Threat-Aware Selection for Target Engagement

Daniel Biediger and Aaron T. Becker

***Abstract*— This paper investigates the scheduling problem related to engaging a swarm of attacking drones with a single defensive turret. The defending turret must turn, with a limited slew rate, and remain facing a drone for a dwell time to eliminate it. The turret must eliminate all the drones in the swarm before any drone reaches the turret. In 2D, this is an example of a Traveling Salesman Problem with Time Windows (TSPTW) where the turret must visit each target during the window. In 2D, the targets and turret are restricted to a plane and the turret rotates with one degree of freedom. In 3D, the turret can pan and tilt, while the drones attempt to reach a safe zone anywhere along the vertical axis above the turret. This 3D movement makes the problem more challenging, since the azimuth angles of the turret to the drones vary as a function of time. This paper investigates the theoretical optimal solution for simple swarm configurations. It compares heuristic approaches for the path scheduling problem in 2D and 3D using a simulation of the swarm behavior. It provides results for an improved heuristic approach, the Threat-Aware Nearest Neighbor.**

I. Introduction

Quadcopter UAVs (Unmanned Aerial Vehicles or drones) have become ubiquitous among technology enthusiasts, hobbyists, filmmakers, public safety departments, and military interests. These small drones have many potential military uses, from squad-level reconnaissance to direct attacks against military targets. In addition, it is no longer just the military that has access to these types of drones. Civilian drones can be used, out of the box, for illicit reconnaissance of military activity, or can be equipped with improvised explosive devices, chemical weapons, or incendiary weapons. This allows for inexpensive and improvised aerial strike capabilities. Incidents have already arisen from rogue drones performing illegal, unauthorized, and possibly nefarious flybys of military assets. Recently, an incident involving the USS Kidd [1] saw a swarm of 4-6 drones under unknown control fly by a US Destroyer group off the coast of California, constituting a significant breach in security.

One solution to the problem of hostile drones is engagement by land or ship-based gun turrets that fire either projectiles or directed energy. An example of the former is a CIWS, or Close In Weapon System [2]. A standard CIWS consists of a gun turret, typically a multi-barrel rotary cannon, which allows this defensive system to generate a high fire rate (above 4,000 rounds per minute), paired with sophisticated sensor arrays, such as radar or infrared, for tracking of targets, and computerized targeting algorithms to allow for friend or foe identification. These systems are designed to track and engage airborne targets that present a potential threat to the safety of a ship. While many quadcopter drones are significantly smaller than most targets a CIWS engages, they are still valid targets for a CIWS. The efficiency and implementation of tracking and engagement algorithms differ greatly between standard targets, such as cruise missiles and aircraft, and slower, but more maneuverable drones. Much current research work is focused on directed energy weapons, where high intensity energy (often laser light) is directed at a target [3], [4]. A directed energy system requires a non-zero dwell time during which the energy is focused on the target.

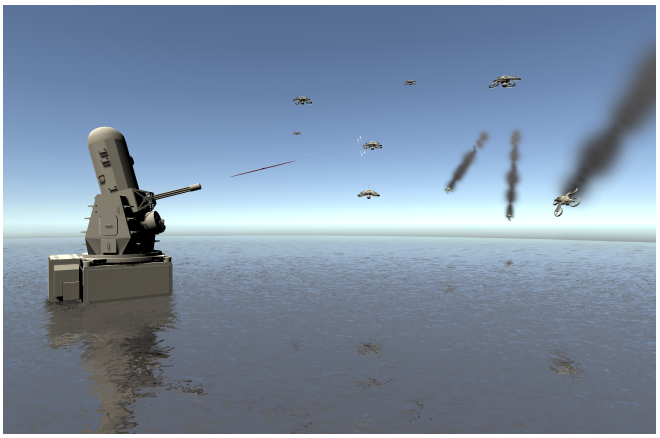

Fig. 1. This work is motivated by the challenge of a defensive turret engaging a swarm of attacking drones.

II. Related Work

Control of anti-aircraft artillery has a long and rich history. Of note is Norbert Wiener's seminal work on predicting the movement of targets [5], and related research during World War II [6]. Our paper investigates a variant of this problem, where the trajectories of the drones are known in advance, but the time window for engaging the drones is limited. In 2D, this allows representing the problem as a Travelling Salesman Problem with Time Windows (TSPTW). If each time window is from 0 to the time when the drone reaches the turret, the minimum-time successful path for the turret is given by the solution to the TSPTW. The time constraints make this problem more difficult than solving a TSP. The best heuristic results for the Traveling Salesman Problem with Time Windows are given by Boland et al. in [7]. They use an integer programming technique using partially time-expanded networks. These networks are used interactively to contract upper as well as lower bounds until the optimal

This work was supported by the Alexander von Humboldt Foundation and the National Science Foundation under Grant Nos. [IIS-1553063, 2130793, 1849303].

Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA {debiediger,atbecker}@uh.edu.

solution is reached, or a computation time limit is reached. Their solver produces close to optimal results on a variety of benchmark problems (up to 200 cities) with a computation time limit of five hours. An approach based on time buckets, presented in [8], provides a method for finding lower bounds for Branch-Cut algorithms.

In the cases where the turret can survive visits by drones, or where the turret has insufficient time to visit all the targets, the turret's goal is to maximize the number of drones visited. This problem (in 2D) can be represented as a Prize Collecting Travelling Salesman with Time Windows (PCTSPTW). Recent work using heuristics to solve by Dogan and Alkaya [9] find good solutions for up to 60-city PCTSPTW problems.

If more than one turret engages the drones, this problem can be modelled as a Vehicle Routing Problem with Time Windows (VRPTW). Google OR-Tools [10] provides code for solving this problem. These algorithms' performances scale with the number of drones, and computation times are measured in minutes or hours, making them currently unsuited for online problems. Moreover, in 3D unless the trajectories of the drones are straight lines directly into the turret, the relative azimuth to the drones will change with time. This changes the amount the turret must move in elevation to travel between targets over time. These time-varying distances are beyond the abilities of [7], [9], [10]. Such trajectories can arise when the drones use an indirect attack strategy. There exists a cylindrical safe-zone extending along the z-axis of the turret where a drone becomes unreachable. We showed in [11] that this is due to a kinematic singularity in the turret and it is reasonable for the drones to exploit it. Similar challenges exist for other drone trajectories, such as drones executing evasive maneuvers or if multiple turrets exist, due to parallax. Considering these challenges, and with a desire for an online solution, this paper focuses on heuristic approaches to the challenge.

III. Problem Statement

A. Optimal targeting sequence: 2D

Consider a top-down, planar version of the targeting problem, like the one shown in Fig. 2. The n drones are confined to the plane and are all moving radially inward at velocity v m/s. The turret can rotate at angular velocity ω rad/s and must visit all n drones before any can reach the turret's position at the origin. Assume that visiting a drone requires a dwell time of τ seconds and that only one drone can be visited at a time.

If the initial position of each drone relative to the turret is angle ϕ_i and radial distance from the turret is some nominal distance d^* plus an individual distance d_i for each drone in the swarm, such that the minimum d_i is zero. Given $d^* + d_i$, for $i \in [1, n], d_i \in [0, \infty), \phi_i \in [-\pi, \pi]$, what visiting sequence minimizes the required d^*?

Equivalently, find the minimum $d^* \geq 0$ over all visiting sequences, since this correlates to the minimum time to complete the sequence.

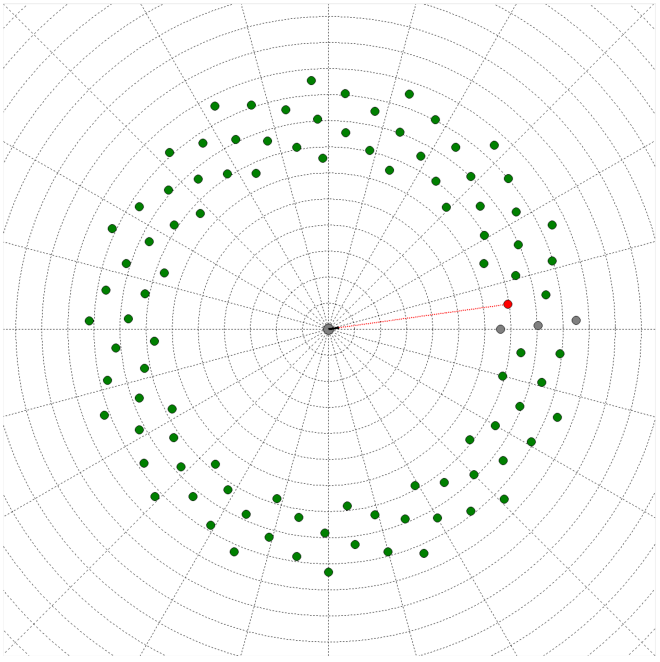

Fig. 2. One frame from a 2D drone swarm simulation with $n = 100$ drones. The center turret defends against drones in a 2D multi-spiral wave formation. The red colored drone is currently engaged and the gray drones have already been visited.

The base case, if $n = 1$ is

$$\frac{\phi_i}{\omega} + \tau = \frac{d^* + d_i}{v}, \tag{1}$$

$$d^* = \left(\frac{\phi_i}{\omega} + \tau\right) v - d_i. \tag{2}$$

The d^* can be lower bounded by the drone that reaches the turret first:

$$d_c^* \geq \max_{i \in [1,n]} \left(\left(\frac{\phi_i}{\omega} + \tau\right) v - d_i \right). \tag{3}$$

The d^* can also be lower bounded because each drone must be visited, requiring a dwell time for each drone and a variable time to transition between the drones in the sequence. Without loss of generality, assume that the ϕ angles are sorted from $[0, 2\pi]$. A path that visits all the drones can then either turn counterclockwise and end at angle ϕ_n for a total angular path length of ϕ_n, or turn clockwise and end at angle ϕ_1 for a total angular path length of $2\pi - \phi_1$. Alternately, a short path could take advantage of a gap in the drone angles: the turret could turn counterclockwise to drone i, then reverse and continue clockwise until it reaches drone $i+1$, for a total path length of $2\phi_i + (2\pi - \phi_{i+1})$. Or the turret could turn clockwise to drone $i+1$, then reverse and continue clockwise until it reaches drone i, for a total path length of $2(2\pi - \phi_{i+1}) + \phi_i$. The lower bound on the time to visit all the drones is then

$$t_\phi = n\tau + \frac{1}{\omega} \min \left\{ \begin{array}{l} 2\pi - \phi_1, \\ \phi_n, \\ \min_{i \in [1,n-1]} 2\phi_i + (2\pi - \phi_{i+1}), \\ \min_{i \in [1,n-1]} 2(2\pi - \phi_{i+1}) + \phi_i \end{array} \right\} \tag{4}$$

Given an ordering for visiting the drones, we can determine the minimum d^*. Let the time of visitation of drone i be t_i. Then $t_i+\tau \leq \frac{d^*+d_i}{v}$, which can be rearranged to $(t_i+\tau)\,v - d_i \leq d^*$. Given a set of n drones and an ordering sequence σ that assigns a visit time t_i to each drone, then

$$d^* = \max_{i\in[1,n]} \Big((t_i+\tau)\,v - d_i \Big). \tag{5}$$

Given a sequence σ of drones to visit, the turret always turns the shortest angular distance between successive drones. Consider a set of drones evenly spaced in a circle around the turret, as in Fig. 8. From (5), because the time required to visit each of the drones is bounded, there is a definite starting distance, beyond which the swarm can always be defeated. For swarms starting closer than this distance, the turret does not have enough time to complete a tour.

For a set of drones arrayed in a complex spiral pattern, like the one in Fig. 2 the boundary is not as clear. If all the drones in this swarm start beyond the distance in (5), then the turret is assured success by sweeping through the formation. Similarly, if all the drones in the swarm start closer than this distance, the turret will not have sufficient time to visit them all before the swarm reaches it. In the circle formation, all drones start at the same radial distance, but for the spiral waves, the drones start at different radial distances. This leads to the possibility that some of the drones may start closer than the corresponding d^* for a circular formation, while others start farther away. In this case, we cannot guarantee the turret's victory, and may need more complex strategies to ensure its success.

There are two basic strategies to approach this problem: select the next closest proximal threat to the turret, or select the next target that is nearest to the current turret orientation. The Proximal Threat (PT) approach focuses on the next immediate threat to the turret; the drone that will arrive the soonest. The Nearest Neighbor (NN) approach ensures that the next target will be engaged quickly and without passing over any targets. In 2D, it functions like a sweep over all drones, selecting the shortest gap to the next target.

To pursue the PT targeting approach, select from all n drones the drone j that reaches the turret first according to (3). This is the base case. $\pi(1) = j$, and d^* is given by (3). Next, we visit j, and from there, determine which drone will reach the turret first according to an updated (3) with drone j removed and all radial distances and angles updated. We designate this drone k, and add it to the end of π; updating, if needed, the required d^*.

By continuing to grow π, selecting the next drone that increases d^*, every drone can be visited by the resulting d^*. This is not dynamic programming, because it assumes that we can always append the next drone to our existing sequence (at the end), without needing to permute the drones already in our sequence.

B. Planar Turret against infinite lines of drones

We can compare these strategies by considering a number of infinite lines of drones, radiating away from the turret, at set angles from the turret. Each drone is separated by s meters radial distance from the next drone in line, see Fig. 8. The initial distances for the drones in each line is important. Let $d_0(t)$ be the distance from turret at time t for the first drone at angle 0 radians, and let $d_\pi(t)$ be the distance from turret at time t for first drone at angle π radians. To defeat a single line of drones, the first drone must be $d^* \geq \tau v$ distance from the turret so that it can be visited before it reaches the turret. By the same argument, $s \geq \tau v$.

The nearest neighbors method fails to defeat two infinite lines of drones with equal spacing s, with one line of drones at 0 radians and the other at π radians. The next nearest target is the one just behind the current target in line. Because there are infinite drones, the turret fails at time $v \cdot d_\pi(0)$, when the first ignored drone behind it arrives.

Selecting targets based on their proximal threat will cause the turret to switch between the lines after each visit. If

$$s < 2v\left(\tau + \frac{\pi}{\omega}\right), \tag{6}$$

the proximal threat protocol will also fail; it spends more time moving back between targets than is does in dwell time on the targets.

A better strategy is to switch lines after visiting k drones in a line. The turret must be able to visit k drones in the first line, rotate to the other line and visit k drones, and then rotate back before the $(k+1)$th drone in the first line has moved closer than the first drone's initial position.

The times for these actions is given by the following relation

$$2\left(k\tau + \frac{\pi}{\omega}\right) \leq \frac{ks}{v}. \tag{7}$$

This bounds the minimum separation distance s.

$$2v\left(\tau + \frac{\pi}{k\omega}\right) \leq s \tag{8}$$

There is also a bound on the initial minimum distance for the second line of drones $d_\pi(0)$. The turret must visit k drones, then turn π radians, and then visit the next drone before it arrives

$$\left(k\tau + \tau + \frac{\pi}{\omega}\right) v \leq d_\pi(0). \tag{9}$$

A plot that compares the minimum values for s and $d_\pi(0)$ as a function of k for representative τ, ω, and v values is shown in Fig. 3. The distance $d_\pi(0)$ grows linearly with k and the s asymptotes toward $2\tau v$.

IV. Notation

Any path π is a sequence of targets for the turret to engage. Each target is selected, in turn, and the turret rotates to point at the drone. This is followed by tracking the target for a dwell time. This marks the respective drone as destroyed and removes it from the set of alive drones.

The problem is then defined as follows. Given the initial bearing from the turret to the drone ϕ_i, and initial distances d_i, and drone velocities v_i, for $i \in [1, n]$, find the path π that maximizes the desired criteria.

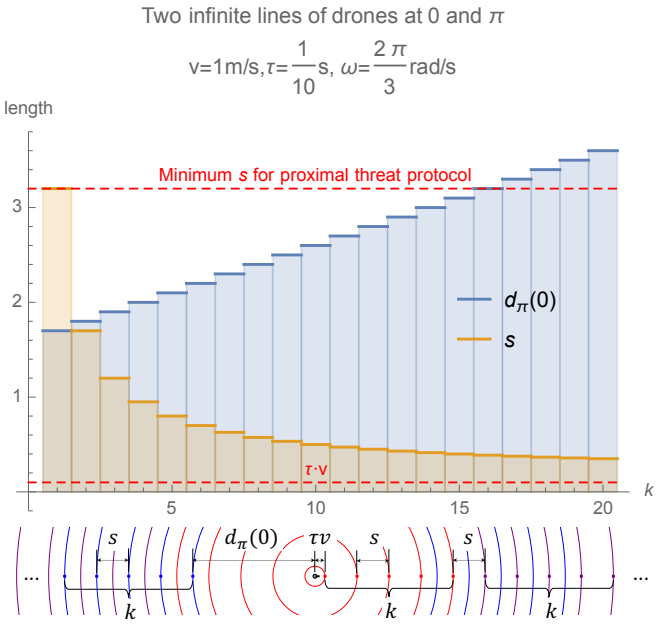

Fig. 3. The required separation distance s decreases inversely with k, the number of drones visited between switching lines. The initial starting distance for the second line $d_\pi(0)$ grows linearly with k.

For a given path π, the visit time for drone i is $T_v(\pi, i)$. The indicator function $\mathbb{1}(\pi, i)$ is 1 only if the turret visits drone i before the drone reaches the turret, and $T_1(\pi)$ is the arrival time of the first drone to reach the turret without being visited.

$$T_v(\pi, i) = \begin{cases} \infty & \pi \text{ visits drone } i \text{ before } d_i/v_i \\ d_i/v_i & \text{else} \end{cases} \tag{10}$$

$$\mathbb{1}(\pi, i) = \begin{cases} 1 & \pi \text{ visits drone } i \text{ before } d_i/v_i \\ 0 & \text{else} \end{cases} \tag{11}$$

$$T_1(\pi) = \min_{i \in [1,n]} T_v(\pi, i) \tag{12}$$

A. Success conditions for metrics

Given these definitions, we can now define the optimal policy under different success conditions. (1) maximize life is π_{ML}, (2) maximize visits is π_{MV}, and (3) just visit is π_{JV}.

$$\pi_{ML} = \underset{\pi}{\operatorname{argmax}} \left(\min_{i \in [1,n]} (T_v(\pi, i)) \right) \tag{13}$$

$$\pi_{MV} = \underset{\pi}{\operatorname{argmax}} \left(\sum_{i \in [1,n]} \begin{cases} \mathbb{1}(\pi, i) & d_i/v_i \le T_1(\pi) \\ 0 & \text{else} \end{cases} \right) \tag{14}$$

$$\pi_{JV} = \underset{\pi}{\operatorname{argmax}} \left(\sum_{i \in [1,n]} \mathbb{1}(\pi, i) \right) \tag{15}$$

There are two simple heuristic approaches, as alternatives to the optimal solver: Proximal Threat (PT) and Nearest Neighbor (NN). The PT approach selects the target that is closest to reaching its goal as the next target. It seeks to remove the most urgent time threats and prolong the life of the turret. The NN approach selects the next target that is easiest to reach from the current position; the nearest neighbor in the state space. It spends the least time moving to the next target and seeks to maximize visits. Both approaches are fast and deterministic, but are greedy and easily exploited by different swarm configurations. In addition there is a strategy that extends and improves upon these approaches: Threat-Aware Nearest Neighbor.

Algorithm 1 NEAREST NEIGHBOR(T)

```
n ← |T|                                  ▷ T are the pending targets
Set the working path π ← ∅
for iteration ← 1 to n do
    Select the nearest target j, move, and engage
    Update drone simulation
    if a drone reaches the turret then
        return No Feasible Path
    else
        Remove j from T and append it to π
    end if
end for
return Success with Path π
```

B. Threat-Aware Nearest Neighbor

The algorithm for Threat-Aware Nearest Neighbor (TANN) is written in Alg. 2. It relies on generating an initial feasible path π using the NN approach, and makes adjustments to address failures in the path plan. Targets are visited in turn as a simulation of the movements of the targets proceeds. If the path generation is successful, the successful path is returned. If the turret loses in the simulation, the first failed target is added to the end of the path and a previous target is removed from the path. The target selected for removal is the one that would have had the highest laxity, if it had not been engaged. This includes the time required to return to the target after completing path Π. As in real-time computing literature, *laxity* is defined as the maximum time a task can be delayed and still complete within its deadline. This corresponds to the previously-engaged target that is most easily overlooked and postponed until later. The simulation and planning then restarts, using the current path as a starting point for the simulation. We count one iteration as *looking ahead* once, and can repeat the process of looking ahead to the next target that causes the path planning to fail.

V. 2D SIMULATION

To simulate the behavior of drones and turrets, we developed a simple 2D simulation in Python. We model the drones as points, moving at their top speed of $v = 10\,\mathrm{m/s}$ directly toward the turret. The turret moves with a limited slew rate of $\omega = \pi/2$ rad/s, but no inertia, to face the drones. It then holds on a drone for a $\tau = 0.5\,\mathrm{s}$ dwell time, to simulate an energy-weapon discharge. The simulation uses an event-based system to mark an uneven passage of time. The events include: a turret rotating to face a new target, a turret tracking a target, and drones arriving at the turret. Each time the turret completes a "visit" to a drone, it must select a

Algorithm 2 THREAT-AWARE NEAREST NEIGHBOR(m)

Set the working path $\Pi \leftarrow \emptyset$
for $iteration \leftarrow 1$ to m **do**
  $\pi \leftarrow [\Pi, \text{NearestNeighbor}([1, \ldots, n] \backslash \Pi)]$
  if Simulation fails due to drone i **then**
    Include i in Π
    Select $j \in \Pi$, s.t. j has highest laxity
    Remove j from Π
  else
    return Feasible Path π
  end if
end for
return No Feasible Path

new target to pursue. It can apply one of the target selection policies: Nearest Neighbor, Proximal Threat, Threat-Aware Nearest Neighbor, or follow the path pre-computed by an off-line solver.

The optimal solution for visiting all the targets in the shortest time is the solution to a TSP problem with Time Windows (TSPTW). For easy time window constraints, the computation is quick using the OR-tools, in [10]. Unfortunately, as the time window constraints become more strict, the computational effort increases. This makes the optimal solver marginally useful for 2D problems: it finds an shortest path, but the computation time often exceeds the simulated time of an engagement.

A. 2D Simulation Results

For all simulations, we determine the minimum d^* by using a binary search. The range for d^* is initialized to $[v\tau, v{\cdot}n(\pi/\omega{+}\tau)]$. The binary search continues to the nearest meter.

The performance of NN and PT for a set of n drones arranged in a vertical line at $x = d^*$, arranged from angle $\phi \in [-\pi/4, \pi/4]$, evenly spaced in Cartesian coordinates, is shown in Fig. 4. These show that NN is clearly superior for this distribution because the PT approach swivels back-and-forth between the two nearest drones, while the NN eliminates all drones from $\phi = [0, \pi/4]$ before engaging the rest. As the number of drones increases, the time required to engage the swarm increases. This allows for additional movement time for the drones and moves the nominal starting distance required for the turret to win farther away.

The slopes of the lines in Fig. 4 indicate the effectiveness of the target selection strategies. The Nearest Neighbor approaches show a more shallow slope than the PT approach. The NN policies select the closest targets with the minimum movement cost to the turret. In contrast, the PT approach can rotate a significant amount to engage the next proximal threat. This additional variable cost is reflected in the steeper slope and in some variability in the exact path length. As a result, the PT spends more time moving between targets allowing the swarm more time to reach their target. In order to succeed using the PT targeting, the same number of drones must be further away when compared to NN.

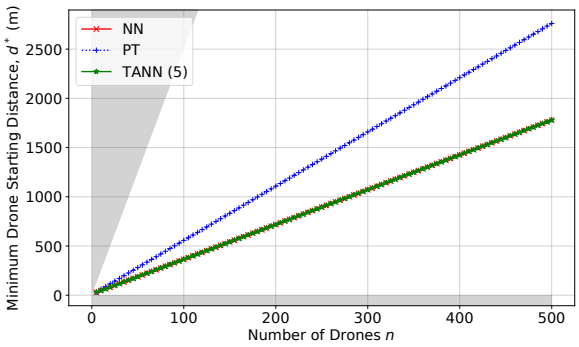

Fig. 4. The minimum starting distance for a turret using different targeting approaches against a 2D linear formation (see schematic in Fig. 8).

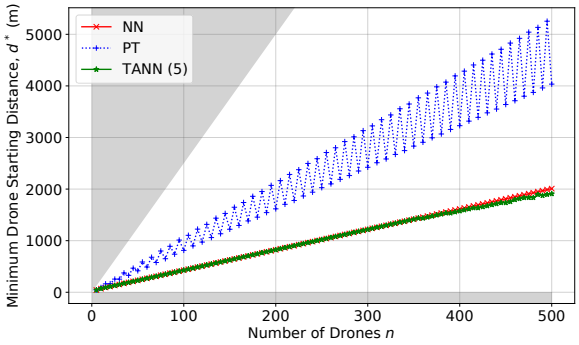

Fig. 5. The minimum starting distance for a turret using different targeting approaches against a 2D double-line linear formation (see schematic in Fig. 8).

Interestingly, there is a similar behavior in Fig. 5, but with two different slopes. In this case the drones are arranged with half at $x = d^*$, arranged from angle $\phi \in [-\pi/4, \pi/4]$ evenly spaced in Cartesian coordinates, and the remaining half at $x = -d^*$, arranged from angle $\phi \in [\pi 3/4, \pi 5/4]$ evenly spaced in Cartesian coordinates. The difference appears between swarms with even and odd numbers of drones. Due to the symmetry of the even numbers of drones have successive nearby proximal targets. Odd numbers of drones require constant switching between the lines in front and behind. This leads to significantly more time spent moving for the odd-numbered swarms as the PT solution rotates almost 2π to eliminate four drones at a time. The Threat-Aware Nearest Neighbor with a look-ahead of 5 shows slightly better performance than the NN. Here, the most present threat are the drones closer to the positive and negative x-axis. Once past these critical drones, the others are farther away and can be dispatched easier. By focusing on these targets slightly out of order it is possible to improve over the purely NN solution.

The minimum starting distance for a turret using different targeting approaches against a multi-spiral wave formation are shown in Fig. 6. These results show the minimum starting distances for drone swarms of up to 500 in increments of 5. It includes the NN, PT, and TANN targeting policies with different amounts of look-ahead. Placing the drones in a multi-spiral wave formation causes Nearest Neighbor

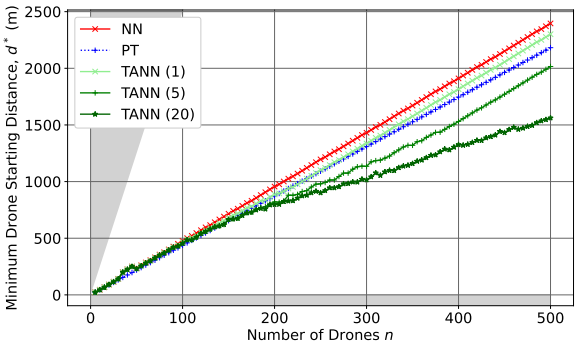

Fig. 6. The minimum starting distance for a turret using different targeting approaches against a multi-spiral wave formation (see schematic in Fig. 2).

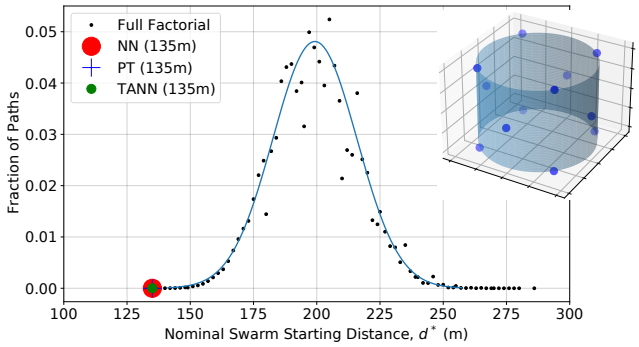

Fig. 7. Path length histogram for all possible permutations of sequences for 12 drones in a 3D cylinder formation (479 million paths). The optimal path sequence will visit all drones for a swarm starting approximately 133 m nominal distance. The total computation time was approximately one hour. The heuristic solutions for NN, PT, and TANN can handle a swarm starting at 135 m with computation times from less than one second to a few seconds.

to perform worse than Proximal Threat. The NN approach ignores the threat of drones approaching it and progresses around to each nearest angular target, including those that are in the outer bands of the formation. The PT approach performs much better because it explicitly targets the next threatening drone. The trade-off is more time spent moving to engage targets.

The Threat-Aware Nearest Neighbor approach balances the performance of the two. Adding a single look-ahead step improves the performance over simple NN, for no noticeable change in computation time. Using five steps of look-ahead, improves the performance to be better than the PT approach. It requires less than half a second on a 3.6 GHz Intel processor to execute the path planning for 100 drones. This increases to about two seconds of computation for 200 drones, as the operation is $O(n^2)$. Including 20 steps of look-ahead improves the performance further, especially as the number of targets increases. It provides up a 50 percent increase in performance over the base NN approach; reducing the minimum starting distance by a third.

B. 3D Simulation

In 3D, the optimization problem is even more difficult than the 2D. As the drones move in indirect attack, seeking to exploit the safety region above a turret, their azimuth angles change. This means that the angular distance between targets changes as the targets move. This means that offline solvers will not generate the overall optimal solution. Fig 7 shows the distribution of starting distances for the full factorial set of all possible paths through a 12-drone swarm in a cylindrical formation. The inset in this figure is a schematic of this swarm in three dimensions on the surface of a cylinder. The possible paths through the drones were computed and simulated to determine the best path providing the smallest d^* where the turret wins. After approximately one hour of computation on an 3.6 GHz Intel processor, the optimal path allowed for a nominal swarm starting radius of 133 m. The OR-tools offline computation, assuming that the drones would not change relative position, generated a path allowing for a a nominal swarm starting radius of 135 m after approximately 180 s. The NN, PT, and TANN approaches found a similar solution for a starting radius of 135 m after less than a second for the two former cases and a few seconds for the latter case.

C. 3D Simulation Results

The addition of the third dimensions opens up an additional challenge for path planning. Fig. 10 shows a schematic arrangement of drones along the bottom half of a hemisphere around a turret. These drones are placed according to a Fibonacci spiral [12]. While they are all the same radial distance from the turret, the drones at higher latitudes are closer to the axis above the turret where they can exploit a weakness in the pan-tilt turret. Similar to the 2D weave pattern, this results in a difference in the threat profile of the different drones. As these drones proceed in horizontal flight, their azimuth angle from the turret will change, complicating the path selection, as the distances between targets change with time. Applying PT selection, the turret consistently favors the high latitude drones and proceeds down the rings of drones. This accounts for the smoothness of the line as the number of drones increases. It also causes the turret to bypass intermediate targets in favor of clearing rings of targets. This also means that the turret spends a significant portion of the engagement moving between targets. In contrast, the NN and TANN approaches show variations as the specifics of the paths vary slightly with slight changes in the number of drones. The precise details of the spacing cause slightly different paths, and the approaches can find shorter paths when the number of drones increases. The NN and TANN (5) approaches perform significantly better than the PT approach. They tend to find paths along the diagonal spirals, but can miss the final drone at the highest latitude, as its azimuth angle moves it out of consideration. While the TANN approach can handle some of these drones, there is only small advantage of the TANN approach over NN.

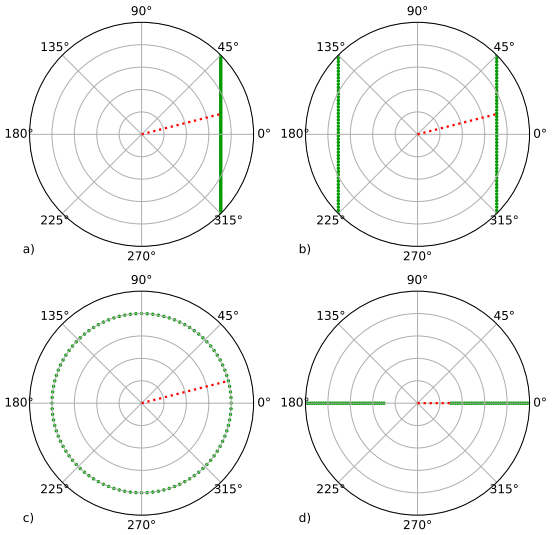

Fig. 8. Swarm formations in 2D with $n = 100$ drones. Line (a), Double Line (b), Circle (c), and Two Radial Lines.

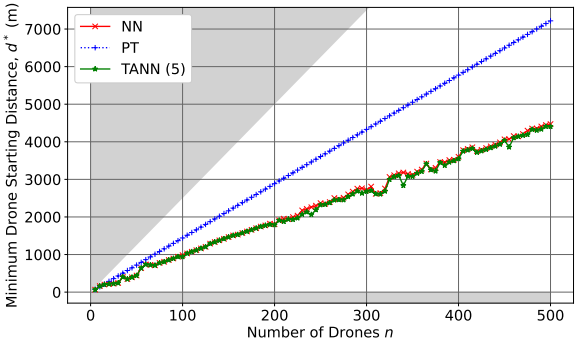

Fig. 9. The minimum starting distance for a turret using different targeting approaches against a truncated hemisphere with a Fibonacci spiral. (See schematic in Fig. 10.)

VI. Conclusions and Future Work

This work introduced a problem related to the TSPTW, where a turret must visit and neutralize a set of drones that are trying to reach the turret. We produced optimal results for a canonical arrangement of targets, and then built 2D and 3D event-based simulators. These simulators were used to test and compare several heuristics against representative 2D and 3D swarms. For the most simple swarm configurations and small numbers of drones, a greedy approach selecting the nearest target is fast and produces optimal or near-optimal results. For other swarms, accounting for the proximal threats yields better performance, with more look-ahead providing better results. These heuristics can serve as performance benchmarks for additional approaches that include additional look-ahead, local optimization, and branch and bound. Future work will improve the heuristics, and investigate performance for drone paths that represent evasive maneuvers as in [13].

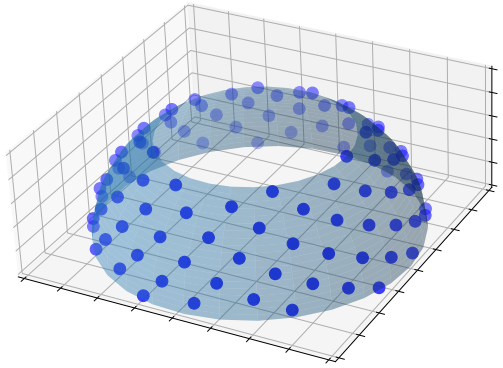

Fig. 10. Targets arranged in a truncated hemisphere using a Fibonacci spiral [12].

References

[1] D. Hambling, "Mystery drones kept buzzing U.S. destroyers during exercises off california," Mar 2021. [Online]. Available: https://www.forbes.com/sites/davidhambling/2021/03/24/mystery-drones-kept-buzzing-us-destroyers-during-exercises-off-california/

[2] K. Fong, "CIWS: The last-ditch defence," *Asian Defence Journal*, vol. Naval Forces, July/August 2008.

[3] H. T. Obering, "Directed energy weapons are real ... and disruptive," *PRISM*, vol. 8, no. 3, pp. 36–47, 2019. [Online]. Available: https://www.jstor.org/stable/26864275

[4] C. Lyu and R. Zhan, "Global analysis of active defense technologies for unmanned aerial vehicle," *IEEE Aerospace and Electronic Systems Magazine*, vol. 37, no. 1, pp. 6–31, 2022.

[5] P. Galison, "The ontology of the enemy: Norbert Wiener and the cybernetic vision," *Critical inquiry*, vol. 21, no. 1, pp. 228–266, 1994.

[6] D. Mindell, "Automation's finest hour: Bell labs and automatic control in World War II," *IEEE Control Sys Mag*, vol. 15, no. 6, p. 72, 1995.

[7] N. Boland, M. Hewitt, D. M. Vu, and M. Savelsbergh, "Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks," in *Integration of AI and OR Techniques in Constraint Programming*, D. Salvagnin and M. Lombardi, Eds. Cham: Springer International Publishing, 2017, pp. 254–262.

[8] S. Dash, O. Gunluk, A. Lodi, and A. Tramontani, "A time bucket formulation for the traveling salesman problem with time windows," *INFORMS Journal on Computing*, vol. 24, pp. 132–147, 02 2012.

[9] O. Dogan and A. F. Alkaya, "A novel method for prize collecting traveling salesman problem with time windows," in *Intelligent and Fuzzy Techniques for Emerging Conditions and Digital Transformation*. Cham: Springer International Publishing, 2022, pp. 469–476.

[10] L. Perron and V. Furnon, "OR-Tools," Google. [Online]. Available: https://developers.google.com/optimization/

[11] D. Biediger, L. Popov, and A. T. Becker, "The pursuit and evasion of drones attacking an automated turret," in *IEEE/RSJ IROS*, 2021, pp. 9677–9682.

[12] D. P. Hardin, T. Michaels, and E. B. Saff, "A comparison of popular point configurations on s2," *arXiv preprint arXiv:1607.04590*, 2016.

[13] D. Biediger, A. Mahadev, and A. T. Becker, "Investigating the survivability of drone swarms with flocking and swarming flight patterns using virtual reality," in *IEEE CASE*. IEEE, 2019, pp. 1718–1723.