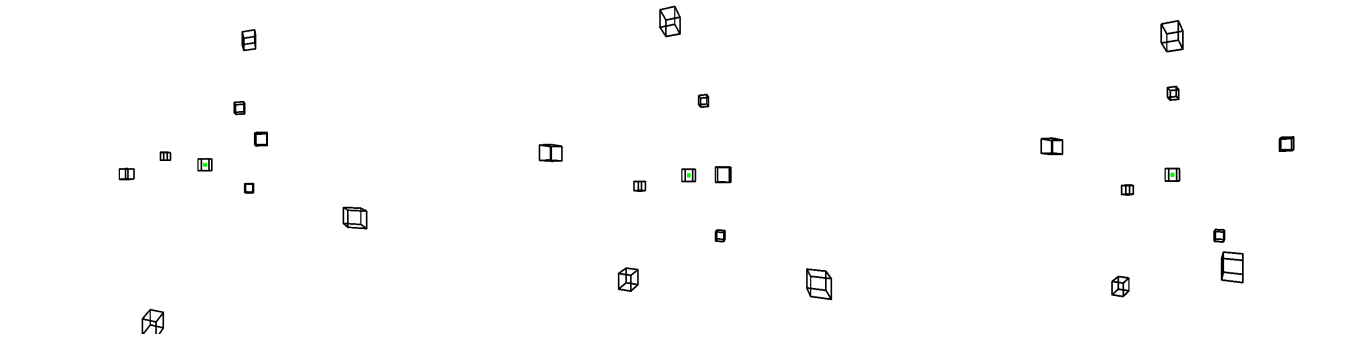
Figure 1a shows the positions of UASs in a swarm with random positions at the start of a mission. The box in the center containing represents the central node around which the peripheral nodes cluster over tiem. This can be thought of the atomic nucleus with the single boxes showing the peripheral (electron) nodes. Figures 1b and 1c show the process of swarm convergence with the application of virtual forces using our approach.

Typically, virtual forces algorithms are proposed as omnipotent algorithms which have the global knowledge of the position and internal state of each node. When operating such algorithms on real hardware, or in a realistic simulation, each node must determine when to communicate its position and how to compute the new forces based on the other nodes in the swarm. Hence, the first step in applying this approach in a realistic communications environment is transforming the algorithm so that it works separately on each individual node and it uses network packets for communication.

We use a periodic time interval $t_{transmit}$, after which each node broadcasts its position to the network. Each node receives position information from other nodes as they arrive and cache the last known position of every other node for use in calculations. Every $t_{calculate}$ interval, each node walks the position cache and compute the force it should apply based on the virtual forces calculation. This force is then applied to its velocity. It is possible to keep $t_{transmit}$ constant to ensure constant bandwidth usage. However, we decrease $t_{calculate}$ so that the forces will be applied in smaller increments and more regularly based on the calculation of the node's new position even if it has not yet received the new positions of its neighbors.

Algorithm 1 is used to determine the virtual forces to be applied and executed on every peripheral node in the swarm every $t_{calculate}$ interval. The algorithm requires access to the position of the node in 3D space, as well as access to persistent position and node type caches, for information about the neighbor nodes.

Algorithm 1 starts by checking the newly received packets and puts new neighbor positions in the $neighbor\_cache$ based on the received information. For each neighbor in the $neighbor\_cache$, the distance, direction and the attrac-



(a) Swarm with initial random positions. (b) Virtual forces refining positions. (c) Swarm with stabilized positions.

Fig. 1: Swarm position stabilization.

Algorithm 1: Force Calculation

Data: $\{positions, kinds, node\_position\}$=Node state
for position p received from node n, kind $kind$ **do**
  $positions[n] = p$
  $kinds[n] = kind$
end for
$force = 0, 0, 0$
for n in $keys(positions)$ **do**
  $b = positions[n]$
  $r = |node\_position - b|$ Distance between current node and n
  $d = normalize(a - b)$ Unit vector toward node n
  $F_a = d * r$ Attraction force
  $F_r = -d/r$ Repulsion force
  if $kinds[n] == CentralNode$ **then**
    $force = A * F_a + force$
  end if
  if $kinds[n] == PeripheralNode$ **then**
    $force = R * F_r + force$
  end if
end for
Apply net force $force$ to UAS
Data: $\{positions, node\_position, node\_kind\}$=Node state

tion/repulsion forces are calculated. For force calculations, we introduce two constants: a and r. These constants are used along with the distances to find the total force to be applied to the node. At every $t_{transmit}$ interval, each node also sends its position ($node\_position$) and type ($node\_kind$) information to the other nodes in the swarm.

B. Swarm Stability Based Optimization

One of the main goals of the approach is to position all nodes in the swarm at a stable location. SOSPUAS uses parameters a and r to determine swarm cohesion and repulsion at every step of the swarm formation. Hence, these parameters are critical for the stable swarming behavior.

We use a TPE-based optimization to decide on the a and r values for the swarm. The TPE drives the process forward by providing new swarm parameters to try via ask and $tell$ in each one of a high number of simulations. Once a swarm with those parameters is simulated, the error for that run is reported back to the TPE. TPE matches the requirements of our approach as it explores multidimensional workloads efficiently and also provides the parallel search capabilities [5].

We integrated TPE with NS-3 discrete-event network simulator [6] for using a realistic communications environment while optimizing the swarming protocol parameters. We have chosen NS-3 due to its full software simulation of the OSI model, 802.11 WiFi, RF gain and latency. It also features mobility models that can be used for mobile network nodes. Communication protocols such as cellular mobile framework have been shown to work for larger distances and exotic swarm structures [24]. We choose to use 802.11 WiFi because of the ubiquity and low cost of wifi devices [25].

The TPE initially generates pseudorandom values for a and r. The value of each parameter is then passed to NS-3, where a full simulation is run. This produces a log file of the position of each UAS every 100 milliseconds. This file is parsed, and the statistics are calculated to determine the error score, which is passed back to the TPE so that better parameters can be generated.

We use an error function which incorporates three parameters: A central distance cost, a peripheral distance cost, and a velocity cost. Overall we want a swarm that is able to assume a stable symmetrical structure quickly and reliably. The central distance cost is based on the difference between a target distance value and the distance between the central node and all peripheral nodes. The peripheral distance cost is the mean absolute deviation among all peripheral nodes, given as follows:

$$\sum_{i=1}^{n} |x_i - m(X)| \tag{1}$$

where x_i is the distance between the central and peripheral node and $m(X)$ is the mean central-peripheral node distance for the entire simulation.

Equation 2 is used to calculate the mean velocity for all nodes across the simulation.

$$\frac{1}{n} \times \sum_{i=1}^{n} v_i \tag{2}$$

where v_i is velocity for a given node at one time step.

Once the targeted formation is reached, UASs should remain the same distance away from the central node and not move. The optimization program analyzes data in simulation steps of 0.1 seconds. At each simulation tick, the position of each UAS is interpolated based on the data exported by running the simulation, and the velocity between this step and the last step is calculated. The distance between the central node and each peripheral node on this step is also calculated, and added to a list along with velocity so that a mean can be computed as needed. After the data points are collected for each simulation step in range of the time positions are recorded for, the error of a particular simulation run is calculated as the sum of the three previously mentioned costs, multiplied by the fixed constants used for tuning.

The simulations with the least error cluster in a line around $a = Mr + B$. This is because if one force parameter is increased without change to the other, the swarm is unbalanced, and acquires a tendency to collapse in or expand out too far. Either resulting in increased velocity, or by moving distance away from the target peripheral distance. Both of which increase error. If either a or r are multiplied by a constant N, the other force must also be multiplied by N to maintain equilibrium when the swarm is in a given formation, leading to this linear relationship.

It's important to note that the optimization program can be extended with new optimization parameters and the error function. Therefore the approach can be modified to suit the needs of new systems or swarms while retaining the logic for executing simulations and collecting results. The simulation code is open-source and publicly available [26].

IV. Results

In this section, we present the results of our performance evaluation study on the proposed approach.

Figure 2 shows the results of the optimization when using parameters $t_{calculate}$ =0.01 seconds, $t_{transmit}$ =0.3 seconds and the target peripheral distance is 3. The swarm parameters r and a are plotted on the x axis and y axis respectively. Every individual simulation that is run by the optimization program is captured as a point on the graph where color determines error. Error is sorted in descending order where light green indicates the run with the lowest error and dark red represents most error. The color for all other runs is interpolated based on its error percentile in the sorted list.

An shown in Fig. 2, all of the most successful simulations arise on a specific line when a vs r ($a = 0.402r - 0.0034$) is graphed. Since the TPE explores the entire search space, it is extremely unlikely other areas of the graph would exist that would allow for successful simulations.

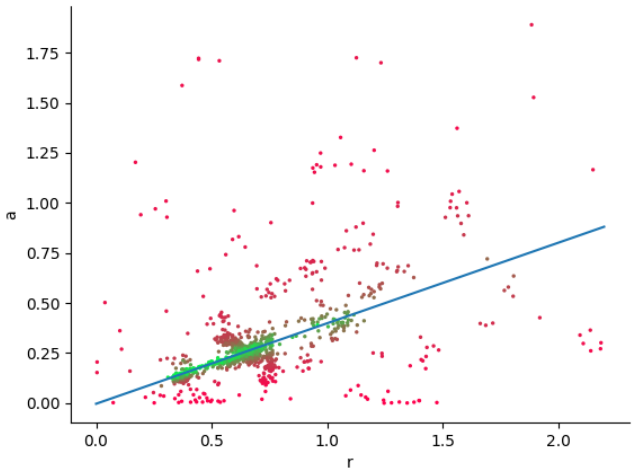


Fig. 2: Swarm error behavior with $d = 3, t_{transmit} = 0.3s$.

A stable swarm is defined as one in equilibrium, where the net forces acting on it are zero. With attraction and repulsion as the primary forces, there is only one ratio of a and r coefficients that causes the forces to reach equilibrium. Because these coefficients are lineally applied, if a is doubled, r would also need to double to maintain equilibrium.

Figure 3 shows the results of the optimization when using parameters $t_{calculate}$ =0.01 seconds, $t_{transmit}$ =1 seconds and the target peripheral distance is 5. Similar to Figure 2, the successful simulations arise on a specific line.

Figure 4 shows the error of each simulation over time. The simulation starts with some error as our optimization approach tries to find better parameter values. As parameters with less error are found, points around these are likely to be chosen, resulting in a lower error. At certain times, vastly new parameters are tried in an attempt to increase the search space, temporally increasing error. After some time, the algorithm either concludes that the new parameter region has too much error and gives up, or it discovers a new region with acceptable

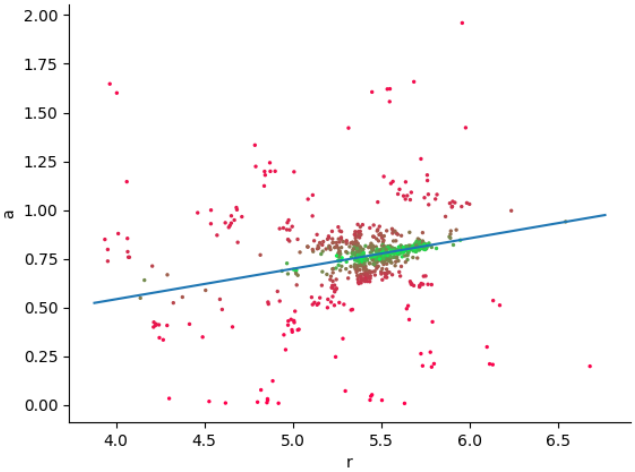


Fig. 3: Swarm error behavior with $d = 5$, $t_{transmit} = 1s$.

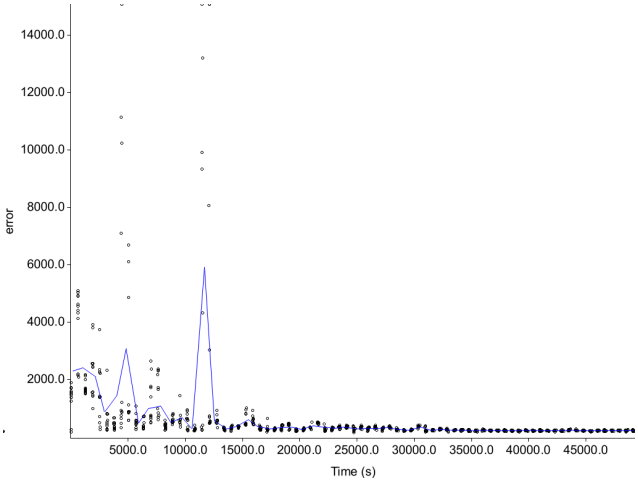


Fig. 4: Error function over time for $d = 5, t_{transmit} = 0.3s$.

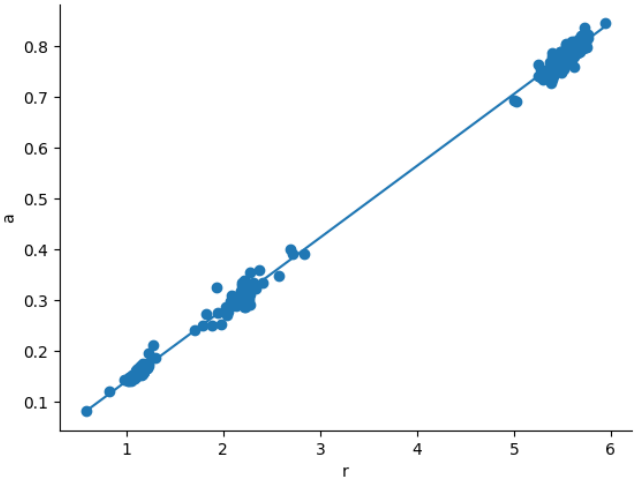


Fig. 6: All data points for $d = 5$ in the bottom 20^{th} percentile.

error, leading to cyclical fluctuating error.

Figure 5 shows the optimization results of executing a matrix of t_x and d for different r and a values. For readability, only simulations in the bottom 20^th percentile of error are shown. The results demonstrate the relationship among t_x, d, a and r. According to these results, while d is correlated strongly with the slope of a vs r, t_x seems to have a smaller impact.

Figure 6 shows the results of the regression to determine an equation for the ratio of a and r to d. To obtain the data points in this figure, we first gather all the data points for all values of d into one graph. This is repeated for $d = 3, 7.5, 10, 12.5, and 15$ data points.

When we run linear regression on these graphs to determine $\frac{da}{dr}$, (ratio between a and r), and plot $\frac{da}{dr}$ over the values of d to obtain a final exponential relationship, as given in Figure 7, we obtain the following equation:

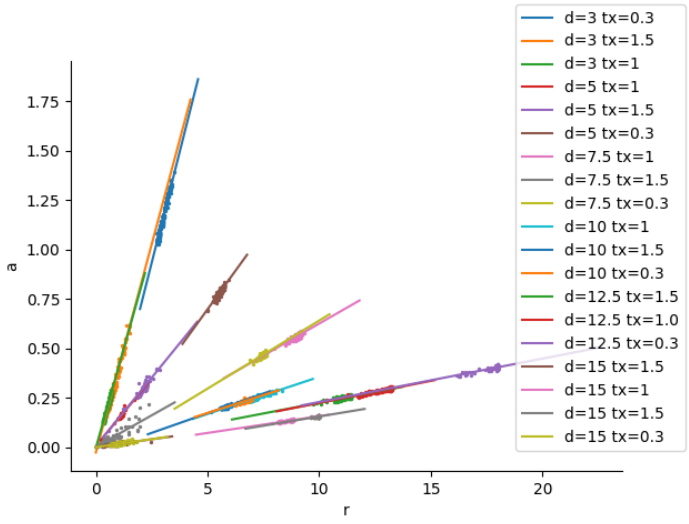


Fig. 5: All optimization results for $t_{transmit} = \{0.3, 1, 1.5\}$, and d=$\{3, 5, 7.5, 10, 12.5, 15\}$.

$$\frac{da}{dr} = 1.81044 * e^{-0.53441d} + 0.02145 \quad (3)$$

Equation 3 can be used to calculate the a/r ratio that a stable swarm needs for a given swarm size d. As shown above, for a given swarm size there is a ratio of a to r that yields the most stable swarm. Equation 3 above can be used by a user of the virtual forces algorithm to find a reasonable ratio of a to r without running any simulations.

Figure 8 is obtained by taking the average location on the a vs r graph, based on a weighted average where the weight is the inverse of the error for that point. The magnitude of this point on the a vs r graph is then plotted against t_x and d. The figure shows that there is little relation between central distance and the magnitude of a and r. As stated before, the linear relationship between a and r means that as long as attraction and repulsion forces are balanced, a swarm will be stable. Therefore a swarm can be stable with any magnitude of a and r as long as the ratio of a/r fits with Equation 3.

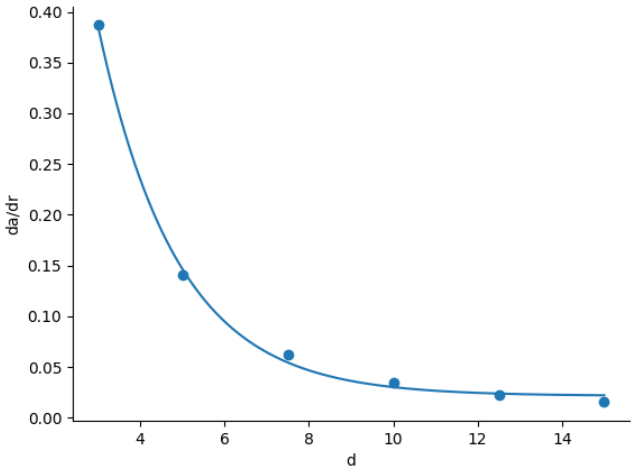


Fig. 7: Relationship between $\frac{da}{dr}$ and d.

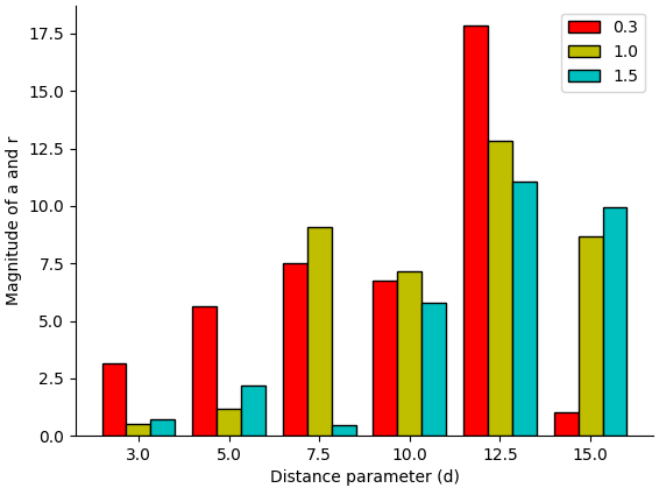


Fig. 8: Relationship of t_x, d with the magnitude of a and r.

V. Conclusion

A realistic implementation and optimization of the virtual forces algorithm is the general topic we are interested in and this paper is about. In this paper, we propose an approach for the implementation and optimization of the virtual forces algorithm in a realistic environment. The performance evaluation results show that the approach can identify the relationship among the virtual forces for given positional properties of a UAS swarm.

The possible future work to extend this work can be listed as implementing additional error functions, extending it for other types of swarming algorithms, and experimenting it using different communication protocols. The error function presented here provided a good middle ground between stability and formation time. Different constraints for future UAS swarms might benefit using a different error function to optimize a/r ratios. There are many swarming algorithms besides Virtual Forces which contain hyperparameters which could be optimized using our approach, provided that such algorithms can be simulated and analyzed.

Acknowledgment

This work relates to Department of Navy award N00014-20-1-2515 issued by the Office of Naval Research. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein.

References

[1] İ. Bekmezci, Ö. K. Şahingoz, and Ş. Temel, "Flying Ad-Hoc Networks (FANETs): A Survey," *Elsevier Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.

[2] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2624–2661, 2016.

[3] M. İ. Akbaş and D. Turgut, "APAWSAN: Actor Positioning for Aerial Wireless Sensor and Actor Networks," in *Proceedings of the IEEE Conference on Local Computer Networks*, Oct. 2011, pp. 567–574.

[4] M. İ. Akbaş, G. Solmaz, and D. Turgut, "Molecular Geometry Inspired Positioning for Aerial Networks," *Elsevier Computer Networks*, vol. 98, pp. 72–88, 2016.

[5] J. Bergstra, D. Yamins, and D. Cox, "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," in *Proceedings of the International Conference on Machine Learning*, vol. 28, no. 1. PMLR, Jun 2013, pp. 115–123.

[6] G. F. Riley and T. R. Henderson, "The NS-3 network simulator," in *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34.

[7] Ö. K. Şahingoz, "Networking Models in Flying Ad-Hoc Networks (FANETs): Concepts and Challenges," *Springer Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, p. 513, 2014.

[8] M. Y. Arafat and S. Moh, "A survey on cluster-based routing protocols for unmanned aerial vehicle networks," *IEEE Access*, vol. 7, pp. 498–516, 2018.

[9] J. Jiang and G. Han, "Routing protocols for unmanned aerial vehicles," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 58–63, 2018.

[10] Z. Yuan, J. Jin, L. Sun, K.-W. Chin, and G.-M. Muntean, "Ultra-reliable iot communications with uavs: A swarm use case," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 90–96, 2018.

[11] O. S. Oubbati, A. Lakas, F. Zhou, M. Güneş, and M. B. Yagoubi, "A survey on position-based routing protocols for flying ad hoc networks (fanets)," *Vehicular Communications*, vol. 10, pp. 29–56, 2017.

[12] L. Sultan, M. Anjum, M. Rehman, S. Murawwat, and H. Kosar, "Communication Among Heterogeneous Unmanned Aerial Vehicles (UAVs): Classification, Trends, and Analysis," *IEEE Access*, vol. 9, pp. 118 815–118 836, 2021.

[13] X. Chen, J. Tang, and S. Lao, "Review of unmanned aerial vehicle swarm communication architectures and routing protocols," *Applied Sciences*, vol. 10, no. 10, p. 3661, 2020.

[14] S. I. Muna, S. Mukherjee, K. Namuduri, M. Compere, M. I. Akbas, P. Molnár, and R. Subramanian, "Air Corridors: Concept, Design, Simulation, and Rules of Engagement," *Sensors*, vol. 21, no. 22, p. 7536, 2021.

[15] H. Kang, J. Joung, J. Kim, J. Kang, and Y. S. Cho, "Protect your sky: A survey of counter unmanned aerial vehicle systems," *IEEE Access*, vol. 8, pp. 168 671–168 710, 2020.

[16] M. İ. Akbaş, K. A. Adkins, and M. D. Compere, "Real-Time Urban Observations for Aviation," in *AIAA Aviation 2021 Forum*, 2021, p. 2359.

[17] Z. Zhao and T. Braun, "Topology Control and Mobility Strategy for UAV Ad-hoc Networks: A Survey," in *Proceedings of the Joint ERCIM eMobility and MobiSense Workshop*, 2012, pp. 27–32.

[18] J. Rentrope and M. I. Akbaş, "Spatially adaptive positioning for molecular geometry inspired aerial networks," in *Proceedings of the ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications*, 2017, pp. 1–8.

[19] G. Zeng and X. Li, "An artifical physics-based 3D swarm control strategy," in *Proceedings of the IEEE International Conference on Electric Information and Control Engineering*, 2011, pp. 148–151.

[20] S. Bhandari, X. Wang, and R. Lee, "Mobility and location-aware stable clustering scheme for uav networks," *IEEE Access*, vol. 8, pp. 106 364–106 372, 2020.

[21] J. A. Devens, T. Bakker, and R. H. Klenke, "Autonomous navigation with obstacle avoidance for unmanned aircraft systems using milp," Ph.D. dissertation, Virginia Commonwealth University, 2016.

[22] K. Toyoshima, T. Oda, M. Hirota, K. Katayama, and L. Barolli, "A dqn based mobile actor node control in wsan: Simulation results of different distributions of events considering three-dimensional environment," in *International Conference on Emerging Internetworking, Data & Web Technologies*. Springer, 2020, pp. 197–209.

[23] D. Elmazi, M. Cuka, M. Ikeda, and L. Barolli, "Effect of size of giant component for actor node selection in wsans: A comparison study," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 8, p. e5106, 2020.

[24] M. Campion, P. Ranganathan, and S. Faruque, "Uav swarm communication and control architectures: a review," *Journal of Unmanned Vehicle Systems*, vol. 7, no. 2, pp. 93–106, 2018.

[25] J. Pojda, A. Wolff, M. Sbeiti, and C. Wietfeld, "Performance analysis of mesh routing protocols for uav swarming applications," in *International Symposium on Wireless Communication Systems*. IEEE, 2011, pp. 317–321.

[26] T. Neubauer, "SOSUAS Source code," https://github.com/AkbasLab/SOSPUAS, 2022.