V-FORMATION AS MODEL PREDICTIVE CONTROL

RADU GROSU, ANNA LUKINA, SCOTT A. SMOLKA, ASHISH TIWARI, VASUDHA VARADARAJAN, AND XINGFANG WANG

Cyber-Physical Systems Group, Technische Universität Wien, Austria
e-mail address: radu.grosu@tuwien.ac.at

Institute of Science and Technology Austria
e-mail address: anna.lukina@ist.ac.at

Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
e-mail address: sas@cs.stonybrook.edu

Microsoft Research, USA
e-mail address: Ashish.Tiwari@microsoft.com

Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
e-mail address: vvaradarajan@cs.stonybrook.edu

Department of Computer Science, Stony Brook University, Stony Brook, NY, USA
e-mail address: wxingfang@cs.stonybrook.edu

Key words and phrases: V-Formation, Model Predictive Control, Markov Decision Processes, Controller-Attacker Games.

Preprint submitted to
Logical Methods in Computer Science

Abstract. We present recent results that demonstrate the power of viewing the problem of V-formation in a flock of birds as one of Model Predictive Control (MPC). The V-formation-MPC marriage can be understood in terms of the problem of synthesizing an optimal plan for a continuous-space and continuous-time Markov decision process (MDP), where the goal is to reach a target state that minimizes a given cost function.

The first result we consider is ARES, an efficient approximation algorithm for generating optimal plans (action sequences) that take an initial state of an MDP to a state whose cost is below a specified (convergence) threshold. ARES uses Particle Swarm Optimization, with *adaptive sizing* for both the receding horizon and the particle swarm. Inspired by Importance Splitting, the length of the horizon and the number of particles are chosen such that at least one particle reaches a *next-level* state, i.e., a state where the cost decreases by a required delta from the previous-level state. The level relation on states and the plans constructed by ARES implicitly define a Lyapunov function and an optimal policy, respectively, both of which could be explicitly generated by applying ARES to all states of the MDP, up to some topological equivalence relation. We assess the effectiveness of ARES by statistically evaluating its rate of success in generating optimal plans for V-formation.

ARES can alternatively be viewed as a model-predictive control (MPC) algorithm that utilizes an adaptive receding horizon, a technique we refer to as Adaptive MPC (AMPC). We next present Distributed AMPC (DAMPC), a distributed version of AMPC that works with local neighborhoods. We introduce adaptive neighborhood resizing, whereby the neighborhood size is determined by the cost-based Lyapunov function evaluated over a global system state. Our approach applies to reachability problems for any collection of entities that seek convergence from an arbitrary initial state to a desired goal state, where a notion of distance to the goal state(s) can be suitably defined. Our experimental evaluation shows that DAMPC can perform almost as well as centralized AMPC, while using only local information and a form of distributed consensus in each time step.

Finally, inspired by security attacks on cyber-physical systems (CPS), we introduce *controller-attacker games*, where two players, a controller and an attacker, have antagonistic objectives. To highlight the power of adaptation, we formulate a special case of controller-attacker games called V-formation games, where the attacker's goal is to prevent the controller from attaining V-formation. We demonstrate how adaptation in the design of the controller helps in overcoming certain attacks.

1. Introduction

Cyber-physical systems (CPSs) comprised of multiple computing agents are often highly distributed and may exhibit emergent behavior. V-formation in a flock of birds is a quintessential example of emergent behavior in a (stochastic) multi-agent system. V-formation brings numerous benefits to the flock. It is primarily known for being energy-efficient due to the *upwash benefit* a bird in the flock enjoys from its frontal neighbor. It also offers a *clear view* benefit, as no bird's field of vision is obstructed by another bird in the formation. Moreover, its collective spatial flock mass can be intimidating to potential predators. It is therefore not surprising that interest in V-formation is on the rise [Con17,Blo]. Because of V-formation's intrinsic appeal, it is important to (i) understand its control-theoretic foundations, (ii) devise efficient algorithms for the problem, and (iii) identify the vulnerabilities in these approaches to cyber-attacks.

This paper brings together our recent results on V-formation that show how the problem can be formulated in terms of Model Predictive Control (MPC), both centralized and distributed. It also shows how an MPC-based formulation of V-formation can be used as a comprehensive framework for investigating cyber-attacks on this formation.

We first consider *Adaptive Receding-Horizon Synthesis of Optimal Plans* (ARES) [LEH+17], an efficient approximation algorithm for generating optimal plans (action sequences) that take an initial state of an MDP to a state whose cost is below a specified (convergence) threshold. ARES uses Particle Swarm Optimization (PSO), with *adaptive sizing* for both the receding horizon and the particle swarm. Inspired by Importance Splitting, a sampling technique for rare events, the length of the horizon and the number of particles are chosen such that at least one particle reaches a *next-level* state, that is, a state where the cost decreases by a required delta from the previous-level state. The level relation on states and the plans constructed by ARES implicitly define a Lyapunov function and an optimal policy, respectively, both of which could be explicitly generated by applying ARES to all states of the MDP, up to some topological equivalence relation.

We assess the effectiveness of ARES by statistically evaluating its rate of success in generating optimal plans that bring a flock from an arbitrary initial state to a state exhibiting a single connected V-formation. For flocks with 7 birds, ARES is able to generate a plan that leads to a V-formation in 95% of the 8,000 random initial configurations within 63 seconds, on average. ARES can be viewed as a model-predictive controller (MPC) with an adaptive receding horizon, which we also call adaptive MPC (AMPC). We provide statistical guarantees of convergence. To the best of our knowledge, our adaptive-sizing approach is the first to provide *convergence guarantees* in receding-horizon techniques.

We next present DAMPC [LTSG19], a distributed, adaptive-horizon and adaptive-neighborhood algorithm for solving the stochastic reachability problem in multi-agent systems; specifically the flocking problem modeled as an MDP. In DAMPC, at each time step, every agent first calls a centralized, adaptive-horizon model-predictive control (AMPC) algorithm to obtain an optimal solution for its local neighborhood. Second, the agents derive the flock-wide optimal solution through a sequence of consensus rounds. Third, the neighborhood is adaptively resized using a flock-wide cost-based Lyapunov function. In this way DAMPC improves efficiency without compromising convergence. The proof of statistical global convergence is non-trivial and involves showing that V follows a monotonically decreasing trajectory despite potential fluctuations in cost and neighborhood size.

We evaluate DAMPC's performance using statistical model checking, showing that DAMPC achieves considerable speed-up over AMPC (two-fold in some cases) with only a slightly lower convergence rate. Smaller average neighborhood size and lookahead horizon demonstrate the benefits of the DAMPC approach for stochastic reachability problems involving any controllable multi-agent system that possesses a cost function.

Inspired by the emerging problem of CPS security, we lastly introduce the concept of *controller-attacker games* [TSE+17]: a two-player stochastic game involving a controller and an attacker, which have antagonistic objectives. A controller-attacker game is formulated in terms of an MDP, with the controller and the attacker jointly determining the MDP's transition probabilities. We also introduce *V-formation games*, a class of controller-attacker games where the goal of the controller is to maneuver the plant (a simple model of flocking dynamics) into a V-formation, and the goal of the attacker is to prevent the controller from doing so. Controllers in V-formation games utilize AMPC, giving them extraordinary power: we prove that under certain controllability conditions, an AMPC controller can attain V-formation with probability 1.

We evaluate AMPC's performance on V-formation games using statistical model checking. Our results show that (a) as we increase the power of the attacker, the AMPC controller

adapts by suitably increasing its horizon, and thus demonstrates resiliency to a variety of attacks; and (b) an intelligent attacker can significantly outperform its naive counterpart.

The rest of the paper is organized as follows. Section 2 provides background content in the form of our dynamic model of V-formation, stochastic reachability, and PSO. Sections 3-5 present the ARES algorithm, the DAMPC algorithm, and controller-attacker games for V-formation, respectively. Section 7 offers our concluding remarks.

This paper was written on the occasion of Jos Baeten's retirement as general director of CWI and professor of theory of computing of ILLC. Jos was a highly influential collaborator of the third author (Smolka), and remains a good friend and colleague. Jos's feedback to Smolka on the invited talk he gave on V-formation at CONQUEST 2016 was an important impetus for moving the work forward.

2. Background

This section introduces the basic concepts and techniques needed to formulate and derive our results.

2.1. Dynamic Model for V-formation.

In our flocking model, each bird in the flock is modeled using four variables: a 2-dimensional vector $\boldsymbol{x}$ denoting the position of the bird in a 2D space, and a 2-dimensional vector $\boldsymbol{v}$ denoting the velocity of the bird. We use $s = \{\boldsymbol{x}_i, \boldsymbol{v}_i\}_{i=1}^{B}$ to denote a state of a flock with B birds. The *control actions* of each bird are 2-dimensional accelerations $\boldsymbol{a}$ and 2-dimensional position displacements $\boldsymbol{d}$ (see discussion of $\boldsymbol{a}$ and $\boldsymbol{d}$ below). Both are random variables.

Let $\boldsymbol{x}_i(t), \boldsymbol{v}_i(t), \boldsymbol{a}_i(t)$, and $\boldsymbol{d}_i(t)$ respectively denote the position, velocity, acceleration, and displacement of the i-th bird at time t, $1 \leqslant i \leqslant B$. The behavior of bird i in discrete time is modeled as follows:

$$\begin{aligned} \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t) + \boldsymbol{d}_i(t) \\ \boldsymbol{v}_i(t+1) &= \boldsymbol{v}_i(t) + \boldsymbol{a}_i(t) \end{aligned} \tag{2.1}$$

The next state of the flock is jointly determined by the accelerations and the displacements based on the current state following Eq. 2.1.

Every bird in our model [GPR+14] moves in 2-dimensional space performing acceleration actions determined by a global controller. When there is no external disturbance, the displacement term is zero and the equations are:

$$\begin{aligned} \boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t) \\ \boldsymbol{v}_i(t+1) &= \boldsymbol{v}_i(t) + \boldsymbol{a}_i(t) \end{aligned} \tag{2.2}$$

The controller detects the positions and velocities of all birds through sensors, and uses this information to compute an optimal acceleration for the entire flock. A bird uses its own component of the solution to update its velocity and position.

We extend this discrete-time dynamical model to a (deterministic) MDP by adding a cost (fitness) function[1] based on the following metrics inspired by [YGST16]:

[1] A classic MDP [RN10] is obtained by adding sensor/actuator or wind-gust noise, which are the case we are addressing in the follow-up work.

- *Clear View* (CV). A bird's visual field is a cone with angle θ that can be blocked by the wings of other birds. We define the clear-view metric by accumulating the percentage of a bird's visual field that is blocked by other birds. Fig. 1 (left) illustrates the calculation of the clear-view metric. Let $B_{ij}(\boldsymbol{x}_i, \boldsymbol{v}_i, \boldsymbol{x}_j)$ be the part of the angle subtended by the wing of Bird j on the eye of Bird i that intersects with Bird i's visual cone with angle θ. Then, the clear view for Bird i, $CV_i(\boldsymbol{x}, \boldsymbol{v})$, is defined as $|\cup_{j\neq i} B_{ij}(\boldsymbol{x}_i, \boldsymbol{v}_i, \boldsymbol{x}_j)|/\theta$, and the total clear view, $CV(\boldsymbol{x}, \boldsymbol{v})$, is defined as $\sum_i CV_i(\boldsymbol{x}, \boldsymbol{v})$. The optimal value in a V-formation is $CV^*=0$, as all birds have a clear view. Note that the value B_{ij} can be computed using Bird i's velocity and position, and Bird j's position using standard trigonometric functions.
- *Velocity Matching* (VM). The accumulated differences between the velocity of each bird and all other birds, summed up over all birds in the flock defines VM. Fig. 1 (middle) depicts the values of VM in a velocity-unmatched flock. Formally, $VM(\boldsymbol{x}, \boldsymbol{v}) = \sum_{i>j}(||\boldsymbol{v}_i - \boldsymbol{v}_j||/(||\boldsymbol{v}_i|| + ||\boldsymbol{v}_j||))^2$. The optimal value in a V-formation is $VM^*=0$, as all birds will have the same velocity (thus maintaining the V-formation).
- *Upwash Benefit* (UB). The trailing upwash is generated near the wingtips of a bird, while downwash is generated near the center of a bird. We accumulate all birds' upwash benefits using a Gaussian-like model of the upwash and downwash region, as shown in Fig. 1 (right) for the right wing. Let h_{ij} be the projection of the vector $\boldsymbol{x}_j - \boldsymbol{x}_i$ along the wing-span of Bird i. Similarly, let g_{ij} be the projection of $\boldsymbol{x}_j - \boldsymbol{x}_i$ along the direction of $\boldsymbol{v}_i$. Specifically, the upwash benefit UB_{ij} for Bird i coming from Bird j is given by

$$UB_{ij} = \begin{cases} \alpha S(|h_{ij}|)G(h_{ij}, g_{ij}, \mu_1, \Sigma_1) & \text{if } |h_{ij}| \geq \frac{(4-\pi)w}{8} \wedge g_{ij} > 0 \\ S(|h_{ij}|)G(h_{ij}, g_{ij}, \mu_1, \Sigma_1) & \text{if } |h_{ij}| < \frac{(4-\pi)w}{8} \wedge g_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

  where $S(z) = \texttt{erf}(2\sqrt{2}(z - \frac{(4-\pi)w}{8}))$ is the error function, which is a smooth approximation of the sign function, $G_1(\vec{z}, \Sigma) = e^{(-\frac{1}{2}(\vec{z}^T\Sigma^{-1}\vec{z}))}$ is a 2D-Gaussian with mean at the origin, and $G(y, z, \mu, \Sigma) = G_1([|y|, |z|] - \mu, \Sigma)$ is a 2D-Gaussian shifted so that the mean is μ. The parameter w is the wing span, and $\mu_1 = [(12 + \pi)w/16, 1]$ is the relative position where upwash benefit is maximized. The total upwash benefit, UB_i, for Bird i is $\sum_{j\neq 1} UB_{ij}$. The maximum upwash a bird can obtain is upper-bounded by 1. Since we are working with cost (that we want to minimize), we define $UB(\boldsymbol{x}, \boldsymbol{v}) = \sum_i(1 - \min(UB_i, 1))$. The optimal value for UB in a V-formation is $UB^*=1$, as the leader does not receive any upwash.

Finding smooth and continuous formulations of the fitness metrics is a key element of solving optimization problems. The PSO algorithm has a very low probability of finding an optimal solution if the fitness metric is not well-designed.

Let $\boldsymbol{c}(t) = \{\boldsymbol{c}_i(t)\}_{i=1}^b = \{\boldsymbol{x}_i(t), \boldsymbol{v}_i(t)\}_{i=1}^b \in \mathbb{R}$ be a flock configuration at time-step t. Given the above metrics, the overall fitness (cost) metric J is of a sum-of-squares combination of VM, CV, and UB defined as follows:

$$J(\boldsymbol{c}(t), \boldsymbol{a}^h(t), h) = (CV(\boldsymbol{c}^h_{\boldsymbol{a}}(t)) - CV^*)^2 + (VM(\boldsymbol{c}^h_{\boldsymbol{a}}(t)) - VM^*)^2 + (UB(\boldsymbol{c}^h_{\boldsymbol{a}}(t)) - UB^*)^2, \tag{2.3}$$

where h is the receding prediction horizon (RPH), $\boldsymbol{a}^h(t) \in \mathbb{R}$ is a sequence of accelerations of length h, and $\boldsymbol{c}^h_{\boldsymbol{a}}(t)$ is the configuration reached after applying $\boldsymbol{a}^h(t)$ to $\boldsymbol{c}(t)$. Formally, we

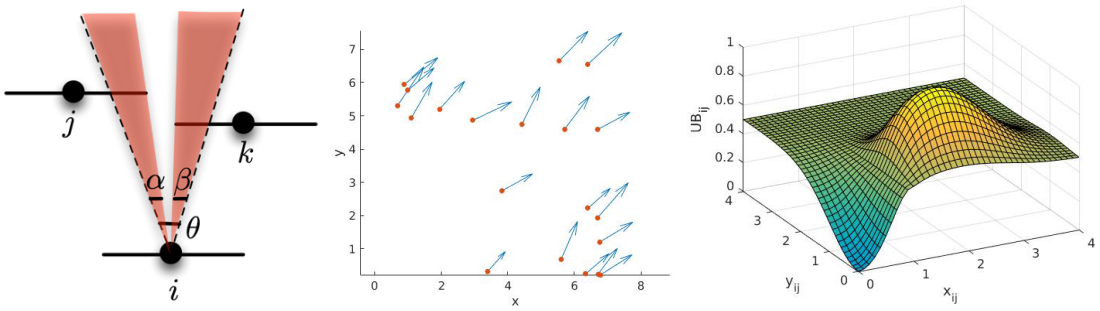

Figure 1: Illustration of the clear view (CV), velocity matching (VM), and upwash benefit (UB) metrics. Left: Bird i's view is partially blocked by birds j and k. Hence, its clear view is $CV = (\alpha + \beta)/\theta$. Middle: A flock and its unaligned bird velocities results in a velocity-matching metric $VM = 6.2805$. In contrast, $VM = 0$ when the velocities of all birds are aligned. Right: Illustration of the (right-wing) upwash benefit bird i receives from bird j depending on how it is positioned behind bird j. Note that bird j's downwash region is directly behind it.

have

$$\boldsymbol{c}_{\boldsymbol{a}}^h(t) = \{\boldsymbol{x}_{\boldsymbol{a}}^h(t), \boldsymbol{v}_{\boldsymbol{a}}^h(t)\} = \{\boldsymbol{x}(t) + \sum_{\tau=1}^{h(t)} \boldsymbol{v}(t+\tau), \boldsymbol{v}(t) + \sum_{\tau=1}^{h(t)} \boldsymbol{a}^\tau(t)\}, \tag{2.4}$$

where $\boldsymbol{a}^\tau(t)$ is the τth acceleration of $\boldsymbol{a}^h(t)$. As discussed further in Section 3, we allow RPH $h(t)$ to be *adaptive* in nature.

The fitness function J has an optimal value of 0 in a perfect V-formation. Thus, there is a need to perform flock-wide minimization of J at each time-step t to obtain an optimal plan of length h of acceleration actions:

$$\textbf{opt-}\boldsymbol{a}^h(t) = \{\textbf{opt-}\boldsymbol{a}_i^h(t)\}_{i=1}^b = \underset{\boldsymbol{a}^h(t)}{\arg\min}\, J(\boldsymbol{c}(t), \boldsymbol{a}^h(t), h). \tag{2.5}$$

The optimization is subject to the following constraints on the maximum velocities and accelerations: $||\boldsymbol{v}_i(t)|| \leqslant \boldsymbol{v}_{max}, ||\boldsymbol{a}_i^h(t)|| \leqslant \rho||\boldsymbol{v}_i(t)|| \;\forall\; i \in \{1, \ldots, b\}$, where $\boldsymbol{v}_{max}$ is a constant and $\rho \in (0, 1)$. The above constraints prevent us from using mixed-integer programming, we might, however, compare our solution to other continuous optimization techniques in the future. The initial positions and velocities of each bird are selected at random within certain ranges, and limited such that the distance between any two birds is greater than a (collision) constant d_{min}, and small enough for all birds, except for at most one, to feel the UB.

2.2. **V-Formation MDP.** This section defines Markov Decision Processes (MDPs) and the corresponding MDP formulated by Lukina et al. [LEH+17] for the V-formation problem.

Definition 1 . *A **Markov decision process** (MDP) is a 5-tuple $\mathcal{M} = (S, A, T, J, I)$ consisting of a set of states S, a set of actions A, a transition function $T : S \times A \times S \mapsto [0, 1]$, where $T(\boldsymbol{s}, a, \boldsymbol{s}')$ is the probability of transitioning from state $\boldsymbol{s}$ to state $\boldsymbol{s}'$ under action $\boldsymbol{a}$, a cost function $J : S \mapsto \mathbb{R}$, where $J(\boldsymbol{s})$ is the cost associated with state $\boldsymbol{s}$, and an initial state distribution $I : S \mapsto [0, 1]$.*

The *MDP $\mathcal{M}$ modeling a flock* of B birds is defined as follows. The set of states S is $S = \mathbb{R}^{4B}$, as each bird has a 2D position and a 2D velocity vector, and the flock contains

B birds. The set of actions A is $A = \mathbb{R}^{2B}$, as each bird takes a 2D acceleration action and there are B birds. The cost function J is defined by Eq. 2.3. The transition function T is defined by Eq. 2.1. As the acceleration vector $\boldsymbol{a}_i(t)$ for bird i at time t is a random variable, the state vector $\boldsymbol{s}_i = \{\boldsymbol{x}_i(t+1),\ \boldsymbol{v}_i(t+1)\}$ is also a random variable. The initial state distribution I is a uniform distribution from a region of state space where all birds have positions and velocities in a range defined by fixed lower and upper bounds.

2.3. **Stochastic Reachability Problem.** Given the stochasticity introduced by PSO, the V-formation problem can be formulated in terms of a reachability problem for the Markov chain induced by the composition of a Markov decision process (MDP) and a controller.

Before we can define traces, or executions, of $\mathcal{M}$, we need to fix a controller, or strategy, that determines which action from A to use at any given state of the system. We focus on randomized strategies. A *randomized strategy (controller)* σ over $\mathcal{M}$ is a function of the form $\sigma : S \mapsto PD(A)$, where $PD(A)$ is the set of probability distributions over A. That is, σ takes a state $\boldsymbol{s}$ and returns an action consistent with the probability distribution $\sigma(\boldsymbol{s})$. Applying a policy σ to the MDP $\mathcal{M}$ defines the Markov chain. $\mathcal{M}_\sigma$. We use the terms strategy and controller interchangeably.

In the bird-flocking problem, a controller would be a function that determines the accelerations for all the birds given their current positions and velocities. Once we fix a controller, we can iteratively use it to (probabilistically) select a sequence of flock accelerations. The goal is to generate a sequence of actions that takes an MDP from an initial state $\boldsymbol{s}$ to a state $\boldsymbol{s}^*$ with $J(\boldsymbol{s}^*) \leqslant \varphi$.

Definition 2 . *Let $\mathcal{M} = (S, A, T, J, I)$ be an MDP, and let $G \subseteq S$ be the set of goal states $G = \{\boldsymbol{s} | J(\boldsymbol{s}) \leqslant \varphi\}$ of $\mathcal{M}$. The **stochastic reachability problem** is to design a controller $\sigma : S \mapsto PD(A)$ for $\mathcal{M}$ such that for a given δ, the probability of the underlying Markov chain $\mathcal{M}_\sigma$ to reach a state in G in m steps (for a given m) starting from an initial state, is at least $1 - \delta$.*

We approach the stochastic reachability problem by designing a controller and quantifying its probability of success in reaching the goal states.

2.4. **Particle Swarm Optimization.** Particle Swarm Optimization (PSO) is a randomized approximation algorithm for computing the value of a parameter minimizing a possibly nonlinear cost (fitness) function. Interestingly, PSO itself is inspired by bird flocking [KE95]. Hence, PSO assumes that it works with a flock of birds.

Note, however, that in our running example, these birds are "acceleration birds" (or particles), and not the actual birds in the flock. Each bird has the same goal, finding food (reward), but none of them knows the location of the food. However, every bird knows the distance (horizon) to the food location. PSO works by moving each bird preferentially toward the bird closest to food.

The work delineated in this paper uses Matlab-Toolbox `particleswarm`, which performs the classical version of PSO. This PSO creates a swarm of particles, of size say p, uniformly at random within a given bound on their positions and velocities. Note that in our example, each particle represents itself a flock of bird-acceleration sequences $\{\boldsymbol{a}_i^h\}_{i=1}^b$, where h is the current length of the receding horizon. PSO further chooses a neighborhood of a random size for each particle j, $j = \{1, \ldots, p\}$, and computes the fitness of each particle. Based on

the fitness values, PSO stores two vectors for j: its so-far personal-best position $\mathbf{x}_P^j(t)$, and its fittest neighbor's position $\mathbf{x}_G^j(t)$. The positions and velocities of each particle j in the particle swarm $1 \leqslant j \leqslant p$ are updated according to the following rule:

$$\begin{aligned}\mathbf{v}^j(t+1) = \omega \cdot \mathbf{v}^j(t) + y_1 \cdot \mathbf{u_1}(t+1) \otimes (\mathbf{x}_P^j(t) - \mathbf{x}^j(t)) \\ + y_2 \cdot \mathbf{u_2}(t+1) \otimes (\mathbf{x}_G^j(t) - \mathbf{x}^j(t))\end{aligned} \tag{2.6}$$

where ω is *inertia weight*, which determines the trade-off between global and local exploration of the swarm (the value of ω is proportional to the exploration range); y_1 and y_2 are *self adjustment* and *social adjustment*, respectively; $\mathbf{u_1}, \mathbf{u_2} \in \text{Uniform}(0,1)$ are randomization factors; and $\otimes$ is the vector dot product, that is, $\forall$ random vector $\mathbf{z}$: $(\mathbf{z}_1, \ldots, \mathbf{z}_b) \otimes (\mathbf{x}_1^j, \ldots, \mathbf{x}_b^j) = (\mathbf{z}_1\mathbf{x}_1^j, \ldots, \mathbf{z}_b\mathbf{x}_b^j)$.

If the fitness value for $\mathbf{x}^j(t+1) = \mathbf{x}^j(t) + \mathbf{v}^j(t+1)$ is lower than the one for $\mathbf{x}_P^j(t)$, then $\mathbf{x}^j(t+1)$ is assigned to $\mathbf{x}_P^j(t+1)$. The particle with the best fitness over the whole swarm becomes a global best for the next iteration. The procedure is repeated until the number of iterations reaches its maximum, the time elapses, or the minimum criteria is satisfied. For our bird-flock example we obtain in this way the best acceleration.

3. Adaptive Receding-Horizon Synthesis of Optimal Plans (ARES)

ARES [LEH+17] is a general *adaptive, receding-horizon synthesis algorithm* (ARES) that, given an MDP and one of its initial states, generates an optimal plan (action sequence) taking that state to a state whose cost is below a desired threshold. ARES implicitly defines an *optimal, online policy-synthesis algorithm*, assuming plan generation can be performed in real-time. ARES can alternatively be viewed as a model-predictive control (MPC) algorithm that utilizes an adaptive receding horizon, a technique we refer to as Adaptive MPC (AMPC).

ARES makes repeated use of PSO [KE95] to effectively generate a plan. This was in principle unnecessary, as one could generate an optimal plan by calling PSO only once, with a maximum plan-length horizon. Such an approach, however, is in most cases impractical, as every unfolding of the MDP adds a number of new dimensions to the search space. Consequently, to obtain adequate coverage of this space, one needs a very large number of particles, a number that is either going to exhaust available memory or require a prohibitive amount of time to find an optimal plan.

3.1. The ARES Algorithm.

One could in principle solve the optimization problem defined in Sections 2.1 and 2.2 by calling PSO only once, with a horizon h in $\mathcal{M}$ equaling the maximum length m allowed for a plan. This approach, however, tends to lead to very large search spaces, and is in most cases intractable. Indeed, preliminary experiments with this technique applied to our running example could not generate any convergent plan.

A more tractable approach is to make repeated calls to PSO with a small horizon length h. The question is how small h can be. *The current practice in model-predictive control (MPC) is to use a fixed h*, $1 \leqslant h \leqslant 3$ (see the outer loop of Fig. 3, where resampling and conditional branches are disregarded). Unfortunately, this forces the selection of *locally-optimal plans* (of size less than three) in each call, and there is *no guarantee of convergence* when joining them together. In fact, in our running example, we were able to find plans leading to a V-formation in only 45% of the time for $10,000$ random initial flocks.

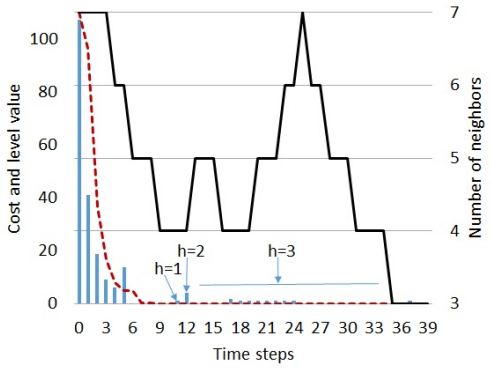

Figure 2: Blue bars are the values of the cost function in every time step. Red dashed line is the cost-based Lyapunov function used for horizon and neighborhood adaptation. Black solid line is neighborhood resizing for the next step given the current cost.

Inspired by Importance Splitting (see Fig. 4 (right) and Fig. 3), we introduce the notion of a *level-based horizon*, where level ℓ_0 equals the cost of the initial state, and level ℓ_m equals the threshold φ. Intuitively, by using an asymptotic cost-convergence function ranging from ℓ_0 to ℓ_m, and dividing its graph in m equal segments, we can determine on the vertical axis a sequence of levels ensuring convergence.

The asymptotic function ARES implements is essentially $\ell_i = \ell_0\,(m-i)/\,m$, but specifically tuned for each particle. Formally, if particle k has previously reached level equaling $J_k(s_{i-1})$, then its next target level is within the distance $\Delta_k = J_k(s_{i-1})/(m-i+1)$. In Fig. 3, after passing the thresholds assigned to them, values of the cost function in the current state s_i are sorted in ascending order $\{\widehat{J}_k\}_{k=1}^{n}$. The lowest cost $\widehat{J}_1$ should be apart from the previous level ℓ_{i-1} at least on its Δ_1 for the algorithm to proceed to the next level $\ell_i := \widehat{J}_1$.

The levels serve two purposes. First, they implicitly define a Lyapunov function, which guarantees convergence. If desired, this function can be explicitly generated for all states, up to some topological equivalence. Second, the levels ℓ_i help PSO overcome local minima (see Fig. 4 (left)). If reaching a next level requires PSO to temporarily pass over a state-cost ridge, then ARES incrementally increases the size of the horizon h, up to a maximum size

Algorithm 1: Simulate $(\mathcal{M}, h, i, \{\Delta_k, J_k(s_{i-1})\}_{k=1}^{n})$

foreach $\mathcal{M}_k \in \mathcal{M}$ **do**
  $[\boldsymbol{a}_k^h, \mathcal{M}_k^h] \leftarrow$ `particleswarm`$(\mathcal{M}_k, p, h)$; *// use PSO in order to determine best next action for the MDP $\mathcal{M}_k$ with RPH h*
  $J_k(s_i) \leftarrow$ `Cost`$(\mathcal{M}_k^h, \boldsymbol{a}_k^h, h)$; *// calculate cost function if applying the sequence of optimal actions of length h*
  if $J_k(s_{i-1}) - J_k(s_i) > \Delta_k$ **then**
    $\Delta_k \leftarrow J_k(s_i)/(m-i)$; *// new level-threshold*
  end
end

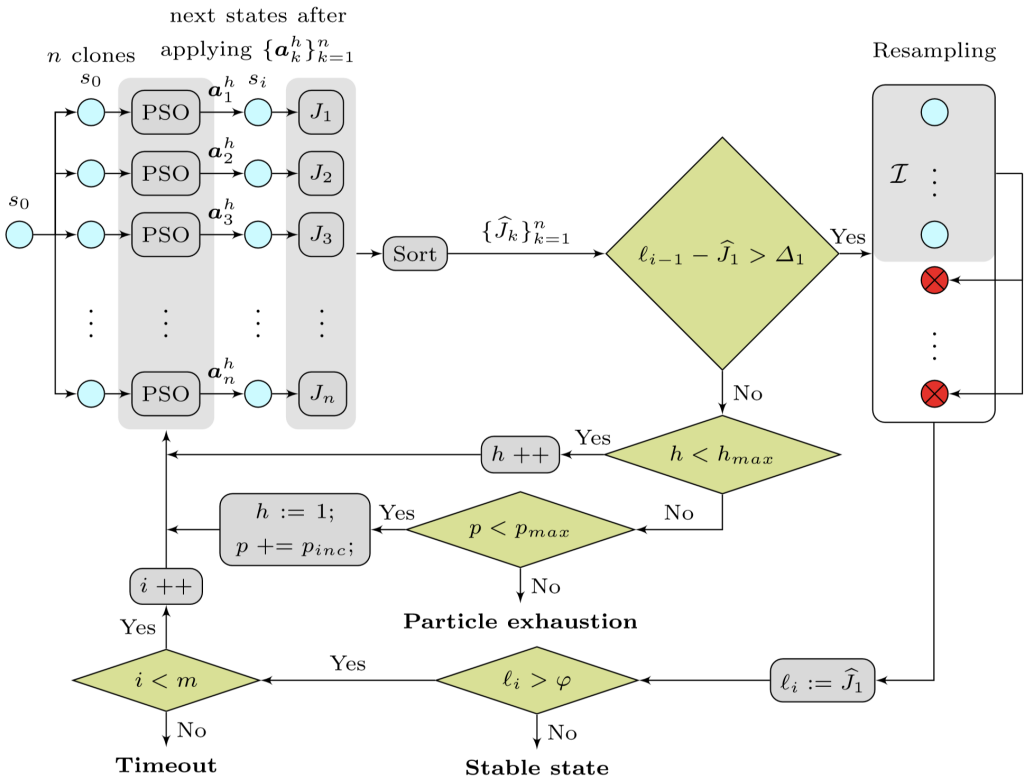

Figure 3: Graphical representation of ARES.

Algorithm 2: Resample $(\{\mathcal{M}_k^h, J_k(s_i)\}_{k=1}^n)$

$\mathcal{I} \leftarrow$ Sort ascending $\mathcal{M}_k^h$ by their current costs; *// find indexes of MDPs whose costs are below the median among all the clones*
for $k = 1$ **to** n **do**
  if $k \notin \mathcal{I}$ **then**
    Sample r uniformly at random from $\mathcal{I}$; $\mathcal{M}_k \leftarrow \mathcal{M}_r^h$;
  else
    $\mathcal{M}_k \leftarrow \mathcal{M}_k^h$; *// Keep more successful MDPs unchanged*
  end
end

h_{max}. For particle k, passing the thresholds Δ_k means that it reaches a new level, and the definition of Δ_k ensures a smooth degradation of its threshold.

Another idea imported from IS and shown in Fig. 3, is to maintain n clones $\{\mathcal{M}_k\}_{k=1}^n$ of the MDP $\mathcal{M}$ (and its initial state) at any time t, and run PSO, for a horizon h, on each h-unfolding $\mathcal{M}_k^h$ of them. This results in an action sequence $\boldsymbol{a}_k^h$ of length h (see Algo. 1). This approach allows us to call PSO for each clone and desired horizon, with a very small number of particles p per clone.

To check which particles have overcome their associated thresholds, we sort the particles according to their current cost, and split them in two sets: the successful set, having the

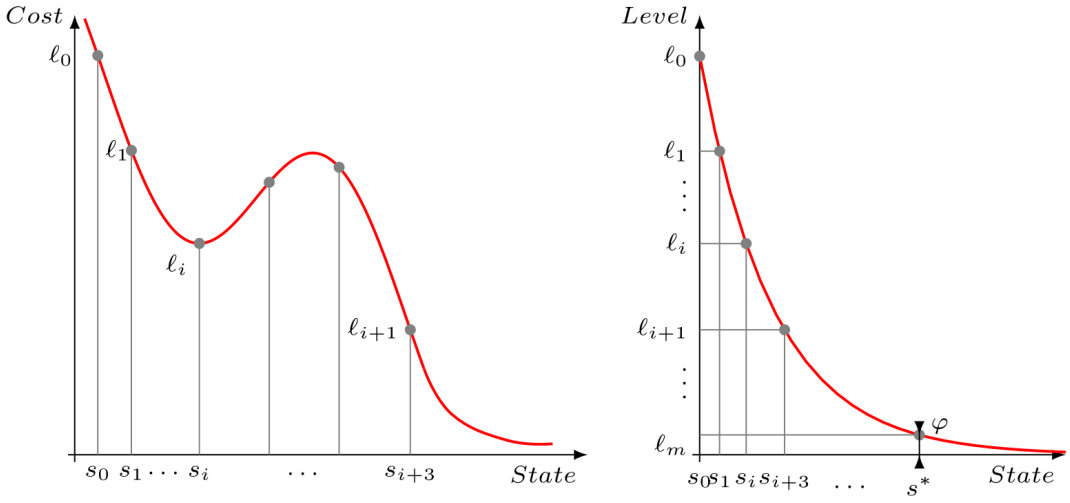

Figure 4: Left: If state s_0 has cost ℓ_0, and its successor-state s_1 has cost less than ℓ_1, then a horizon of length 1 is appropriate. However, if s_i has a local-minimum cost ℓ_i, one has to pass over the cost ridge in order to reach level ℓ_{i+1}, and therefore ARES has to adaptively increase the horizon to 3. Right: The cost of the initial state defines ℓ_0 and the given threshold φ defines ℓ_m. By choosing m equal segments on an asymptotically converging (Lyapunov) function (where the number m is empirically determined), one obtains on the vertical cost-axis the levels required for ARES to converge.

indexes $\mathcal{I}$ and whose costs are lower than the median among all clones; and the unsuccessful set with indexes in $\{1,\ldots,n\}\setminus\mathcal{I}$, which are discarded. The unsuccessful ones are further replenished, by sampling uniformly at random from the successful set $\mathcal{I}$ (see Algo. 2).

The number of particles is increased $p = p + p_{inc}$ if no clone reaches a next level, for all horizons chosen. Once this happens, we reset the horizon to one, and repeat the process. In this way, we adaptively focus our resources on escaping from local minima. From the last level, we choose the state s^* with the minimal cost, and traverse all of its predecessor states to find an optimal plan comprised of actions $\{\boldsymbol{a}^i\}_{1\leqslant i\leqslant m}$ that led MDP $\mathcal{M}$ to the optimal state s^*. In our running example, we select a flock in V-formation, and traverse all its predecessor flocks. The overall procedure of ARES is shown in Algo. 3.

Proposition 1 Optimality and Minimality**.** *(1) Let $\mathcal{M}$ be an MDP. For any initial state s_0 of $\mathcal{M}$, ARES is able to solve the optimal-plan synthesis problem for $\mathcal{M}$ and s_0. (2) An optimal choice of m in function Δ_k, for some particle k, ensures that ARES also generates the shortest optimal plan.*

Sketch. (1) The dynamic-threshold function Δ_k ensures that the initial cost in s_0 is continuously decreased until it falls below φ. Moreover, for an appropriate number of clones, by adaptively determining the horizon and the number of particles needed to overcome Δ_k, ARES always converges, with probability 1, to an optimal state, given enough time and memory. (2) This follows from convergence property (1), and from the fact that ARES always gives preference to the shortest horizon while trying to overcome Δ_k. □

Algorithm 3: ARES

Input : $\mathcal{M}, \varphi, p_{start}, p_{inc}, p_{max}, h_{max}, m, n$
Output : $\{\boldsymbol{a}^i\}_{1 \leqslant i \leqslant m}$ *// synthesized optimal plans*

Initialize $\ell_0 \leftarrow \inf$; $\{J_k(s_0)\}_{k=1}^n \leftarrow \inf$; $p \leftarrow p_{start}$; $i \leftarrow 1$; $h \leftarrow 1$; $\Delta_k \leftarrow 0$;
while $(\ell_i > \varphi) \vee (i < m)$ **do**
  // find and apply best actions with RPH h
  $[\{\boldsymbol{a}_k^h, J_k(s_i), \mathcal{M}_k^h\}_{k=1}^n] \leftarrow$`Simulate`$(\mathcal{M}, h, i, \{\Delta_k, J_k(s_{i-1})\}_{k=1}^n)$;
  $\widehat{J}_1 \leftarrow sort(J_1(s_i), \ldots, J_n(s_i))$; *// find minimum cost among all the clones*
  if $\ell_{i-1} - \widehat{J}_1 > \Delta_1$ **then**
    $\ell_i \leftarrow \widehat{J}_1$; *// new level has been reached*
    $i \leftarrow i + 1$; $h \leftarrow 1$; $p \leftarrow p_{start}$; *// reset adaptive parameters*
    $\{\mathcal{M}_k\}_{k=1}^n \leftarrow$ `Resample`$(\{\mathcal{M}_k^h, J_k(s_i)\}_{k=1}^n)$;
  else
    if $h < h_{max}$ **then**
      $h \leftarrow h + 1$; *// improve time exploration*
    else
      if $p < p_{max}$ **then**
        $h \leftarrow 1$; $p \leftarrow p + p_{inc}$; *// improve space exploration*
      else
        break;
      end
    end
  end
end
Take a clone in the state with minimum cost $\ell_i = J(s_i^*) \leqslant \varphi$ at the last level i;
foreach i **do**
  $\{s_{i-1}^*, \boldsymbol{a}^i\} \leftarrow Pre(s_i^*)$; *// find predecessor and corresponding action*
end

The optimality referred to in the title of the paper is in the sense of (1). One, however, can do even better than (1), in the sense of (2), by empirically determining parameter m in the dynamic-threshold function Δ_k. Also note that ARES is an *approximation algorithm*, and may therefore return non-minimal plans. Even in these circumstances, however, the plans will still lead to an optimal state. This is a V-formation in our flocking example.

3.2. **Evaluation of ARES.** To assess the performance of our approach, we developed a simple simulation environment in Matlab. All experiments were run on an Intel Core i7-5820K CPU with 3.30 GHz and with 32GB RAM available.

We performed numerous experiments with a varying number of birds. Unless stated otherwise, results refer to 8,000 experiments with 7 birds with the following parameters: $p_{start} = 10$, $p_{inc} = 5$, $p_{max} = 40$, $\ell_{max} = 20$, $h_{max} = 5$, $\varphi = 10^{-3}$, and $n = 20$. The initial configurations were generated independently uniformly at random subject to the following constraints:

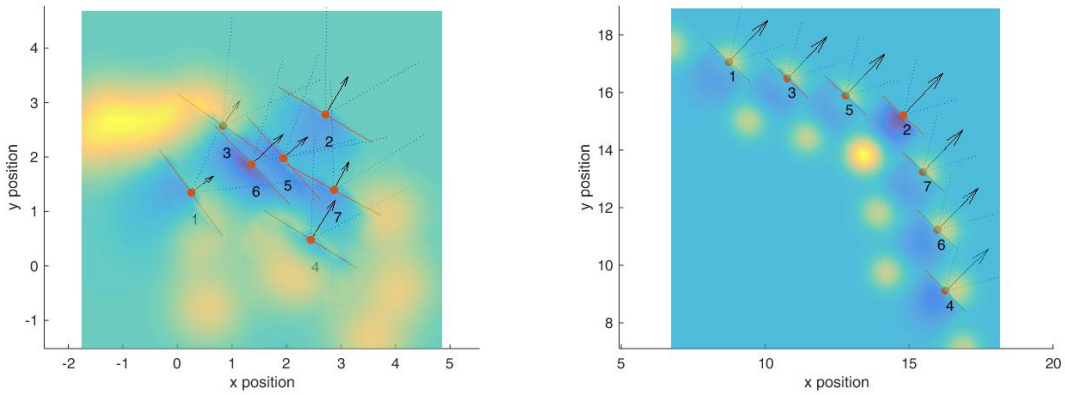

Figure 5: Left: Example of an arbitrary initial configuration of 7 birds. Right: The V-formation obtained by applying the plan generated by ARES. In the figures, we show the wings of the birds, bird orientations, bird speeds (as scaled arrows), upwash regions in yellow, and downwash regions in dark blue.

Table 1: Overview of the results for 8,000 experiments with 7 birds

No. Experiments	Successful				Total			
	7573				8000			
	Min	Max	Avg	Std	Min	Max	Avg	Std
Cost, J	$2.88{\cdot}10^{-7}$	$9{\cdot}10^{-4}$	$4{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$2.88{\cdot}10^{-7}$	1.4840	0.0282	0.1607
Time, t	23.14s	310.83s	63.55s	22.81s	23.14s	661.46s	64.85s	28.05s
Plan Length, i	7	20	12.80	2.39	7	20	13.13	2.71
RPH, h	1	5	1.40	0.15	1	5	1.27	0.17

Table 2: Average duration for 100 experiments with various number of birds

No. of birds	3	5	7	9
Avg. duration	4.58s	18.92s	64.85s	269.33s

(1) Position constraints: $\forall\, i \in \{1, \ldots, 7\}.\ \boldsymbol{x}_i(0) \in [0, 3] \times [0, 3]$.

(2) Velocity constraints: $\forall\, i \in \{1, \ldots, 7\}.\ \boldsymbol{v}_i(0) \in [0.25, 0.75] \times [0.25, 0.75]$.

Table 1 gives an overview of the results with respect to the 8,000 experiments we performed with 7 birds for a maximum of 20 levels. The average fitness across all experiments is 0.0282, with a standard deviation of 0.1654. We achieved a success rate of 94.66% with fitness threshold $\varphi = 10^{-3}$. The average fitness is higher than the threshold due to comparably high fitness of unsuccessful experiments. When increasing the bound for the maximal plan length m to 30 we achieved a 98.4% success rate in 1,000 experiments at the expense of a slightly longer average execution time.

The left plot in Fig. 6 depicts the resulting distribution of execution times for 8,000 runs of our algorithm, where it is clear that, excluding only a few outliers from the histogram, an arbitrary configuration of birds (Fig. 5 (left)) reaches V-formation (Fig. 5 (right)) in around 1 minute. The execution time rises with the number of birds as shown in Table 2.

In Fig. 6, we illustrate for how many experiments the algorithm had to increase RPH h (Fig. 6 (middle)) and the number of particles used by PSO p (Fig. 6 (right)) to improve time and space exploration, respectively.

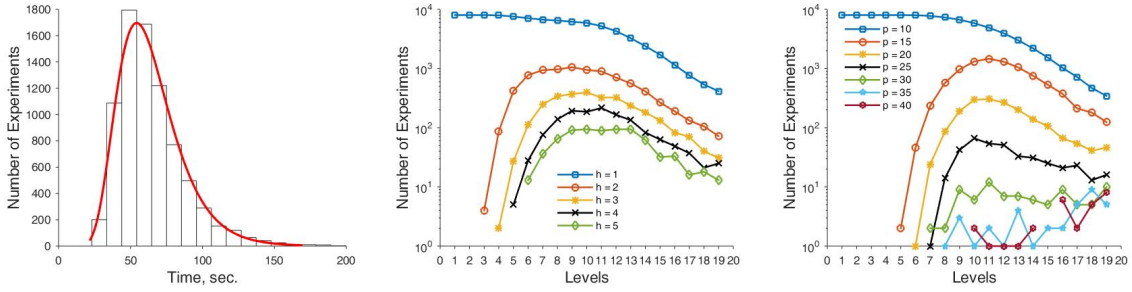

Figure 6: Left: Distribution of execution times for 8,000runs. Middle: Statistics of increasing RPH h. Right: Particles of PSO p for 8,000 experiments

After achieving such a high success rate of ARES for an arbitrary initial configuration, we would like to demonstrate that the number of experiments performed is sufficient for high confidence in our results. This requires us to determine the appropriate number N of random variables $Z_1, ...Z_N$ necessary for the Monte-Carlo approximation scheme we apply to assess efficiency of our approach. For this purpose, we use the additive approximation algorithm as discussed in [GPR+14]. If the sample mean $\mu_Z = (Z_1 + \ldots + Z_N)/N$ is expected to be large, then one can exploit the Bernstein's inequality and fix N to $\Upsilon \propto ln(1/\delta)/\varepsilon^2$. This results in an *additive* or *absolute-error* (ε, δ)*-approximation scheme*:

$$\mathbf{P}[\mu_Z - \varepsilon \leq \widetilde{\mu}_Z \leq \mu_Z + \varepsilon)] \geq 1 - \delta,$$

where $\widetilde{\mu}_Z$ approximates μ_Z with absolute error ε and probability $1 - \delta$.

In particular, we are interested in Z being a Bernoulli random variable:

$$Z = \begin{cases} 1, & \text{if } J(\boldsymbol{c}(t), \boldsymbol{a}(t), h(t)) \leqslant \varphi, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, we can use the Chernoff-Hoeffding instantiation of the Bernstein's inequality, and further fix the proportionality constant to $\Upsilon = 4\, ln(2/\delta)/\varepsilon^2$, as in [HLMP04a].

Hence, for our performed 8,000 experiments, we achieve a success rate of 95% with absolute error of $\varepsilon = 0.05$ and confidence ratio 0.99.

Moreover, considering that the average length of a plan is 13, and that each state in a plan is independent from all other plans, we can roughly consider that our above estimation generated 80,000 independent states. For the same confidence ratio of 0.99 we then obtain an approximation error $\varepsilon = 0.016$, and for a confidence ratio of 0.999, we obtain an approximation error $\varepsilon = 0.019$.

4. Adaptive-Neighborhood Distributed Control

In Section 3, we introduced the concept of Adaptive-Horizon MPC (`AMPC`). `AMPC` gives controllers extraordinary power: we proved that under certain controllability conditions, an `AMPC` controller can attain V-formation with probability 1. We now present `DAMPC` [LTSG19],

a distributed version of AMPC that extends AMPC along several dimensions. First, at every time step, DAMPC runs a *distributed consensus algorithm* to determine the optimal action (acceleration) for every agent in the flock. In particular, each agent i starts by computing the optimal actions for its local subflock. The subflocks then communicate in a sequence of consensus rounds to determine the optimal actions for the entire flock.

Secondly, DAMPC features *adaptive neighborhood resizing* in an effort to further improve the algorithm's efficiency. Like with an adaptive prediction horizon in AMPC, neighborhood resizing utilizes the implicit Lyapunov function to guarantee eventual convergence to a minimum neighborhood size. DAMPC thus treats the neighborhood size as another controllable variable that can be dynamically adjusted for efficiency purposes. This leads to reduced communication and computation compared to the centralized solution, without sacrificing statistical guarantees of convergence such as those offered by its centralized counterpart AMPC. Statistical global convergence can be proven.

4.1. **DAMPC System Model.** We consider a distributed setting with the following assumptions about the system model.

(1) Birds can communicate with each other without delays. As explained below, each bird i adaptively changes its communication radius. The measure of the radius is the number of birds covered, and we refer to it as bird i's local neighborhood N_i, including bird i itself.
(2) All birds use the same algorithm to satisfy their local reachability goals, i.e., to bring the local cost $J(\boldsymbol{s}_{N_i})$, $i \in \{1, \ldots, B\}$, below the given threshold φ.
(3) Birds move in continuous space and change accelerations synchronously at discrete time points.
(4) After executing its local algorithms, each bird broadcasts the obtained solution to its neighbors. In this manner, every bird receives solution proposals, which differ due to the fact that each bird has its own local neighborhood. To achieve consensus, each bird takes as its best action the one with the minimal cost among the received proposals. The solutions for the birds in the considered neighborhood are then fixed. The consensus rounds repeat until all birds have fixed solutions.
(5) At every time step, the value of the global cost function $J(\boldsymbol{s})$ is received by all birds in the flock and checked for improvement. The neighborhood for each bird is then resized based on this global check.
(6) The upwash benefit UB_i for bird i defined in Section 2.1 maintains connectivity of the flock along the computations, while our algorithm manages collision avoidance.

4.2. **The Distributed AMPC Algorithm.** In this section, we solve a stochastic reachability problem in the context of V-formation control, and demonstrate that the algorithm can be used as an alternative hill-climbing, cost-based optimization technique avoiding local minima.

DAMPC (see Alg. 4) takes as input an MDP $\mathcal{M}$, a threshold φ defining the goal states G, the maximum horizon length h_{max}, the maximum number of time steps m, the number of birds B, and a scaling factor β. It outputs a state $\boldsymbol{s}_0$ in I and a sequence of actions $\boldsymbol{a}^{1:m}$ taking $\mathcal{M}$ from $\boldsymbol{s}_0$ to a state in G. The initialization step (Line 1) chooses an initial state $\boldsymbol{s}_0$ from I, fixes an initial level ℓ_0 as the cost of $\boldsymbol{s}_0$, sets the initial time t and number of birds to process k. The outer while-loop (Lines 2-22) is active as long as $\mathcal{M}$ has not

Table 3: Table of Notation

H, h_i	$\triangleq$	Maximum and current local horizon lengths
N_i	$\triangleq$	neighborhood of the i's bird
k	$\triangleq$	the number of birds in the neighborhood ($\lvert N_i \rvert$)
m	$\triangleq$	number of time-steps allowed by the property φ
$\boldsymbol{a}^{1:m}$	$\triangleq$	sequence of synthesized acceleration for all birds for each time-step
?	$\triangleq$	acceleration that has not yet been fixed
1, !	$\triangleq$	superscript for the first and last, respectively, elements in the horizon sequence
$\boldsymbol{a}^{1:!}_{N_i}$, $\boldsymbol{s}^{1:!}_{N_i}$	$\triangleq$	sequence of accelerations and corresponding states of the horizon length reached at time-step t by bird i locally in its neighborhood N_i
Δ_i	$\triangleq$	dynamical threshold defined based on the last achieved local cost $J\left(\boldsymbol{s}^{!}_{N_j}\right)$ in the neighborhood N_j
$\boldsymbol{a}^{!}$, $\boldsymbol{s}^{!}$	$\triangleq$	accelerations and corresponding states for all birds achieved globally as unions of the last elements in the best horizon sequences reached locally in each neighborhood
$\boldsymbol{a}^{1}(t)$, $\boldsymbol{s}^{1}$	$\triangleq$	accelerations and states for all birds achieved globally as unions of the first elements in the best horizon sequences reached locally in each neighborhood
ℓ_t	$=$	$J\left(\boldsymbol{s}^{!}\right)$ – level achieved globally at time-step t after applying $\boldsymbol{a}^{1:!}$ to the current state
Δ	$\triangleq$	dynamical threshold defined based on the last achieved global level

reached G and time has not expired. In each time step, `DAMPC` first sets the sequences of accelerations $\boldsymbol{a}_i^{1:!}(t)$ for all i to ? (not yet fixed), and then iterates lines 4-15 until all birds fix their accelerations through global consensus (Line 10). This happens as follows. First, all birds determine their neighborhood (*subflock*) N_i and the cost decrement Δ_i that will bring them to the next level (Lines 6-7). Second, they call `LocalAMPC` (see Section 4.3), which takes sequences of states and actions fixed so far and extends them such that (line 8) the returned sequence of actions $\boldsymbol{a}^{1:!}_{N_i}$ and corresponding sequence of states $\boldsymbol{s}^{1:!}_{N_i}$ decrease the cost of the subflock by Δ_i. Here notation $1:!$ means the whole sequence including the last element ! (some number, the farthest point in the future where the state of the subflock is fixed), which can differ from one neighborhood to another depending on the length of used horizon. Note that an action sequence passed to `LocalAMPC` as input $\boldsymbol{a}^{1:!}_{N_i}$ contains ? and the goal is to fill in the gaps in solution sequence by means of this iterative process. In Line 10, we use the value of the cost function in the last resulting state $J\left(\boldsymbol{s}^{!}_{N_j}\right)$ as a criterion for choosing the best action sequence proposed among neighbors $j \in R$. Then the acceleration sequences of all birds in this subflock are fixed (Lines 12-14).

After all accelerations sequences are fixed, that is, all ? are eliminated, the first accelerations in this sequence are selected for the output (Line 17). The next state $\boldsymbol{s}^1$ is set to the union of $\boldsymbol{s}^1_{N_i}$ for all neighbors $i = 1:B$, the state of the flock after executing $\boldsymbol{a}(t)$ is set to the union of $\boldsymbol{s}^{!}_{N_i}$. If we found a path that eventually decreases the cost by Δ, we reached the next level, and advance time (Lines 18-20). In that case, we optionally decrease the neighborhood, and increase it otherwise (Line 21).

Algorithm 4: DAMPC

Input : $\mathcal{M} = (S, A, T, J, I), \varphi, h_{max}, m, B, \beta$
Output: $\boldsymbol{s}_0$, $\boldsymbol{a}^{1:m} = [\boldsymbol{a}(t)]_{1 \leqslant t \leqslant m}$

1 $\boldsymbol{s}_0 \leftarrow \text{sample}(I)$; $\boldsymbol{s} \leftarrow \boldsymbol{s}_0$; $\ell_0 \leftarrow J(\boldsymbol{s})$; $t \leftarrow 1$; $k \leftarrow B$; $H \leftarrow h_{max}$;
2 **while** $(\ell_{t-1} > \varphi) \wedge (t < m)$ **do**
3 $\quad \forall i : \boldsymbol{a}_i^{1:!}(t) \leftarrow ?$; `// No bird has a fixed solution yet`
4 $\quad$ **while** $(R \leftarrow \{j \mid \boldsymbol{a}_j(t) = ?\}) \neq \emptyset$ **do**
5 $\quad\quad$ **for** $i \in R$ **do in parallel**
6 $\quad\quad\quad N_i \leftarrow \texttt{Neighbors}(i, k)$; `// k neighbors of i`
7 $\quad\quad\quad \Delta_i \leftarrow J\left(\boldsymbol{s}^!_{N_i}\right)/(m-t)$;
8 $\quad\quad\quad \left(\boldsymbol{s}^{1:!}_{N_i}, \boldsymbol{a}^{1:!}_{N_i}\right) \leftarrow \texttt{LocalAMPC}\left(\mathcal{M}, \boldsymbol{s}^{1:!}_{N_i}, \boldsymbol{a}^{1:!}_{N_i}, \Delta_i, H, \beta\right)$;
9 $\quad\quad$ **end**
10 $\quad\quad i^* \leftarrow \arg\min_{j \in R} J\left(\boldsymbol{s}^!_{N_j}\right)$; `// Best solution in R`
11 $\quad\quad$ `// Fix` i^*`'s neighbors solutions`
12 $\quad\quad$ **for** $i \in \texttt{Neighbors}(i^*, k)$ **do**
13 $\quad\quad\quad \boldsymbol{a}_i^{1:!}(t) \leftarrow \boldsymbol{a}^{1:!}_{N_{i^*}}[i]$; `// The solution for bird i`
14 $\quad\quad$ **end**
15 $\quad$ **end**
16 $\quad$ `// First action and next state`
17 $\quad \boldsymbol{a}(t) \leftarrow \boldsymbol{a}^1(t)$; $\boldsymbol{s}^1 \leftarrow \bigcup_i \boldsymbol{s}^1_{N_i}$; $\boldsymbol{s}^! \leftarrow \bigcup_i \boldsymbol{s}^!_{N_i}$; $\boldsymbol{s} \leftarrow \boldsymbol{s}^1$;
18 $\quad$ **if** $\ell_{t-1} - J\left(\boldsymbol{s}^!\right) > \Delta$ **then**
19 $\quad\quad \ell_t \leftarrow J\left(\boldsymbol{s}^!\right)$; $t \leftarrow t{+}1$; `// Proceed to the next level`
20 $\quad$ **end**
21 $\quad k \leftarrow \texttt{NeighSize}\left(J\left(\boldsymbol{s}^!\right), k\right)$; `// Adjust neighborhood size`
22 **end**

The algorithm is distributed and with a dynamically changing topology. Lines 4, 10, and 18 require synchronization, which can be achieved by broadcasting corresponding information to a central hub of the network. This can be a different bird or a different base station at each time-step.

4.3. **The Local AMPC Algorithm.** `LocalAMPC` is a modified version of the AMPC algorithm [TSE+17], as shown in Alg. 5. Its input is an MDP $\mathcal{M}$, the current state $\boldsymbol{s}^{1:!}_{N_i}$ of a subflock N_i, a vector of acceleration sequences $\boldsymbol{a}^{1:!}_{N_i}$, one sequence for each bird in the subflock, a cost decrement Δ_i to be achieved, a maximum horizon H and a scaling factor β. In $\boldsymbol{a}^{1:!}_{N_i}$ some accelerations may not be fixed yet, that is, they have value ?.

Its output is a vector of acceleration sequences $\boldsymbol{a}^{1:!}_{N_i}$, one for each bird, that decreased the cost of the flock at most, the state $\boldsymbol{s}^{1:!}_{N_i}$ of the subflock after executing all actions. `LocalAMPC` first initializes (Line 1) the number of particles p to be used by PSO, proportionally to the input horizon h_i, to the number of birds B, and the scaling factor β. It then tries to decrement the cost of the subflock by at least Δ_i, as long as the maximum horizon H is not reached (Lines 3-7).

For this purpose it calls PSO (Line 5) with an increasingly longer horizon, and an increasingly larger number of particles. The idea is that the flock might have to first overcome a cost bump, before it gets to a state where the cost decreases by at least Δ_i. PSO

Algorithm 5: LocalAMPC

Input : $\mathcal{M} = (S, A, T, J, I)$, $\boldsymbol{s}_{N_i}^{1:!}$, $\boldsymbol{a}_{N_i}^{1:!}$, Δ_i, H, β
Output : $\boldsymbol{s}_{N_i}^{1:!}$, $\boldsymbol{a}_{N_i}^{1:!}$

1 $p \leftarrow 2 \cdot \beta \cdot B$; // Initial swarm size
2 $h_i \leftarrow 1$; // Initial horizon $\forall j \in N_i : \boldsymbol{a}_j^1 = ?$
3 **repeat**
4 | // Run PSO with local information $\boldsymbol{s}_{N_i}^{1:!}$ and $\boldsymbol{a}_{N_i}^{1:!}$
5 | $\left(t\boldsymbol{s}_{N_i}^{1:!}, t\boldsymbol{a}_{N_i}^{1:!}\right) \leftarrow \texttt{PSO}(\mathcal{M}, \boldsymbol{s}_{N_i}^{1:!}, \boldsymbol{a}_{N_i}^{1:!}, p, h_i)$;
6 | $h_i \leftarrow h_i + 1$; $p \leftarrow 2 \cdot \beta \cdot h_i \cdot B$; // increase horizon, swarm size
7 **until** $\left(J\left(t\boldsymbol{s}_{N_i}^{!}\right) - \ell_{t-1} < \Delta_i\right) \wedge (h_i \leqslant H)$
8 $\boldsymbol{s}_{N_i}^{1:!} \leftarrow t\boldsymbol{s}_{N_i}^{1:!}$; $\boldsymbol{a}_{N_i}^{1:!} \leftarrow t\boldsymbol{a}_{N_i}^{1:!}$; // Return temporary sequences

extends the input sequences of fixed actions to the desired horizon with new actions that are most successful in decreasing the cost of the flock, and it computes from scratch the sequence of actions, for the ? entries. The result is returned in $\boldsymbol{a}_{N_i}^{1:!}$. PSO also returns the states $\boldsymbol{s}_{N_i}^{1:!}$ of the flock after applying the whole sequence of actions. Using this information, it computes the actual cost achieved.

4.4. **Dynamic Neighborhood Resizing.** The key feature of DAMPC is that it *adaptively resizes neighborhoods*. This is based on the following observation: *as the agents are gradually converging towards a global optimal state, they can explore smaller neighborhoods* when computing actions that will improve upon the current configuration.

Adaptation works on lookahead cost, which is the cost that is reachable in some future time. Line 19 of DAMPC is reached (and the level ℓ_t is incremented) whenever we are able to decrease this look-ahead cost. If level ℓ_t is incremented, neighborhood size $k \in [k_{min}, k_{max}]$ is decremented, and incremented otherwise, as follows: $\texttt{NeighSize}(J, k) =$

$$\begin{cases} \min\left(\max\left(k - \left\lceil\left(1 - \frac{J(s^!)}{k}\right)\right\rceil, k_{min}\right), k_{max}\right), & \text{the next level} \\ \min\left(k+1, k_{max}\right), & \text{otherwise.} \end{cases} \tag{4.1}$$

In Fig. 7 we depict a simulation-trace example, demonstrating how levels and neighborhood size are adapting to the current value of the cost function.

4.5. **Local Convergence.**

Lemma 4.1 (Local convergence)**.** *Given $\mathcal{M} = (S, A, T, J, I)$, an MDP with cost function* `cost`*, and a nonempty set of target states $G \subset S$ with $G = \{\boldsymbol{s} \mid J(\boldsymbol{s}) \leqslant \varphi\}$. If the transition relation T is controllable with actions in A for every (local) subset of agents, then there exists a finite (maximum) horizon h_{max} such that* LocalAMPC *is able to find the best actions $\boldsymbol{a}_{N_i}^{1:!}$ that decreases the cost of a neighborhood of agents in the states $\boldsymbol{s}_{N_i}^{1:!}$ by at least a given Δ.*

Proof. In the input to LocalAMPC, the accelerations of some birds in N_i may be fixed (for some horizon). As a consequence, the MDP $\mathcal{M}$ may not be fully controllable within this horizon. Beyond this horizon, however, PSO is allowed to freely choose the accelerations, that is, the MDP $\mathcal{M}$ is fully controllable again. The result now follows from convergence of AMPC (Theorem 1 from [TSE+17]). □

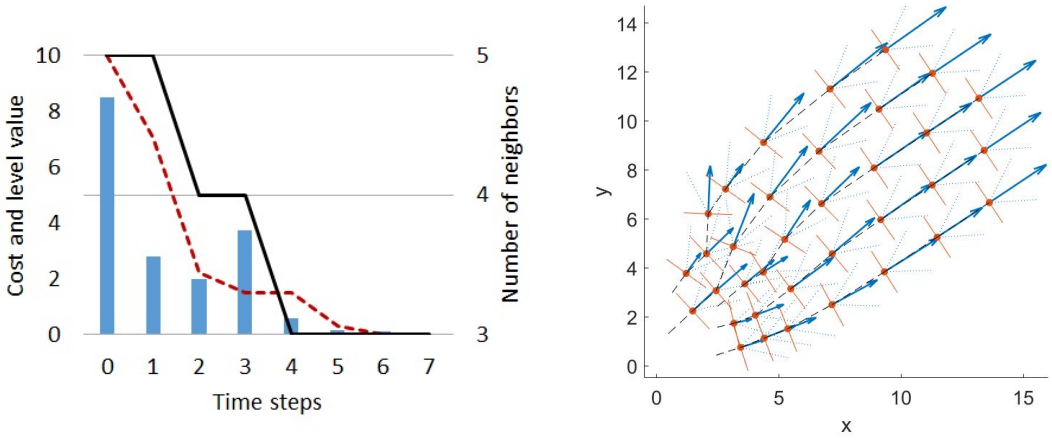

Figure 7: Left: Blue bars are the values of the cost function in every time step. Red dashed line in the value of the Lyapunov function serving as a threshold for the algorithm. Black solid line is resizing of the neighborhood for the next step given the current cost. Right: Step-by-step evolution of the flock from an arbitrary initial configuration in the left lower corner towards a V-formation in the right upper corner of the plot.

4.6. **Global Convergence and Stability.** Global convergence is achieved by our algorithm, where we overcome a local minimum by gradually adapting the neighborhood size to proceed to the next level defined by the Lyapunov function. Since we are solving a nonlinear nonconvex optimization problem, the cost J itself may not decrease monotonically. However, the look-ahead cost – the cost of some future reachable state – monotonically decreases. These costs are stored in level variables ℓ_t in Algorithm `DAMPC` and they define a Lyapunov function V.

$$V(t) = \ell_t \quad \text{for levels } t = 0, 1, 2, \ldots \tag{4.2}$$

where the levels decrease by at least a minimum dynamically defined threshold: $V(t+1) < V(t) - \Delta$.

Lemma 4.2. *$V(t) : \mathbb{Z} \to \mathbb{R}$ defined by (4.2) is a valid Lyapunov function, i.e., it is positive-definite and monotonically decreases until the system reaches its goal state.*

Proof. Note that the cost function $J(\boldsymbol{s})$ is positive by definition, and since ℓ_t equals $J(\boldsymbol{s})$ for some state $\boldsymbol{s}$, V is nonnegative. Line 18 of Algorithm `DAMPC` guarantees that V is monotonically decreasing by at least Δ.

□

Lemma 4.3 (Global consensus)**.** *Given Assumptions 1-7 in Section 4.1, all agents in the system will fix their actions in a finite number of consensus rounds.*

Proof. During the first consensus round, each agent i in the system runs `LocalAMPC` for its own neighborhood N_i of the current size k. Due to Lemma 4.1, $\exists\widehat{h}$ such that a solution, i.e. a set of action (acceleration) sequences of length $\widehat{h}$, will be found for all agents in the considered neighborhood N_i. Consequently, at the end of the round the solutions for at least all the agents in N_{i^*}, where i^* is the agent which proposed the globally best solution, will

be fixed. During the next rounds the procedure recurses. Hence, the set R of all agents with `nfy` values is monotonically decreasing with every consensus round. □

Global consensus is reached by the system during communication rounds. However, to achieve the global optimization goal we prove that the consensus value converges to the desired property.

Definition 3 . *Let $\{\boldsymbol{s}(t) : t = 1, 2, \ldots\}$ be a sequence of random vector-variables and s^* be a random or non-random. Then $\boldsymbol{s}(t)$ **converges with probability one** to s^* if*

$$\mathbb{P}\left[\bigcup_{\varepsilon>0}\bigcap_{N<\infty}\bigcup_{n\geqslant N}|\boldsymbol{s}(t)-\boldsymbol{s}^*|\geqslant\varepsilon\right]=0.$$

Lemma 4.4 (Max-neighborhood convergence)**.** *If* `DAMPC` *is run with constant neighborhood size B, then it behaves identically to centralized AMPC.*

Proof. If `DAMPC` uses neighborhood B, then it behaves like the centralized AMPC, because the accelerations of all birds are fixed in the first consensus round. □

Theorem 4.5 (Global convergence)**.** *Let $\mathcal{M} = (S, A, T, J, I)$ be an MDP with a positive and continuous cost function J and a nonempty set of target states $G \subset S$, with $G = \{\boldsymbol{s} \mid J(\boldsymbol{s}) \leqslant \varphi\}$. If there exists a finite horizon h_{max} and a finite number of execution steps m, such that centralized AMPC is able to find a sequence of actions $\{\boldsymbol{a}(t) : t = 1, \ldots, m\}$ that brings $\mathcal{M}$ from a state in I to a state in G, then* `DAMPC` *is also able to do so, with probability one.*

Proof. We illustrate the proof by our example of flocking. Note that the theorem is valid in the general formulation above for the fact that as global Lyapunov function approaches zero, the local dynamical thresholds will not allow neighborhood solutions to significantly diverge from reaching the state obtained as a result of repeated consensus rounds. Owing to Lemma 4.1, after the first consensus round, Alg. 5 finds a sequence of best accelerations of length h_{i^*}, for birds in subflock N_{i^*}, decreasing their cost by Δ_{i^*}. In the next consensus round, birds j outside N_{i^*} have to adjust the accelerations for their subflock N_j, while keeping the accelerations of the neighbors in $N_{i^*} \cap N_j$ to the already fixed solutions. If bird j fails to decrease the cost of its subflock N_j with at least Δ_j within prediction horizon h_{i^*}, then it can explore a longer horizon h_j up to h_{max}. This allows PSO to compute accelerations for the birds in $N_{i^*} \cap N_j$ in horizon interval $h_j < h \leqslant h_{i^*}$, decreasing the cost of N_j by Δ_j. Hence, the entire flock decreases its cost by Δ (this defines Lyapunov function V in Eq. 4.2) ensuring convergence to a global optimum. If h_{max} is reached before the cost of the flock was decreased by Δ, the size of the neighborhood will be increased by one, and eventually it would reach B. Consequently, using Theorem 1 in [TSE+17], there exists a horizon $h_{\max}$ that ensures global convergence. For this choice of h_{max} and for maximum neighborhood size, the cost is guaranteed to decrease by Δ, and we are bound to proceed to the next level in `DAMPC`. The Lyapunov function on levels guarantees that we have no indefinite switching between "decreasing neighborhood size" and "increasing neighborhood size" phases, and we converge (see Fig. 2). □

The result presented in [TSE+17] applied to our distributed approach, together with Theorem 4.5, ensure the following corollary.

Corollary 1 Global stability**.** *Assume the set of target states $G \in S$ has been reached and one of the following perturbations of the system dynamics has been applied: a) the best next*

action is chosen with probability zero (crash failure); b) an agent is displaced (sensor noise); c) an action of a player with opposing objective is performed. Then applying Algorithm 4 the system converges with probability one from a disturbed state to a state in G.

4.7. **Evaluation of the Distributed `AMPC` Controller.** We comprehensively evaluated `DAMPC` to compute statistical estimates of the success rate of reaching V-formation from an arbitrary initial state in a finite number of steps m. We considered flocks of size $B = \{5, 7, 9\}$ birds. The specific reachability problem we addressed is as follows.

Given a flock MDP $\mathcal{M}$ with B birds and the randomized strategy $\sigma : S \mapsto PD(A)$ of Alg. 4, estimate the probability of reaching a state s where the cost function $J(s) \leqslant \varphi$, starting from an initial state in the underlying Markov chain $\mathcal{M}_\sigma$ induced by σ on $\mathcal{M}$.

Since the exact solution to this stochastic reachability problem is intractable (infinite/continuous state and action spaces), we solve it approximately using statistical model checking (SMC). In particular, as the probability estimate of reaching a V-formation under our algorithm is relatively high, we can safely employ the *additive error* (ε, δ)-Monte-Carlo-approximation scheme [GPR$^+$14]. This requires L i.i.d. executions (up to a maximum time horizon), determining in Z_l if execution l reaches a V-formation, and returning the mean of the random variables $Z_1, \ldots, Z_L$. We compute $\widetilde{\mu}_Z = \sum_{l=1}^{L} Z_l/L$ by using Bernstein's inequality to fix $L \propto ln(1/\delta)/\varepsilon^2$ and obtain $\mathbb{P}[\mu_Z - \varepsilon \leq \widetilde{\mu}_Z \leq \mu_Z + \varepsilon] \geq 1 - \delta$, where $\widetilde{\mu}_Z$ approximates μ_Z with additive error ε and probability $1 - \delta$. In particular, we are interested in a Bernoulli random variable Z returning 1 if the cost $J(s)$ is less than φ and 0 otherwise. In this case, we can use the Chernoff-Hoeffding instantiation of the Bernstein's inequality, and further fix the proportionality constant to $N = 4\, ln(2/\delta)/\varepsilon$ [HLMP04b]. Executing the algorithm 10^3 times for each flock size gives us a confidence ratio $\delta = 0.05$ and an additive error of $\varepsilon = 10^{-2}$.

We used the following parameters: number of birds $B \in \{5, 7, 9\}$, cost threshold $\varphi = 10^{-1}$, maximum horizon $h_{max} = 3$, number of particles in PSO $p = 200 \cdot h \cdot B$. `DAMPC` is allowed to run for a maximum of $m = 60$ steps. The initial configurations are generated independently, uniformly at random, subject to the following constraints on the initial positions and velocities: $\forall\, i \in \{1, \ldots, B\}\ \boldsymbol{x}_i(0) \in [0, 3] \times [0, 3]$ and $\boldsymbol{v}_i(0) \in [0.25, 0.75] \times [0.25, 0.75]$.

To perform the SMC evaluation of `DAMPC`, and to compare it with the centralized AMPC from [TSE$^+$17], we designed the above experiments for both algorithms in C, and ran them on the 2x Intel Xeon E5-2660 Okto-Core, 2.2 GHz, 64 GB platform.

Our experimental results are given in Table 4. We used three different ways of computing the average number of neighbors for successful runs. Assuming a successful run converges after m' steps, we (1) compute the average over the first m' steps, reported as "for good runs until convergence"; (2) extend the partial m'-step run into a full m-step run and compute the average over all m steps, reported as "for good runs over m steps"; or (3) take an average across $> m$ steps, reported as "for good runs after convergence", to illustrate global stability.

We obtain a high success rate for 5 and 7 birds, which does not drop significantly for 9 birds. The average convergence duration, horizon, and neighbors, respectively, increase monotonically when we consider more birds, as one would expect. The average neighborhood size is smaller than the number of birds, indicating that we improve over AMPC [TSE$^+$17] where all birds need to be considered for synthesizing the next action.

Table 4: Comparison of DAMPC and AMPC [TSE+17] on 10^3 runs.

	DAMPC			AMPC		
Number of Birds	5	7	9	5	7	9
Success rate, $\widetilde{\mu}_Z$	0.98	0.92	0.80	0.99	0.95	0.88
Avg. convergence duration, m	7.40	10.15	15.65	9.01	12.39	17.29
Avg. horizon, h	1.35	1.36	1.53	1.29	1.55	1.79
Avg. execution time in sec.	$295s$	$974s$	$\propto 10^3 s$	$644s$	$3120s$	$\propto 10^4 s$
Avg. neighborhood size, k						
for good runs until convergence	3.69	5.32	6.35	5.00	7.00	9.00
for good runs over m steps	3.35	4.86	5.58	5.00	7.00	9.00
for good runs after convergence	4.06	5.79	6.75	5.00	7.00	9.00
for bad runs	4.74	6.43	6.99	5.00	7.00	9.00

We also observe that the average number of neighbors for good runs until convergence is larger than the one for bad runs, except for 5 birds. The reason is that in some bad runs the cost drops quickly to a small value resulting in a small neighborhood size, but gets stuck in a local minimum (e.g., the flock separates into two groups) due to the limitations imposed by fixing the parameters h_{max}, p, and m. The neighborhood size remains small for the rest of the run leading to a smaller average.

Finally, compared to the centralized AMPC [TSE+17], `DAMPC` is faster (e.g., two times faster for 5 birds). Our algorithm takes fewer steps to converge. The average horizon of `DAMPC` is smaller. The smaller horizon and neighborhood sizes, respectively, allow PSO to speed up its computation.

5. Attacking the V (Controller-Attacker Games)

In [TSE+17], we introduced *V-formation games*, a class of controller-attacker games, where the goal of the controller is to maneuver the plant (a simple model of flocking dynamics) into a V-formation, and the goal of the attacker is to prevent the controller from doing so. Controllers in V-formation games use centralized `AMPC`. We define several classes of attackers, including those that in one move can remove a small number of birds from the flock, or introduce random displacement (perturbation) into the flock dynamics, again by selecting a small number of victim agents. We consider both naive attackers, whose strategies are purely probabilistic, and `AMPC`-enabled attackers, putting them on par strategically with the controller.

We describe the specialization of the stochastic-game verification problem to V-formation. In particular, we present the AMPC-based control strategy for reaching a V-formation, and the various attacker strategies against which we evaluate the resilience of our controller.

5.1. Controller's Adaptive Strategies.

Given current state $(\vec{x}(t), \vec{v}(t))$, the controller's strategy σ_C returns a probability distribution on the space of all possible accelerations (for all birds). As mentioned above, this probability distribution is specified implicitly via a randomized algorithm that returns an actual acceleration (again for all birds). This randomized algorithm is the AMPC algorithm, which inherits its randomization from the randomized PSO procedure it deploys.

When the controller computes an acceleration, it assumes that the attacker does *not* introduce any disturbances; i.e., the controller uses Eq. 2.1 where $\boldsymbol{a}(t)$ is the only control variable. Note that the controller chooses its next action $\boldsymbol{a}(t)$ based on the current configuration $(\boldsymbol{x}(t), \boldsymbol{v}(t))$ of the flock using MPC. The current configuration may have been influenced by the disturbance $\vec{d}(t-1)$ introduced by the attacker in the previous time step. Hence, the current state need not to be the state predicted by the controller when performing MPC in step $t-1$. Moreover, depending on the severity of the attacker action $\vec{d}(t-1)$, the AMPC procedure dynamically adapts its behavior, i.e. the choice of horizon h, in order to enable the controller to pick the best control action $\vec{a}(t)$ in response.

5.2. **Attacker's Strategies.** We are interested in evaluating the resilience of our V-formation controller when it is threatened by an attacker that can remove a certain number of birds from the flock, or manipulate a certain number of birds by taking control of their actuators (modeled by the displacement term in Eq. 2.1). We assume that the attack lasts for a limited amount of time, after which the controller attempts to bring the system back into the good set of states. When there is no attack, the system behavior is the one given by Eq. 2.2.

Bird Removal Game. In a BRG, the attacker selects a subset of R birds, where $R \ll B$, and removes them from the flock. The removal of bird i from the flock can be simulated in our framework by setting the displacement $\boldsymbol{d}_i$ for bird i to ∞. We assume that the flock is in a V-formation at time $t=0$. Thus, the goal of the controller is to bring the flock back into a V-formation consisting of $B-R$ birds.

Apart from seeing if the controller can bring the flock back to a V-formation, we also analyze the time it takes the controller to do so.

Definition 4 . *In a* Bird Removal Game *(BRG), the attacker strategy σ_D is defined as follows. Starting from a V-formation of B birds, i.e., $J(s_0) \leqslant \varphi$, the attacker chooses a subset of R birds, $R \ll B$, by uniform sampling without replacement. Then, in every round, it assigns each bird i in the subset a displacement $\boldsymbol{d}_i = \infty$, while for all other birds j, $\boldsymbol{d}_j = 0$.*

Random Displacement Game. In an RDG, the attacker chooses the displacement vector for a subset of R birds uniformly from the space $[0, M] \times [0, 2\pi]$ with $R \ll B$. This means that the magnitude of the displacement vector is picked from the interval $[0, M]$, and the direction of the displacement vector is picked from the interval $[0, 2\pi]$. We vary M in our experiments. The subset of R birds that are picked in different steps are not necessarily the same, as the attacker makes this choice uniformly at random at runtime as well.

The game starts from an initial V-formation. The attacker is allowed a fixed number of moves, say 20, after which the displacement vector is identically 0 for all birds. The controller, which has been running in parallel with the attacker, is then tasked with moving the flock back to a V-formation, if necessary.

Definition 5 . *In a* Random Displacement Game *(RDG), the attacker strategy σ_D is defined as follows. Starting from a V-formation of B birds, i.e., $J(s_0) \leqslant \varphi$, in every round, it chooses a subset of R birds, $R \ll B$, by uniform sampling without replacement. It then assigns each bird i in the subset a displacement $\boldsymbol{d}_i$ chosen uniformly at random from $[0, M] \times [0, 2\pi]$, while for all other birds j, $\boldsymbol{d}_j = 0$. After T rounds, all displacements are set to 0.*

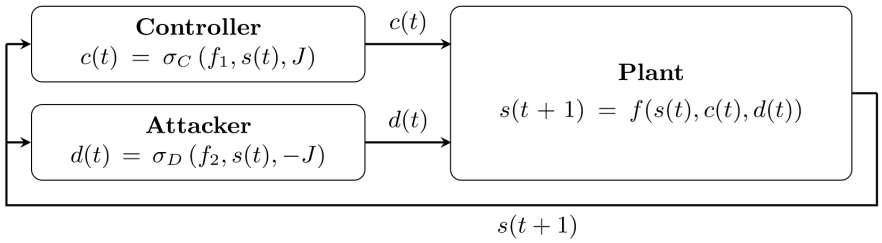

Figure 8: Controller-Attacker Game Architecture. The controller and the attacker use randomized strategies σ_C and σ_D to choose actions $c(t)$ and $d(t)$ based on dynamics, respectively, where $s(t)$ is the state at time t, and f is the dynamics of the plant model. The controller tries to minimize the cost J, while the attacker tries to maximize it.

AMPC Game. An AMPC game is similar to an RDG except that the attacker does not use a uniform distribution to determine the displacement vector. The attacker is advanced and strategically calculates the displacement using the AMPC procedure. See Figure 8. In detail, the attacker applies AMPC, but assumes the controller applies zero acceleration. Thus, the attacker uses the Eq. 2.1 as the model of the flock dynamics.

Note that the attacker is still allowed to have $\boldsymbol{d}_i(t)$ be non-zero for only a small number of birds. However, it gets to choose these birds in each step. It uses the AMPC procedure to simultaneously pick the subset of R birds and their displacements. The objective of the attacker's AMPC is to maximize the cost.

Definition 6 . *In an* AMPC game*, the attacker strategy σ_D is defined as follows. Starting from a V-formation of B birds, i.e., $J(s_0) \leqslant \varphi$, in every round, it uses AMPC to choose a subset of R birds, $R \ll B$, and their displacements $\boldsymbol{d}_i$ for bird i in the subset from $[0, M] \times [0, 2\pi]$; for all other birds j, $\boldsymbol{d}_j = 0$. After T rounds, all displacements are set to 0.*

Theorem 5.1 (AMPC Convergence)**.** *Given an MDP $\mathcal{M} = (S, A, T, J)$ with positive and continuous cost function J, and a nonempty set of target states $G \subset S$ with $G = \{s \mid J(s) \leqslant \varphi\}$. If the transition relation T is controllable with actions in A, then there exists a finite maximum horizon h_{max} and a finite number of execution steps m, such that AMPC is able to find a sequence of actions $a_1, \ldots, a_m$ that brings a state in S to a state in G with probability one.*

Proof. In each (macro-) step of horizon length h, from level ℓ_{i-1} to level ℓ_i, AMPC decreases the distance to φ by $\Delta_i \geqslant \Delta$, where $\Delta > 0$ is fixed by the number of steps m chosen in advance. Hence, AMPC converges to a state in G in a finite number of steps, for a properly chosen m. AMPC is able to decrease the cost in a macro step by Δ_i by the controllability assumption and the fairness assumption about the PSO algorithm. Since AMPC is a randomized algorithm, the result is probabilistic. Note that the theorem is an existence theorem of h_{max} and m whose values are chosen empirically in practice. □

Theorem 5.2 (AMPC resilience in a C-A game)**.** *Given a controller-attacker game, there exists a finite maximum horizon h_{max} and a finite maximum number of game-execution steps m such that AMPC controller will win the controller-attacker game in m steps with probability 1.*

Proof. Since the flock MDP (defined by Eq. 2.1) is controllable, the PSO algorithm we use is fair, and the attack has a bounded duration, the proof of the theorem follows from Theorem 5.1. □

Remark 1 . *While Theorem 5.2 states that the controller is expected to win with probability 1, we expect winning probability to be possibly lower than one in many cases because: (1) the maximum horizon h_{max} is fixed in advance, and so is (2) the maximum number of execution steps m; (3) the underlying PSO algorithm is also run with bounded number of particles and time. Theorem 5.2 is an existence theorem of h_{max} and m, while in practice one chooses fixed values of h_{max} and m that could be lower than the required values.*

5.3. **Statistical MC Evaluation of V-Formation Games.** The stochastic-game verification problem we address in the context of the V-formation-AMPC algorithm is formulated as follows. Given a flock MDP $\mathcal{M}$ (we consider the case of $B=7$ birds), acceleration actions $\boldsymbol{a}$ of the controller, displacement actions $\boldsymbol{d}$ of the attacker, the randomized strategy $\sigma_C : S \mapsto PD(C)$ of the controller (the AMPC algorithm), and a randomized strategy $\sigma_D : S \mapsto PD(D)$ for the attacker, determine the probability of reaching a state s where the cost function $J(s) \leqslant \varphi$ (V-formation in a 7-bird flock), starting from an initial state (in this case this is a V-formation), in the underlying Markov chain induced by strategies σ_C, σ_D on $\mathcal{M}$.

Since the exact solution to this reachability problem is intractable due to the infinite/continuous space of states and actions, we solve it approximately with classical statistical model-checking (SMC). The particular SMC procedure we use is from [GPR+14] and based on an *additive* or *absolute-error* (ε, δ)*-Monte-Carlo-approximation scheme*.

This technique requires running N i.i.d. game executions, each for a given maximum time horizon, determining if these executions reach a V-formation, and returning the average number of times this occurs.

Each of the games described in Section 5.2 is executed 2,000 times. For a confidence ratio $\delta = 0.01$, we thus obtain an additive error of $\varepsilon = 0.1$. We use the following parameters in the game executions: number of birds $B=7$, threshold on the cost $\varphi = 10^{-3}$, maximum horizon $h_{max} = 5$, number of particles in PSO $p = 20hB$. In BRG, the controller is allowed to run for a maximum of 30 steps. In RDG and AMPC game, the attacker and the controller run in parallel for 20 steps, after which the displacement becomes 0, and the controller has a maximum of 20 more steps to restore the flock to a V-formation.

To perform SMC evaluation of our AMPC approach we designed the above experiments in C and ran them on the Intel Core i7-5820K CPU with 3.30 GHz and with 32GB RAM available.

Table 5: Results of 2,000 game executions for removing 1 bird with $h_{max} = 5$, $m = 40$

	Ctrl. success rate, %	Avg. convergence duration	Avg. horizon
Bird 4	99.9	12.75	3.64
Bird 3	99.8	18.98	4.25
Bird 2	100	10.82	3.45

Discussion of the Results. To demonstrate the resilience of our adaptive controller, for each game introduced in Section 5.2, we performed a number of experiments to estimate the probability of the controller winning. Moreover, for the runs where the controller wins, the average number of steps required by the controller to bring the flock to a V-formation is

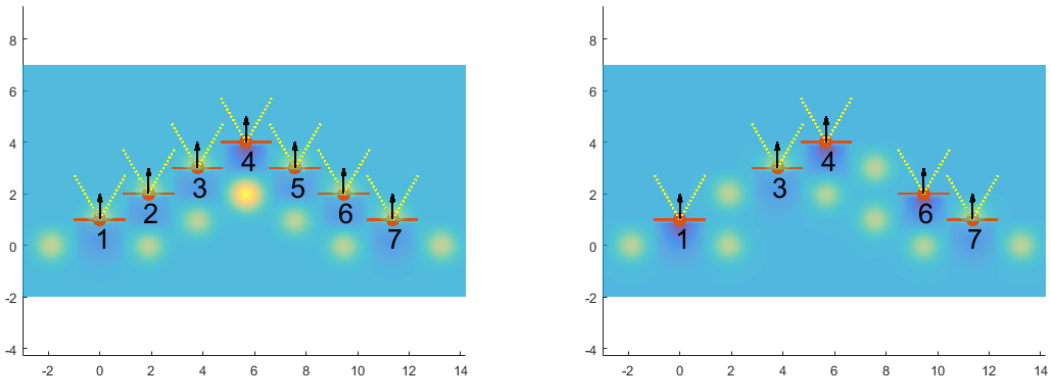

Figure 9: Left: numbering of the birds. Right: configuration after removing Bird 2 and 5. The red-filled circle and two protruding line segments represent a bird's body and wings. Arrows represent bird velocities. Dotted lines illustrate clear-view cones. A brighter/darker background color indicates a higher upwash/downwash.

Table 6: Results of 2,000 game executions for removing 2 birds with $h_{max} = 5$, $m = 30$

	Ctrl. success rate, %	Avg. convergence duration	Avg. horizon
Birds 2 and 3	0.8	25.18	4.30
Birds 2 and 4	83.1	11.11	2.94
Birds 2 and 5	80.3	9.59	2.83
Birds 2 and 6	98.6	7.02	2.27
Birds 3 and 4	2.0	22.86	4.30
Birds 3 and 5	92.8	11.8	3.43

Table 7: Results of 2,000 game executions for random displacement and AMPC attacks with $h_{max} = 5$ and $m = 40$ (attacker runs for 20 steps)

Range of noise	Ctrl. success rate, %	Avg. convergence duration	Avg. horizon
	Random displacement game		
$[0, 0.50] \times [0, 2\pi]$	99.9	3.33	1.07
$[0, 0.75] \times [0, 2\pi]$	97.9	3.61	1.11
$[0, 1.00] \times [0, 2\pi]$	92.3	4.14	1.18
	AMPC game		
$[0, 0.50] \times [0, 2\pi]$	97.5	4.29	1.09
$[0, 0.75] \times [0, 2\pi]$	63.4	5.17	1.23
$[0, 1.00] \times [0, 2\pi]$	20.0	7.30	1.47

reported as *average convergence duration*, and the average length of the horizon used by AMPC is reported as *average horizon*.

The numbering of the birds in Tables 5 and 6 is given in Figure 9. Bird-removal scenarios that are symmetric with the ones in the tables are omitted. The results presented in Table 5 are for the BRG game with $R=1$. In this case, the controller is *almost always* able to bring the flock back to a V-formation, as is evident from Table 5. Note that removing Bird 1 (or 7) is a trivial case that results in a V-formation.

In the case when $R=2$, shown in Table 6, the success rate of the controller depends on *which two birds are removed*. Naturally, there are cases where dropping two birds does not break the V-formation; for example, after dropping Birds 1 and 2, the remaining birds continue to be in a V-formation. Such trivial cases are not shown in Table 6. Note that the scenario of removing Bird 1 (or 7) and one other bird can be viewed as removing one bird in flock of 6 birds, thus not considered in this table. Among the other nontrivial cases, the success rate of controller drops slightly in four cases, and drops drastically in remaining two cases. This suggests that attacker of a CPS system can incur more damage by being prudent in the choice of the attack.

Impressively, whenever the controller wins, the controller needs about the same number of steps to get back to V-formation (as in the one-bird removal case). On average, removal of two birds results in a configuration that has worse cost compared to an BRG with $R=1$. Hence, the adaptive controller is able to make bigger improvements (in each step) when challenged by worse configurations. Furthermore, among the four cases where the controller win rate is high, experimental results demonstrate that removing two birds positioned asymmetrically with respect to the leader poses a stronger, however, still manageable threat to the formation. For instance, the scenarios of removing birds 2 and 6 or 3 and 5 give the controller a significantly higher chance to recover from the attack, 98.6% and 92.8%, respectively.

Table 7 explores the effect of making the attacker smarter. Compared to an attacker that makes random changes in displacement, an attacker that uses AMPC to pick its action is able to win more often. This again shows that an attacker of a CPS system can improve its chances by cleverly choosing the attack. For example, the probability of success for the controller to recover drops from 92.3% to 20.0% when the attacker uses AMPC to pick displacements with magnitude in $[0, 1]$ and direction in $[0, 2\pi]$. The entries in the other two columns in Table 7 reveal two even more interesting facts.

First, in the cases when the controller wins, we clearly see that the controller uses a longer look-ahead when facing a more challenging attack. This follows from the observation that the average horizon value increases with the strength of attack. This gives evidence for the fact that the adaptive component of our AMPC plays a pivotal role in providing resilience against sophisticated attacks. Second, the average horizon still being in the range 1-1.5, means that the adaptation in our AMPC procedure also helps it perform better than a fixed-horizon MPC procedure, where usually the horizon is fixed to $h \geqslant 2$. When a low value of h (say $h=1$) suffices, the AMPC procedure avoids unnecessary calculation that using a fixed h might incur.

In the cases where success rate was low (Row 1 and Row 5 in Table 6, and Row 3 of the AMPC game in Table 7), we conducted additional 500 runs for each case and observed improved success rates (2.4%, 9% and 30.8% respectively) when we increased h_{max} to 10 and m to 40. This shows that success rates of AMPC improves when given more resources, as predicted by Theorem 5.1.

6. Related Work

Organized flight in flocks of birds can be categorized in *cluster flocking* and *line formation* [Hep74]. In cluster flocking the individual birds in a large flock seem to be uncoordinated in general. However, the flock moves, turns, and wheels as if it were one organism. In 1987 Reynolds [Rey87b] defined his three famous rules describing separation, alignment, and cohesion for individual birds in order to have them flock together. This work has been great inspiration for research in the area of collective behavior and self-organization.

In contrast, line formation flight requires the individual birds to fly in a very specific formation. Line formation has two main benefits for the long-distance migrating birds. First, exploiting the generated uplift by birds flying in front, trailing birds are able to conserve energy [LS70, CS94, WMC+01]. Second, in a staggered formation, all birds have a clear view in front as well as a view on their neighbors [BH09]. While there has been quite some effort to keep a certain formation for multiple entities when traveling together [SPH02, GIV05, DH15], only little work deals with a task of achieving this extremely important formation from a random starting configuration [CS11]. The convergence of bird flocking into V-formation has been also analyzed with the use of combinatorial techniques [Cha14].

Compared to previous work, in [CA07] this question is addressed without using any behavioral rules but as problem of *optimal control*. In [YGST16] a cost function was proposed that reflects all major features of V-formation, namely, *Clear View* (CV), *Velocity Matching* (VM), and *Upwash Benefit* (UB). The technique of MPC is used to achieve V-formation starting from an arbitrary initial configuration of n birds. MPC solves the task by minimizing a functional defined as squared distance from the optimal values of CV, VM, and UB, subject to constraints on input and output. The approach is to choose an optimal *velocity adjustment*, as a control input, at each time-step applied to the velocity of each bird by predicting model behavior several time-steps ahead.

The controller synthesis problem has been widely studied [VPT+12]. The most popular and natural technique is Dynamic Programming (DP) [Bel57], which improves the approximation of the functional at each iteration, eventually converging to the optimal one given a fixed asymptotic error. Compared to DP, which considers all possible states of the system and might suffer from state-space explosion in case of environmental uncertainties, approximate algorithms [HMZ+12,BBB+16,MRG03,BBW11,SS12b,SS12a] take into account only the paths leading to a desired target. One of the most efficient ones is Particle Swarm Optimization (PSO) [KE95] that has been adopted for finding the next best step of MPC in [YGST16]. Although it is a very powerful optimization technique, it has not yet been possible to achieve a high success rate in solving the considered flocking problem.

Sequential Monte-Carlo methods prove to be efficient in tackling the question of control for linear stochastic systems [CWL09], in particular, Importance Splitting (IS) [KJL+16]. The approach we propose is, however, the first attempt to combine adaptive IS, PSO, and receding-horizon technique for *synthesis of optimal plans for controllable systems*. We use MPC to synthesize a plan, but use IS to determine the intermediate fitness-based waypoints. We use PSO to solve the multi-step optimization problem generated by MPC, but choose the planning horizon and the number of particles adaptively. These choices are governed by the difficulty to reach the next level.

Adaptive control, and its special case of adaptive model predictive control, typically refers to the aspect of the controller updating its process model that it uses to compute the control action. The field of adaptive control is concerned with the discrepancy between the

actual process and its model used by the controller. In our adaptive-horizon MPC, we adapt the lookahead horizon employed by the MPC, and not the model itself. Hence, the work in this paper is orthogonal to what is done in adaptive control [Nar90, ADG09].

Adaptive-horizon MPC was used in [DE11] to track a reference signal. If the reference signal is unknown, and we have a poor estimate of its future behavior, then a larger horizon for MPC is not beneficial. Thus, the horizon was determined by the uncertainty in the knowledge of the future reference signal. We consider cost-based reachability goals here, which allows us to choose a horizon in a more generic way based on the progress toward the goal. More recently, adaptive horizons were also used in [Kre16] for a reachability goal. However, they chose a large-enough horizon that enabled the system to reach states from where a pre-computed local controller could guarantee reachability of the goal. This is less practical than our approach for establishing the horizon.

Prior work on the V-formation problem has focused on giving combinations of *dynamical flight rules* as driving forces. These approaches tend to be distributed in nature, as flight rules describe how an individual bird maneuvers depending on the positions and velocities of the neighbors within its radius of influence. For instance, in [Fla98], the authors extend Reynolds' flocking model [Rey87a] with a rule that forces a bird to move laterally away from any bird that blocks its view. This can result in multiple V-shaped clusters, but flock-wide convergence is not guaranteed. The work of [DS03] induces V-formations by extending Reynolds' model with a *drag reduction* rule, but the final formation tends to oscillate as birds repeatedly adjust the angle of the V. Another approach, based on three *positioning rules*, is that of [NB08]. It provides an alternative model that produces V-formations. The birds in their model follow three positioning rules: (1) seek the proximity of the nearest bird; (2) seek the nearest position that affords an unobstructed longitudinal view; and (3) attempt to position itself in the upwash of a leading bird. Their model, however, is limited by the assumption that the birds have a constant longitudinal heading. The authors of [SW11] attempt to improve upon this approach by handling *turning movements*. This also forms small clusters of birds, each of which is only moderately V-like.

In [ZL13], the problem of taking an arbitrary initial configuration of n agents to a final configuration where every pair of "neighbors" is a fixed distance d apart, and every agent is stationary (its velocity is zero) is considered. They present centralized and distributed algorithms for this problem, both of which use MPC to determine the next action. The problem addressed in [ZL13] is arguably simpler than the V-formation problem we consider. The cost function being minimized in their case is a quadratic convex function. In [ZL13] the proof of convergence uses the fact of existing sequence of states with monotonically decreasing cost. Our cost function is nonconvex and nonlinear, which requires overcoming local minima by horizon and neighborhood adaptation. Both of these concepts are not required, and hence not addressed, in [ZL13]. In the distributed control procedure of [ZL13], each agent publishes the control value it locally computed, which is then used by other agents to compute their control value. A quadratic number of these "small steps" are performed before each agent fixes its control input for the next time step. Our distributed procedure has at most a linear number of these small steps.

Other related work, including [FD02, DD03, YZS17], focuses on distributed controllers for flight formation that operate in an environment where the (multi-agent) plant is already in the desired formation and the (distributed) controller's objective is to maintain formation in the presence of disturbances (typically on the roll angle of the wing). Moreover, the plants

considered are more physically detailed than our plant model in terms of capturing the dynamics of moving-wing aircraft. We plan to consider models of this nature as future work.

A distinguishing feature of these approaches is the particular formation they are seeking to maintain, including a half-vee [FD02], a ring and a torus [DD03], and a leader-follower formation [YZS17]. In contrast, we use distributed AMPC with dynamic neighborhood resizing to bring a flock from a mostly random initial configuration to a stable V-formation.

In the field of CPS security, one of the most widely studied attacks is *sensor spoofing*. When sensors measurements are compromised, state estimation becomes challenging, which inspired a considerable amount of work on attack-resilient state estimation [FTD14, PDB13, PWB+14, PIW+15, DWJ+16]. In these approaches, resilience to attacks is typically achieved by assuming the presence of redundant sensors, or coding sensor outputs. In our work, we do not consider sensor-spoofing attacks, but assume the attacker gets control of the displacement vectors (for some of the birds/drones). We have not explicitly stated the mechanism by which an attacker obtains this capability, but it is easy to envision ways (radio controller, attack via physical medium, or other channels [CMK+11]) for doing so.

A key focus in CPS security has also been detection of attacks. For example, recent work considers displacement-based attacks on formation flight [NKC16], but it primarily concerned with detecting which UAV was attacked using an unknown-input-observer based approach. We are not concerned with detecting attacks, but establishing that the adaptive nature of our controller provides attack-resilience for free. Moreover, in our setting, for both the attacker the and controller the state of the plant is completely observable. In [SSP+17], a control policy based on the robustness of the connectivity graph is proposed to achieve consensus on the velocity among a team of mobile robots, in the present of non-cooperative robots that communicate false values but execute the agreed upon commands. In contrast, we allow the attacker to manipulate the executed commands of the robots. The cost function we use is also more flexible so that we can encode more complicated objectives.

We are unaware of any work that uses statistical model checking to evaluate the resilience of adaptive controllers against (certain classes of) attacks.

7. Conclusions

We first presented ARES, a very general adaptive, receding-horizon synthesis algorithm for MDP-based optimal plans; ARES can be viewed as a model-predictive controller with an adaptive receding horizon (AMPC). We conducted a thorough performance analysis of ARES on the V-formation problem to obtain statistical guarantees of convergence. For flocks of 7 birds, ARES is able to generate, with high confidence, an optimal plan leading to V-formation in 95% of the 8,000 random initial configurations we considered, with an average execution time of only 63 seconds per plan.

We next presented `DAMPC`, a distributed version of AMPC that uses an adaptive-neighborhood and adaptive-horizon model-predictive control algorithm to generated actions for a controllable MDP so that it eventually reaches a state with cost close to zero, provided that the MDP has such a state. The main contribution of `DAMPC` as a distributed control algorithm is that it adaptively resizes an agent's local neighborhood, while still managing to converge to a goal state with high probability. Our evaluation showed that the value of dynamic neighborhood resizing, where we observed that it can lead to a relatively small average neighborhood size while successfully reaching a goal state.

Finally, to demonstrate the resilience of our adaptive controllers, we introduced a variety of controller-attacker games and carried out a number of experiments to estimate the probability of the controller winning. Our analysis demonstrated the effectiveness of adaptive controllers in overcoming certain kinds of controller-targeted attacks.

Acknowledgment

The authors gratefully acknowledge the significant contributions of Ezio Bartocci, Lukas Esterle, Christian Hirsch, and Junxing Yang to this work.

References

[ADG09] Veronica Adetola, Darryl DeHaan, and Martin Guay. Adaptive model predictive control for constrained nonlinear systems. *Systems & Control Letters*, 58(5):320–326, 2009.

[BBB+16] Ezio Bartocci, Luca Bortolussi, Tomáš Brázdil, Dimitrios Milios, and Guido Sanguinetti. Policy learning for time-bounded reachability in continuous-time markov decision processes via doubly-stochastic gradient ascent. In *Proc. of QEST 2016: the 13th International Conference on Quantitative Evaluation of Systems*, volume 9826, pages 244–259, 2016.

[BBW11] J. Baxter, P. L. Bartlett, and L. Weaver. Experiments with infinite-horizon, policy-gradient estimation. *J. Artif. Int. Res.*, 15(1):351–381, 2011.

[Bel57] Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[BH09] Iztok Lebar Bajec and Frank H. Heppner. Organized flight in birds. *Animal Behaviour*, 78(4): 777–789, 2009.

[Blo] Boeing Copies Flying Geese to Save Fuel. `https://www.bloomberg.com/news/articles/2017-08-08/boeing-nasa-look-to-flying-geese-in-chase-for-jet-fuel-savings`.

[CA07] E. F. Camacho and C. B. Alba. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer, 2007.

[Cha14] Bernard Chazelle. The Convergence of Bird Flocking. *Journal of the ACM*, 61(4):21:1–21:35, 2014.

[CMK+11] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis an, Franziska Roesner, and Tadayoshi Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security*, 2011.

[Con17] J. Condliffe. A 100-drone swarm, dropped from jets, plans its own moves. *MIT Technology Review*, January 2017.

[CS94] C Cutts and J Speakman. Energy savings in formation flight of pink-footed geese. *Journal of Experimental Biology*, 189(1):251–261, 1994.

[CS11] F. S. Cattivelli and A. H. Sayed. Modeling bird flight formations using diffusion adaptation. *IEEE Transactions on Signal Processing*, 59(5):2038–2051, 2011.

[CWL09] Yuguo Chen, Bin Wu, and Tze Leung Lai. *Fast Particle Filters and Their Applications to Adaptive Control in Change-Point ARX Models and Robotics*. INTECH Open Access Publisher, 2009.

[DD03] R. D'Andrea and G. E. Dullerud. Distributed control design for spatially interconnected systems. *IEEE Transactions on Automatic Control*, 48(9), 2003.

[DE11] Greg Droge and Magnus Egerstedt. Adaptive time horizon optimization in model predictive control. In *American Control Conference (ACC), 2011*, pages 1843–1848. IEEE, 2011.

[DH15] A. D. Dang and J. Horn. Formation control of autonomous robots following desired formation during tracking a moving target. In *Proceedings of the International Conference on Cybernetics*, pages 160–165. IEEE, 2015.

[DS03] G Dimock and M Selig. The Aerodynamic Benefits of Self-Organization in Bird Flocks. *Urbana*, 51:1–9, 2003.

[DWJ+16] Drew Davidson, Hao Wu, Robert Jellinek, Thomas Ristenpart, and Vikas Singh. Controlling UAVs with sensor input spoofing attacks. In *Proceedings of WOOT'16, 10th USENIX Workshop on Offensive Technologies*. Austin, TX, August 2016.

[FD02] J. M. Fowler and R. D'Andrea. Distributed control of close formation flight. In *Proc. of 41st IEEE Conference on Decision and Control*, December 2002.

[Fla98] Gary William Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MIT Press, 1998.

[FTD14] Hamza Fawzi, Paulo Tabuada, and Suhas N. Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Trans. Automat. Contr.*, 59(6):1454–1467, 2014. URL `http://dx.doi.org/10.1109/TAC.2014.2303233`.

[GIV05] M. C. De Gennaro, L. Iannelli, and F. Vasca. Formation Control and Collision Avoidance in Mobile Agent Systems. In *Proceedings of the International Symposium on Control and Automation Intelligent Control*, pages 796–801. IEEE, 2005.

[GPR+14] R. Grosu, D. Peled, C. R. Ramakrishnan, S. A. Smolka, S. D. Stoller, and J. Yang. Using statistical model checking for measuring systems. In *Proceedings of the International Symposium Leveraging Applications of Formal Methods, Verification and Validation*, volume 8803 of *LNCS*, pages 223–238. Springer, 2014.

[Hep74] Frank H Heppner. Avian flight formations. *Bird-Banding*, 45(2):160–169, 1974.

[HLMP04a] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation*, 2004.

[HLMP04b] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In *International Workshop on Verification, Model Checking, and Abstract Interpretation*, pages 73–84. Springer, 2004.

[HMZ+12] David Henriques, Joao G. Martins, Paolo Zuliani, Andre Platzer, and Edmund M. Clarke. Statistical model checking for markov decision processes. In *Proc. of QEST 2012: the Ninth International Conference on Quantitative Evaluation of Systems*, QEST'12, pages 84–93. IEEE Computer Society, 2012.

[KE95] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. of 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[KJL+16] Kenan Kalajdzic, Cyrille Jégourel, A. Lukina, Ezio Bartocci, Axel Legay, Scott A. Smolka, and Radu Grosu. Feedback Control for Statistical Model Checking of Cyber-Physical Systems. In *Proceedings of the International Symposium Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, LNCS, pages 46–61. Springer, 2016.

[Kre16] Arthur J Krener. Adaptive horizon model predictive control. *arXiv preprint arXiv:1602.08619*, 2016.

[LEH+17] Anna Lukina, Lukas Esterle, Christian Hirsch, Ezio Bartocci, Junxing Yang, Ashish Tiwari, Scott A. Smolka, and Radu Grosu. ARES: Adaptive receding-horizon synthesis of optimal plans. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017*, volume 10206 of *LNCS*, pages 286–302, 2017.

[LS70] PBS Lissaman and Carl A Shollenberger. Formation flight of birds. *Science*, 168(3934):1003–1005, 1970.

[LTSG19] Anna Lukina, Ashish Tiwari, Scott A. Smolka, and Radu Grosu. Distributed adaptive-neighborhood control for stochastic reachability in multi-agent systems. In Chih-Cheng Hung and George A. Papadopoulos, editors, *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 914–921. ACM, 2019. URL `https://doi.org/10.1145/3297280.3297370`.

[MRG03] S. Mannor, R. Y. Rubinstein, and Y. Gat. The cross entropy method for fast policy search. In *ICML*, pages 512–519, 2003.

[Nar90] Kumpati S. Narendra. Adaptive control using neural networks. In *Neural networks for control*, pages 115–142. MIT Press, 1990.

[NB08] Andre Nathan and Valmir C Barbosa. V-like Formations in Flocks of Artificial Birds. *Artificial Life*, 14(2):179–188, 2008.

[NKC16] Lebsework Negash, Sang-Hyeon Kim, and Han-Lim Choi. An unknown-input-observer based approach for cyber attack detection in formation flying UAVs. In *AIAA Infotech*, 2016.

[PDB13] F. Pasqualetti, F. Dorfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Trans. on Automatic Control*, 58(11):2715–2729, 2013.

[PIW+15] Junkil Park, Radoslav Ivanov, James Weimer, Miroslav Pajic, and Insup Lee. Sensor attack detection in the presence of transient faults. In *6th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2015.

[PWB+14] Miroslav Pajic, James Weimer, Nicola Bezzo, Paulo Tabuada, Oleg Sokolsky, Insup Lee, and George J. Pappas. Robustness of attack-resilient state estimators. In *5th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2014.

[Rey87a] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *ACM Siggraph Computer Graphics*, volume 21, pages 25–34. ACM, 1987.

[Rey87b] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Computer Graphics*, 21(4):25–34, 1987.

[RN10] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 3rd edition, 2010.

[SPH02] P. Seiler, A. Pant, and K. Hedrick. Analysis of bird formations. In *Proceedings of the Conference on Decision and Control*, volume 1, pages 118–123 vol.1. IEEE, 2002.

[SS12a] F. Stulp and O. Sigaud. Path integral policy improvement with covariance matrix adaptation. *arXiv preprint arXiv:1206.4621*, 2012. URL `http://arxiv.org/abs/1206.4621`.

[SS12b] F. Stulp and O. Sigaud. Policy improvement methods: Between black-box optimization and episodic reinforcement learning, 2012. URL `http://hal.upmc.fr/hal-00738463/`.

[SSP+17] Kelsey Saulnier, David Saldana, Amanda Prorok, George J Pappas, and Vijay Kumar. Resilient flocking for mobile robot teams. *IEEE Robotics and Automation Letters*, 2(2):1039–1046, 2017.

[SW11] F. Stonedahl and U. Wilensky. Finding forms of flocking: Evolutionary search in ABM parameter-spaces. In *Multi-Agent-Based Simulation XI*, pages 61–75. Springer, 2011.

[TSE+17] Ashish Tiwari, Scott A. Smolka, Lukas Esterle, Anna Lukina, Junxing Yang, and Radu Grosu. Attacking the v: On the resiliency of adaptive-horizon mpc. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 446–462. Springer International Publishing, Cham, 2017. ISBN 978-3-319-68167-2.

[VPT+12] G. Verfaillie, C. Pralet, F. Teichteil, G. Infantes, and C. Lesire. Synthesis of plans or policies for controlling dynamic systems. *AerospaceLab*, (4):p. 1–12, 2012.

[WMC+01] Henri Weimerskirch, Julien Martin, Yannick Clerquin, Peggy Alexandre, and Sarka Jiraskova. Energy Saving in Flight Formation. *Nature*, 413(6857):697–698, 2001.

[YGST16] Junxing Yang, Radu Grosu, Scott A Smolka, and Ashish Tiwari. Love Thy Neighbor: V-Formation as a Problem of Model Predictive Control. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 59. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[YZS17] D. Ye, J. Zhang, and Z. Sun. Extended state observer-based finite-time controller design for coupled spacecraft formation with actuator saturation. *Advances in Mechanical Engineering*, 9 (4):1–13, 2017.

[ZL13] Jingyuan Zhan and Xiang Li. Flocking of multi-agent systems via model predictive control based on position-only measurements. *IEEE Trans. Industrial Informatics*, 9(1):377–385, 2013.