Intercepting Unmanned Aerial Vehicle Swarms with Neural-Network-Aided Game-Theoretic Target Assignment

Nicholas G. Montalbano and Todd E. Humphreys
Aerospace Engineering & Engineering Mechanics
The University of Texas at Austin
Austin, TX, USA
nick30075@utexas.edu
todd.humphreys@utexas.edu

***Abstract*—This paper examines the use of neural networks to perform low-level control calculations within a larger game-theoretic framework for drone swarm interception. As unmanned aerial vehicles (UAVs) become more capable and less expensive, their malicious use becomes a greater public threat. This paper examines the problem of intercepting rogue UAV swarms by exploiting the underlying game-theoretic nature of large-scale pursuit-evasion games to develop locally optimal profiles for target assignment. It paper also examines computationally efficient means to streamline this process.**

***Index Terms*—pursuit-evasion game; neural network; UAV swarm control.**

I. Introduction

In January, 2015, the U.S. White House was briefly locked down when a stray unmanned aerial vehicle (UAV) entered its grounds [1]. In December 2018, London's Gatwick airport was shut down for nearly three days by deliberate drone incursions [2]. In an age of highly capable and increasingly autonomous UAVs purchasable for a few hundred dollars, future such intrusions can be expected to occur in various forms at sensitive sites across the globe. Electronic countermeasures, an obvious first response to such incursions, may not be effective against deliberate attacks [3], and directed energy defenses may be overwhelmed by a swarm attack. This paper contemplates a scenario in which a hostile UAV swarm is parried by a swarm of intercepting UAVs.

Interception of hostile UAV swarms has been studied from a variety of perspectives. Playbook-based approaches in which a human operator chooses from a set of predefined strategies for a swarm to autonomously execute have been examined both for effectiveness and for burden on the human operator [4], [5]. Such a system, while adequate for simple interception tasks, scales poorly to larger swarms or complex environments in which an operator has difficulty comprehending the full state space of the interception. Furthermore, while the human involvement necessary is limited, the system is not fully autonomous and thus not provably optimal.

Some approaches seeking full autonomy have treated the motion of members of the target swarm as a Markov decision process in a discretized space, distributing interceptors as necessary to guarantee successful interception [6]. This approach fails to consider which behaviors the attacker deems optimal; for example, an attacker known to have a target destination can be more easily intercepted by a defending interceptor who cuts off the attacker's intended path. Other work has examined the case in which interceptors seek to herd a target rather than destroy it [7], [8]. These techniques present guarantees of eventual success but either decline to include optimality criteria for interceptors (such as minimizing time or fuel usage) or only consider optimality in cases with tightly-constrained action sets.

Other solutions treat interception as a game of surveillance, with interceptors searching a space (discretized or continuous) for a target (or swarm of targets). Some treat the problem as a surveillance game adapted from the traveling salesman problem with the assumption that all targets or defended areas are of equal importance [9]. In this school of thought, the interception is considered complete when the pursuer is either well-localized or herded to some final area. This approach, again, ignores attacker priorities.

Decentralized swarm control with regret monitoring has emerged as a popular control scheme for directing UAV swarms [10], [11]. In regret monitoring, instead of directly cooperating on a strategy, interceptors communicate to their neighbors which strategies they wish that had played at the previous time step. This scheme has been shown to converge to the Nash equilibrium of all interceptor costs, given sufficient time. Though this family of techniques can be generalized to consider the likelihood of successfully defending targets, the attacker's priorities are again neglected. Speed of convergence can also be a concern for fast-paced applications [11].

A solution that does not plan around the attacker's goals may not be optimal in a Nash sense for real-world applications. The problem that this work will consider consists of two swarms of UAVs with known high-level objectives and costs. The attacking (defending) swarm desires to evade (hasten) capture and move towards (herd the other swarm away from) a set of predefined objectives with costs known to all

May 2020 preprint of paper submitted for publication

parties. This mutual knowledge of costs is key to streamlining behavior, anticipating that some strategies offer limited benefit to one swarm and conserving resources that would otherwise be spent countering an unlikely action.

The underlying game-theoretic problem is likely computationally intractable. It can be solved using dynamic programming, but such solutions are only guaranteed to converge if the game is zero-sum [12]. This assumption may not be reasonable for real-world applications; a zero-sum game requires that one player's loss is explicitly represented in the other player's cost function as a reward. Consider the case of an interception game surrounding a set of ground targets. The defender may place more value on defending a target than the attacker does on attacking it due to differences in the perceived value of the target and what role each swarm believes that target may play in future interactions. The resulting game is non-zero sum.

This paper splits the swarm interception problem into a high-level game of assigning a defensive interceptor to each attacker and a low-level game of pursuit-evasion played out by each pairing. The result can be thought of as the game-theoretic equivalent of a local minimum of the underlying problem, one created by finding the optimal solutions to two subproblems.

This paper assumes known attacker objectives to produce a defensive solution that anticipates the behavior of the attacker swarm and more efficiently plans actions for defenders using this knowledge. It seeks a Nash equilibrium of an underlying game of reduced complexity with some simplifications for computational efficiency. Neural networks are applied to develop control profiles for the low-level game and enhance cost generation of the high-level game, reducing the overall system's response time, thus improving scalability. Finally, this paper examines the usefulness of heuristic pruning of strategies to reduce the game's search space.

The remainder of this paper is organized as follows. Following a brief review of background material in Section II, Section III outlines the problem and introduces a split between the high- and low-level parts of the swarm interception problem. Section IV provides a simple outline for neural network training procedures and evaluation for the high-level problem. Heuristics using iterative strict dominance to bound swarm costs and eliminate non-optimal strategies are introduced in the latter half of Section IV. Finally, Section V provides simulation results for the proposed algorithms.

II. Background: Game Theory and Nash Equilibria

In an n-player perfect-information game, two or more players must choose from a finite set of strategies to minimize some mutually-known costs. A classic example is the Prisoner's Dilemma, a game in which two conspirators face a choice after being arrested: the prisoners may either cooperate with one another by remaining silent, or may defect by confessing to the crime. If both cooperate, they are convicted of a lesser charge and sent to prison for 2 years; if both defect, they are sentenced to 4 years. If one defects and the other cooperates, the defector goes free on a plea deal whereas the cooperator spends 5 years in prison. The dilemma can be cast as a bimatrix non-zero-sum game as shown in Table I.

TABLE I: A standard two-player two-action game, with costs to Players 1 & 2 represented in the table entries as (C_1, C_2).

	Player 2	
Player 1	Cooperate	Defect
Cooperate	(2, 2)	(5, 0)
Defect	(0, 5)	(4, 4)

A Nash equilibrium is the strategy pair for which each player's best response to the opponent's action is to play the equilibrium pair, essentially ensuring convergence on a given strategy driven by mutual self-interest. The Prisoner's Dilemma equilibrium is (Defect, Defect), which is counterintuitive, as the (Cooperate, Cooperate) pair produces the best joint outcome. But a player who knows that the other will cooperate would rather defect in order to minimize his own cost and fear of the other player defecting prompts rational, self-interested players to defect, thus making (Defect, Defect) the equilibrium action of the game. In terms of costs J_1 and J_2 as a function of actions u_1 and u_2 for Players 1 and 2, the Nash equilibrium can be expressed as the action pair (u_1^*, u_2^*) from which neither player profits by unilateral deviation:

$$J_1(u_1^*, u_2^*) \leq J_1(u_1, u_2^*)$$
$$J_2(u_1^*, u_2^*) \leq J_2(u_1^*, u_2)$$

Uncertainty in costs complicates the underlying problem. For simple games played in the presence of noise, if multiple equilibria emerge, the risk-dominant equilibrium (lowest product of costs of deviation from equilibrium) proves to be the most robust [13] [14].

III. Problem Formulation

This paper examines the problem of interception of individual UAVs in an attacker-swarm-versus-defender-swarm engagement between two parties consisting of M defending interceptors and N attackers, with $M > N$. The dynamics of all vehicles are assumed known, as are the costs (see Section III-A). While it may be impractical for real-world problems to assume that all cost parameters are known, it is standard within the literature to assume large amounts of mutual knowledge [15]. An evaluation of the necessity of this assumption is left to future work.

Each attacker UAV also has a mutually known ground target, with a reward to the attacking swarm if the relevant UAV arrives at its location and a penalty to the defending swarm for permitting this. These costs and values are not necessarily zero-sum; one side, for example, may prioritize long-term goals over short-term goals so protecting/damaging easily-replaced resources may be more valuable to one side's higher-level plans than the other's. However, this concession adds a layer of complexity to the problem–as the game is now necessarily non-zero sum, many standard numerical solution techniques no longer apply [16].

Finally, each side receives a discount to its cost function for leaving forces in reserve, rewarding it for preserving forces for a later engagement. If the weight of this discount were sufficiently high, a refusal by both parties to sortie would represent a Pareto-optimal equilibrium (a Nash equilibrium with the lowest costs to each player) of the underlying game. It is assumed that the discount factor is sufficiently high to motivate some UAVs to stay in reserve, but not the entire swarm. UAVs can be withdrawn from service at any time during the engagement.

A. Costs and Dynamics

Denoting the current position/velocity state of the i^{th} defender as x_i, the time index as k, the control action applied by player i at time k as $u_i(k)$, each defending UAV has mutually known dynamics

$$x_i(k+1) = f_i(k, x_i(k), u_i(k), w_i(k)) \tag{1}$$

where $w_i(k)$ is normally distributed Gaussian white noise sampled from a known covariance matrix W_i. The covariance is assumed sufficiently small to promote noise-independent selection of Nash equilibria [14]. Using the subscript j to denote attacker states, the attacker UAVs have dynamics

$$x_j(k+1) = f_j(k, x_j(k), u_j(k), w_j(k)) \tag{2}$$

The cost function of the defending swarm consists of a sum of quadratic fuel costs, the sum of the proximity of each sortied attacker to the nearest interceptor, a penalty for losing one of the T targets, and the discount for maintaining a reserve force.

$$\begin{aligned} J_d = &\sum_{i}^{M} u_i^T R_d u_i \\ &+ \sum_{j}^{N} (x_j - x_{d,n})^T Q_d (x_j - x_{d,n}) \\ &- \sum_{k}^{T} P_d V_{d,k} - M_{rsv} V_{rsv} \end{aligned} \tag{3}$$

Similarly, the attacker's cost function is

$$\begin{aligned} J_a = &\sum_{j}^{N} u_j^T R_a u_j \\ &- \sum_{j}^{N} (x_j - x_{d,n})^T Q_a (x_j - x_{d,n}) \\ &- \sum_{k}^{T} P_a V_{a,k} - N_{rsv} V_{rsv} \end{aligned} \tag{4}$$

R_a, Q_a, R_d, $Q_d > 0$. The signs of the proximity terms are opposite for each cost function so as to capture the competing goals of the defending and attacking swarms–the defender wishes to reduce the distance between UAVs while the attacker wishes to increase this distance.

B. Solution Techniques

The players seek a Nash equilibrium set of controls (u_d^*, u_a^*) of the paired cost functions defined in Equations 3 and 4 such that $J_d(u_d^*, u_a^*) < J_d(u_d, u_a^*)$ and $J_a(u_d^*, u_a^*) < J_a(u_d^*, u_a)$. Determining this equilibrium may be difficult. The game is both stochastic and non-zero sum, which disqualifies conversion of the problem to one amenable to dynamic programming as outlined in [16]. Optimal control techniques performed in an iterative best response framework could be used to solve this problem but the large state space of this problem makes this approach computationally intractable [17].

Instead, this paper seeks to decompose the swarm-versus-swarm game into two sets of games. The first is a large number of low-level games between paired attackers and interceptors. The higher-level game determines these pairings (assignments) and whether or not to hold a given UAV in reserve. Existing solutions for solving low-level pursuit-evasion games require significant assumptions, ranging from ignoring the effects of noise to sharing raw sensor data [15], [18]. This paper uses a stochastic mixed trajectory technique which discretizes the control space of each party and solves for the Nash equilibrium of the underlying stochastic non-zero sum game. This approach represents the local equivalent to a global Nash equilibrium; while it lacks guarantees of optimality a global sense, this approach provides an avenue for real-time implementability.

C. The Low-Level Game

Each low-level game is treated as a standard pursuit-evasion game [19]. The UAVs belonging to the attacker and defender swarms have costs derived from the swarm cost function, with the cost function weights for the attacker UAV corresponding to the cost function weights of the attacker UAV within the higher-level swarm game.

Solving these games is computationally intensive. For the case of linear dynamics and quadratic costs, a pair of coupled differential (algebraic) Riccati equations can be solved for the control value pair that represents the Nash equilibrium of the continuous-time (discrete-time) game [20]. A defender could precompute low-level controls to improve real-time implementability, but this requires either advance knowledge of all cost functions or interpolation over a very large set of precomputed game matrices. This approach is too restrictive for real-world implementation.

This lack of advance knowledge of cost functions invites investigation of alternative techniques. Though a discretized mixed trajectory approach is computationally burdensome [20], its ease of adaptation to games with nonlinear dynamics and non-LQR costs makes it ideal for this use case. The portion of the algorithm that involves shooting of trajectories is easily parallelized and can be used as a fall back in limited cases [12]. This paper will continue by using a modified mixed trajectory approach that handles noise by examining risk-dominance of equilibria in cases in which a pure strategy Nash equilibrium does not emerge. In the deterministic case, this approach reduces to [20]'s mixed trajectories.

To further enable real-time implementability, this work instead uses a neural network to approximate the solutions of these low-level games.

IV. Algorithms

Algorithms and potential heuristic modifications will be presented here.

A. Solving the Low-Level Game

Denote the simulation horizon as k_s and planning horizon as k_h, the cost function structure as J_s, admissible range of cost function parameters as J_c, actuator constraints $u_{\max}$, number of training simulations as k_S, game dynamics f, controller (producing a (u_A^*, u_D^*) pair) g_u, and initial state constraints $x_{\max}$. The resulting low-level controller is presented in Algorithm 1.

Algorithm 1: Train/predict controls

Input : k_s, J_s, J_c, $x_{\max}$, $u_{\max}$, k_s
Output: N

Training Time:

$D = [\,]^{k_S k_h}$
$S = [\,]^{k_S k_h}$
$n = 0$

for *i = 1:*k_S **do**
  $J_p = \texttt{rand}(J_c)$
  $x_A, x_D = \texttt{rand}(x_{\max})$
  for *j = 1:*k_s **do**
    $(u_A, u_D) = g_u(x_A, x_D, u_{\max}, k_h)$
    $D(n) = \{x_A, x_D, J_p\}$
    $S(n) = \{u_A, u_D\}$
    $x_A, x_D = f(x_A, u_A, x_D, u_D)$
    $n++$
  end
end

$n_o = \texttt{randsample}(1 : k_s k_s)$
$D = D(n_o)$
$S = S(n_o)$

$N = \texttt{new\_neural\_network}$
$N = \texttt{train\_network}(D, S)$

Evaluation Time:

$(u_A, u_D) = \texttt{predict}(N, x_A, x_D, J_p)$

The initial states and cost function parameters are randomized at each new simulation iteration to ensure that the neural network can be used to handle a broader range of interception games without requiring specialized advance knowledge. Training data is randomized after generation in order to ensure that each training batch represents a variety of cost parameters as well as a variety of states representative of a late-stage pursuit. This provides a mild performance improvement in practice. This may be a result of the learning rate discount providing decreasing weight to batches learned later in training; varying the cost parameters in batches seen early in training will improve performance.

B. Solving the High-Level Game

During selection of target assignment, the above neural network is evaluated to provide estimates for control pairs in order to enhance evaluation speed. Pairings are evaluated against a global cost function at the swarm level. Low-level pursuit-evasion games are governed by cost matrices representing the proximity costs of the global cost functions. Denote the lists of attackers and defenders as L_A and L_D, respectively, the list of attacker targets and values as X_a and V_a, the costs to the defender of a lost target as J_t, swarm cost functions as J_a and J_d, cost parameters for the low-level games J_p, and inputs to Algorithm 1 as S. Denoting the output set of swarmwide controls for each party as (u_A^*, u_D^*), the resulting target assignment algorithm is presented in Algorithm 2.

Algorithm 2: Target assignment

Input : L_A, L_D, X_a, V_a, J_a, S, N, J_p
Output: (u_A^*, u_D^*)

$U_A, U_D = [\,]$

for $i = 1 : \texttt{length}(L_A)$ **do**
  $(U_A(i),\ .\ ) = \texttt{predict}(N, x_A, x_D, J_p)$
end

for $j = 1 : \texttt{length}(L_D)$ **do**
  $(\ .\ , U_D(j)) = \texttt{predict}(N, x_A, x_D, J_p)$
end

$J_A, J_D = [\,]$

for $i = 1 : \texttt{length}(L_A)$ **do**
  for $j = 1 : \texttt{length}(L_D)$ **do**
    $J_A(i,j) = J_a(U_A(i), U_D(j), x_A, x_D, X_a, V_a)$
    $J_D(i,j) = J_d(U_D(i), U_D(j), x_A, x_D, X_a, J_t)$
  end
end

$(u_A^*, u_D^*) = \texttt{Nash}(J_A, J_D)$

Cost function terms related to ground targets are evaluated over the active sets of sortied attackers. Aside from the control generation and `Nash` steps, the runtime of Algorithm 2 is second-order in the mean list size. It may be advantageous to develop some heuristics with which to prune the assignment lists in order to reduce computational costs.

C. Heuristics and Computation Time

The `Nash` step of Algorithm 2 can be computationally intensive for large swarms. To generate a non-reserve list of possible launches of a set of n drones creates $\Sigma_{i=1}^{n}\binom{n}{i}$ items, which then must be paired with possible launches of a similar size from the other swarm. The possible pairing matrix which must be solved in the `Nash` step grows as $\texttt{min}(n_{\texttt{attacker}}, n_{\texttt{defender}})^2$. Solution algorithms for this matrix, such as the Lemke-Howson algorithm, are roughly seventh-order in matrix size for standard implementations [21]. This is computationally impractical for real-world applications.

As distance between paired attacker and defender UAVs becomes large, the proximity cost and value terms ($\Delta x^T Q \Delta x$) come to dominate the control cost terms ($u^T R u$) and inputs

reach their maximum allowable limits. A simple heuristic could lower- and upper-bound the costs to the attacker of each pairing, with the lower-bound cost for the attacker being the case in which the defender moves towards the attacker with maximum thrust and the upper-bound cost for the attacker being the case in which the defender flees with maximum thrust. Modifying a standard velocity matching interception technique [22] with motion prediction based on the above bounds permits rapid shooting of trajectories to calculate asuccession of states. From here, iterative strict dominance (iteratively eliminating strategies whose lower-bounded costs exceed another strategy's upper-bounded costs for all pairings) can be used to prune the cost matrices and reduce the search space for possible assignments. Denoting the maximum control inputs of the attacker and defender as v_a and v_d, respectively, a proposed algorithm for heuristic pruning an attacker's input control set L_A to a reduced set l_A is presented in Algorithm 3.

Algorithm 3: Attacker strategy pruning

Input : L_A, L_D, X_a, V_a, J_a, S, v_a, v_d
Output: l_A

$U_A, U_D^-, U_D^+ = [\,]$
for $i = 1 : \texttt{length}(L_A)$ **do**
  for $j = 1 : \texttt{length}(L_D)$ **do**
    $\hat{u}_m = \texttt{unit\_vector}(\texttt{heading}(\texttt{velocity\_match}(X_a, X_d)))$
    $U_A(i) = v_a \hat{u}_m$
    $U_D^+(j) = v_d \hat{u}_m$
    $U_D^-(j) = -v_d \hat{u}_m$
  end
end
$J_A^-, J_A^+ = [\,]$
for $i = 1 : \texttt{length}(L_A)$ **do**
  for $j = 1 : \texttt{length}(L_D)$ **do**
    $J_A^-(i,j) = J_A(U_A(i), U_D^-(j), x_A, x_D, X_a, V_a)$
    $J_A^+(i,j) = J_A(U_A(i), U_D^+(j), x_A, x_D, X_a, V_a)$
  end
end
$l_A = L_A$
for $i = 1 : \texttt{length}(L_A)$ **do**
  for $j = 1 : \texttt{length}(L_A), j \neq i$ **do**
    $D = J_A^-(i,:) - J_A^+(j,:)$
    $s = \texttt{signs}(D)$
    if $\texttt{min}(s) \geq 0$ **then**
      $l_A = l_A(:\neq i)$
    end
  end
end

The defending swarm may apply a similar pruning approach, assuming that the assigned defender UAV applies maximum input towards the attacker UAV and that the attacker UAV applies maximum input towards (away from) the defender, thus forming an upper (lower) bound on costs.

These two pruning algorithms can be applied iteratively in alternating attacker-defender order in order to iteratively reduce the size of pairing lists necessary to examine. This step is $O(n^2)$ in the number of UAVs launched by either swarm but it reduces the size of the input to the required $O(n^7)$ solution algorithm.

Algorithm 4: Defender strategy pruning

Input : L_A, L_D, X_a, V_a, J_a, S, v_a, v_d
Output: l_D

$U_A^-, U_A^+, U_D = [\,]$
for $i = 1 : \texttt{length}(L_A)$ **do**
  for $j = 1 : \texttt{length}(L_D)$ **do**
    $\hat{u}_m = \texttt{unit\_vector}(\texttt{heading}(\texttt{velocity\_match}(X_a, X_d)))$
    $U_A^+(i) = v_a \hat{u}_m$
    $U_A^-(i) = -v_a \hat{u}_m$
    $U_D^-(j) = v_d \hat{u}_m$
  end
end
$J_D^-, J_D^+ = [\,]$
for $i = 1 : \texttt{length}(L_A)$ **do**
  for $j = 1 : \texttt{length}(L_D)$ **do**
    $J_D^-(i,j) = J_D(U_A^-(i), U_D(j), x_A, x_D, X_a, V_a)$
    $J_D^+(i,j) = J_D(U_A^+(i), U_D(j), x_A, x_D, X_a, V_a)$
  end
end
$l_D = L_D$
for $i = 1 : \texttt{length}(L_D)$ **do**
  for $j = 1 : \texttt{length}(L_D), j \neq i$ **do**
    $D = J_D^-(:,i) - J_D^+(:,j)$
    $s = \texttt{signs}(D)$
    if $\texttt{min}(s) \geq 0$ **then**
      $l_D = l_D(:\neq i)$
    end
  end
end

V. Results

To validate Algorithm 1, a neural network was trained on 1500 full simulations of simple pursuit-evasion games with double integrator dynamics with an additional aerodynamic drag term proportional to velocity $\dot{v} = -0.3v$. Iterative testing found that a neural network composed of six layers provided the best performance for these data, shown in Figure 1.

Splitting the neural network into two networks, one designed to predict pursuer controls and a second designed to predict evader controls, provides further insight into the behavior of the neural network trained for this problem, shown in Figures 2 and 3.

An identically structured seven-layer neural network was used for each case. The discrepancy between neural network quality for pursuer and evader cases is left to later research. Simulations with the neural networks provided worse performance than a hand-tuned velocity matching controller; this may be the result of velocity matching providing controls close to the Nash-optimal values for low control costs. This work will proceed using Algorithm 2 as outlined above, though

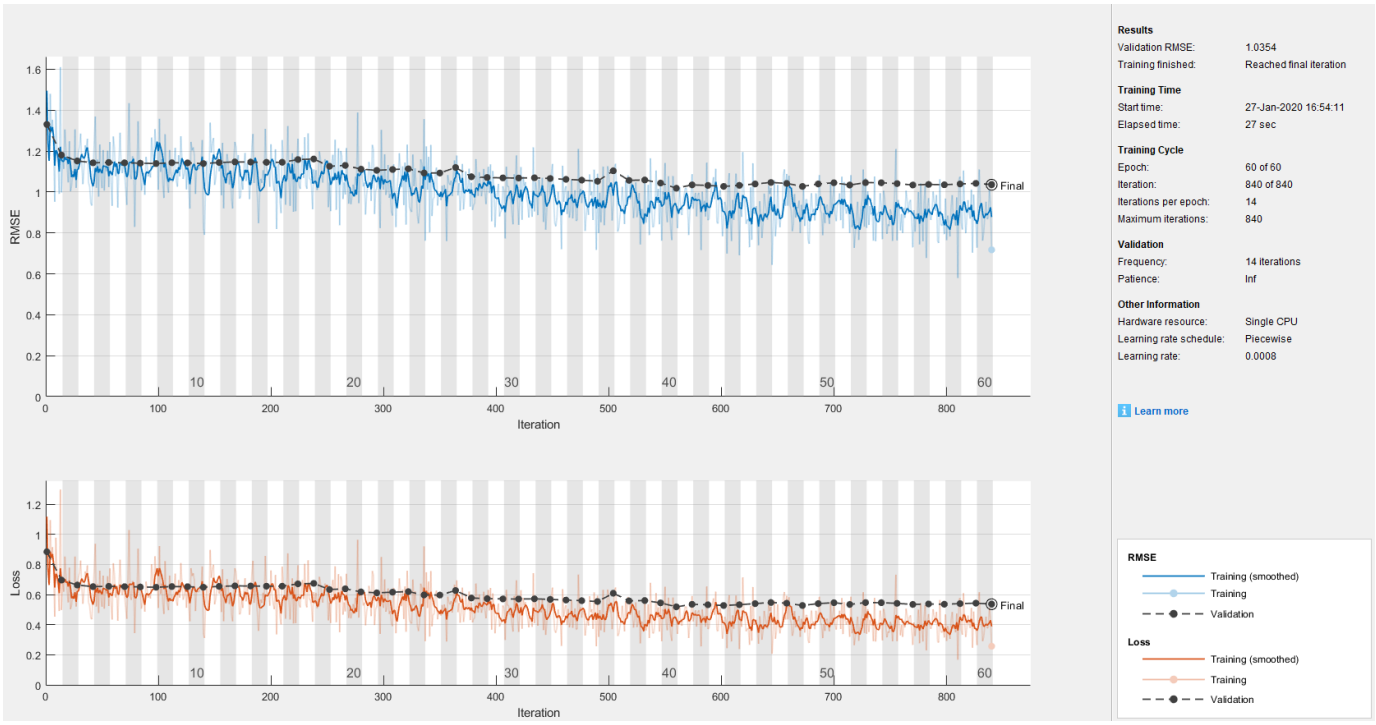

Fig. 1: Matlab-generated training summary. The six-layer neural network converges to its final error range after approximately 600 training steps. A final RMSE value of 1.0354 is obtained; this is the RMSE value of the concatenated attacker/defender control vector with a control input range varying on $[-2, 2]$. In the worst-case scenario that error is constant across all output states, this represents an error of approximately 7% compared to the standard algorithm.

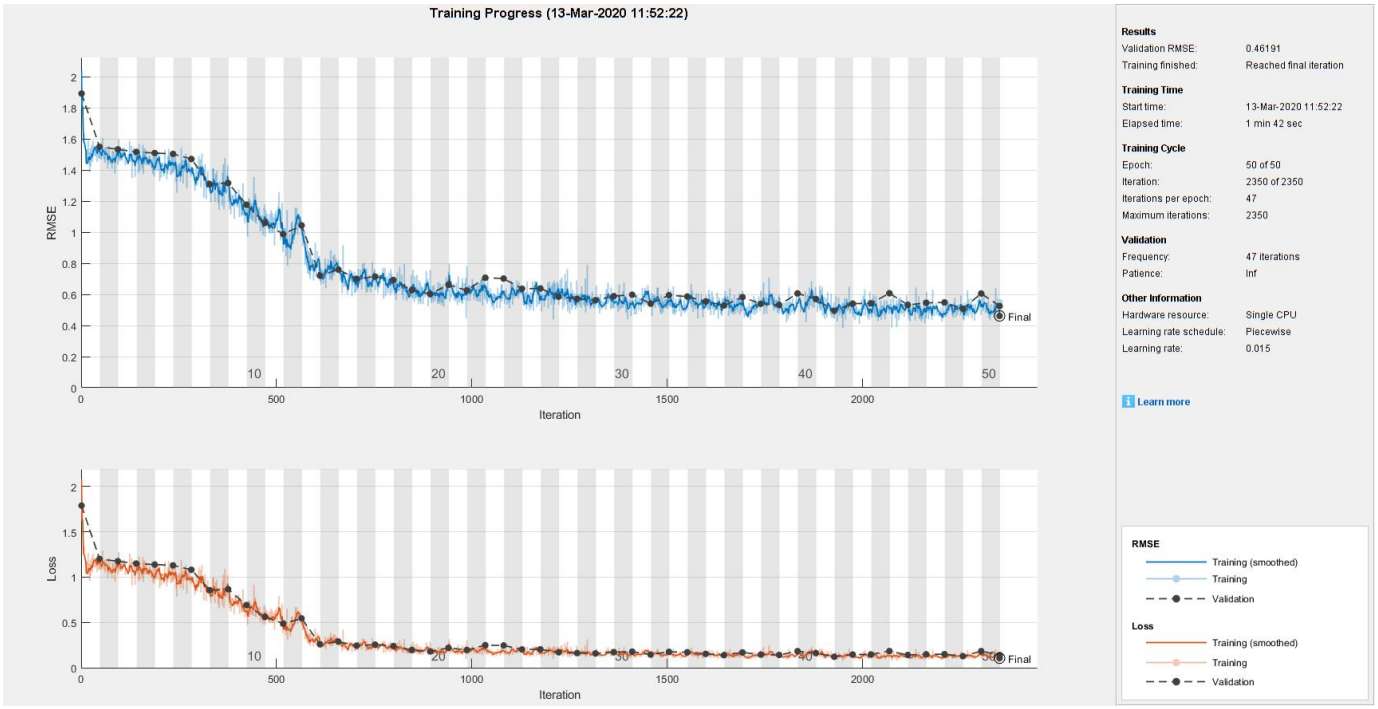

Fig. 2: Matlab-generated training summary for neural network approximating pursuer controls only. The network provides better accuracy (lower RMSE) for pursuer controls alone than it does for simultaneous generation of pursuer and evader controls.

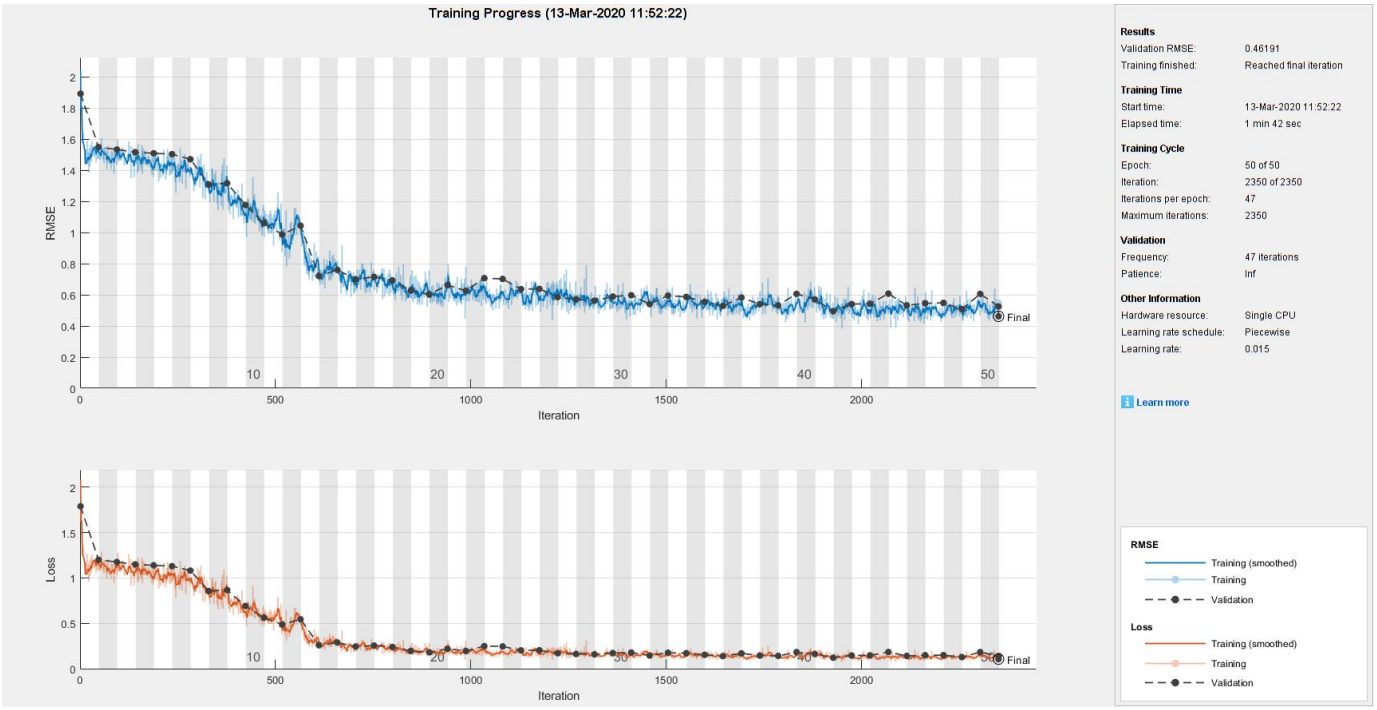

Fig. 3: Matlab-generated training summary for neural network approximating evader controls only. The network provides worse accuracy (higher RMSE) for evader controls alone than it does for simultaneous generation of pursuer and evader controls.

for simple problems, velocity matching may prove to produce lower errors (and thus, lower costs) than a neural network.

To validate Algorithm 2, a simulation was performed using randomized cost weights and attacker and defender swarms initialized in randomized clusters. Each swarm consisted of three UAVs.

Algorithms 3 and 4 were then implemented and added to the simulation.

The heuristic successfully reduced the number of evaluations of the low-level controller. However, the increased number of evaluations of the high-level cost functions increased total computational time. It is likely that, with a less efficient low-level controller, Algorithms 3 and 4 will provide substantial performance gains.

VI. Conclusions

Four algorithms have been presented to efficiently generate control actions, assign targets, and prune target assignments for attacking and defending swarms involved in a large-scale attack on a series of defended targets. This approach is capable of exploiting knowledge of attacker optimality in order to avoid behaviors that may lead to failures in longer engagements. A series of simulation experiments have been performed to validate four proposed algorithms, though unexpectedly poor performance was observed with the latter two algorithms.

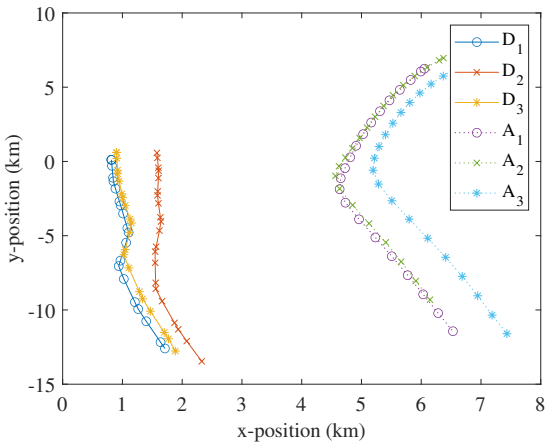

Fig. 4: Attacker and defender swarm trajectories plotted over a 20 time step attack/defense simulation. The attacking swarm initially moves from its initial location in the upper right towards the origin, where their targets are clustered, but the defending swarm (initialized near the origin) moves to head them off. The attacking swarm drops several UAVs from its active set (represented by the UAVs which begin to coast in a nearly straight line). The attacking attempts to withdraw them but their momentum forces the defenders to treat them as hostile in order to prevent a potential reactivation. This eventually drives the game into a stalemate in which all players on each swarm are active.

References

[1] M. S. Schmidt and M. D. Shear, "A drone, too small for radar to detect, rattles the White House," Jan. 2015, https://www.nytimes.com/2015/01/27/us/white-house-drone.html.

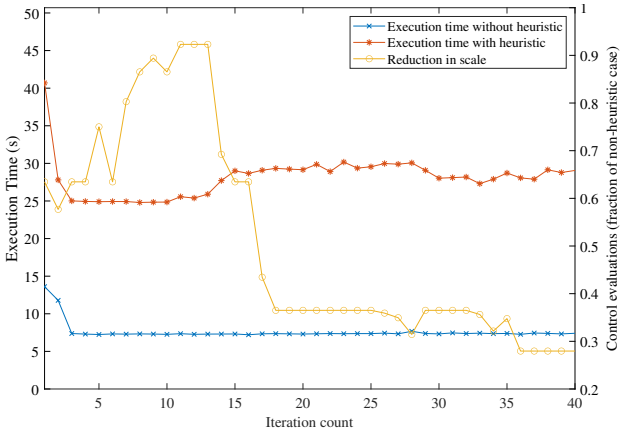

Fig. 5: Execution time and low-level control evaluations with and without the heuristic. For this simple case with a neural network generating low-level control commands, the velocity matching technique only reduces the computational time of the low-level controller slightly. The increase in evaluations of the cost function (as heuristics must be evaluated at least three times) increase the total computational burden. The total evaluations of the low-level controller are reduced by more than 60% through a long interception. If a low-level controller with a high computational cost (with a runtime of three to four times the computational time of evaluating the costs of the high-level game), the heuristic may drastically reduce computational time.

[2] B. Mueller and A. Tsang, "Gatwick airport shut down by 'deliberate' drone incursions," Dec. 2018, https://www.nytimes.com/2018/12/20/world/europe/gatwick-airport-drones.html.

[3] T. E. Humphreys, "Congressional testimony: Statement on the security threat posed by unmanned aerial systems and possible countermeasures," Mar. 2015.

[4] R. P. Goldman, K. Z. Haigh, D. J. Musliner, and M. J. Pelican, "Macbeth: a multi-agent constraint-based planner [autonomous agent tactical planner]," in *Proceedings. The 21st Digital Avionics Systems Conference*, vol. 2. IEEE, 2002, pp. 7E3–7E3.

[5] R. Simmons, D. Apfelbaum, D. Fox, R. P. Goldman, K. Z. Haigh, D. J. Musliner, M. Pelican, and S. Thrun, "Coordinated deployment of multiple, heterogeneous robots," in *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113)*, vol. 3. IEEE, 2000, pp. 2254–2260.

[6] M. Sinay, N. Agmon, O. Maksimov, G. Levy, M. Bitan, and S. Kraus, "UAV/UGV search and capture of goal-oriented uncertain targets."

[7] S. Nardi, F. Mazzitelli, and L. Pallottino, "A game theoretic robotic team coordination protocol for intruder herding," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4124–4131, 2018.

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

[9] M. Pavone, E. Frazzoli, and F. Bullo, "Adaptive and distributed algorithms for vehicle routing in a stochastic and dynamic environment," *IEEE Transactions on Automatic Control*, vol. 56, no. 6, pp. 1259–1274, 2010.

[10] M. Khosravi, H. Khodadadi, H. Rivaz, and A. G. Aghdam, "Cooperative control for multi-target interception with sensing and communication limitations: A game-theoretic approach," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 1048–1053.

[11] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: A game-theoretical formulation," 2007.

[12] H. Zhang, Q. Wei, and D. Liu, "An iterative adaptive dynamic programming method for solving a class of nonlinear zero-sum differential games," *Automatica*, vol. 47, no. 1, pp. 207–214, 2011.

[13] H. Carlsson and E. Van Damme, "Global games and equilibrium selection," *Econometrica: Journal of the Econometric Society*, pp. 989–1018, 1993.

[14] D. M. Frankel, S. Morris, and A. Pauzner, "Equilibrium selection in global games with strategic complementarities," *Journal of Economic Theory*, vol. 108, no. 1, pp. 1–44, 2003.

[15] P. Kumar and J. Van Schuppen, "On Nash equilibrium solutions in stochastic dynamic games," *IEEE Transactions on Automatic Control*, vol. 25, no. 6, pp. 1146–1149, 1980.

[16] E. N. Barron, L. C. Evans, and R. Jensen, "Viscosity solutions of Isaacs' equations and differential games with Lipschitz controls," *Journal of Differential Equations*, vol. 53, no. 2, pp. 213–233, 1984.

[17] D. Fridovich-Keil, E. Ratner, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," *arXiv preprint arXiv:1909.04694*, 2019.

[18] Y. Yavin, "Stochastic pursuit-evasion differential games in the plane," *Journal of optimization theory and applications*, vol. 50, no. 3, pp. 495–523, 1986.

[19] W. Willman, "Formal solutions for a class of stochastic pursuit-evasion games," *IEEE Transactions on Automatic Control*, vol. 14, no. 5, pp. 504–509, 1969.

[20] T. Basar and G. J. Olsder, *Dynamic noncooperative game theory*. SIAM, 1999, vol. 23.

[21] B. Codenotti, S. De Rossi, and M. Pagan, "An experimental analysis of Lemke-Howson algorithm," *arXiv preprint arXiv:0811.3247*, 2008.

[22] F. Kunwar and B. Benhabib, "Rendezvous-guidance trajectory planning for robotic dynamic obstacle avoidance and interception," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1432–1441, 2006.