ARES: Adaptive Receding-Horizon Synthesis of Optimal Plans

Anna Lukina[1], Lukas Esterle[1], Christian Hirsch[1], Ezio Bartocci[1],
Junxing Yang[2], Ashish Tiwari[3], Scott A. Smolka[2], and Radu Grosu[1,2]

[1] Cyber-Physical Systems Group, Technische Universität Wien, Austria
[2] Department of Computer Science, Stony Brook University, USA
[3] SRI International, USA

Abstract. We introduce ARES, an efficient approximation algorithm for generating optimal plans (action sequences) that take an initial state of a Markov Decision Process (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, 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 also assess the effectiveness of ARES by statistically evaluating its rate of success in generating optimal plans. The ARES algorithm resulted from our desire to clarify if flying in V-formation is a flocking policy that optimizes energy conservation, clear view, and velocity alignment. That is, we were interested to see if one could find 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 also be easily customized into a model-predictive controller (MPC) with an adaptive receding horizon and 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.

1 Introduction

Flocking or swarming in groups of social animals (birds, fish, ants, bees, etc.) that results in a particular global formation is an emergent collective behavior that continues to fascinate researchers [1, 8]. One would like to know if such a formation serves a higher purpose, and, if so, what that purpose is.

One well-studied flight-formation behavior is *V-formation*. Most of the work in this area has concentrated on devising simple dynamical rules that, when followed by each bird, eventually stabilize the flock to the desired V-formation [12, 13, 26]. This approach, however, does not shed very much light on the overall purpose of this emergent behavior.

In previous work [35,36], we hypothesized that flying in V-formation is nothing but an optimal policy for a flocking-based Markov Decision Process (MDP) $\mathcal{M}$. States of $\mathcal{M}$, at discrete time t, are of the form $(\boldsymbol{x}_i(t), \boldsymbol{v}_i(t))$, $1 \leqslant i \leqslant N$, where $\boldsymbol{x}_i(t)$ and $\boldsymbol{v}_i(t)$ are N-vectors (for an N-bird flock) of 2-dimensional positions and velocities, respectively. $\mathcal{M}$'s transition relation, shown here for bird i is simply and generically given by

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

where $\boldsymbol{a}_i(t)$ is an action, a 2-dimensional acceleration in this case, that bird i can take at time t. $\mathcal{M}$'s cost function reflects the energy-conservation, velocity-alignment and clear-view benefits enjoyed by a state of $\mathcal{M}$ (see Section 2).

In this paper, we not only confirm this hypothesis, but we also devise a very 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. In fact, ARES implicitly defines an *optimal, online-policy, synthesis algorithm* that could be used in practice if plan generation can be performed in real-time.

ARES makes repeated use of Particle Swarm Optimization (PSO) [22] 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 an 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.

A simple solution to this problem would be to use a short horizon, typically of size two or three. This is indeed the current practice in Model Predictive Control (MPC) [14]. This approach, however, has at least three major drawbacks. First, and most importantly, it does not guarantee convergence and optimality, as one may oscillate or become stuck in a local optimum. Second, in some of the steps, the window size is unnecessarily large thereby negatively impacting performance. Third, in other steps, the window size may be not large enough to guide the optimizer out of a local minimum (see Fig. 1 (left)). One would therefore like to find the proper window size adaptively, but the question is how one can do it.

Inspired by Importance Splitting (IS), a sequential Monte-Carlo technique for estimating the probability of rare events, we introduce the notion of a *level-based horizon* (see Fig. 1 (right)). Level ℓ_0 is the cost of the initial state, and level ℓ_m is the desired threshold. By using a state function, asymptotically converging to the desired threshold, we can determine a sequence of levels, ensuring convergence of ARES towards the desired optimal state(s) having a cost below $\ell_m = \varphi$.

The levels serve two purposes. First, they implicitly define a Lyapunov function, which guarantees convergence. If desired, this function can be explicitly

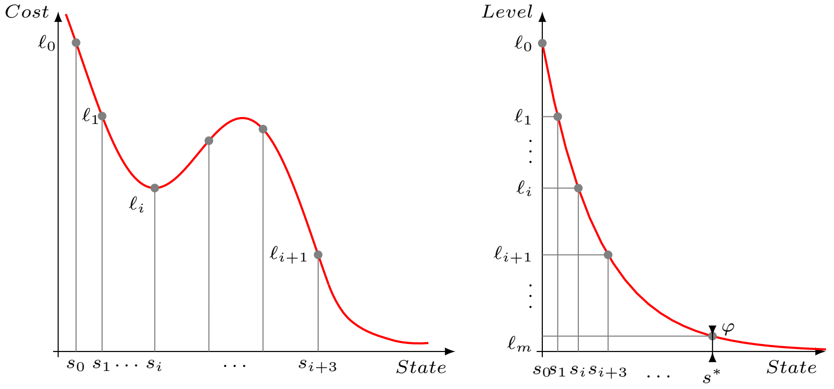

Fig. 1. Left: If state s_0 has cost ℓ_0, and its successor-state s_1 has cost less than ℓ_1, then a horizon of length 1 is appropriate. However, if s_i has a local-minimum cost ℓ_i, one has to pass over the cost ridge in order to reach level ℓ_{i+1}, and therefore ARES has to adaptively increase the 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 asympthotically converging (Lyapunov) function (where the number m is empirically determined), one obtains on the vertical cost-axis the levels required for ARES to converge.

generated for all states, up to some topological equivalence. Second, the levels help PSO overcome local minima (see Fig. 1 (left)). If reaching a next level requires PSO to temporarily pass over a state-cost ridge, ARES incrementally increases the size of the horizon, up to a maximum length.

Another idea imported from IS is to maintain n clones of the initial state at a time, and run PSO on each of them (see Fig. 3). This allows us to call PSO for each clone and desired horizon, with a very small number of particles per clone. Clones that do not reach the next level are discarded, and the successful ones are resampled. The number of particles is increased 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. At the last level, we choose the optimal particle (a V-formation in case of flocking) and traverse its predecessors to find a plan.

We asses the rate of success in generating optimal plans in form of an (ε, δ)-approximation scheme, for a desired error margin ε, and confidence ratio $1-\delta$. Moreover, we can use the state-action pairs generated during the assessment (and possibly some additional new plans) to construct an explicit (tabled) optimal policy, modulo some topological equivalence. Given enough memory, one can use this policy in real time, as it only requires a table look-up.

To experimentally validate our approach, we have applied ARES to the problem of V-formation in bird flocking (with a deterministic MDP). The cost function to be optimized is defined as a weighted sum of the (flock-wide) clear-view, velocity-alignment, and upwash-benefit metrics. Clear view and velocity alignment are more or less obvious goals. Upwash optimizes energy savings. By flapping its wings, a bird generates a trailing upwash region off its wing tips; by using this upwash, a bird flying in this region (left or right) can save energy.

Note that by requiring that at most one bird does not feel its effect, upwash can be used to define an analog version of a connected graph.

We ran ARES on 8,000 initial states chosen uniformly and at random, such that they are packed closely enough to feel upwash, but not too close to collide. We succeeded to generate a V-formation 95% of the time, with an error margin of 0.05 and a confidence ratio of 0.99. These error margin and confidence ratio dramatically improve if we consider all generated states and the fact that each state within a plan is independent from the states in all other plans.

The rest of this paper is organized as follows. Section 2 reviews our work on bird flocking and V-formation, and defines the manner in which we measure the cost of a flock (formation). Section 3 revisits the swarm optimization algorithm used in this paper, and Section 4 examines the main characteristics of importance splitting. Section 5 states the definition of the problem we are trying to solve. Section 6 introduces ARES, our adaptive receding-horizon synthesis algorithm for optimal plans, and discusses how we can extend this algorithm to explicitly generate policies. Section 7 measures the efficiency of ARES in terms of an (ε, δ)-approximation scheme. Section 8 compares our algorithm to related work, and Section 9 draws our conclusions and discusses future work.

2 V-Formation MDP

We represent a flock of birds as a dynamically evolving system. Every bird in our model [17] moves in 2-dimensional space performing acceleration actions determined by a global controller. Let $\boldsymbol{x}_i(t), \boldsymbol{v}_i(t)$ and $\boldsymbol{a}_i(t)$ be 2-dimensional vectors of positions, velocities, and accelerations, respectively, of bird i at time t, where $i \in \{1, \ldots, b\}$, for a fixed b. The discrete-time behavior of bird i is then

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

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[4] based on the following metrics inspired by [35]:

- *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. 2 (left) illustrates the calculation of the clear-view metric. The optimal value in a V-formation is $CV^* = 0$, as all birds have a clear view.
- *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. 2 (middle) depicts the values of VM in a velocity-unmatched flock.

[4] A classic MDP [28] is obtained by adding sensor/actuator or wind-gust noise.

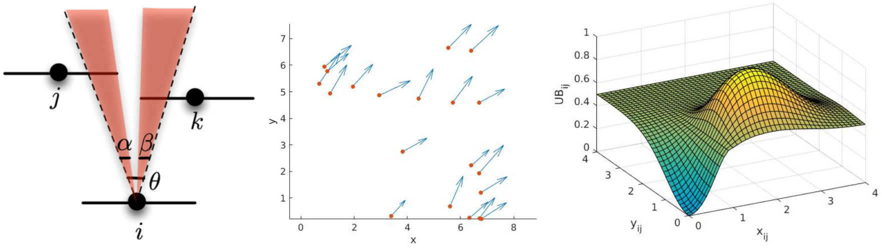

Fig. 2. Illustration of the clear view (CV), velocity matching (VM), and upwash benefit (UB) metrics. Left: Bird i's view is partially blocked by birds j and k. Hence, its clear view is $CV = (\alpha + \beta)/\theta$. Middle: A flock and its unaligned bird velocities results in a velocity-matching metric $VM = 6.2805$. 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.

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. 2 (right) for the right wing. The maximum upwash a bird can obtain has an upper bound of 1. For bird i with UB_i, we use $1 - UB_i$ as its upwash-benefit metric, because the optimization algorithm performs minimization of the fitness metrics. The optimal value 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 $\boldsymbol{c}(t) = \{\boldsymbol{c}_i(t)\}_{i=1}^{b} = \{\boldsymbol{x}_i(t), \boldsymbol{v}_i(t)\}_{i=1}^{b}$ 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:

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

where h is the receding prediction horizon (RPH), $\boldsymbol{a}^h(t)$ 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 have

$$\boldsymbol{c}^h_{\boldsymbol{a}}(t) = \{\boldsymbol{x}^h_{\boldsymbol{a}}(t), \boldsymbol{v}^h_{\boldsymbol{a}}(t)\} = \{\boldsymbol{x}(t) + \sum_{\tau=1}^{h(t)} \boldsymbol{v}(t+\tau), \boldsymbol{v}(t) + \sum_{\tau=1}^{h(t)} \boldsymbol{a}^\tau(t)\}, \qquad (3)$$

where $\boldsymbol{a}^\tau(t)$ is the τth acceleration of $\boldsymbol{a}^h(t)$. A novelty of this paper is that, as described in Section 6, 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. The main goal of ARES is to compute the sequence of acceleration actions that lead the flock from a random initial configuration towards a controlled V-formation characterized by optimal fitness in order to conserve energy during flight including optimal combination of a clear visual field along with visibility of lateral neighbors. Similar to the centralized version of the approach given in [35], ARES performs a single flock-wide minimization of J at each time-step t to obtain an optimal plan of length h of acceleration actions:

$$\textbf{opt-}\boldsymbol{a}^h(t) = \{\textbf{opt-}\boldsymbol{a}_i^h(t)\}_{i=1}^b = \operatorname*{arg\,min}_{\boldsymbol{a}^h(t)} J(\boldsymbol{c}(t), \boldsymbol{a}^h(t), h). \tag{4}$$

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 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. In the following sections, we demonstrate how to generate optimal plans taking the initial state to a stable state with optimal fitness.

3 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 [22]. 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.

ARES 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:

$$\begin{aligned}\mathbf{v}^j(t+1) = \omega \cdot \mathbf{v}^j(t) + y_1 \cdot \mathbf{u_1}(t+1) \otimes (\mathbf{x}_P^j(t) - \mathbf{x}^j(t))\\ + y_2 \cdot \mathbf{u_2}(t+1) \otimes (\mathbf{x}_G^j(t) - \mathbf{x}^j(t)),\end{aligned} \tag{5}$$

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.

4 Importance Splitting

Importance Splitting (IS) is a sequential Monte-Carlo approximation technique for estimating the probability of rare events in a Markov process [7]. The algorithm uses a sequence $S_0, S_1, S_2, \ldots, S_m$ of sets of states (of increasing "importance") such that S_0 is the set of initial states and S_m is the set of states defining the rare event. The probability p, computed as $\mathbf{P}(S_m \mid S_0)$ of reaching S_m from the initial set of states S_0, is assumed to be extremely low (thus, a rare event), and one desires to estimate this probability [16]. Random sampling approaches, such as the additive-error approximation algorithm described in Section 7, are bound to fail (are intractable) in this case, as they would require an enormous number of samples to estimate p with low-variance.

Importance splitting is a way of decomposing the estimation of p. In IS, the sequence $S_0, S_1, \ldots$ of sets of states is defined so that the conditional probabilities $p_i = \mathbf{P}(S_i \mid S_{i-1})$ of going from one level, S_{i-1}, to the next one, S_i, are considerably larger than p, and essentially equal to one another. The resulting probability of the rare event is then calculated as the product $p = \prod_{i=1}^{k} p_i$ of the intermediate probabilities. The levels can be defined adaptively [23].

To estimate p_i, IS uses a swarm of particles of size N, with a given initial distribution over the states of the stochastic process. During stage i of the algorithm, each particle starts at level S_{i-1} and traverses the states of the stochastic process, checking if it reaches S_i. If, at the end of the stage, the particle fails to reach S_i, the particle is discarded. Suppose that K_i particles survive. In this case, $p_i = K_i/N$. Before starting the next stage, the surviving particles are resampled, such that IS once again has N particles. Whereas IS is used for estimating probability of a rare event in a Markov process, we use it here for synthesizing a plan for a *controllable* Markov process, by combining it with ideas from controller synthesis (receding-horizon control) and nonlinear optimization (PSO).

5 Problem Definition

Definition 1. *A* ***Markov decision process (MDP)*** $\mathcal{M}$ *is a sequential decision problem that consists of a set of states S (with an initial state s_0), a set of* *actions A, a transition model T, and a cost function J. An MDP is* ***deterministic*** *if for each state and action, $T: S \times A \rightarrow S$ specifies a unique state.*

Definition 2. *The* ***optimal plan synthesis problem*** *for an MDP $\mathcal{M}$, an arbitrary initial state s_0 of $\mathcal{M}$, and a threshold φ is to synthesize a sequence of actions $\boldsymbol{a}^i$ of length $1 \leqslant i \leqslant m$ taking s_0 to a state s^* such that cost $J(s^*) \leqslant \varphi$.*

Section 6 presents our adaptive receding-horizon synthesis algorithm (ARES) for the optimal plan synthesis problem. In our flocking example (Section 2), ARES is used to synthesize a sequence of acceleration-actions bringing an arbitrary bird flock s_0 to an optimal state of V-formation s^*. We assume that we can easily extend such an optimal plan to maintain the cost of successor states below φ ad infinitum (optimal stability).

6 The ARES Algorithm for Plan Synthesis

As mentioned in Section 1, one could in principle solve the optimization problem defined in Section 5 by calling the PSO only once, with a horizon h in $\mathcal{M}$ equaling the maximum length m allowed for a plan. This approach, however, tends to explode the search space, and is therefore 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.

Inspired by IS (see Fig. 1 (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. 1 (left)). If reaching a next level

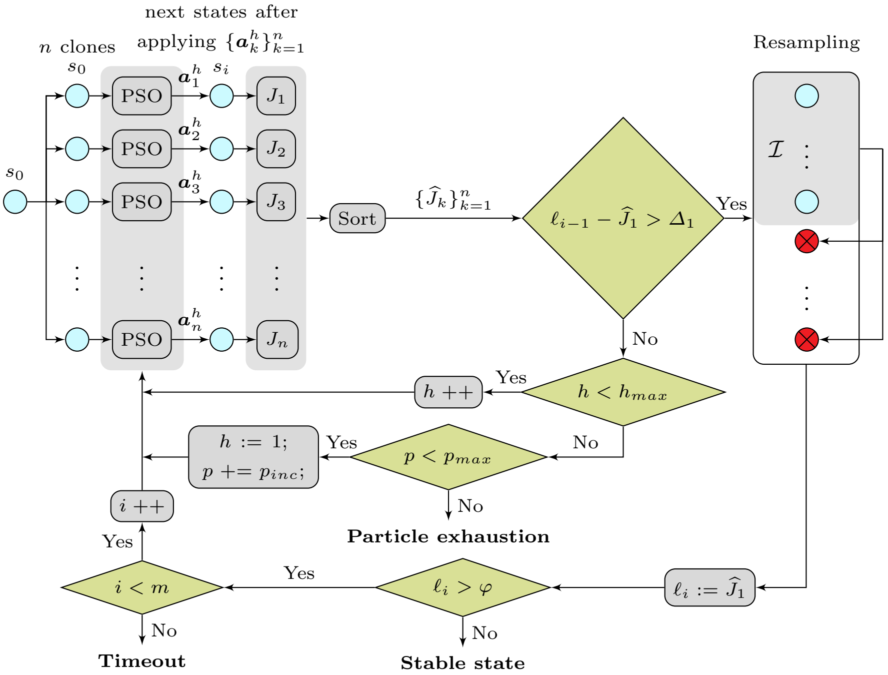

Fig. 3. Graphical representation of ARES.

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 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.

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
```

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
```

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 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.*

Proof (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.

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.* As a consequence, it might 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.

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
```

7 Experimental Results

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:

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 at 0.0282 with a standard deviation of 0.1654. We

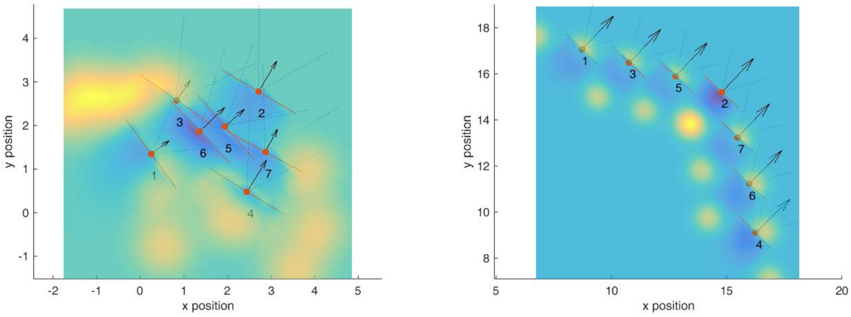

Fig. 4. Left: Example of an arbitrary initial configuration of 7 birds. Right: The V-formation obtained by applying the plan generated by ARES. In the figures, we show the wings of the birds, bird orientations, bird speeds (as scaled arrows), upwash regions in yellow, and downwash regions in dark blue.

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

	Successful				Total			
No. Experiments	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

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. 5 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. 4 (left)) reaches V-formation (Fig. 4 (right)) in around 1 minute. The execution time rises with the number of birds as shown in Table 2.

In Fig. 5, we illustrate for how many experiments the algorithm had to increase RPH h (Fig. 5 (middle)) and the number of particles used by PSO p (Fig. 5 (right)) to improve time and space exploration, respectively.

After achieving such a high success rate of ARES for an arbitrary initial configuration, we would like to demonstrate that the number of experiments

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

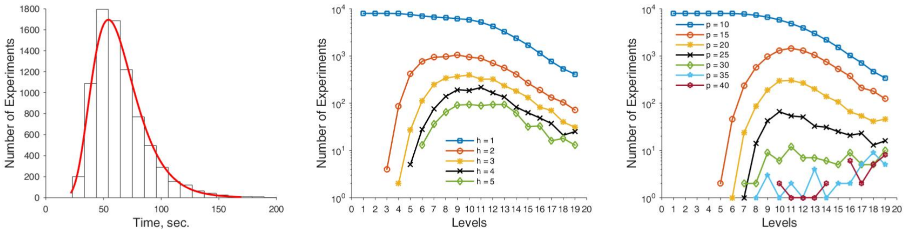

Fig. 5. Left: Distribution of execution times for 8,000 runs. Middle: Statistics of increasing RPH h. Right: Particles of PSO p for 8,000 experiments

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 [17]. 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 [20]. 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 $\varepsilon = 0.019$.

8 Related Work

Organized flight in flocks of birds can be categorized in *cluster flocking* and *line formation* [19]. 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 [27] 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 lift by birds flying in front, trailing birds are able to conserve energy [10,24,34]. Second, in a staggered formation, all birds have a clear view in front as well as a view on their neighbors [1]. While there has been quite some effort to keep a certain formation for multiple entities when traveling together [11, 15, 30], only little work deals with a task of achieving this extremely important formation from a random starting configuration [6]. The convergence of bird flocking into V-formation has been also analyzed with the use of combinatorial techniques [8].

Compared to previous work, in [5] this question is addressed without using any behavioral rules but as problem of *optimal control.* In [35] 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 [33]. The most popular and natural technique is Dynamic Programming (DP) [4] that 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 the possible states of the system and might suffer from state-space explosion in case of environmental uncertainties, approximate algorithms [2, 3, 18, 25, 31, 32] take into account only the paths leading to desired target. One of the most efficient ones is Particle Swarm Optimization (PSO) [22] that has been adopted for finding the next best step of MPC in [35]. 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 proved to be efficient in tackling the question of control for linear stochastic systems [9], in particular, Importance Splitting (IS) [23]. 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.

9 Conclusion and Future Work

In this paper, we have presented ARES, a very general adaptive, receding-horizon synthesis algorithm for MDP-based optimal plans. Additionally, ARES can be readily converted into a model-predictive controller with an adaptive receding horizon and statistical guarantees of convergence. We have also conducted a very thorough performance analysis of ARES based on the problem of V-formation in a flock of birds. For flocks of 7 birds, ARES is able to generate an optimal plan leading to a V-formation in 95% of the 8,000 random initial configurations we considered, with an average execution time of only 63 seconds per plan.

The execution time of the ARES algorithm can be even further improved in a number of ways. First, we currently do not parallelize our implementation of the PSO algorithm. Recent work [21, 29, 37] has shown how Graphic Processing Units (GPUs) are very efficient at accelerating PSO computation. Modern GPUs, by providing thousands of cores, are well-suited for implementing PSO as they enable execution of a very large number of particles in parallel, which can improve accuracy of the optimization procedure. Likewise, the calculation of the fitness function can also be run in parallel. The parallelization of these steps should significantly speed up our simulations.

Second, we are currently using a static approach to decide how to increase our prediction horizon and the number of particles used in PSO. Specifically, we first increase the prediction horizon from 1 to 5, while keeping the number of particles unchanged at 10; if this fails to find a solution with fitness $\widehat{J_1}$ satisfying $\ell_{i-1} - \widehat{J_1} > \varDelta_1$, we then increase the number of particles by 5. Based on our results, we speculate that in the initial stages, increasing the prediction horizon is more beneficial (leading rapidly to the appearance of cost-effective formations), whereas in the later stages, increasing the number of particles is more helpful. As future work, we will use machine-learning approaches to decide on the prediction horizon and the number of particles deployed at runtime given the current level and state of the MDP.

Third, in our approach, we always calculate the number of clones for resampling based on the current state. An alternative approach would rely on statistics built up over multiple levels in combination with the rank in the sorted list to determine whether a configuration should be used for resampling or not.

Finally, we are currently using our approach to generate plans for a flock to go from an initial configuration to a final V-formation. Our eventual goal is to achieve formation flight for a robotic swarm of (bird-like) drones. A real-world example is parcel-delivering drones that follow the same route to their destinations. Letting them fly together for a while could save energy and increase flight time. To achieve this goal, we first need to investigate the wind dynamics of multi-rotor drones. Then, the fitness function needs to be adopted to the new wind dynamics. Lastly, a decentralized approach of this method needs to be implemented and tested on the drone firmware.

Acknowledgments. The first author and the last author would like to thank Jan Kretinsky for very valuable feedback. This work was partially supported by the Doctoral Program Logical Methods in Computer Science funded by the Austrian Science Fund (FWF), and the Austrian National Research Network (nr. S 11405-N23 and S 11412-N23) SHiNE funded by FWF.

References

1. Bajec, I.L., Heppner, F.H.: Organized flight in birds. Animal Behaviour 78(4), 777–789 (2009)
2. Bartocci, E., Bortolussi, L., Brázdil, T., Milios, D., Sanguinetti, G.: 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. vol. 9826, pp. 244–259 (2016)
3. Baxter, J., Bartlett, P.L., Weaver, L.: Experiments with infinite-horizon, policy-gradient estimation. J. Artif. Int. Res. 15(1), 351–381 (2011)
4. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
5. Camacho, E.F., Alba, C.B.: Model Predictive Control. Advanced Textbooks in Control and Signal Processing, Springer (2007)
6. Cattivelli, F.S., Sayed, A.H.: Modeling bird flight formations using diffusion adaptation. IEEE Transactions on Signal Processing 59(5), 2038–2051 (2011)
7. Cérou, F., Guyader, A.: Adaptive multilevel splitting for rare event analysis. Stochastic Analysis and Applications 25, 417–443 (2007)
8. Chazelle, B.: The Convergence of Bird Flocking. Journal of the ACM 61(4), 21:1–21:35 (2014)
9. Chen, Y., Wu, B., Lai, T.L.: Fast Particle Filters and Their Applications to Adaptive Control in Change-Point ARX Models and Robotics. INTECH Open Access Publisher (2009)
10. Cutts, C., Speakman, J.: Energy savings in formation flight of pink-footed geese. Journal of Experimental Biology 189(1), 251–261 (1994)
11. Dang, A.D., Horn, J.: Formation control of autonomous robots following desired formation during tracking a moving target. In: Proceedings of the International Conference on Cybernetics. pp. 160–165. IEEE (2015)
12. Dimock, G., Selig, M.: The Aerodynamic Benefits of Self-Organization in Bird Flocks. Urbana 51, 1–9 (2003)
13. Flake, G.W.: The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation. MIT Press (1998)
14. García, C.E., Prett, D.M., Morari, M.: Model predictive control: Theory and practice – a survey. Automatica 25(3), 335–348 (1989)
15. Gennaro, M.C.D., Iannelli, L., Vasca, F.: Formation Control and Collision Avoidance in Mobile Agent Systems. In: Proceedings of the International Symposium on Control and Automation Intelligent Control. pp. 796–801. IEEE (2005)
16. Glasserman, P., Heidelberger, P., Shahabuddin, P., Zajic, T.: Multilevel Splitting for Estimating Rare Event Probabilities. Operations Research 47(4), 585–600 (1999)
17. Grosu, R., Peled, D., Ramakrishnan, C.R., Smolka, S.A., Stoller, S.D., Yang, J.: Using statistical model checking for measuring systems. In: Proceedings of the International Symposium Leveraging Applications of Formal Methods, Verification and Validation. LNCS, vol. 8803, pp. 223–238. Springer (2014)
18. Henriques, D., Martins, J.G., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical model checking for markov decision processes. In: Proc. of QEST 2012: the Ninth International Conference on Quantitative Evaluation of Systems. pp. 84–93. QEST'12, IEEE Computer Society (2012)
19. Heppner, F.H.: Avian flight formations. Bird-Banding 45(2), 160–169 (1974)
20. Hérault, T., Lassaigne, R., Magniette, F., Peyronnet, S.: Approximate probabilistic model checking. In: Proceedings of the International Conference on Verification, Model Checking, and Abstract Interpretation (2004)

21. Hung, Y., Wang, W.: Accelerating parallel particle swarm optimization via gpu. Optimization Methods and Software 27(1), 33–51 (2012)
22. James, K., Russell, E.: Particle swarm optimization. In: Proceedings of 1995 IEEE International Conference on Neural Networks. pp. 1942–1948 (1995)
23. Kalajdzic, K., Jégourel, C., Lukina, A., Bartocci, E., Legay, A., Smolka, S.A., Grosu, R.: 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. pp. 46–61. LNCS, Springer (2016)
24. Lissaman, P., Shollenberger, C.A.: Formation flight of birds. Science 168(3934), 1003–1005 (1970)
25. Mannor, S., Rubinstein, R.Y., Gat, Y.: The cross entropy method for fast policy search. In: ICML. pp. 512–519 (2003)
26. Nathan, A., Barbosa, V.C.: V-like Formations in Flocks of Artificial Birds. Artificial Life 14(2), 179–188 (2008)
27. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Computer Graphics 21(4), 25–34 (1987)
28. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice-Hall, 3rd edn. (2010)
29. Rymut, B., Kwolek, B., Krzeszowski, T.: GPU-Accelerated Human Motion Tracking Using Particle Filter Combined with PSO. In: Proceedings. of the International Conference on Advanced Concepts for Intelligent Vision Systems. LNCS, vol. 8192, pp. 426–437. Springer (2013)
30. Seiler, P., Pant, A., Hedrick, K.: Analysis of bird formations. In: Proceedings of the Conference on Decision and Control. vol. 1, pp. 118–123 vol.1. IEEE (2002)
31. Stulp, F., Sigaud, O.: Path integral policy improvement with covariance matrix adaptation. arXiv preprint arXiv:1206.4621 (2012), http://arxiv.org/abs/1206.4621
32. Stulp, F., Sigaud, O.: Policy improvement methods: Between black-box optimization and episodic reinforcement learning (2012), http://hal.upmc.fr/hal-00738463/
33. Verfaillie, G., Pralet, C., Teichteil, F., Infantes, G., Lesire, C.: Synthesis of plans or policies for controlling dynamic systems. AerospaceLab (4), p. 1–12 (2012)
34. Weimerskirch, H., Martin, J., Clerquin, Y., Alexandre, P., Jiraskova, S.: Energy Saving in Flight Formation. Nature 413(6857), 697–698 (2001)
35. Yang, J., Grosu, R., Smolka, S.A., Tiwari, A.: Love Thy Neighbor: V-Formation as a Problem of Model Predictive Control. In: LIPIcs-Leibniz International Proceedings in Informatics. vol. 59. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
36. Yang, J., Grosu, R., Smolka, S.A., Tiwari, A.: V-Formation as Optimal Control. In: Proceedings of the Biological Distributed Algorithms Workshop 2016 (2016)
37. Zhou, Y., Tan, Y.: GPU-based Parallel Particle Swarm Optimization. In: Proceedings of the Congress on Evolutionary Computation. pp. 1493–1500. IEEE (2009)