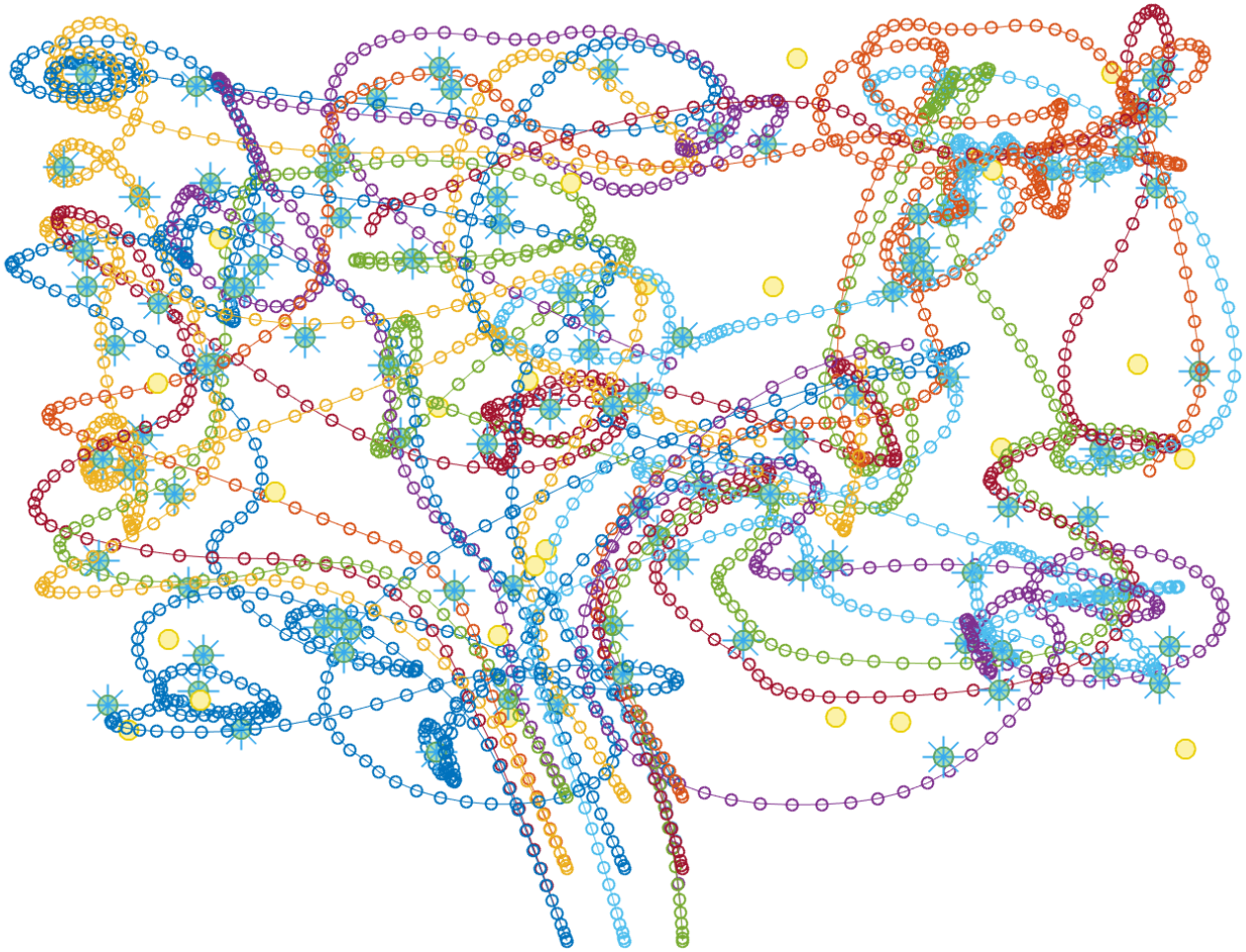
OPTIMIZATION OF AUTONOMOUS SWARM

Will Sharpless
UC Berkeley
October 2019



Contents

0.1 Deliverables 2
0.2 Introduction 3
0.3 Model and Methods 3
0.4 Results and Discussion 7
0.5 Conclusion 11
0.6 Appendix 11

0.1 Deliverables

I wrote the following technical report in a format that I could publish in my portfolio so I integrated the required deliverables throughout the text. The following references link to the start of the sentence where I answer the question.

- Introduction 1: 0.2
- Introduction 2: 0.2
- Introduction 3: 0.2
- Background and Theory 1: 0.3
- Background and Theory 2: 0.3
- Background and Theory 3: 0.3
- Background and Theory 4: 0.3
- Background and Theory 5: 0.3
- Background and Theory 6: 0.3
- Procedure and Methods 1: 0.3
- Procedure and Methods 2: 0.3
- Procedure and Methods 3: 0.3
- Procedure and Methods 4: 0.3
- Procedure and Methods 5: 0.3
- Procedure and Methods 6: 0.3
- Results and Discussion 1: 1
- Results and Discussion 2: 2a 2b 2c
- Results and Discussion 3: 1
- Results and Discussion 4: 0.4
- Results and Discussion 5: 3
- Conclusion 1 0.5
- Appendix 1 0.6

0.2 Introduction

In the modern age, small vehicles are cheap and powerful. With lightweight and fast microprocessors, units may be operated via community control protocols and deployed autonomously. In the following project, I sought to find optimal design variables for autonomous control of a swarm of unmanned aerial vehicles (drones) through genetic optimization of a simulated model. The desired function of the swarm was to rapidly map desired targets while avoiding obstacles and crashing. First, I outline the methods including the model, simulation and genetic algorithm in detail, second, I discuss the results, and finally, I conclude with a brief summary and takeaways. The work demonstrates a scheme for discovering nontrivial optimization parameters and expediting the industrial engineering process.

0.3 Model and Methods

We model the drones in a fixed Cartesian base with the following generic equations for position, velocity and acceleration,

$$\boldsymbol{r} = r_1\boldsymbol{e}_1 + r_2\boldsymbol{e}_2 + r_3\boldsymbol{e}_3$$

$$\boldsymbol{v} = \dot{\boldsymbol{r}} = \dot{r}_1\boldsymbol{e}_1 + \dot{r}_2\boldsymbol{e}_2 + \dot{r}_3\boldsymbol{e}_3$$

$$\boldsymbol{a} = \ddot{\boldsymbol{r}} = \ddot{r}_1\boldsymbol{e}_1 + \ddot{r}_2\boldsymbol{e}_2 + \ddot{r}_3\boldsymbol{e}_3$$

We idealize the masses of the drones (and targets and obstacles) as points because their size, probably less than an eighth of a cubic meter, is irrelevant in the context of the arena which is nearly 11e9 cubic meters (page 6). It is unlikely that the perimeters of the objects would collide while their centers did not at this scale and to be safe we include a crash radius of 5 meters.

The drones are subjected to forces of their own propulsion and drag, but immune to the effects of gravity, lift and buoyancy. We assume the drones are light, but we exclude these forces because they are orthogonal to drag and would require additional parameters in the GA to control a separate dimension of motion and possibly, hinder the ability of the genetic algorithm within the context of 100 generations.

$$m_i\boldsymbol{a}_i = \boldsymbol{\Psi}_i^{tot}$$

$$\underbrace{\Psi_i^{tot}}_{\text{net force}} = \underbrace{\boldsymbol{F}_{p,i}}_{\text{prop. force}} + \underbrace{\boldsymbol{F}_{d,i}}_{\text{drag force}}$$

We define the propulsion and drag forces as the following (note the norm is the euclidean distance),

$$\boldsymbol{F}_{p,i} = F_{p,i}\boldsymbol{n}_i^*$$

$$\boldsymbol{F}_{d,i} = \frac{1}{2}\rho_a C_{d,i} A_i \|\boldsymbol{v}_a - \boldsymbol{v}_i\| (\boldsymbol{v}_a - \boldsymbol{v}_i)$$

Within the simulation the agents have a fixed propulsion $F_{p,i}$ and therefore, reach and maintain flight at the terminal velocity 36.14 m/s (note this has subtle but important implications for behavior),

$$v_i = \sqrt{\frac{2F_{p,i}}{\rho_a C_{d,i} A_i}} = 36.14 \text{ m/s}$$

The propulsion component of each agent is computed as the normalized sum of three types of interactions with itself and all other objects, member-target, member-member and member-obstacle, weighted with priority parameters W_{mt}, W_{mm}, and W_{mo} that are optimized by the GA.

$$\boldsymbol{n}_i^* = \frac{\boldsymbol{N}_i^{tot}}{\left\|\boldsymbol{N}_i^{tot}\right\|}$$

$$\boldsymbol{N}_i^{tot} = W_{mt}\boldsymbol{N}_i^{mt} + W_{mo}\boldsymbol{N}_i^{mo} + W_{mm}\boldsymbol{N}_i^{mm}$$

Control principally manifests through the interaction vector $\hat{\boldsymbol{n}}_{i\to j}^{mt}$ which for member in position $\boldsymbol{r}_i$ and target position $\boldsymbol{T}_j$ is given by the unit vector normal $\boldsymbol{n}_{i\to j}^{mt}$ weighted with the difference of two decreasing exponentials; one term to allow attraction and the other to allow repulsion each with optimized parameters which control the weight and spatial decay rates of the attraction and repulsion. We maintain that the parameters affecting the rates of decay are positive, hence that they amount to rates of decay and not growth as that would undesirably make agents more "aware" of objects at a distance than the objects nearer promoting crashes and apathy for easily mappable targets. Note,that while other control framework or functions could be used, decreasing exponentials are well suited for the problem as they are smooth, decrease exponentially, and we expect them to work for weights within 0 to 1 despite that they may result in sub-optimal scores.

$$\boldsymbol{n}_{i\to j}^{mt} = \frac{\boldsymbol{T}_j - \boldsymbol{r}_i}{\left\| \boldsymbol{T}_j - \boldsymbol{r}_i \right\|}$$

$$\hat{\boldsymbol{n}}_{i\to j}^{mt} = (\underbrace{w_{t1} e^{-a_1 d_{ij}^{mt}}}_{\text{attraction}} - \underbrace{w_{t2} e^{-a_2 d_{ij}^{mt}}}_{\text{repulsion}}) \boldsymbol{n}_{i\to j}^{mt}$$

The total member-target interaction vector for member i, $\boldsymbol{N}_i^{mt}$ is the sum of all interaction vectors for member i and targets $j \in (1, N_T)$.

$$\boldsymbol{N}_i^{mt} = \sum_{j=1}^{N_t} \hat{\boldsymbol{n}}_{i\to j}^{mt}$$

The interaction form is similar for the interactions between member i and obstacles $j \in (1, N_o)$ in positions $\boldsymbol{O}_j$,

$$\boldsymbol{n}_{i\to j}^{mo} = \frac{\boldsymbol{O}_j - \boldsymbol{r}_i}{\left\| \boldsymbol{O}_j - \boldsymbol{r}_i \right\|}$$

$$\hat{\boldsymbol{n}}_{i\to j}^{mo} = \left(w_{o1} e^{-b_1 d_{ij}^{mo}} - w_{o2} e^{-b_2 d_{ij}^{mo}} \right) \boldsymbol{n}_{i\to j}^{mo}$$

$$\boldsymbol{N}_i^{mo} = \sum_{j=1}^{N_o} \hat{\boldsymbol{n}}_{i\to j}^{mo}$$

and also for the interactions between member i and members $j \in (1, N_m)$,

$$\boldsymbol{n}_{i\to j}^{mm} = \frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{\left\| \boldsymbol{r}_j - \boldsymbol{r}_i \right\|}$$

$$\hat{\boldsymbol{n}}_{i\to j}^{mm} = \left(w_{m1} e^{-c_1 d_{ij}^{mm}} - w_{m2} e^{-c_2 d_{ij}^{mm}} \right) \boldsymbol{n}_{i\to j}^{mm}$$

$$\boldsymbol{N}_i^{mm} = \sum_{j=1, j\neq i}^{N_m} \hat{\boldsymbol{n}}_{i\to j}^{mm}$$

After each of the interactions have been computed and thus, the forces on each drone, we may compute the drones change in velocity and position for $t + \Delta t$. We solve the simulation with the following Forward Euler scheme. The Forward Euler method is well suited for the problem because it has a low computation speed while retaining an accuracy of $O(\Delta t^2) = O(0.04)$. However, if one increased the step size, we might expect the simulation computation might speed up because of less time step calculations but it would diverge from reality rapidly because of the higher $O(\Delta t^2)$.

$$\boldsymbol{v}_i(t + \Delta t) \doteq \boldsymbol{v}_i(t) + \boldsymbol{a}_i(t)\Delta t = \boldsymbol{v}_i(t) + \boldsymbol{\Psi}_i^{tot}(t)\frac{\Delta t}{m_i}$$

$$\boldsymbol{r}_i(t+\Delta t) \doteq \boldsymbol{r}_i(t) + \boldsymbol{v}_i(t)\Delta t$$

The swarm is composed of 20 drones and it has 60 seconds to map 100 targets whilst dodging 25 obstacles. Targets and obstacles are initialized randomly in the domain

$$(|x| \leq 100m), \quad (|y| \leq 100m), \quad (|z| \leq 10m)$$

while drones are initialized within the boundary,

$$(-150m \leq x \leq -110m), \quad (|y| \leq 10m), \quad (|z| \leq 10m)$$

I initialized the drones in this region in three parallel yz-planes with 5 drones in a plane where one drone is in the center and one on each corner of the plane; this organization looks like three X's separated by 20 meters each. I chose this formation as it had the greatest minimum distance (I could think of) with centers 14m from each corner and all other points separated by 20m. A drone will "crash" if it leaves the defined boundary

$$(|x| \leq 150m), \quad (|y| \leq 150m), \quad (|z| \leq 60m)$$

Each drone must be within 5m of the target to map it and if a drone and an obstacle or another drone come within 2m, they will crash. Within my simulation, I maintain activity matricies (initialized at ones(N_m,1)) to track wether each agent has been destroyed and which targets have been mapped along with clauses which cause an interaction calculation loop to skip over a computation if the activity of the other object is zero; with running sums for the overall interaction component, this method neglects analysis and incorporation of interaction with inactive members. Note, I chose to make each position matrix 3D where the third dimension represents steps in time; while this costs time to store t times as much data (still runs 10 generations/min), it allows the user to plot the flights of the drones for a better understanding of the flight mechanics and behavior (Figure 4)

Each design variable $\boldsymbol{\Lambda}$ is the following string of undetermined variables that govern the swarm's behavior (random guesses $\in (0,2)$ to begin),

$$\boldsymbol{\Lambda}^i \stackrel{\text{def}}{=} \{W_{mt}, W_{mo}, W_{mm}, w_{t1}, w_{t2}, w_{o1}, w_{o2}, w_{m1}, w_{m2}, a_1, a_2, b_1, b_2, c_1, c_2\}^i$$

After each simulation we score the performance of the design variable $\boldsymbol{\Lambda}_i$ with the cost function Π (with fixed/non-optimized weights w_1, w_2 and w_3) and metrics $M*$, $T*$,

and $L*$,

$$\Pi = w_1 M^* + w_2 T^* + w_3 L^*$$

$$M^* = \frac{(\text{ Unmapped targets })}{(\text{ Total targets })}, T^* = \frac{(\text{ Used time })}{(\text{ Total time})}, L^* = \frac{(\text{Crashed agents})}{(\text{Total agents})}$$

$$w_1 = 70, w_2 = 10, w_3 = 20$$

The genetic algorithm begins with random guesses for $\boldsymbol{\lambda}$ then it scores, sorts and removes all but 6 top scoring parents which are bred to make 6 new children (pairwise with random portions of each parent). With the parents, children and new random guesses, the algorithm iterates and begins a new generation. This optimization method is better suited than a gradient-based method not only because of its speed but because the objective function is discontinuous and undifferentiable as it is a function of discrete amounts of events (targets mapped, agents crashing) and thus, will lead Newton's method to error at many points.

0.4 Results and Discussion

In 100 generations, the Genetic Algorithm is able to minimize Π from 90 to 6.27 absolute units in which 0 agents crash and all targets are hit within 37.4 seconds. The convergence plot demonstrates a history of the population, parents and best $\boldsymbol{\lambda}$'s that lead to the aforementioned minimization (Figure 1).

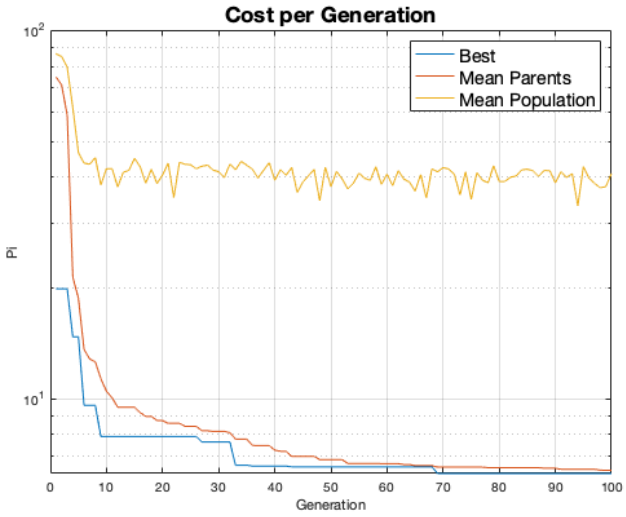
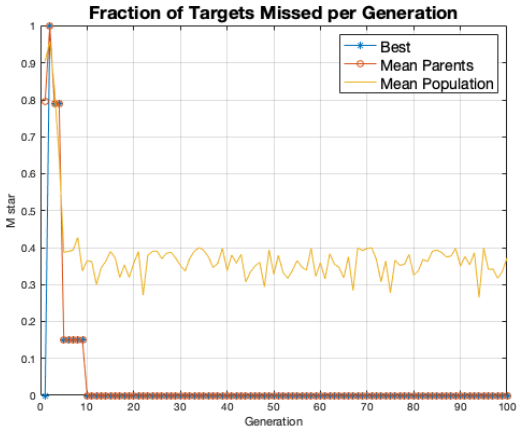


Figure 1: Convergence of Π Final score of Pi is 6.27

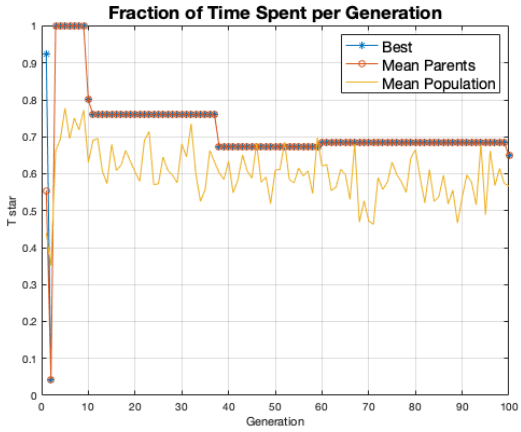
We may note from the convergence, that the final score was approximately reached in

30 generations with the following 70 resulting in less than a 1 unit improvement. Once confident of the GA's minimum region, one might in future designs change the random guesses to center around the λ_k of the best design in order to further minimize the score.

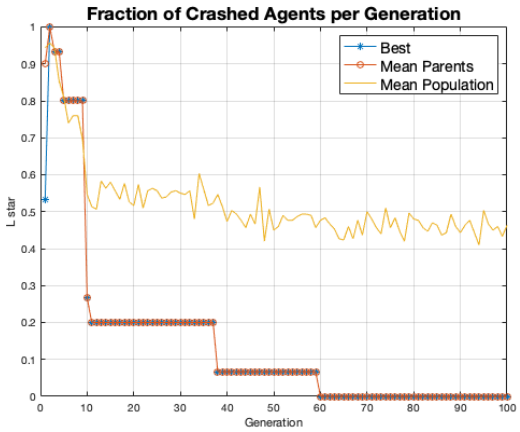
We may gain further understanding from the convergence of M*, L* and T* that correspond the aforementioned minimization that achieved a Π of 6.9 (Figures 2a,2b,2c). The plots do not follow a strict minimizing landscape and are not always in sync; minimization of the overall Pi often comes in pushes on one of the three fonts independently. It is clear from the relative size of w_1 why M star minimizes first and most rapidly, but it is interesting that parents and best score nearly always track together (I originally interpreted this as a mistake in the code but have verified it); perhaps this is due to the low resolution of score changes relative to changes in design variables.



(a) Final score of M* is 0



(b) Final score of T* is 0.65



(c) Final score of L* is 0

Figure 2: Convergence of M*, T* and L*

The top four scoring lambda are nearly identical with ranges of difference no greater

DESIGN	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6	λ_7	λ_8
1	0.3227	0.2093	1.3475	0.9781	1.5450	1.1611	0.8536	1.5942
2	0.3227	0.2094	1.3472	0.9780	1.5482	1.1617	0.8540	1.5940
3	0.3227	0.2093	1.3472	0.9781	1.5444	1.1611	0.8545	1.5940
4	0.3227	0.2092	1.3476	0.9781	1.5426	1.1607	0.8541	1.5943
	λ_9	λ_{10}	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{15}	$\mathbf{\Pi}$
1	1.2656	0.0745	1.4512	1.0933	0.5194	0.5762	0.9197	6.2667
2	1.2650	0.0745	1.4602	1.0933	0.5194	0.5763	0.9198	6.3000
3	1.2659	0.0745	1.4517	1.0933	0.5194	0.5764	0.9197	6.3667
4	1.2653	0.0745	1.4513	1.0933	0.5194	0.5763	0.9197	6.4667

Table 1: Top four scoring Λ design strings

than 0.01 (Table 1); this suggests that all design vectors converged to the same minimum region of Π. We might infer from their similarity that λ's 1, 10, 12, and 13, which correspond to W_{mt},a_1,b_1 and b_2 respectively, are most important to the minimal score. This makes sense for a_1 and b_2 as they are the parameters which control decay rate of attraction to targets (nearly zero meaning it only weakly decays) and decay rate of repulsion to other drones. The consistency of b_1, decay of attraction to obstacles, is a non-obvious result: while it is a significant value, the algorithms inability to raise the value might be an artifact of the density of the obstacles and targets such that too strong a repulsion from objects would also lead to repulsion from targets. The consistently low target priority weight W_{mt}, is a bizarre result that suggests that the drones don't need to prioritize movement based on interactions towards targets, or if we notice W_{mo}/λ_2, nor obstacles, rather, interactions with other members constitute the majority of movement (high W_{mm}/λ_3). One of the most interesting results is the nearly consistent of a low c_1/λ_{14}, the decay rate of attraction to other drones: the weak decay value coupled with the strong weight w_{m1}/λ_8 implies the swarm gains a benefit when it sticks together, and that degree is strictly defined such that any stronger and they crash into one another and any less and they lose the synergistic power. The latter two results elucidate the uncanny ability of swarm and how like a multi-cellular organism, the social cooperation of units results in a "hive mind" processor which has greater sensing range, higher resolution and grander power than any individual unit.

Finally, snapshots of the full simulation that results in $\mathbf{\Pi} = 6.27$ may be found below. In the subtitle one can find the time, targets mapped and agents crashed. Each drone is a small, different colored, ring while objects are yellow circles and targets are green circles, which receive a blue cross once they have been mapped. The viewing domain contains

all targets and obstacles but does not contain the entire crash range in order to see more finely the behavior of the drones. Also, note the viewing domain rotates through the duration of the gif to better demonstrate the action.

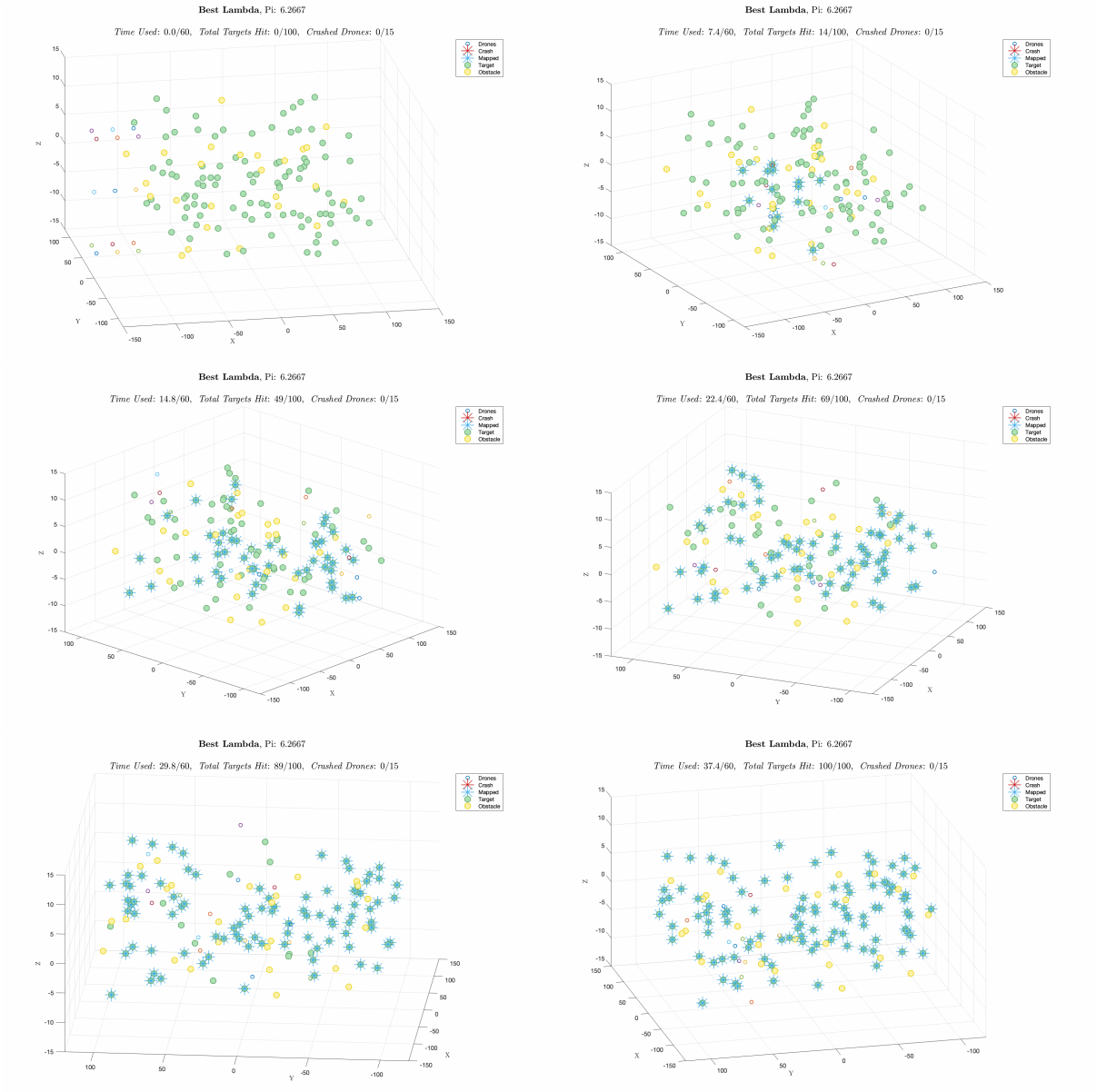


Figure 3: Simulation of Best Lambda The swarm requires 39 seconds to map 100 targets without crashing ($\Pi = 6.27$)

0.5 Conclusion

In summary, a Genetic Algorithm is well suited for the optimization of drone swarms; within 100 generations, the GA produced a solution that was greater than 10x better than random, and additionally, demonstrated nontrivial behaviors and mechanics of flight control like the need for swarm cohesion (). From the simulation, we can gleam the parameters which are meaningless and valuable, and direction for augmenting future designs with other control laws. Above all the method costed only a couple weeks worth of time and little to no physical expenses while simultaneously saving months worth of costs in building experimental methods.

0.6 Appendix

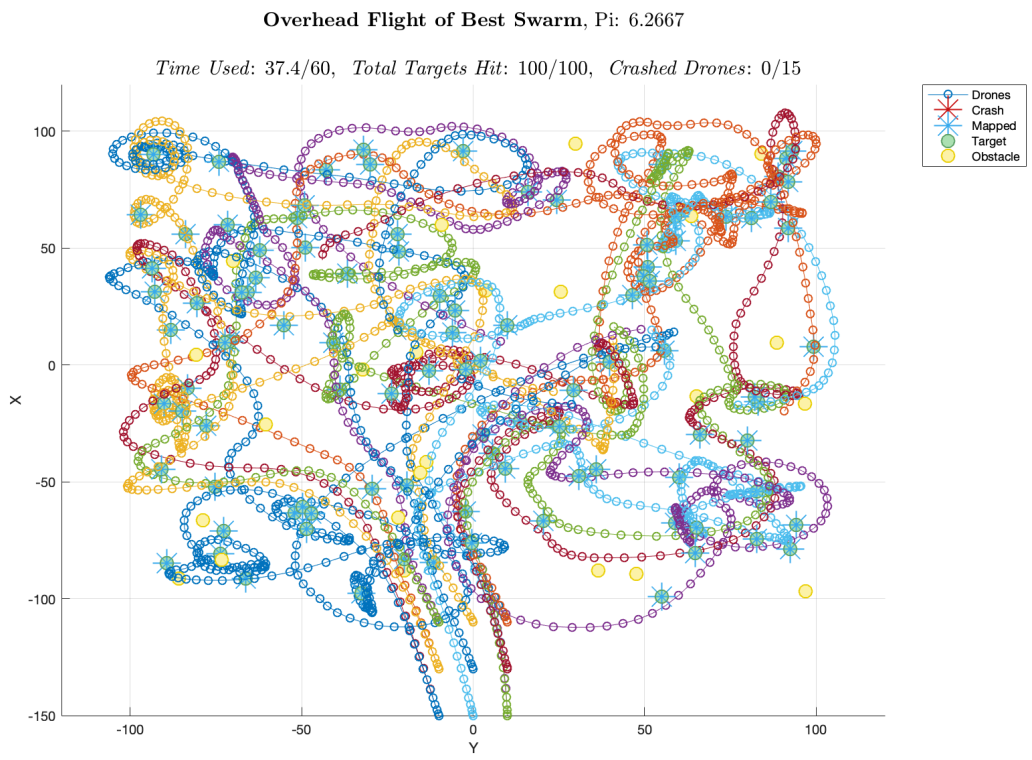


Figure 4: Supplemental Figure: Overhead paths of drones throughout simulation. It is possible to observe subtle behaviors like orbiting (upper left) from these plots