Swarm Intelligence (2023) 17:63–87
https://doi.org/10.1007/s11721-022-00216-x

Drone flocking optimization using NSGA-II and principal component analysis

Jagdish Chand Bansal[1] · Nikhil Sethi[1] · Ogbonnaya Anicho[2] · Atulya Nagar[2]

Received: 27 January 2022 / Accepted: 2 September 2022 / Published online: 26 October 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
Individual agents in natural systems like flocks of birds or schools of fish display a remarkable ability to coordinate and communicate in local groups and execute a variety of tasks efficiently. Emulating such natural systems into drone swarms to solve problems in defense, agriculture, industrial automation, and humanitarian relief is an emerging technology. However, flocking of aerial robots while maintaining multiple objectives, like collision avoidance, high speed etc., is still a challenge. This paper proposes optimized flocking of drones in a confined environment with multiple conflicting objectives. The considered objectives are collision avoidance (with each other and the wall), speed, correlation, and communication (connected and disconnected agents). Principal Component Analysis (PCA) is applied for dimensionality reduction and understanding of the collective dynamics of the swarm. The control model is characterized by 12 parameters which are then optimized using a multi-objective solver (NSGA-II). The obtained results are reported and compared with that of the CMA-ES algorithm. The study is particularly useful as the proposed optimizer outputs a Pareto Front representing different types of swarms that can be applied to different scenarios in the real world.

Keywords Drone swarm · Multi-objective optimization · PCA · NSGA-II · Drone swarm simulator · Collective dynamics

✉ Jagdish Chand Bansal
jcbansal@sau.ac.in

Nikhil Sethi
sethi.nirvil@gmail.com

Ogbonnaya Anicho
anichoo@hope.ac.uk

Atulya Nagar
atulya.nagar@hope.ac.uk

[1] South Asian University, New Delhi, India

[2] Liverpool Hope University, Liverpool, UK

1 Introduction

Collective behavior is pervasive in nature and is frequently observed in diverse organisms ranging from microscopic bacteria (Allison and Hughes 1991) to large-scale flocking of birds and insects (Nagy et al. 2010; Czaczkes et al. 2015). While researchers still hypothesize the underlying mechanisms behind such behavior, moving in groups can offer several advantages like avoiding predators or carrying collective cargo (Ron et al. 2018). Emulating these natural systems has gained popularity in the past few years, and the development of a robust, fault-tolerant and generalized swarm of robots is now a widely regarded problem among researchers (Saffre et al. 2021; Coppola et al. 2020). Aerial swarms, owing to their high maneuverability and speed, find a number of applications in various industries. They can be deployed as counter-drone measures (Brust et al. 2017) or for basic surveillance operations in a defense scenario. In Abraham et al. (2019), the authors utilize the sensing capability of multiple robots to yield topographical and population density maps of a disaster-afflicted area. In Tosato et al. (2019), a centralized swarm architecture was proposed for measuring air pollution. A system like this could reduce measurement error due to the bigger sample size and distributed data points over the coverage volume. Mixed aerial and ground swarms have also been used to automate construction tasks (Krizmancic et al. 2020). In Ju and Son (2018), multiple UAVs have been shown to outperform a single UAV for tasks like agricultural sensing and monitoring by measuring multiple metrics like energy consumption, flight time, and area coverage. In general, drone swarms can be classified into three categories in order of increasing complexity (Kumar 2020):

- Coordinated: This refers to the collective movement with basic environmental awareness and collision avoidance.
- Cooperative: Here, the robots work together to achieve a particular goal using fewer resources than a single drone.
- Collaborative: This refers to multiple drones working together irrespective of their nature, i.e., heterogeneous collaboration.

This paper proposes a methodology to solve the drone swarm coordination problem with multiple conflicting objectives. This document uses the terms drone, UAV, and aerial robot interchangeably.

Developing a robust velocity controller that allows multiple drones to self-organize comes with its challenges. According to the taxonomy defined in Coppola et al. (2020), the control of velocities is one of the methods under classical Swarming Behavior, i.e., deciding on a high-level control policy with shared information across each agent's neighbors. The challenge is to take this shared information (the agent's state as well as states of neighbors) and come up with controllers (functions) that output an instantaneous velocity vector for each agent. Over time, each agent's velocity gives rise to various patterns and mutual interactions that can potentially emerge into self-organizing behavior. Conventionally, the first-of-its-kind algorithm by Reynolds was based on simple rules for each agent and has been successfully applied in many fields (Hauert et al. 2011; Dewi et al. 2011; Moere 2004). In Vásárhelyi et al. (2018), the authors address

this problem by defining a single fitness function and optimizing it through the CMA-ES algorithm. However, the study does not consider multiple conflicting objectives, the priority of which can vary depending on the scenario. A comprehensive study in Fine and Shell (2013) formalizes flocking behaviors and unifies literature by presenting a data-flow template for various flocking stages. In Márquez-Vega et al. (2021), a multi-objective solution for quad-rotors is proposed. However, the swarm size is limited, and the full range of solutions considering the relations among the fitnesses is not explored. We explore these relationships using unsupervised learning and extend our findings to highlight the use of obtaining a non-dominating solution set for drone flocking.

Modeling natural processes through simulation often needs to be complemented by an in-depth qualitative understanding of the performance measures. Unsupervised learning can help understand and cluster data, especially in high-dimensional spaces that cannot be visualized. It is widely used in experiments where abundant data is available such as mapping vulnerability indices (Abson et al. 2012), understanding relationships between economic and environmental objectives in a chemical supply chain (Pozo et al. 2012), understanding global motions of atoms in proteins (Loeffler and Kitao 2009), and most commonly for dimensionality reduction in evolutionary algorithms (Deb and Saxena 2006). Like many natural systems, the solution to an optimization problem depends on various factors. Often, these factors or objectives are conflicting and they cannot be solved simultaneously without compromising the overall fitness. In the case of flocking, we consider six objectives from Vásárhelyi et al. (2018) (referred to by the Vásárhelyi et al. model from here onwards):

- Collision avoidance with the wall.
- Collision avoidance with each other.
- Average speed of the swarm.
- Average velocity alignment or correlation.
- Total number of connected agents.
- Total number of disconnected agents.

We use PCA to understand the collective dynamics of multi-agent systems and therefore reduce the multi-objective optimizer's objective space. To the best of our knowledge, this work is the first attempt that involves using PCA to reduce the objective functions for a drone flocking optimization problem.

These objectives are then optimized via a well-established multi-objective optimizer (NSGA-II) to yield a Pareto front that can guide decision-making and trade-offs under various situations. We report the results and show that the results at the extremities of the Pareto front perform better than that of the CMA-ES algorithm. We conclude by giving some practical examples of such abstract mathematical formalism for real-time decision-making with a flock of UAVs.

In short, in this research, we create a drone swarm simulator integrated with a multi-objective solver, use PCA to understand the collective dynamics of swarms, and give a Pareto front that represents different swarms that can be used in real-world scenarios. The rest of the paper is organized as follows: Sect. 2 presents the background of Principal Component Analysis (PCA) and multi-objective optimizer (NSGA-II). A drone flocking

optimization problem is formulated in Sect. 3. In Sect. 4, PCA is used to reduce the number of the objective functions and a discussion on the correlations is followed. Section 5 presents the experimental setup and the numerical results and discussions are given in Sect. 6. The research is concluded by giving some potential use cases and possible future work.

2 Background

2.1 Principal component analysis

A high-dimensional objective space suffers from poor selection pressure and convergence (Deb and Saxena 2006). It is also challenging to visualize the space and gain intuition which is often required for appropriate decision-making. Principal component analysis, a technique under the domain of unsupervised learning, may help understand the underlying structure of the data without explicit labels. The idea is to search for the eigenvectors of an m-dimensional covariance matrix (K) which is then used to decide the redundant objectives. Here, m is the number of objectives. This covariance matrix is symmetric, and its elements give the relations between the design variables on which the analysis has been run. Such an analysis of the objectives of an optimization problem can provide insights into their correlations and can help in understanding their qualitative aspects. The process to determine K is given in Appendix A.

2.2 Non-dominating sorting genetic algorithm-II

NSGA-II is a multi-objective optimization algorithm based on ranking each solution in the population according to their fitness and progressively producing better solutions using genetic operators like reproduction and mutation. We use NSGA-II in our work to trade off the reduced fitnesses with each other. This trade-off is represented by Pareto Fronts, which are made up of non-dominated solutions within an evolutionary population. The entire algorithm is explained in detail in Deb et al. (2002). However, a brief explanation covering the salient features of NSGA-II is also explained in Appendix A.

3 Drone flocking optimization problem

A completely decentralized flocking swarm is based on simple rules like Separation, Alignment, and Cohesion. When defined using a velocity control algorithm, these rules have specific parameters that can be tuned to flock optimally. In this section, these parameters are introduced, and a simulation framework capable of handling artificial sensor noise is created. The algorithm used for flocking is based on the Vásárhelyi et al. model and Reynold's Flocking model (Reynolds 1987). Some subtle modifications have been incorporated to handle a multi-objective optimization framework. We use vectorized versions of the equations to leverage fast computation with matrix computation libraries.

To simulate a multi-agent system, there must be a mechanism to share information across the agents. In the case of a decentralized system, this information is shared in each agent's neighborhood $\mathcal{N}_o$. Moreover, real systems are characterized by stochastic uncertainty and noise, which are incorporated into the position($\mathbf{r}$) and velocity($\mathbf{v}$) vectors of the drones. The model for simulating the noise and environmental effects is taken from Virágh et al. (2014). The relative position ($\mathbf{r}_{ji}$) and velocity ($\mathbf{v}_{ji}$) at time t is then found using the following equations:

$$\mathbf{r}_{ji}(t) = (\mathbf{r}_j(t - t_{\text{del}}^{\text{comm}}) + \mathbf{r}_j^{gps}) - \mathbf{r}_i(t) - \mathbf{r}_i^{gps} \tag{1}$$

$$\mathbf{v}_{ji}(t) = (\mathbf{v}_j(t - t_{\text{del}}^{\text{comm}}) + \mathbf{v}_j^{gps}) - \mathbf{v}_i(t) - \mathbf{v}_i^{gps} \tag{2}$$

$$R_j^{\text{rel}} = \mathbf{r}_{ji}(t) \tag{3}$$

$$V_j^{\text{rel}} = \mathbf{v}_{ji}(t) \tag{4}$$

where, $\mathbf{r}_{ji} \equiv$ Relative position vector of jth agent with respect to ith agent at time t, $\mathbf{v}_{ji} \equiv$ Relative velocity vector of jth agent with respect to ith agent at time t, $R_j^{\text{rel}} \equiv j$th row of the Relative position matrix for agent $i \; \forall \; j = 1, 2, ..., \mathcal{N}_o$, $V_j^{\text{rel}} \equiv j$th row of the Relative velocity matrix for agent $i \; \forall \; j = 1, 2, ..., \mathcal{N}_o$, $t_{\text{del}}^{\text{comm}} \equiv$ Simulated communication delay, $\mathbf{r}^{gps} \equiv$ Simulated GPS noise for position, $\mathbf{v}^{gps} \equiv$ Simulated GPS noise for velocity

3.1 Decision variables

The flocking rules are explained in the following sections based on the above modification for the relative position and velocities. These rules give rise to certain parameters which are used as decision variables for the drone flocking optimization problem. Note that all the flocking operations are developed for the ith agent and are carried out for all N agents. All operations involving vectors/matrices and scalars are performed element-wise unless mentioned otherwise.

3.1.1 Separation

To flock effectively without collisions, the agents must have a mechanism for repulsion. A spring-like mechanism is used, which is activated at short ranges of inter-agent distance in the flock. The model for the same is taken from Vásárhelyi et al. (2018) and is delineated in Appendix B.

3.1.2 Alignment

Vásárhelyi et al. Vásárhelyi et al. (2018) realized that effective control of *both* the magnitude and direction of velocities as a function of inter-agent distances could yield the best alignment with scalable velocities. We use the same model in our work as well. The equations for alignment are explained in Appendix B for the sake of completeness.

3.1.3 Wall collisions

To account for collisions at walls, the Vásárhelyi et al. model proposes virtual "shill" agents at the walls which the actual agents can try to align their velocities with. These shill agents have no gain, so repulsion at walls takes place to the maximum extent ($c^{\text{shill}} = 1$). This makes sense while flocking in confined environments because one of the primary goals is to avoid the wall at any cost. In our research, however, while seeking a non-dominated set of solutions (ref. section 6), we characterize the *elasticity* of the virtual geo-fence using a shill gain (c^{shill}) parameter. The following equations are used to find a shill velocity vector from each wall to align with it. $\mathbf{r}_{ci}$ is the relative position vector from the agent to the arena's center $\mathbf{r}_c$. This vector is used to find the distances to the walls in Eq. (6). Eq. (8) gives a m x m sized matrix, with rows as the shill vector from each wall. We assume a square geo-fence with the center of the square at (0,0) in our research, but trivial modifications to Eq. (5) and (6) can generalize it to other shapes as well.

$$\mathbf{r}_{ci} = \mathbf{r}_c - \mathbf{r}_i \tag{5}$$

$$\mathbf{r}_s^{\text{mag}} = L_c/2 - |\mathbf{r}_{ci}| \tag{6}$$

$$\mathbf{v}_i^{shillmax} = D(\mathbf{r}_s^{\text{mag}} - r_0^{\text{shill}}, a^{\text{shill}}, p^{\text{shill}}) \tag{7}$$

$$V_s = (v^{\text{shill}}.\frac{\mathbf{r}_{ci}}{|\mathbf{r}_{ci}|}) \odot I \tag{8}$$

$$\mathbf{v}_s^{\text{mag}} = \|V_s - \mathbf{v}_i\|_{\top \mathbf{v}_i^{shillmax}} \tag{9}$$

$$V^{\text{shill}} = c^{\text{shill}}.(\mathbf{v}_s^{\text{mag}} - \mathbf{v}_i^{shillmax}).\frac{V_s}{\mathbf{v}_s^{\text{mag}}} \tag{10}$$

$$\mathbf{v}_i^{\text{shill}} = \sum_{k=1}^{m} V_k^{\text{shill}} \tag{11}$$

where,
$\mathbf{r}_c \equiv$ Absolute position of the center of the arena
$L_c \equiv$ Side length of the arena
$\mathbf{r}_{ci} \equiv$ Relative position of the center with respect to the agent
$p^{\text{shill}} \equiv$ Slope for the linear part of the decay curve (user-dependent parameter)
$a^{\text{shill}} \equiv$ Acceleration for the non-linear part of the decay curve (user-dependent parameter)
$c^{\text{shill}} \equiv$ Overall Gain for shilling alignment (user-dependent parameter)
$v^{\text{shill}} \equiv$ Speed of shilling agents (user-dependent parameter)
$r_0^{\text{shill}} \equiv$ Alignment cutoff distance for maximum alignment (user-dependent parameter)
$V^{\text{shill}} \equiv \mathcal{N}_o$ x 2 sized matrix of scaled shilling velocities for all dimensions
$V_k^{\text{shill}} \equiv$ Alignment velocity of kth dimension i.e. kth row of V^{shill}

$\mathbf{v}_i^{\text{shill}}$ ≡ Desired collective shilling vector for ith agent.

Here, I is the identity matrix, and $\odot$ is the Hadamard product. Eqs. (9) - (11) have the same velocity alignment procedure done in section 1, but here it's done for each wall's shill velocity instead of each agent neighbor.

The above three velocities (3.1.1 - 3.1.3) along with the normalized flocking velocity are summed up and normalized again to give the desired velocity for the respective agent. This desired velocity is further used to update the current velocity and position sequentially using a first-order Euler integration method.

$$\mathbf{v}_i^{desired} = \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|} v^{\text{flock}} + \mathbf{v}_i^{\text{rep}} + \mathbf{v}_i^{\text{frict}} + \mathbf{v}_i^{\text{shill}} \tag{12}$$

$$\mathbf{v}_i^{desired} \longleftarrow min\{v^{max}, \|\mathbf{v}_i^{desired}\|\} \frac{\mathbf{v}_i^{desired}}{\|\mathbf{v}_i^{desired}\|} \tag{13}$$

Finally, the set of resulting 12 parameters to optimize is:

$$x = \{r_0^{\text{sep}}, p^{\text{rep}}, r_0^{\text{frict}}, a^{\text{frict}}, p^{\text{frict}}, v^{\text{frict}}, c^{\text{frict}}, r_0^{\text{shill}}, v^{\text{shill}}, a^{\text{shill}}, p^{\text{shill}}, c^{\text{shill}}\}$$

3.2 Fitness functions

Order parameters are defined and passed through transfer functions to get the fitnesses and measure the performance of one simulation run.

$$\begin{aligned}
\mathcal{F}^{\text{speed}} &= \mathbf{F}_1(\phi^{\text{vel}}, v^{\text{flock}}, v^{tol}) \\
\mathcal{F}^{\text{coll}} &= \mathbf{F}_3(\phi^{\text{coll}}, a^{tol}) \\
\mathcal{F}^{\text{wall}} &= \mathbf{F}_2(\phi^{\text{wall}}, r^{tol}) \\
\mathcal{F}^{\text{corr}} &= \Theta(\phi^{\text{corr}})\phi^{\text{corr}} \\
\mathcal{F}^{\text{disc}} &= \mathbf{F}_3(\phi^{\text{disc}}, N/5) \\
\mathcal{F}^{\text{cluster}} &= \mathbf{F}_3(\phi^{\text{cluster}}, N/5)
\end{aligned} \tag{14}$$

Here, the order parameters $\phi^{\text{vel}}, \phi^{\text{coll}}, \phi^{\text{corr}}, \phi^{\text{wall}}$, and transfer functions $\mathbf{F}_1, \mathbf{F}_2, \mathbf{F}_3$ are taken from Vásárhelyi et al. (2018) the Vásárhelyi et al. model, and Θ is the Heaviside step function. Parameters r^{tol}, a^{tol}, and v^{tol} are explained in Section 5. Order parameters for disconnected agents (ϕ^{disc}) and the minimum connected agents (ϕ^{cluster}) are explained below. These parameters are calculated locally in r^{cluster} sized clusters:

$$r^{\text{cluster}} = r^{\text{rep}} + r^{\text{frict}} + \tilde{D}(v^{\text{flock}}, a^{\text{frict}}, p^{\text{frict}}) \tag{15}$$

$\tilde{D}$ is the braking distance r for which D(r, a, p) = v for any agent.

3.2.1 Disconnected agents

This parameter measures the average number of completely disconnected agents throughout the simulation. Eq. (18) gives the number of agents within r^{cluster} distance of each agent

at any given moment. Eq. (19) is then used to determine the number of agents throughout the simulation with zero connected agents, i.e., disconnected.

$$\Theta(x) = \begin{cases} 1 \text{ if } x \geq 0 \\ 0 \text{ if } x < 0 \end{cases} \tag{16}$$

$$n_i^{\text{cluster}}(t) = \sum_{j \neq i}^{N-1} \Theta(r^{\text{cluster}} - r_{ij}(t)) \tag{17}$$

$$\phi^{\text{disc}} = \frac{1}{T} \int_0^T \sum_{i=1}^{N} \Theta(n_i^{\text{cluster}}(t) - 1) \tag{18}$$

3.2.2 Minimum connected agents

This parameter measures the minimum number of connected agents averaged throughout the simulation and is therefore dependent on time. Since the drones start at random positions, it was observed that keeping this parameter time-dependent instead of steady state (the global minimum throughout the simulation) gave a better idea of the robustness of the communication graph throughout the simulation.

$$\begin{aligned} \phi^{\text{cluster}}(t) = \frac{1}{T} \int_0^T min\{n_1^{\text{cluster}}, n_2^{\text{cluster}} \ldots \ n_i^{\text{cluster}}\}(t) \\ \forall\, i : 1, 2, ..., N \end{aligned} \tag{19}$$

Finally, to optimize these fitness functions given in Eq. (14) simultaneously, the six objectives must be analyzed for correlations among them so that the system can be represented with fewer objectives, preferably two. In the next section, Principal Component Analysis (PCA) is used for dimensionality reduction so that the multi-objective optimizer NSGA-II can be used effectively.

4 Dimensionality reduction using PCA

A data set of the six objectives discussed in Section 3 is collected to reduce the number of objective functions. This data is just the result of 500 random simulations without any heuristic to cover the entire search space. This search space is the same as the one used for optimization in section 6. Note that we use the fitness values after being passed through the transfer functions as the data for PCA. This can be done directly on the order parameters as well. Both processes would give different correlations depending on the nature of the transfer function. We prefer the former method as it gives a more accurate representation of the matrix components and the exact fitnesses functions used for optimization. This data is

	1^{st} component (w)	Covariance matrix (K)					
$\mathcal{F}^{wall}$	0.191	1.002	0.2821	−0.172	−0.115	−0.094	−0.131
$\mathcal{F}^{speed}$	0.329	0.282	1.002	−0.285	−0.283	−0.301	−0.259
$\mathcal{F}^{corr}$	−0.495	−0.172	−0.285	1.002	0.253	0.5815	0.682
$\mathcal{F}^{coll}$	−0.285	−0.115	−0.283	0.253	1.002	0.278	0.204
$\mathcal{F}^{disc}$	−0.509	−0.094	−0.301	0.581	0.278	1.002	0.748
$\mathcal{F}^{cluster}$	−0.518	−0.131	−0.259	0.682	0.204	0.748	1.002

Fig. 1 Matrices obtained from Principal component analysis

used to create the covariance matrix and principal components is shown in Section 2, followed by a qualitative discussion on the correlations.

In Fig. 1, the matrices obtained from the application of PCA on the objective space are given. The matrices show some interesting results. Some insights are discussed as follows:

K_{23} is negative, implying that a higher velocity doesn't necessarily imply higher cohesion. This might be false in situations where the UAVs have very high-velocity magnitudes while traveling long distances or have a large turn radius (as in the case of fixed-wing drones). But in a confined environment speeds must be reduced to maintain cohesion at the edges (where the flock gets broken up most). This is also a consequence of a limited acceleration which aligns with actual physical systems.

K_{13} also confirms the above statement regarding confined environments. To maintain cohesion at walls, the UAVs can either slow down or skip the wall altogether. A combination of slowing down and breaching the geo-fence makes the above movement the most efficient. Note that intuition would suggest that as speed increases, it would be easier to decrease the wall fitness as there is indeed a limited acceleration/deceleration available. Upon running numerous simulations and making covariance matrices, it was found that this is because $\mathcal{F}^{\text{wall}}$ itself is time-dependent. This means that the fitness is inversely proportional to the number of time frames the drones spend outside the wall. Since $\mathcal{F}^{\text{corr}}$, $\mathcal{F}^{\text{disc}}$, and $\mathcal{F}^{\text{cluster}}$ collectively maintain cohesion and connectivity wherever possible at the expense of ϕ^{vel} and $\mathcal{F}^{\text{wall}}$, whenever the drones slow down, they naturally spend more time frames outside and turn slowly irrespective of the acceleration. This makes $\mathcal{F}^{\text{wall}}$ and $\mathcal{F}^{\text{speed}}$ directly correlated with each other on average. The elements w_{11} and w_{12} and K_{12} represent this fact.

Drones naturally collide less with each other when their velocities are aligned and they are well connected. This is because the time it takes for velocity changes to travel throughout the communication network is much lesser. Although, when this network is strongly connected, the agent has to sum up through many velocity differences in its neighborhood. While this is advantageous when the neighbors are moving in similar directions, it can be detrimental when there is a lot of noise and the inter-agent velocity differences point in different directions. As a result, the summed-up alignment velocity for the concerned agent gets dampened by canceling out. This results in inter-agent friction and makes the entire flock sluggish (slow to react). This is clearly shown by the elements w_2 and w_3, which are strongly uncorrelated. It is also worth pointing out that

elements w_3 through w_6 are strongly correlated, which confirms the association of velocity cohesion and collisions with the communication network. As expected, the cluster parameters for disconnection and minimum number of connected UAVs are strongly correlated (K_{65}) as they are both direct functions of the communication network.

Using a single objective can result in the loss of important information as the final fitness is just the collective product or weighted sum. Notably, in noisy dynamical systems such as multi-agent robotics, efforts need to be made to retain as much information as possible and use it intelligently to guide the decision-making process. We propose a multi-objective methodology for optimization of the swarm's fitness to tackle this problem.

The principal component (w) for the maximum variance captures all the above relations and shows them how they relate with each other on average. The sign of the elements indicates correlation which gives rise to the following features/objectives:

$$\mathcal{F}_1 = -(\mathcal{F}^{\text{wall}} \,.\, \mathcal{F}^{\text{speed}}) \tag{20}$$

$$\mathcal{F}_2 = -(\mathcal{F}^{\text{corr}} \,.\, \mathcal{F}^{\text{coll}} \,.\, \mathcal{F}^{\text{disc}} \,.\, \mathcal{F}^{\text{cluster}}) \tag{21}$$

Unlike traditional PCA, we do not use just the non-redundant objectives. Each objective captures tangible physical information about the simulation and therefore, we multiply the two sets individually to retain that information and also make it easier to draw a comparison with the single objective CMA-ES optimizer as given in Sect. 6. Apart from quantitative comparison, multiplying the $\mathcal{F}$'s also eased the qualitative analysis of collective behaviors. For example, a sudden drop in $\mathcal{F}_2$ could be attributed to a specific fitness within the product, and an appropriate analysis could be carried out. One might also use weights for each objective and their respective correlations to determine this final objective function. Note that we consider the product's negative as we minimize the objective functions.

It is worth mentioning that the above covariance matrix is dependent on the number of agents and the size of the confined arena. While some parameters like the cluster connectivity and cohesion remain the same because they are independent of the above parameters, a different non-redundant set of objectives was obtained upon changing the size of the geofence. The simulations dictated that the same number of agents in a larger space took more time to align with the shill agents due to the stronger inter-agent alignment over long distances. The covariance matrix for the same is not shown here for brevity. In Vásárhelyi et al. (2018), there is a certain ambiguity in the size of the geo-fence. While the authors mentioned that they used a side length of 250m for the square arena, the averaged results on their open-source simulator were closer to the claimed ones when a radius of 250m (or side length of 500m) was used for the arena. To make comparisons easier, we also continue with the latter definition for our study.

The reduced objectives are passed to the multi-objective solver NSGA-II (Deb et al. 2002), and the results are summarized below.

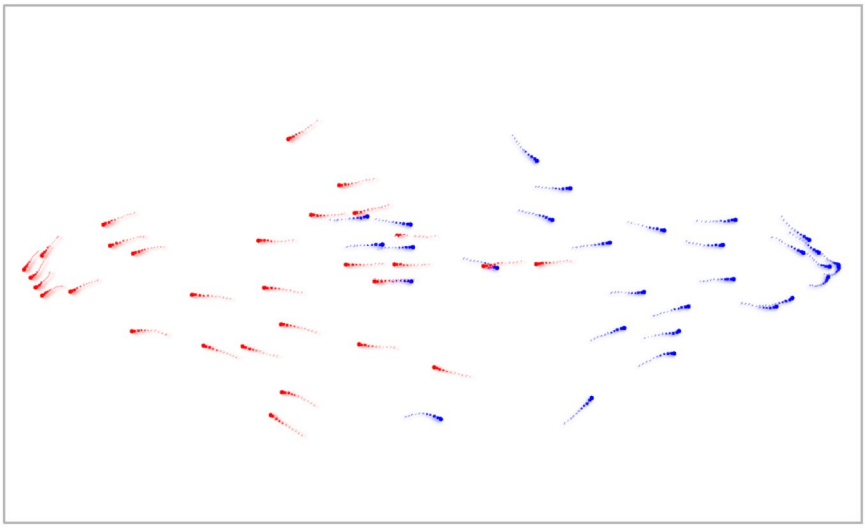


Fig. 2 MOflock simulation screenshot. The blue and red swarms are non-interacting. See supplementary material S2 (2022) for a video of the simulation

5 Numerical experiments

A custom simulator MOflock was created in the Python programming language to test the proposed algorithm and for future work. The ease of use in setting up multiple processes and leveraging optimization and machine learning libraries was a major influence in choosing Python. The simulator is highly object-oriented and modular. It has the drone agent abstracted at various levels and allows experimenting with both single (Bot) and multiple collaborative agents (CoBot). The class diagram for the same is given in Fig. 7 of Appendix C. It was kept in mind that error between RobotSim (Vásárhelyi et al. 2018) and the current work should remain under a threshold of 5-10%. This is done so that MOflock can be validated against state-of-the-art simulators which have been tested on drone hardware. The code repository link is available at supplementary material S1 (2022), and a screenshot of the simulation is shown in Fig. 2. All the experiments are carried out with a flocking velocity (v^{flock}) and maximum velocity (v^{max}) of 6 m/s. However, no changes were made in the algorithm to avoid disrupting the scalability in velocity. Artificial GPS noise is added using the Brownian noise model used in Virágh et al. (2014). Communication delays are integral to the result of optimization as they simulate a kind of inertia at the walls and with neighbors as well. Without these delays and noises, the drones favor high gain and short-range repulsion as opposed to the model optima. Note that the experiments in this paper are carried out in two-dimensional vector space.

After analyzing the covariance matrix for correlations (Sect. 4), the objectives are combined accordingly and passed to the multi-objective optimizer. A good multi-objective optimization algorithm should contain the following characteristics:

Table 1 Optimization parameters

	Lower bound	Upper bounds
r_0^{rep}	30.8 m	51 m
p^{rep}	0.02 1/*s*	0.10 1/*s*
r_0^{frict}	58.5 m	100 m
a^{frict}	5.04 m/s^2	10.0 m/s^2
p^{frict}	0.38 1/*s*	9.67 1/*s*
v^{frict}	0.3 *m*/*s*	2.7 *m*/*s*
c^{frict}	0.03	0.22
r_0^{shill}	−10 m	0 m
v^{shill}	10.0 *m*/*s*	15.0 *m*/*s*
a^{shill}	1.54 m/s^2	6.55 m/s^2
p^{shill}	0.48 1/*s*	9.96 1/*s*
c^{shill}	0.3	1

Table 2 Simulation parameters

Parameter	Value
v^{flock}	6 m/s
v^{max}	6 m/s
t_{del}^{gps}	0.2 s
N	30
L^{arena}	500 m
σ_{inner}	0.005 m^2/s^2
t_{del}^{comm}	1 s
r^{coll}	3 m
v^{tol}	3.75 m/s
a^{tol}	0.0003
r^{tol}	5 m

1. Guide the solutions to an optimal Pareto front
2. Maintain solution diversity

NSGA-II is proven to be one of the best-performing algorithms in this regard. The ‘pymoo’ (Blank and Deb 2020) python library is used with default parameters.

It is imperative to set up the optimization problem so that there is enough diversity in the search space to find "good enough" solutions through heuristic methods. For the sake of exploration, a test run is conducted using the CMA-ES algorithm without any bounds on the parameters. The solution for this setup revealed that the flock only moves in a tight circle around the center and does not collide with the walls at all. While such a solution is mathematically optimal, it does not encapsulate the physical limitations and logical constraints on the variable bounds. This happens because the correlation and wall fitnesses

become abnormally high. For instance, this solution has a very large r_0^{frict} value much greater than the communication range of the drones. This allows each drone to have velocity correlation to the maximum extent and increases $\mathcal{F}^{\text{corr}}$ drastically.

To avoid such solutions which disregard environmental constraints, either explicitly known bounds can be set on the variables which are realistic and relevant to the physics of a UAV, or another objective that maximizes the search area covered in minimum time can be incorporated into the optimization process. For this study, the former approach is used without any loss of generality. An iterative process is used to find the optimization bounds. To begin with, the exact optimization bounds from Vásárhelyi et al. (2018) are used and tweaked progressively according to our use case. We do this by relaxing or constraining the bounds so that the specific behaviors we wanted to analyze through PCA (colliding/surpassing the wall, coming very close to each other etc.) could be incorporated. For instance, r_0^{shill} and c^{shill} are relaxed so that walls could be ignored sometimes. When all behaviors could be observed, we started optimizing in this space. In cases where the optimizer does't explore enough or there is a major change in flocking physics, we tweak the bounds, conduct PCA again, obtain the covariance matrix and optimize on the newly obtained bounds. Some bounds were large enough and did not need any change (eg: separation parameters). The bounds used for the variables are shown in Table 1 and some miscellaneous simulation parameters including certain tolerance parameters r^{tol}, a^{tol}, and v^{tol} for the transfer

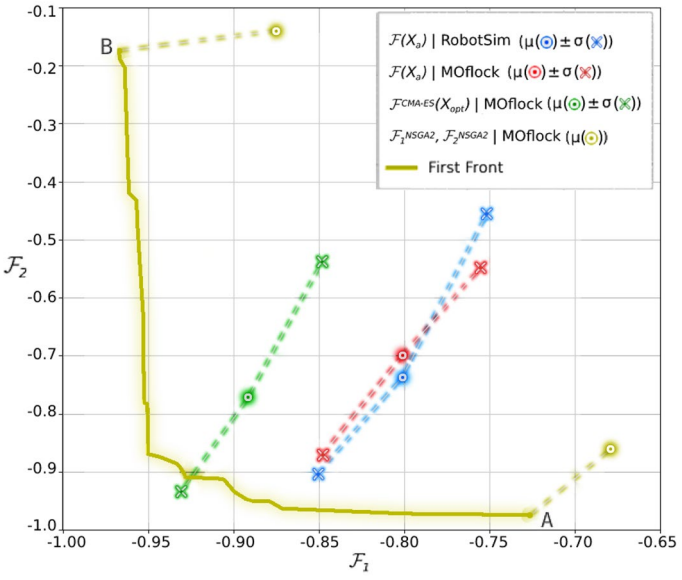


Fig. 3 Statistical evaluations (mean ± standard deviation) of different configurations with respect to the optimal Pareto front. The blue and red curves are comparisons of MOflock and RobotSim at the model optima(X_a) for $v^{\text{flock}} = 6m/s$. $\mathcal{F}(X_a)|RobotSim$ is the multi-objective fitness for the model optima X_a taken from Vásárhelyi et al. (2018) and evaluated on RobotSim itself. $\mathcal{F}(X_a)|MOflock$ is the fitness for X_a evaluated on MOflock. $\mathcal{F}^{CMA-ES}(X_{opt})|MOflock$ is the optimized fitness result on our simulator using the CMA-ES algorithm. The highest ranked Pareto front for the last generation using the NSGA-II algorithm is also shown $\mathcal{F}_1^{NSGA-II}$, $\mathcal{F}_2^{NSGA-II}|MOflock$ are mean values for the extreme points on this Pareto Front

functions in section 3.2 are given in Table 2. Appropriate values for these tolerance parameters promote a better search of solutions and gradient directions.

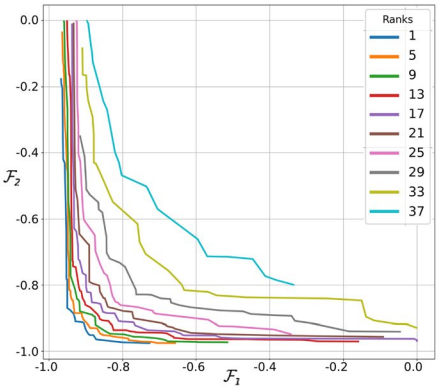
We use simulated binary crossover and polynomial mutation with default parameters from the library Blank and Deb (2020), and the optimization is run for a total of 40 generations and a population size of 50. All the experiments were performed on a machine with the AMD Ryzen 7 4800H 16 core CPU and 16 GB of RAM. The results are reported in Sect. 6.

6 Results and discussions

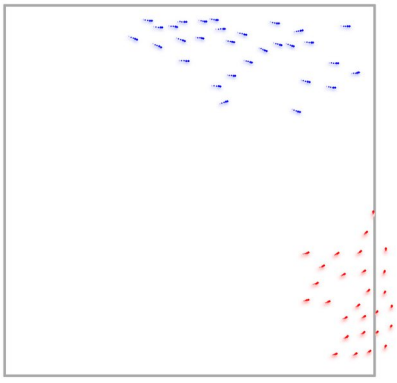
The results of the optimization procedure are analyzed and discussed in this section.

Fig. 3 shows 100 simulations for different points. As targeted, the error on mean fitnesses between both simulators at the model optima in Vásárhelyi et al. (2018) is 4.28%. Note that $\mathcal{F}^{CMA-ES}(X_{opt})|MOflock$ was not evaluated using a multi-objective algorithm but was separated into $\mathcal{F}_1$ and $\mathcal{F}_2$ according to Sect. 4. This is done so that comparisons can be drawn easily between the single objective and multi-objective results.

Since the single objective fitness is the product of all individual fitnesses, it follows that neither of the six fitnesses can be close to zero or even guaranteed to be maximum if there exists a negative correlation between some. As a result, when optimizing a single objective function, a 'best of both' situation is sought after. However, in the case of multiple conflicting objectives, this can be forgiven for better performance on the separated fitness functions. This also explains why the CMA-ES point lies around the knee of the Pareto fronts. It should be noted, however, that the CMA-ES optimum on our simulator

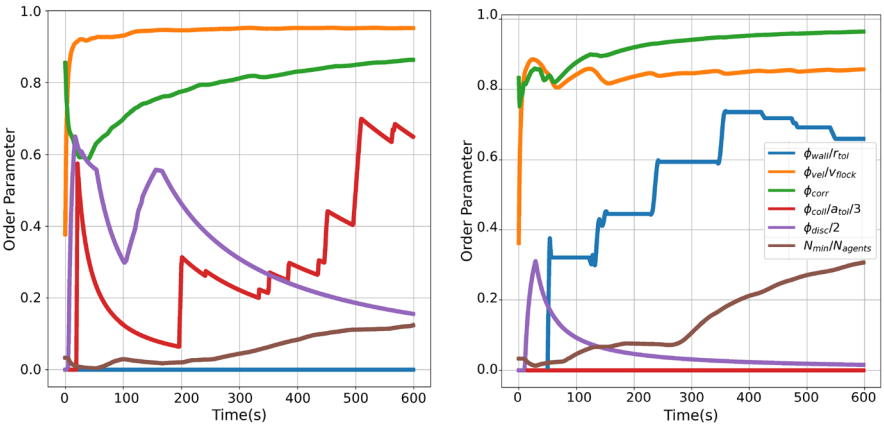


(a) Every 4^{th} Pareto front from the last generation. The Pareto front with rank 1 contains solutions that are non-dominated within themselves but dominate all other solutions in the population. Here rank is calculated using the formula in appendix A



(b) Swarms for extreme points displayed in the same arena. Point A: Blue agents represent the agile flock which minimizes $\mathcal{F}_1$ in the highest ranked Pareto front from the last generation; Point B: Red agents represent the cohesive flock which minimizes $\mathcal{F}_2$ in the highest ranked Pareto front from the last generation.

Fig. 4 Optimization results



(a) Point A: A more agile but less cohesive flock. ϕ^{wall} remains 0 throughout, but ϕ^{corr} is lesser than Point B

(b) Point B: A cohesive but sluggish swarm. ϕ^{wall} increases progressively as the swarm spends more time frames outside the geo-fence. N_{min} is greater than Point A because more agents are present within $r^{cluster}$ radius

Fig. 5 Cumulative order parameters for points A and B. The order parametershave been scaled equally using simulation parameters to accommodate themon a (0,1) range ordinate

outperforms the Pareto front at its knee. This is owed to the high degree of automation and robustness of the CMA-ES algorithm.

While the user can now choose amongst any of the points depending on the scenario and relative importance, there are two interesting points on the optimal front corresponding to the extreme situations when either one of the two solutions is compromised for the other. They are given by points A and B in Fig. 3. The values of the variables and fitnesses at the above points are summarized in Table 3. Fig. 4a shows every 4th Pareto front from

Table 3 Optimization results

	Point A	Point B
$\mathcal{F}_1(\mu \pm \sigma)$	-0.890 ± 0.039	-0.065 ± 0.039
$\mathcal{F}_2(\mu \pm \sigma)$	-0.112 ± 0.09	-0.896 ± 0.179
r_0^{rep}	33.69 m	33.45 m
p^{rep}	0.023 1/*s*	0.028 1/*s*
r_0^{frict}	59.26 m	58.95 m
a^{frict}	5.38 m/s^2	8.223 m/s^2
p^{frict}	4.62 1/*s*	2.67 1/*s*
v^{frict}	1.73 *m/s*	3.00 *m/s*
c^{frict}	0.035	1.84
r_0^{shill}	-2.45 *m*	-0.21 *m*
v^{shill}	12.93 *m/s*	12.93 *m/s*
a^{shill}	4.84 m/s^2	2.57 m/s^2
p^{shill}	4.83 1/*s*	1.30 1/*s*
c^{shill}	0.55	0.43

the last generation. This spacing was only chosen to display the spread and convergence in a neat manner. A snapshot of the relevant simulations for both points is also shown in Fig. 4b along with the graphs for their order parameters in Fig. 5. They can be qualitatively understood as follows:

Point A: A weaker cluster-dependent fitness shows that multiple clusters can coexist in the same environment when cohesion and speed is sacrificed.

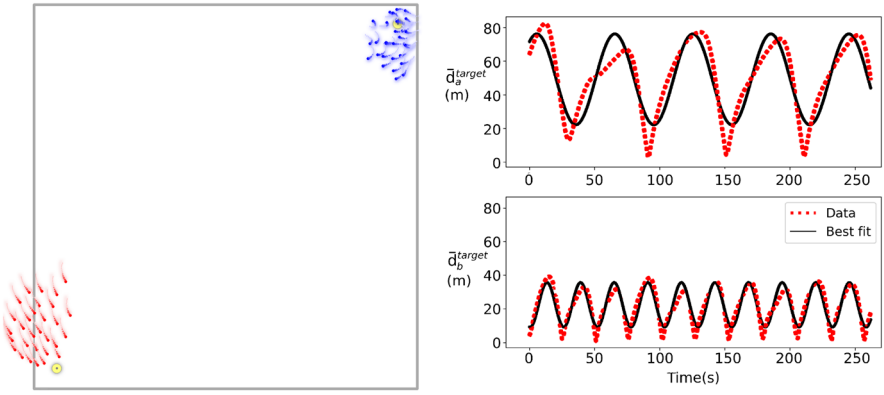
Point B: Similarly, the other point clearly skips the geo-fence and/or slows down to maintain a good cohesion and compensate for the damping caused by inter-agent friction and pressure at the walls.

The generation of the above two points directly results from the physical and environmental restrictions imposed on the swarm. The limited acceleration does not allow the entire swarm to turn sharply without slowing down. The confined walls don't allow agents to flock together when moving at high speeds without losing some cohesion. These statements are a testament to the complex dynamics that multi-agent systems exhibit. A video showing the above interactions is available at supplementary material S2 (2022). Better mathematical formalism and high-fidelity simulations can be developed to realize such intertwined relationships.

The trend in the order parameters in Fig. 5 also confirms the covariance matrix elements in Section 4. Note that the graph is scaled to the (0,1) interval with the relevant maximum feasible values for each parameter, and cumulative values are shown for the curves.

Further statistical analysis of the data from the optimization shows that there is a lot of redundancy in the decision variables. The following observations indicate this finding:

- Even though points A and B are far apart on the Pareto front, their respective parameters for repulsion are very similar.



(a) Target following simulation. Each group follows its own target point in yellow. The red group is cohesive and minimizes $\mathcal{F}_2$. The blue group is agile and minimizes $\mathcal{F}_1$

(b) $\bar{d}^{target}(t)$ for Point A and B. The Y axis shows the average distance of the center of mass from the target point with respect to time.

Fig. 6 Target order parameter comparison

- It was observed that the right combination of r_0^{shill} and a^{shill} gives similar fitnesses and order parameters even with a constant shilling velocity.
- The introduced shill gain (c^{shill}) does not take its maximum possible value (1.0) even when seeking the best $\mathcal{F}_1$, which is highly dependent on this parameter.

Note that a complete PCA correlation analysis on the decision variables can be performed to confirm the above observation and reduce the dimension of the input space as well.

The above results are more consequential than just a Pareto front. Real-life missions and the inherently stochastic nature of the environment demand a range of potential solutions from which a human in the loop can choose in an ad-hoc manner. A typical mission profile consists of cruise, loiter, surveillance, and occasionally a payload drop. A brief description of the use of the practical applications of the Pareto optimal points are shown below.

- Target search and loitering is a common phase in surveillance missions. A snapshot of an extreme case where the target is located at a corner of the geo-fence is shown in Fig. 6a. The flock breaks at corners and walls to loiter around the target. To make this observation mathematically sound, another order parameter called ϕ^{target} is created. The target following physical model is taken from Virágh et al. (2014).

$$\mathbf{x}^{COM}(t) = \frac{\sum_{i=1}^{N} \mathbf{r}_i(t)}{N} \tag{22}$$

$$\bar{d}^{target}(t) = \|\mathbf{x}^{target} - \mathbf{x}^{COM}(t)\| \tag{23}$$

where,
$\mathbf{x}^{COM}(t) \equiv$ Center of mass of the swarm at time t
$\bar{d}^{target}(t) \equiv$ Mean distance to target (over all N agents) at time t

This parameter includes two performance measures- the closeness of the entire flock to the target on average (Eq. 23) and the 'Loiter Frequency (ω)'. This frequency measures how fast the flock can loiter around the target and turn around as a whole. As opposed to the other parameters, the steady state version of this parameter is measured. Since the motion is circular and periodic, the time series is fit to a sinusoidal wave similar to an audio signal.

$$\phi^{target}(t) = a.\sin(\omega.\bar{d}^{target}(t) + \psi) + c \tag{24}$$

$$F = FFT(\phi^{target}) \tag{25}$$

$$\overline{\bar{d}^{target}} = \frac{1}{T}\sum_{t=0}^{T-1} \bar{d}^{target}(t) \tag{26}$$

$$a_o = \sqrt{\frac{2}{T}\sum_{t=0}^{T-1}(\bar{d}^{target}(t) - \overline{\bar{d}^{target}})^2} \tag{27}$$

$$f_o = |f^s_{argmax(|A_k|)}| \tag{28}$$

$$\psi_o = 0 \tag{29}$$

$$c_o = \overline{\bar{d}^{target}} \tag{30}$$

$$a, \omega, \psi, c = LSF(\phi^{target}, \bar{d}^{target}, a_o, f_o, \psi_o, c_o) \tag{31}$$

where,
$\underline{F} \equiv$ Fourier transform output
$\bar{d}^{target} \equiv$ Mean of the average distance throughout the simulation
$f^s \equiv$ Sample frequencies for the time series data
$\bar{d}^{target} \equiv$ Average distance throughout the simulation
$f_o \equiv$ Initial guess of frequency for $\phi^{target}(t)$ corresponding to the maximum F
$\psi_o \equiv$ Initial guess of phase for $\phi^{target}(t)$
$a_o \equiv$ Initial guess of amplitude for $\phi^{target}(t)$
$c_o \equiv$ Initial guess of offset for $\phi^{target}(t)$

This is done by first getting an estimate of the initial coefficients, namely amplitude (a_o), phase (ψ_o), offset (c_o), and frequency ($f_o = \omega_o/2\pi$)) via a Fast Fourier Transform (FFT) on the data (Eq. 24–25) and then passing this estimate for Least Squares curve Fit represented by LSF (Eq. 31). The final order parameter is just the angular frequency divided by the amplitude.

$$\mathcal{F}^{target} = \omega/a \tag{32}$$

The analysis shows that point A on the Pareto front has a lower loiter frequency because of the extra inter-agent friction created to maintain the flock cohesion. Point B, on the other hand, has almost half the amplitude and double the frequency because of the higher velocity, loosely correlated flock with more collisions. These curves and an accompanying simulation screenshot are shown in Fig. 6. A full video showing the target tracking and fitness analysis is available at supplementary material S3 (2022).

- There have been recent studies in which collisions are handled explicitly through physical boundaries and mechanisms instead of an implicit algorithm (Mulgaonkar et al. 2017). The idea is to allow for some collisions as long as agility is maintained and the drones reach their target. Point A on the front is akin to such a situation. The flock does not give much attention to inter-agent separation or cohesion in local clusters. Instead, speed is given a higher priority. This is especially useful when tiny drones need to overcome narrow passages and crevices without acting as a fully connected flock but get through the region as fast as possible, with each drone acting for itself.
- Point B naturally resembles a good flock where connectivity and cohesion is concerned. The decentralized neighbor architecture makes the flock very desirable where robustness and swarm health is an absolute requirement, and the entire swarm needs to travel long distances as a fully connected cluster.

While developing the methodology for this work, several characteristics of collective behavior were noticed in the multi-agent simulations. For instance, the parameters which characterize

the swarm changed drastically based on factors like communication delay and the arena size. These two variables affect the swarm as a whole because any control action for an agent close to the wall is propagated throughout the swarm with the appropriate communication delay. Naturally, every PCA analysis with different simulation parameters yielded unique objectives and, therefore, a different Pareto front. The advantage of separating the objective function into multiple grouped objectives is that global swarm behavior can be controlled by choosing a point on the Pareto front instead of tuning parameters manually or running an offline optimization for each possible situation that the swarm would encounter. Therefore, it follows that this swarm behavior can be controlled by a supervisor with access to the appropriate Pareto front. Moreover, such distinctions and swarm gradients are more consequential when we start attributing human-readable names to these fitnesses. In our case, the swarm which minimizes $\mathcal{F}_1$ acts as an 'agile' swarm whereas minimizing $\mathcal{F}_2$ constitutes a 'cohesive' swarm. A human operator with access to the Pareto front and such terminology could potentially direct global swarm behavior according to the situation at hand. Similar notions of 'stable' vs 'sensitive' swarms have been developed in the past (Balázs et al. 2020) and show that the problem of generalizing a semi-autonomous swarm based on various scenarios is often difficult to handle with just online learning algorithms. A compromise between both, wherein we can control large-scale behavior through multiple objectives and individual decision-making through reinforcement learning can be sought after to solve the generalization problem.

7 Conclusion

In this paper, we proposed a methodology to address the problem of drone flocking. First, a simulator with an integrated optimizer was designed to test the algorithm. The decision variables that characterize the flocking operators and fitness functions that indicate the performance swarm's performance were derived from the Vásárhelyi et al. model and modified accordingly. To use the multi-objective optimizer effectively, the six-dimensional objective space was reduced to two dimensions using Principal Component Analysis. The correlation analysis revealed that fitness functions for both speed and wall avoidance could be treated separately from the cohesive movement of the entire flock. This process also provided insight into the various complex relationships that multi-agent systems can exhibit. Further, the so formed two objective optimization problem is optimized using NSGA-II and the results are compared with the single objective CMA-ES optimization algorithm. It is found that while CMA-ES performs better with respect to the knee of the Pareto front (in situations where a 'best of all' configuration is required), NSGA-II outperforms CMA-ES on the extreme points as it offers an entire range of solutions to choose from. The study also discussed the use cases of such a Pareto front to guide the decision-making process in real-world scenarios. Incorporating algorithms like Reinforcement Learning with the proposed methodology can be future research agenda.

Appendix A: Background

Principal component analysis

The covariance matrix K is formulated as follows: Let X be an n x m design matrix with n rows as the samples and m columns as objectives. A pre-processing step often carried out is

the normalization of the design matrix which brings the mean of samples for each objective to 0.0 and the variance to 1.0 (Eq. A1). The covariance matrix is then calculated by taking the mean of all samples of the pairwise products for each objective (Eq. A2). In a vectorized format, this is equivalent to taking the matrix product of the design matrix X with its transpose (Eq. A3).

$$X_{ij}^{norm} = \frac{X_{ij} - \mu_j}{\sigma_j} \tag{A1}$$

$$K_{ij} = \frac{1}{n} \sum_{k=1}^{n} X_{ki} X_{kj} \tag{A2}$$

$$K = \frac{1}{n} (X^{norm})^T X^{norm} \tag{A3}$$

where,
X_{ij} ≡ Element of X at ith row and jth column
X_{ij}^{norm} ≡ X_{ij} normalized to 0.0 mean and 1.0 standard deviation
μ_j ≡ Mean of all n samples of jth objective
σ_j ≡ Standard deviation of all n samples of jth objective
n ≡ Number of samples
m ≡ Number of objectives
K ≡ Covariance matrix
K_{ij} ≡ Element of K at ith row and jth column

Non-dominated sorting genetic algorithm

Let P_o be an N sized initial random population. This population is sorted based on non-domination according to the following rules: An individual X_1 in the population is said to be dominated by individual X_2 if satisfies both of the following conditions:

- All fitnesses of X_1 must be less than or equal to that of X_2 particle.
- At least one fitness of X_1 must be strictly less than that of X_2.

Mathematically, individual X_1 dominates X_2 if $d = 1$, and the individuals are non-dominated if $d = 0$.

Where, $d = \{\forall m\ F(X_1)^m \leq F(X_2)^m\} \cap \{\exists m\ F(X_1)^m < F(X_2)^m\}$

This method divides the population into dominating and non-dominating solutions, which is a heuristic used to guide the population towards better solutions through the generations. Each solution in this population is also ranked based on the number of other members it is dominated by, and accordingly, it is assigned a front rank. Next, an offspring population Q is created from the sorted population by applying tournament selection, recombination, and mutation operators. A new 2N sized population is made using $P \cup Q$ and is again sorted and ranked to retain the best solution across generations (elitism). To make the next population P_{t+1} from this combined set, solutions are taken in order of their

front ranking. In case the number of solutions belonging to a front exceeds the amount that can be accommodated into the new N sized population, the remaining solutions in that front are ranked based on a crowding operator as follows:

Let $\mathcal{F}^k$ be the set of solutions on the kth ranked Pareto front. The crowding distance (c_i^m) for the mth objective for ith solution on this front is defined as the normalized distance between the two nearest solutions i.e. $(i+1)$th and $(i-1)$th (Eq. A4). The overall crowding distance (c_i) is the sum taken for each objective (Eq. A5).

$$\forall X_i \in \mathcal{F}^k \; : \; c_i^m = \frac{F^m(X_{i+1}) - F^m(X_{i-1})}{F_{max}^m - F_{min}^m} \tag{A4}$$

$$c_i = \sum_{m=1}^{M} c_i^m \tag{A5}$$

This crowding operator ensures that the Pareto Front is uniformly distributed, and the range of each objective value is minimized as the search progresses. The remaining solutions are ranked according to c_i and the new population P_{t+1} moves forward to the next generation. NSGA-II is faster than NSGA-I and has a worst-case complexity of $O(MN^2)$.

Appendix B: Decision variables

Separation

The following two equations (B7) and (B8) are used to find a repulsion vector for agent i after scaling it according to the relative distances in $\mathbf{r}_i^{\text{mag}}$ and a gain p^{rep}.

$$\mathbf{r}^{\text{mag}} = \|R^{rel}\|^{\perp r_0^{\text{rep}}} \tag{B6}$$

$$V^{\text{rep}} = p^{\text{rep}}.(\mathbf{r}^{\text{mag}} - r_0^{\text{rep}}).\frac{R^{rel}}{\mathbf{r}^{\text{mag}}} \tag{B7}$$

$$\mathbf{v}_i^{\text{rep}} = \sum_{j=1}^{\mathcal{N}_o} V_j^{\text{rep}} \tag{B8}$$

where,
$a_{\top b} = max(a, b)$ i.e. a is at least b
$a^{\perp c} = min(a, c)$ i.e. a is at most c
$\mathbf{r}^{\text{mag}} \equiv \mathcal{N}_o$ x 1 sized vector containing inter-agent distances
r_0^{rep} $\equiv$ Repulsion cutoff distance (user-dependent parameter)
p^{rep} $\equiv$ Repulsion gain (user-dependent parameter)
V^{rep} $\equiv \mathcal{N}_o$ x 2 sized matrix of scaled repulsion velocities for all neighbors
V_j^{rep} $\equiv$ Repulsion velocity of jth neighbor i.e. jth row of V^{rep}
$\mathbf{v}_i^{\text{rep}}$ $\equiv$ Desired collective repulsion vector

Note that the upper bound of $\mathbf{r}^{\text{mag}}$ is the parameter r_0^{rep} to enable short-range effects. The matrix norm in Eq. (B6) is only taken along the row axis, i.e. for each neighbor. An Nx2 matrix (R^{rel}) of position vectors divided by the distance vector (r^{mag}) yields unit position vectors. V^{rep} contains all the corresponding scaled repulsion velocities, and the division and multiplication in Eq. (B7) is done element-wise.

Alignment

The equations are similar to separation with one major difference: the upper bound for the velocity magnitude ($\mathbf{v}^{frictmax}$) is now calculated dynamically with decay function D in Eq. (B9), which is dependent on the inter-agent distance (Vásárhelyi et al. 2018). Eqs. (B10) - (B12) describe the process of finding out the agent's combined alignment vector.

$$\mathbf{v}^{frictmax} = D(\mathbf{r}_i^{\text{mag}} - r_0^{\text{frict}} - r_0^{\text{rep}}, a^{\text{frict}}, p^{\text{frict}})_{\top v^{\text{frict}}} \tag{B9}$$

$$\mathbf{v}^{\text{mag}} = \|V^{rel}\|_{\top \mathbf{v}^{frictmax}} \tag{B10}$$

$$V^{\text{frict}} = c^{\text{frict}}.(\mathbf{v}^{\text{mag}} - \mathbf{v}^{frictmax}).\frac{V^{rel}}{\mathbf{v}_{mag}} \tag{B11}$$

$$\mathbf{v}_i^{\text{frict}} = \sum_{j=1}^{\mathcal{N}_o} V_j^{\text{frict}} \tag{B12}$$

where,
D is the velocity decay function from the Vásárhelyi et al. model and takes a vector as the first argument
$p^{\text{frict}} \equiv$ Slope for the linear part of the decay curve (user-dependent parameter)
$a^{\text{frict}} \equiv$ Acceleration for the non-linear part of the decay curve (user-dependent parameter)
$c^{\text{frict}} \equiv$ Overall Gain for alignment (user-dependent parameter)
$v^{\text{frict}} \equiv$ Velocity slack for alignment (user-dependent parameter)
$r_0^{\text{frict}} \equiv$ Alignment cutoff distance for maximum alignment (user-dependent parameter)
$V^{\text{frict}} \equiv \mathcal{N}_o$ x 2 sized matrix of scaled alignment velocities for all neighbors
$V_j^{\text{frict}} \equiv$ Alignment velocity of jth neighbor i.e. jth row of V^{frict}
$\mathbf{v}_i^{\text{frict}} \equiv$ Desired collective alignment vector

Eq. (B9) gives a vector composed of each neighbor's maximum allowable velocity differences. The maximum is proportional to the inter-agent distance. This ensures that the alignment for two agents in close proximity is larger and vice-versa. Also, the maximum allowable difference is lower bound by an optimization parameter v^{frict} so the agents do not strive for perfect alignment, and there is some slack. Eq. (B11) compensates for the velocity difference for each neighbor, and Eq. (B12) sums the alignment velocities for each neighbor.

Appendix C: Software

See Figure 7.

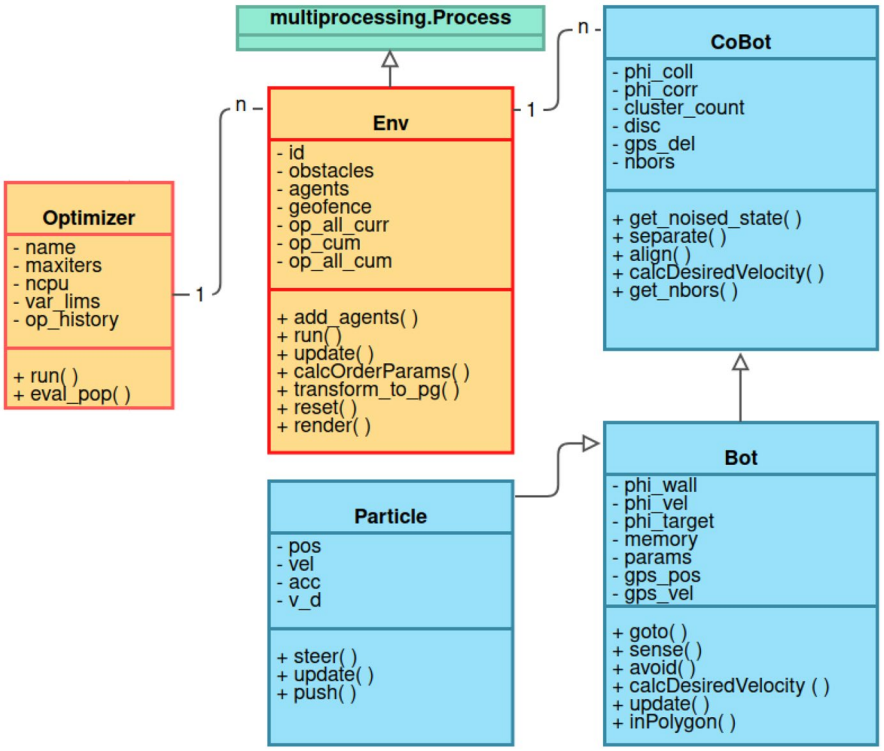


Fig. 7 Class diagram for the Multi-Objective flocking simulator. The *Env* class has a *1* to *n* relationship with the CoBot as each environment can contain multiple CoBots. Similarly, the *Optimizer* class can run multiple environments on multiple cores/processes. The complete simulator is available on GitHub (Supplementary material S1, 2022)

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11721-022-00216-x) contains supplementary material, which is available to authorized users.

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest This work was supported by the funding received in the international joint research project titled "Design and Applications of Swarm Intelligence based algorithms for drone swarm and COVID19 Spread Prediction" funded by Liverpool Hope University UK. The authors declare that there is no conflict of interest.

References

Abraham, L., Biju, S., Biju, F., et al. (2019). Swarm robotics in disaster management. In *2019 International conference on innovative sustainable computational technologies (CISCT)* (pp. 1–5). IEEE.

Abson, D. J., Dougill, A. J., & Stringer, L. C. (2012). Using principal component analysis for information-rich socio-ecological vulnerability mapping in Southern Africa. *Applied Geography, 35*(1–2), 515–524.

Allison, C., & Hughes, C. (1991). Bacterial swarming: an example of prokaryotic differentiation and multicellular behaviour. *Science Progress, 75*(298(Pt 3–4)), 403–422.

Balázs, B., Vásárhelyi, G., & Vicsek, T. (2020). Adaptive leadership overcomes persistence-responsivity trade-off in flocking. *Journal of the Royal Society Interface, 17*(167), 20190853.

Blank, J., & Deb, K. (2020). Pymoo: Multi-objective optimization in python. *IEEE Access, 8,* 89497–89509.

Brust, M. R., Danoy, G., Bouvry, P., Gashi, D., Pathak, H., & Gonçalves, M. P. (2017). Defending against intrusion of malicious UAVs with networked UAV defense swarms. In *2017 IEEE 42nd Conference on Local Computer Networks Workshops (LCN Workshops)* (pp. 103–111).

Coppola, M., McGuire, K. N., De Wagter, C., et al. (2020). A survey on swarming with micro air vehicles: Fundamental challenges and constraints. *Frontiers in Robotics and AI, 7,* 18.

Czaczkes, T. J., Grüter, C., & Ratnieks, F. L. (2015). Trail pheromones: An integrative view of their role in social insect colony organization. *Annual Review of Entomology, 60*(1), 581–599.

Deb, K., & Saxena, D. (2006). Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the world congress on computational intelligence (WCCI-2006)* (pp. 3352–3360).

Deb, K., Pratap, A., Agarwal, S., et al. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.

Dewi, M., Hariadi, M., & Purnomo, M. H. (2011). Simulating the movement of the crowd in an environment using flocking. In *2011 2nd international conference on instrumentation, communications, information technology, and biomedical engineering. IEEE, Bandung, West Java, Indonesia* (pp. 186–191).

Fine, B. T., & Shell, D. A. (2013). Unifying microscopic flocking motion models for virtual, robotic, and biological flock members. *Autonomous Robots, 35*(2), 195–219.

Hauert, S., Leven, S., & Varga, M., et al. (2011). Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. In *2011 IEEE/RSJ international conference on intelligent robots and systems, IEEE* (pp. 5015–5020).

Ju, C., & Son, H. I. (2018). Multiple uav systems for agricultural applications: control, implementation, and evaluation. *Electronics, 7*(9), 162.

Krizmancic, M., Arbanas, B., & Petrovic, T., et al. (2020). Cooperative aerial-ground multi-robot system for automated construction tasks. *IEEE Robotics and Automation Letters*, *5*(2), 798–805.

Kumar, V. (2020). Co-ordination, co-operation, collaboration. https://robohub.org/coordination-cooperation-and-collaboration/ [Robohub Podcast].

Loeffler, H. H., & Kitao, A. (2009). Collective dynamics of periplasmic glutamine binding protein upon domain closure. *Biophysical Journal, 97*(9), 2541–2549.

Márquez-Vega, L. A., Aguilera-Ruiz, M., & Torres-Treviño, L. M. (2021). Multi-objective optimization of a quadrotor flock performing target zone search. *Swarm and Evolutionary Computation, 60*(100), 733.

Moere, A. (2004). Time-varying data visualization using information flocking boids. In *IEEE Symposium on Information Visualization* (pp. 97–104).

Mulgaonkar, Y., Makineni, A., Guerrero-Bonilla, L., et al. (2017). Robust aerial robot swarms without collision avoidance. *IEEE Robotics and Automation Letters, 3*(1), 596–603.

Nagy, M., Akos, Z., Biro, D., & Vicsek, T. (2010). Hierarchical group dynamics in pigeon flocks. *Nature, 464*(7290), 890–893.

Pozo, C., Ruiz-Femenia, R., Caballero, J., et al. (2012). On the use of principal component analysis for reducing the number of environmental objectives in multi-objective optimization: Application to the design of chemical supply chains. *Chemical Engineering Science, 69*(1), 146–158.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput Graph, 21*(4), 25–34.

Ron, J. E., Pinkoviezky, I., Fonio, E., et al. (2018). Bi-stability in cooperative transport by ants in the presence of obstacles. *PLoS Computational Biology, 14*(5), e1006068.

Saffre, F., Hildmann, H., & Karvonen, H. (2021). The design challenges of drone swarm control. In *International conference on human-computer interaction* (pp. 408–426). Springer.

Tosato, P., Facinelli, D., Prada, M., et al. (2019). An autonomous swarm of drones for industrial gas sensing applications. In *2019 IEEE 20th international symposium on "a world of wireless, mobile and multimedia networks" (WoWMoM)* (pp. 1–6).

Vásárhelyi, G., Virágh, C., Somorjai, G., et al. (2018). Optimized flocking of autonomous drones in confined environments. *Science Robotics, 3*(20), eaat3536.
Virágh, C., Vásárhelyi, G., Tarcai, N., et al. (2014). Flocking algorithm for autonomous flying robots. *Bioinspiration and Biomimetics, 9*(2), 025012.

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

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.