GOAL-DIRECTED GRAPH NEURAL CELLULAR AUTOMATA FOR SWARM NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Cellular automata (CA) are a group of computational models in which each cell repeatedly updates its state through a local rule that depends on its own state and the states of its neighbors. After sufficient iterations, CA are able to exhibit emergent global complexity. Graph cellular automata (GCA) are a generalized version of CA that live and operate on an arbitrary graph rather than a lattice structure. The advantage of GCA is that they allow for dynamic edges between graph nodes instead of fixed distances between cells on a lattice, a feature with strong implications for control-oriented dynamical systems. Prior work used graph neural networks (GNNs) to learn the transition rules of graph cellular automata for flocking imitation. Our central contribution is to extend this framework beyond flocking to goal-directed swarm navigation in both two- and three-dimensional environments. First, we train GNNs to imitate an expert flocking controller that guides swarms from randomized initial configurations through a fixed sequence of waypoints. Then, we train GNNs to imitate the same controller under dynamically updated waypoints sampled throughout the spatial domain. We further evaluate robustness to the removal of up to half of the agents from the middle of the flock during execution, which changes both swarm size and graph topology on the fly. Together, these experiments broaden the role of learned GCA from reproducing dynamics to enabling task-directed collective behavior, highlighting their potential as learned policies for decentralized systems.

1 INTRODUCTION

Cellular automata (CA) are discrete computational models in which a grid of cells evolves through the repeated application of a local transition rule, with each cell updating its state based only on its own state and the states of its neighbors (von Neumann & Burks, 1966). Even simple local rules can give rise to remarkably complex global patterns when applied repeatedly, and decades after their introduction, CA remain an active subject of study in dynamical systems theory (Wolfram, 1983; Bhattacharjee et al., 2020). Graph cellular automata (GCA) extend this framework by replacing the regular lattice with an arbitrary graph (Marr & Hütt, 2009), which makes them a natural model for control-oriented systems such as swarms. A lattice fixes the distance between neighboring cells, whereas the agents of a swarm move through continuous space at varying distances from one another, so the graph connecting each agent to its current neighbors changes over time.

Because the global behavior of a CA only emerges from many repeated local updates, designing a transition rule that produces a desired collective behavior is difficult, and deep learning offers a way to learn such rules instead (Mordvintsev et al., 2020). The inspiration for this work comes from Grattarola et al. (2021), who used graph neural networks (GNNs) to learn GCA transition rules, a setting they refer to as graph neural cellular automata (GNCA). Among other systems, they trained a GNCA to reproduce Boids (Reynolds, 1987), a model of bird flocking.

In this work, we extend GNCA from reproducing flocking, which keeps a swarm together without steering it toward any particular location, to learning goal-directed swarm navigation in two and three dimensions. To do so, we train GNCA to imitate a goal-conditioned version of Boids (Figure 1) and evaluate them on two navigation tasks and under the removal of agents during execution. Our contributions are as follows:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

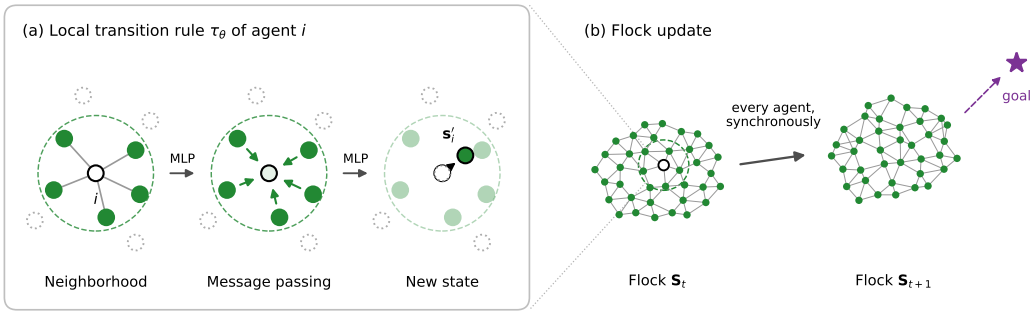

Figure 1: Overview of our goal-directed GNCA. (a) A simplified view of the local transition rule τ_θ of agent i. The state of the agent is processed by an MLP, combined with the messages of the neighbors within its perception radius through message passing, and mapped by a second MLP to its new state $\mathbf{s}_i'$ (the full rule, which also processes relative positions, is given in Section 3.2). Agents outside the perception radius (dotted) do not interact with agent i. In dynamic-waypoint navigation, the state of each agent also includes its displacement to the active goal. (b) Every agent applies the same rule synchronously, updating the whole flock from $\mathbf{S}_t$ to $\mathbf{S}_{t+1}$ as it moves toward the goal, and the neighborhood graph changes as the agents move.

- **Fixed-Waypoint Navigation.** GNCA that guide a swarm through the same ordered sequence of waypoints without receiving any goal input, so the route is encoded entirely in the learned transition rule. In both two and three dimensions, the GNCA complete nearly every rollout while holding a flock as compact as that of the expert.
- **Dynamic-Waypoint Navigation.** GNCA conditioned on each agent's displacement to the active goal, which steer the swarm toward goals that are sampled at random and revealed online. In both two and three dimensions, the GNCA complete every rollout while holding a flock close in compactness to that of the expert.
- **Robustness to Agent Removal.** An evaluation showing that the GNCA maintain task completion when a slab containing up to half of the swarm is cut out of the flock during execution, even though they were never trained on agent removal.

Together, these results demonstrate that graph cellular automata can learn decentralized rules for goal-directed navigation in fully continuous state and action spaces.

2 Related Work

Learning Cellular Automata with Neural Networks Neural networks have long been used to learn the transition rules of cellular automata. Wulff & Hertz (1992) trained a small network to imitate one- and two-dimensional binary CA, and Gilpin (2019) showed that any CA rule can be represented by a convolutional neural network (CNN) and learned end-to-end with gradient descent. Mordvintsev et al. (2020) introduced neural cellular automata (NCA), which use a CNN to learn a transition rule over continuous, multi-dimensional cell states, and trained them to grow and regenerate a target image. NCA have since been applied to tasks such as classifying MNIST digits through local agreement between cells (Randazzo et al., 2020). All of these works operate on regular lattices, where each cell has a fixed position and a fixed set of neighbors.

Goal-Directed Cellular Automata Most NCA converge to a single behavior that is fixed during training. Sudhakaran et al. (2022) proposed goal-guided NCA (GoalNCA), which steer a lattice NCA by adding an encoded goal to the hidden state of every living cell at each step. The goals are drawn from a small discrete set of commands, such as target images to morph into or movement directions (stay, up, down, left, and right) for a pattern on a fixed grid. Because the cells themselves remain in place, movement in GoalNCA consists of a pattern shifting across the grid, and goals specify directions rather than locations to reach.

Learned Swarm Controllers and Graph Cellular Automata GNNs have been used to learn decentralized flocking controllers by imitating a centralized expert that aligns agent velocities while avoiding collisions (Tolstaya et al., 2020). These controllers do not navigate toward goal locations, and the expert they imitate relies on the positions and velocities of all agents rather than on local information alone. Graph cellular automata (Marr & Hütt, 2009), in contrast, are defined by transition rules that are local by construction. Grattarola et al. (2021) learned GCA transition rules with GNNs, including a GNCA that imitates the decentralized Boids flocking model (Reynolds, 1987). Gala et al. (2024) later made GNCA equivariant to rotations and translations and applied them to Boids flocking in three dimensions.

Building on the goal conditioning of GoalNCA and the swarm GNCA of Grattarola et al. (2021), we ask whether a learned cellular automaton can navigate a swarm to a sequence of goal locations in continuous two- or three-dimensional space while keeping it together as a flock. Section 3 formalizes this setting, beginning with the definition of a GCA.

3 METHODS

3.1 GRAPH CELLULAR AUTOMATA

We adopt the graph cellular automaton (GCA) formulation of Grattarola et al. (2021), in which a GCA is the tuple $(\mathcal{G}, \mathcal{S}, \mathcal{N}, \tau)$. Here, $\mathcal{G} = (V, E)$ is a graph whose $n = |V|$ nodes act as cells and whose edges $E \subseteq V \times V$ determine which cells interact, $\mathcal{S}$ is the set of admissible cell states, $\mathcal{N}(i) = \{j \mid (j, i) \in E\}$ is the neighborhood of cell i, and τ is a local transition rule. Each cell i holds a state $\mathbf{s}_i \in \mathcal{S}$, which τ updates using only the cell's own state and the states of its neighbors:

$$\tau(\mathbf{s}_i) : \{\mathbf{s}_i\} \cup \{\mathbf{s}_j \mid j \in \mathcal{N}(i)\} \mapsto \mathbf{s}'_i. \tag{1}$$

Collecting all cell states into $\mathbf{S} \in \mathcal{S}^n$, one step of the automaton applies τ synchronously to every cell, which we write as $\mathbf{S}' = \tau(\mathbf{S})$.

3.2 GRAPH NEURAL NETWORKS AS TRANSITION RULES

Graph neural networks (GNNs) (Gori et al., 2005; Scarselli et al., 2009) operate on graph-structured data through message passing (Gilmer et al., 2017; Battaglia et al., 2018), in which each node aggregates messages computed from its neighbors and combines the result with its own representation. Given node features $\mathbf{h}_i$, a message-passing layer computes

$$\mathbf{h}'_i = \gamma\Big(\mathbf{h}_i, \bigoplus_{j \in \mathcal{N}(i)} \phi\left(\mathbf{h}_i, \mathbf{h}_j\right)\Big), \tag{2}$$

where ϕ is a message function, $\bigoplus$ is a permutation-invariant aggregation, and γ is an update function. When ϕ and γ are differentiable, the layer can be trained end-to-end with gradient descent.

Equation 2 shares two defining properties with the GCA transition rule in Equation 1. The output for node i depends only on $\mathbf{h}_i$ and its neighbors, and the same parameters are applied at every node. A GNCA (Grattarola et al., 2021) exploits this correspondence by implementing τ with a GNN. We denote its transition rule by τ_θ, with learnable parameters θ. We follow their design, which draws on the GNN design space of You et al. (2020), but replace its sum over neighbors with a mean, a choice we motivate below. Our message-passing layer takes the form

$$\mathbf{h}'_i = \mathbf{h}_i \,\Big\|\, \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathrm{ReLU}\left(\mathbf{W}\mathbf{h}_j + \mathbf{b}\right), \tag{3}$$

where $\|$ denotes concatenation and $\mathbf{W}$ and $\mathbf{b}$ are learnable parameters. In terms of Equation 2, the message function is $\phi = \mathrm{ReLU}(\mathbf{W}\mathbf{h}_j + \mathbf{b})$, the aggregation is the mean, and the update γ concatenates the aggregated message with $\mathbf{h}_i$. The message-passing layer is preceded and followed by multi-layer perceptrons (MLPs) that learn node-wise transformations of the features. We use a single message-passing layer, so that, as in Equation 1, the update of each agent depends only on its immediate neighbors.

The aggregation $\bigoplus$ determines how a node summarizes its neighborhood. Common choices include sum (Xu et al., 2019), mean and max (Hamilton et al., 2017), and combinations of several aggregators (Corso et al., 2020). We use the mean for every aggregation in our GNCA. Unlike a sum,

which scales with the number of neighbors, the mean makes each agent's update depend only on the states of its neighbors, not on how many there are. This property is important for swarms, where the number of neighbors changes continuously as agents move.

We adopt the architecture of Grattarola et al. (2021) that learns Boids by combining two message-passing branches that together predict how the velocity of each agent changes (Figure 1a). The first branch applies Equation 3, together with its preceding and following MLPs, to the full state $\mathbf{s}_i$ of each agent, which contains its position and velocity (and, in the dynamic-waypoint setting, its goal displacement defined in Section 3.3). It produces a velocity update Δ_i^{state}. The second branch operates only on relative positions, which carry the spacing information on which separation and cohesion depend,

$$\Delta_i^{\text{rel}} = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \psi\big(\mathbf{p}_i - \mathbf{p}_j,\ \|\mathbf{p}_i - \mathbf{p}_j\|_2\big), \tag{4}$$

where ψ is an MLP that serves as the message function of this branch. For an agent without neighbors, both aggregated messages are zero, so its update depends on its own state alone. Both updates are added to the current velocity, and a final MLP f maps the result to the next velocity, which then advances the position,

$$\mathbf{v}_i' = f\big(\mathbf{v}_i + \Delta_i^{\text{state}} + \Delta_i^{\text{rel}}\big), \qquad \mathbf{p}_i' = \mathbf{p}_i + \mathbf{v}_i'. \tag{5}$$

The final MLP is intended to learn the speed and turn limits of the expert, and integrating the position explicitly keeps the predicted positions consistent with the predicted velocities. Layer sizes and training settings are given in Appendix A.1.

3.3 Learning Goal-Conditioned Transition Rules

The Boids model (Reynolds, 1987) steers each agent with three local rules, namely separation from nearby agents, alignment with their velocities, and cohesion toward their positions. Following Grattarola et al. (2021), training data is obtained by evolving the Boids system from randomized initial positions, and the GNCA is trained to reproduce each observed transition $\mathbf{S}_{t+1} = \tau(\mathbf{S}_t)$, where the state of each agent consists of its position and velocity.

Goal-Conditioned Expert To learn goal-directed behavior, we condition the Boids expert on a goal $\mathbf{g}_k$, the currently active waypoint, by adding a fourth steering term to the three Boids rules. Each agent i steers toward the goal from the centroid of its local neighborhood,

$$\mathbf{a}_i^{\text{goal}} = \text{clamp}\left(\mathbf{g}_k - \mathbf{c}_i\right), \qquad \mathbf{c}_i = \frac{1}{|\mathcal{N}(i)| + 1}\Big(\mathbf{p}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j\Big), \tag{6}$$

where $\text{clamp}(\cdot)$ caps the magnitude of the goal term at the same maximum used for the separation, alignment, and cohesion terms. Steering from the local centroid rather than from each agent's own position moves neighborhoods toward the goal as a group, and including agent i in $\mathbf{c}_i$ allows an isolated agent to navigate on its own. The resulting acceleration is a weighted sum of the four terms, and the full procedure is given in Algorithm 1. Each agent senses the agents within its perception radius r, which defines its neighborhood $\mathcal{N}(i)$, and is repelled only by those within the smaller crowding radius r_c. Because the goal term depends only on the local neighborhood and the shared goal, the expert remains a local, decentralized rule and therefore a valid GCA transition rule for the GNCA to learn.

Each training trajectory of the goal-conditioned expert continues until the swarm has reached every waypoint. When the waypoint sequence is the same for every trajectory, the GNCA receives only agent positions and velocities, so it must implicitly encode the sequence in its learned transition rule. When waypoints vary between trajectories, the state of each agent is augmented with its displacement to the active goal, $\mathbf{d}_{i,g} = \mathbf{g}_k - \mathbf{p}_i$, so that the same GNCA can be steered toward any goal at execution time.

Imitation Objective Reaching a goal and redirecting toward the next one are the behaviors that distinguish goal-directed navigation from flocking, yet they occupy only a small fraction of each

Algorithm 1 Goal-conditioned Boids. Additions to the original Boids update are highlighted.

Require: N agents; waypoints $\mathbf{g}_1, \dots, \mathbf{g}_K$; perception radius r; crowding radius r_c; weights $\beta_s, \beta_a, \beta_c, \beta_g$; arrival radius ε; initial velocity $\mathbf{v}_0$

Initialize positions $\mathbf{p}_i$ at random and set velocities $\mathbf{v}_i \leftarrow \mathbf{v}_0$ for $i = 1, \dots, N$
$k \leftarrow 1$
while $k \le K$ **do**
  for each agent i **do**
    $\mathcal{N}(i) \leftarrow \{j \neq i : \|\mathbf{p}_j - \mathbf{p}_i\|_2 < r\}$
    $\mathbf{a}_i^{\text{sep}} \leftarrow \text{clamp}\left(\sum_{j \in \mathcal{N}(i),\, \|\mathbf{p}_j - \mathbf{p}_i\|_2 < r_c} (\mathbf{p}_i - \mathbf{p}_j)\right)$
    $\mathbf{a}_i^{\text{ali}} \leftarrow \text{clamp}\left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{v}_j - \mathbf{v}_i\right)$
    $\mathbf{a}_i^{\text{coh}} \leftarrow \text{clamp}\left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j - \mathbf{p}_i\right)$
    $\mathbf{c}_i \leftarrow \frac{1}{|\mathcal{N}(i)|+1}\left(\mathbf{p}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{p}_j\right)$
    $\mathbf{a}_i^{\text{goal}} \leftarrow \text{clamp}(\mathbf{g}_k - \mathbf{c}_i)$
    $\mathbf{a}_i \leftarrow \beta_s \mathbf{a}_i^{\text{sep}} + \beta_a \mathbf{a}_i^{\text{ali}} + \beta_c \mathbf{a}_i^{\text{coh}} + \beta_g \mathbf{a}_i^{\text{goal}}$
    $\mathbf{v}_i \leftarrow \text{limit}(\mathbf{v}_i + \mathbf{a}_i)$ ▷ speed and turn limits
    $\mathbf{p}_i \leftarrow \mathbf{p}_i + \mathbf{v}_i$
  if $\|\bar{\mathbf{p}} - \mathbf{g}_k\|_2 \le \varepsilon$ **then** ▷ $\bar{\mathbf{p}}$: swarm centroid
    $k \leftarrow k + 1$ ▷ dynamic setting: $\mathbf{g}_{k+1}$ is sampled here

trajectory. We therefore upweight samples in which the swarm is near a goal. Given a dataset $\mathcal{D}$ of expert transitions $(\mathbf{S}_t, \mathbf{S}_{t+1})$, we minimize

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{S}_t, \mathbf{S}_{t+1}) \in \mathcal{D}} w_t \, \frac{1}{N_t} \sum_{i=1}^{N_t} \|\tau_\theta(\mathbf{S}_t)_i - \mathbf{s}_{i,t+1}\|_2^2 , \tag{7}$$

$$w_t = \begin{cases} \lambda, & \text{if } \min\left(\|\bar{\mathbf{p}}_t - \mathbf{g}_k\|_2,\ \|\bar{\mathbf{p}}_t - \mathbf{g}_{k-1}\|_2\right) < \rho, \\ 1, & \text{otherwise,} \end{cases}$$

where $\bar{\mathbf{p}}_t$ is the centroid of the expert swarm, $\mathbf{g}_{k-1}$ is the previously reached waypoint, ρ is a proximity radius, and $\lambda > 1$ is the upweighting factor.

Training State Perturbations Because the GNCA is applied repeatedly to its own predictions, small errors can accumulate during execution and cause individual agents to drift away from the flock. To teach the GNCA to correct such drift, we optionally perturb the expert during data generation so that the GNCA is trained on noisy states, following the idea of training cellular automata on damaged states (Mordvintsev et al., 2020). Every T_p steps of a trajectory, with T_p spanning hundreds of steps, a small random subset of agents receives bounded uniform noise on its position and velocity,

$$\tilde{\mathbf{p}}_i = \mathbf{p}_i + \boldsymbol{\epsilon}_p, \quad \tilde{\mathbf{v}}_i = \mathbf{v}_i + \boldsymbol{\epsilon}_v, \qquad \boldsymbol{\epsilon}_p \sim U(-\delta_p, \delta_p)^d, \quad \boldsymbol{\epsilon}_v \sim U(-\delta_v, \delta_v)^d, \tag{8}$$

where δ_p and δ_v bound the perturbation and d is the spatial dimension. The expert then continues from the perturbed state, so the GNCA learns to exert the same restorative forces that pull drifting agents back and maintain flock compactness. Perturbed trajectories are mixed with unperturbed ones. We apply this procedure only to three-dimensional dynamic-waypoint navigation and report its settings in Section 4.2.

4 Experiments

We evaluate the GNCA on fixed- and dynamic-waypoint navigation (Section 4.2) and on robustness to agent removal (Section 4.3), each in two and three dimensions. This section presents the main results, and additional figures and implementation details are provided in the appendix.

4.1 Experimental Protocol & Metrics

Setup We consider two navigation tasks. In fixed-waypoint navigation, every rollout visits the same ordered sequence of waypoints. In dynamic-waypoint navigation, each rollout consists of 5 waypoints sampled randomly from the spatial domain, with a minimum separation of 2.5 units between consecutive goals, and each waypoint becomes active only once the previous one is reached. This switching is handled by the simulator rather than by the GNCA. All experiments use swarms of $N = 100$ agents initialized from randomized configurations. The GNCA operates on the same neighborhood graph as the expert, connecting agents within the perception radius $r = 0.1$ (the complete parameter settings of our goal-conditioned Boids implementation are given in Appendix A.2), and every GNCA is trained with the loss in Equation 7 using $\lambda = 2.5$ and $\rho = 2.5$. During evaluation, a waypoint counts as reached when the centroid of the agents present comes within $\varepsilon = 0.5$ of it (Algorithm 1), and each rollout is capped at $5{,}000$ steps. A rollout fails either by timeout, when it does not reach every waypoint within this limit, or by numerical divergence, when the predicted state becomes non-finite. Every GNCA rollout is paired with a Boids rollout from the same initial state and waypoints. Since our goal is to learn the behavior of the expert as a cellular automaton rather than to outperform existing controllers for goal-directed swarm navigation, we compare the GNCA only with the expert it imitates.

Evaluation Metrics Beyond task completion, we measure three properties of the flock at every step t. Flock compactness is quantified by the radial score

$$r(t) = Q_{0.95}\left(\{\|\mathbf{p}_i(t) - \bar{\mathbf{p}}(t)\|_2\}_{i=1}^{N_t}\right), \tag{9}$$

the 95th percentile of agent distances to the centroid $\bar{\mathbf{p}}(t)$ of the N_t agents present at time t, where lower values indicate a tighter flock. Directional alignment is measured by the polarization order parameter of Vicsek et al. (1995), $P(t) = \left\| \frac{1}{|\mathcal{M}_t|} \sum_{i \in \mathcal{M}_t} \mathbf{v}_i(t)/\|\mathbf{v}_i(t)\|_2 \right\|_2$, where $\mathcal{M}_t$ is the set of moving agents and values near 1 indicate aligned motion. Speed consistency is measured by the coefficient of variation of agent speeds, $\mathrm{CV}_v(t) = \mathrm{std}_{i \in \mathcal{M}_t} \|\mathbf{v}_i(t)\|_2 \,/\, \mathrm{mean}_{i \in \mathcal{M}_t} \|\mathbf{v}_i(t)\|_2$, where lower values indicate more uniform speeds. For each rollout, we compute the temporal mean of each metric, denoted $\bar{r}$, $\bar{P}$, and $\overline{\mathrm{CV}}_v$, and we summarize these values across rollouts by their median and interquartile range, which are robust to the few rollouts that time out.

4.2 Goal-Directed Navigation

Training For fixed-waypoint navigation, the 2D GNCA was trained on $1{,}000$ expert trajectories (500 unique starting centers with 2 repeats, balanced across all four spatial quadrants). The 3D GNCA was trained with a staged curriculum over spatial octants, using experience replay to prevent catastrophic forgetting of previously learned regions (Appendix A.3).

For dynamic-waypoint navigation, the GNCA additionally receives the goal displacement $\mathbf{d}_{i,g}$ (Section 3.3). The 2D GNCA was trained on $6{,}400$ trajectories ($1{,}600$ initialized from each spatial quadrant), where each trajectory consists of two sequential waypoints. This two-waypoint design exposes the GNCA not only to single-goal navigation but also to the dynamic transition phase between consecutive active goals. For the 3D GNCA, plain imitation proved insufficient. GNCA trained on $8{,}000$ and $16{,}000$ unperturbed trajectories completed only 13 and 49 of 80 rollouts, and most of their failures (67 and 28, respectively) were numerical divergence of the predicted state (Appendix B.4). We therefore train the 3D GNCA with state perturbations (Section 3.3), using $T_p = 500$, a per-agent selection probability of 0.1 with at most 10 agents per event, and position and velocity perturbation bounds of $\delta_p = 0.1$ and $\delta_v = 0.01$. The 3D GNCA was trained on $8{,}000$ unperturbed and $4{,}000$ perturbed trajectories.

Results

Table 1: Task completion and continuous flock metrics for fixed- and dynamic-waypoint navigation. Continuous values are the median and, in brackets, the interquartile range of the per-rollout temporal means.

Task	Setting	Controller	Task completion	$\overline{r}$	$\overline{P}$	$\overline{\mathrm{CV}}_v$
Fixed	2D	**GNCA**	**40/40**	**0.1512** [0.1442, 0.1611]	**0.9997** [0.9994, 0.9998]	**0.0032** [0.0028, 0.0037]
		Boids	40/40	0.1945 [0.1745, 0.2399]	0.9985 [0.9978, 0.9988]	0.0335 [0.0304, 0.0356]
Fixed	3D	**GNCA**	**77/80**	**0.1188** [0.1092, 0.1388]	**0.9991** [0.9967, 0.9994]	**0.0042** [0.0035, 0.0095]
		Boids	80/80	0.1608 [0.1515, 0.1730]	0.9993 [0.9992, 0.9993]	0.0088 [0.0083, 0.0098]
Dynamic	2D	**GNCA**	**40/40**	**0.1808** [0.1583, 0.1958]	**0.9994** [0.9958, 0.9997]	**0.0053** [0.0038, 0.0094]
		Boids	40/40	0.2379 [0.2057, 0.2633]	0.9978 [0.9974, 0.9983]	0.0358 [0.0337, 0.0393]
Dynamic	3D	**GNCA**	**80/80**	**0.1442** [0.1249, 0.1607]	**0.9998** [0.9994, 0.9998]	**0.0039** [0.0025, 0.0065]
		Boids	80/80	0.1172 [0.1095, 0.1304]	0.9988 [0.9987, 0.9989]	0.0136 [0.0130, 0.0144]

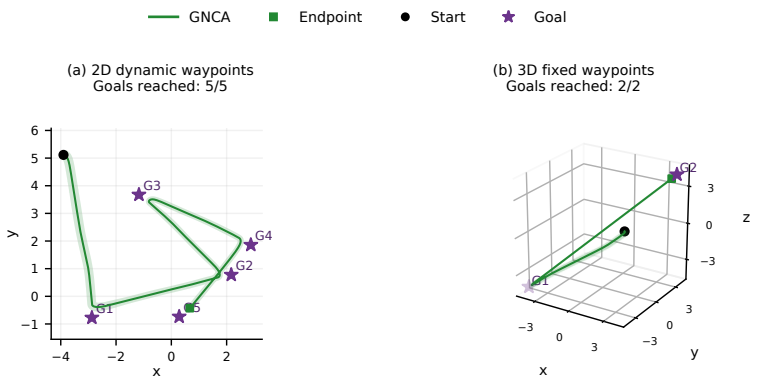

Figure 2: Representative GNCA trajectories for (a) 2D dynamic-waypoint navigation across five online-sampled goals and (b) 3D fixed-waypoint navigation through the ordered waypoint sequence. Solid lines represent the centroids of trajectories. Shaded/tubular regions depict the 95th-percentile agent radial score around each centroid path.

The GNCA completes every rollout in three of the four settings and 77 of 80 rollouts in 3D fixed-waypoint navigation (Table 1). In most settings, the GNCA also holds a tighter flock than Boids while keeping polarization above 0.99. Speed variation is small for both controllers, since agents in both move close to a common speed, but it is lower still for the GNCA, most visibly in 2D. This suggests that the GNCA does not reproduce every small speed adjustment of the expert, but instead learns its general response to a given local configuration, which results in slightly smoother motion. Grattarola et al. (2021) similarly observed that their GNCA produced smoother trajectories than the Boids system it imitated.

Because the GNCA is applied to its own predictions, small errors accumulate over the thousands of steps in a rollout, and once agents begin to drift from the flock, the drift tends to grow. How much this matters depends on how closely the states met during execution resemble those in the training data. In fixed-waypoint navigation, every trajectory follows the same route, so execution stays close to the training distribution, and drift causes only three timeouts in 3D. In dynamic-waypoint navigation, goals can appear in any direction, so the range of states is far larger. Without perturbations, the accumulated errors caused the predicted state of the 3D GNCA to diverge numerically in most rollouts. Training on perturbed states removes all failures while keeping flock compactness close to that of Boids (Table 1).

Both 3D settings required interventions that their 2D counterparts did not, namely a curriculum over octants for fixed waypoints and perturbed training for dynamic waypoints. Adding a dimension enlarges the space of start positions, goal directions, and neighborhood configurations that a finite set of trajectories must cover, so each region of the state space is represented by fewer examples. Additionally, it gives errors an additional direction in which to accumulate. We support this with an ablation over the size of the unperturbed 3D dynamic-waypoint training set, in which even after doubling the number of trajectories from $8{,}000$ to $16{,}000$, 31 of 80 rollouts still failed, most of them through numerical divergence.

4.3 Robustness to Agent Removal

Task & Evaluation Setup We test the robustness of the GNCA to the removal of agents during execution. At a random time step between 500 and $1{,}000$ in each rollout, we draw a random unit vector $\mathbf{u}$ and remove the k agents with the smallest distance $|(\mathbf{p}_i - \bar{\mathbf{p}})^\top \mathbf{u}|$ to the plane that passes through the flock centroid $\bar{\mathbf{p}}$ and is perpendicular to $\mathbf{u}$. This slab removal cuts a band out of the middle of the flock and can split it into two separate groups (Figure 7 in Appendix B.5). The matched Boids rollout uses the same removal step and the same vector $\mathbf{u}$. We remove 10, 25, and 50 agents with each of the four GNCA models and evaluate them with the protocol of Section 4.1.

Results

Table 2: Task completion and continuous flock metrics under slab removal. Continuous values are the median and, in brackets, the interquartile range of the per-rollout temporal means.

Agents removed	Controller	Task completion	$\overline{r}$	$\overline{P}$	$\overline{\mathrm{CV}}_v$
Fixed 2D					
10	**GNCA**	**40/40**	**0.1523** [0.1465, 0.1603]	**0.9997** [0.9995, 0.9998]	**0.0032** [0.0029, 0.0036]
	Boids	40/40	0.2023 [0.1774, 0.2337]	0.9986 [0.9980, 0.9988]	0.0305 [0.0292, 0.0332]
25	**GNCA**	**40/40**	**0.1509** [0.1457, 0.1618]	**0.9997** [0.9994, 0.9998]	**0.0033** [0.0030, 0.0037]
	Boids	40/40	0.1988 [0.1749, 0.2344]	0.9987 [0.9980, 0.9990]	0.0283 [0.0265, 0.0305]
50	**GNCA**	**40/40**	**0.1500** [0.1451, 0.1638]	**0.9995** [0.9988, 0.9997]	**0.0038** [0.0032, 0.0042]
	Boids	40/40	0.1959 [0.1666, 0.2313]	0.9988 [0.9980, 0.9992]	0.0240 [0.0204, 0.0278]
Fixed 3D					
10	**GNCA**	**77/80**	**0.1184** [0.1075, 0.1353]	**0.9989** [0.9976, 0.9992]	**0.0043** [0.0033, 0.0083]
	Boids	80/80	0.1674 [0.1539, 0.1792]	0.9993 [0.9993, 0.9994]	0.0077 [0.0071, 0.0080]
25	**GNCA**	**76/80**	**0.1159** [0.1073, 0.1387]	**0.9987** [0.9972, 0.9991]	**0.0044** [0.0034, 0.0081]
	Boids	80/80	0.1682 [0.1545, 0.1790]	0.9994 [0.9994, 0.9995]	0.0064 [0.0057, 0.0069]
50	**GNCA**	**72/80**	**0.1238** [0.1080, 0.1780]	**0.9977** [0.9918, 0.9989]	**0.0051** [0.0035, 0.0134]
	Boids	80/80	0.1715 [0.1571, 0.1820]	0.9995 [0.9995, 0.9996]	0.0043 [0.0037, 0.0050]
Dynamic 2D					
10	**GNCA**	**40/40**	**0.1857** [0.1729, 0.2131]	**0.9982** [0.9932, 0.9994]	**0.0075** [0.0053, 0.0118]
	Boids	40/40	0.2240 [0.2063, 0.2478]	0.9982 [0.9976, 0.9985]	0.0329 [0.0304, 0.0349]
25	**GNCA**	**39/40**	**0.1922** [0.1699, 0.2195]	**0.9981** [0.9932, 0.9994]	**0.0083** [0.0053, 0.0116]
	Boids	40/40	0.2093 [0.1924, 0.2401]	0.9984 [0.9981, 0.9986]	0.0304 [0.0273, 0.0327]
50	**GNCA**	**38/40**	**0.1954** [0.1705, 0.2517]	**0.9981** [0.9938, 0.9993]	**0.0074** [0.0059, 0.0132]
	Boids	40/40	0.1902 [0.1757, 0.2252]	0.9988 [0.9983, 0.9990]	0.0247 [0.0214, 0.0265]
Dynamic 3D					
10	**GNCA**	**79/80**	**0.1327** [0.1189, 0.1637]	**0.9998** [0.9996, 0.9998]	**0.0033** [0.0025, 0.0057]
	Boids	80/80	0.1141 [0.1074, 0.1301]	0.9988 [0.9988, 0.9990]	0.0127 [0.0120, 0.0134]
25	**GNCA**	**79/80**	**0.1362** [0.1202, 0.1701]	**0.9998** [0.9995, 0.9998]	**0.0035** [0.0025, 0.0060]
	Boids	80/80	0.1129 [0.1048, 0.1291]	0.9990 [0.9989, 0.9990]	0.0109 [0.0100, 0.0116]
50	**GNCA**	**79/80**	**0.1405** [0.1221, 0.1801]	**0.9998** [0.9994, 0.9998]	**0.0039** [0.0025, 0.0070]
	Boids	80/80	0.1070 [0.0998, 0.1269]	0.9992 [0.9991, 0.9993]	0.0077 [0.0071, 0.0084]

Although the GNCA was not trained with agent removal, it completes nearly every rollout after up to half of the swarm is cut out of the flock (Table 2). Compared with navigation without removal (Table 1), completion drops by at most two rollouts in every setting except 3D fixed-waypoint navigation with 50 agents removed, where it falls from 77 to 72 of 80 rollouts. All failures are timeouts rather than numerical divergence. This robustness follows from the structure of the GNCA. The same transition rule is shared by every agent and applied to whatever graph is present, so removing agents only removes nodes and edges and requires no change to the model.

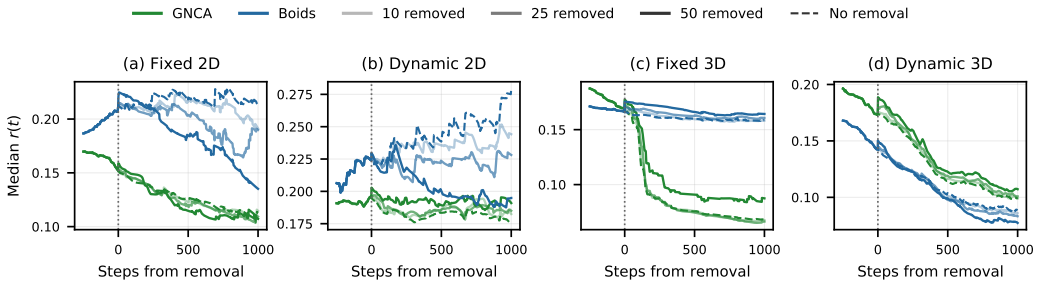

Figure 3: Median radial score $r(t)$ around slab removal for the GNCA and the matched Boids expert. Rollouts are aligned on their removal step (dotted line). Solid lines show 10, 25, and 50 agents removed, and dashed lines show the same rollouts without removal.

Figure 3 shows how the flock responds over time. Slab removal visibly disrupts the flock, and the radial score of both controllers rises at the removal step. In 2D and in 3D dynamic-waypoint navigation, the GNCA then returns close to the radial score of the same rollouts without removal within a few hundred steps, for every k. In 3D fixed-waypoint navigation, the sharp drop after the removal step also occurs without removal and therefore reflects the route rather than the removal. Only with 50 agents removed does this flock settle at a larger radius, which matches its lower completion. Across all four settings, the median flock metrics of the GNCA after removal remain close to those without removal (Tables 1 and 2).

5 Conclusions

We extended graph neural cellular automata from reproducing flocking to goal-directed swarm navigation in two and three dimensions. Trained on a goal-conditioned Boids expert, a single local transition rule guides the swarm through fixed and dynamic waypoints while keeping it together as a flock, and it continues to complete nearly every rollout when up to half of the swarm is removed during execution. These results show that the CA formulation, in which global behavior emerges from repeated local updates, extends from reproducing collective dynamics to accomplishing goal-directed tasks in continuous space.

Our approach has two main limitations. First, the GNCA learns from an expert, so it can only acquire behaviors for which a local rule already exists and cannot yet discover new collective behaviors on its own. Training the transition rule with reinforcement learning, using a reward for reaching goals while keeping the flock together, would remove this dependence on an expert. Second, the switching between waypoints is handled by the simulator based on the swarm centroid, which no individual agent observes. Learning to recognize from local information when a goal has been reached would make the entire navigation process decentralized, and it is a first step toward GNCA that plan at the level of the task, for example by deciding on their own which goal to pursue next. In nature, flocks and schools move as a group through simple interactions between neighbors, and swarm robotics relies on the same kind of local coordination (Brambilla et al., 2013). By showing that such coordination can be learned and directed toward goals, this work suggests that learned GCA could represent a new line of research for real-world swarm control applications.

AI USE STATEMENT

In this work, we used generative AI tools for research execution, to aid and polish the writing, and for the retrieval and discovery of related work. All research ideas, including the methodology, the training procedures, and the design of the experiments, are our own, and we used large language models (LLMs) to generate the code that executes these ideas in our codebase, including the code that produces the figures and tables. We also used LLMs to polish text that we wrote, by correcting grammatical mistakes, improving word choice, and rephrasing sentences for clarity. Finally, we used LLMs for a partial literature search. The works that our method directly builds on were known to us beforehand, and LLMs helped us find further related work. We have not used generative AI tools to generate data sets or to formulate or prove mathematical claims. We have reviewed all AI-assisted work, checked the AI-generated code, and verified every reported number against the underlying evaluation outputs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The GNCA architecture and the training settings of each model are given in Appendix A.1, the parameters of the goal-conditioned Boids expert in Appendix A.2, and the 3D fixed-waypoint curriculum in Appendix A.3. The training data, state perturbations, and imitation objective are described in Section 3.3, and the evaluation protocol, failure definitions, and metrics in Section 4.1. Anonymized code for data generation, training, and evaluation is provided in the supplementary material.

REFERENCES

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Kamalika Bhattacharjee, Nazma Naskar, Souvik Roy, and Sukanta Das. A survey of cellular automata: types, dynamics, non-uniformity and applications. *Natural Computing*, 19(2):433–461, 2020.

Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, volume 33, pp. 13260–13271, 2020.

Gennaro Gala, Daniele Grattarola, and Erik Quaeghebeur. E(n)-equivariant graph neural cellular automata. *Transactions on Machine Learning Research*, 2024.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1263–1272, 2017.

William Gilpin. Cellular automata as convolutional neural networks. *Physical Review E*, 100(3): 032402, 2019.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 2, pp. 729–734, 2005.

Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Learning graph cellular automata. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20983–20994, 2021.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1024–1034, 2017.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Carsten Marr and Marc-Thorsten Hütt. Outer-totalistic cellular automata on graphs. *Physics Letters A*, 373(5):546–549, 2009.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023.

Ettore Randazzo, Alexander Mordvintsev, Eyvind Niklasson, Michael Levin, and Sam Greydanus. Self-classifying MNIST digits. *Distill*, 2020. doi: 10.23915/distill.00027.002.

Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 25–34, 1987.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Shyam Sudhakaran, Elias Najarro, and Sebastian Risi. Goal-guided neural cellular automata: Learning to control self-organising systems. In *ICLR 2022 Workshop on From Cells to Societies: Collective Learning Across Scales*, 2022.

Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 671–682, 2020.

Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Physical Review Letters*, 75(6):1226, 1995.

John von Neumann and Arthur W. Burks. *Theory of Self-Reproducing Automata*. University of Illinois Press, Urbana, 1966.

Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3): 601–644, 1983.

N. H. Wulff and J. A. Hertz. Learning cellular automaton dynamics with neural networks. In *Advances in Neural Information Processing Systems*, volume 5, pp. 631–638, 1992.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Jiaxuan You, Rex Ying, and Jure Leskovec. Design space for graph neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17009–17021, 2020.

A Implementation Details

A.1 GNCA Architecture and Training

The state branch of the GNCA (Section 3.2) computes

$$\mathbf{h}_i \leftarrow \mathrm{MLP}_{\mathrm{pre}}(\mathbf{s}_i), \quad \mathbf{h}_i \leftarrow \mathbf{h}_i \,\Big\|\, \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathrm{ReLU}\left(\mathbf{W}\mathbf{h}_j + \mathbf{b}\right), \quad \Delta_i^{\mathrm{state}} \leftarrow \mathrm{MLP}_{\mathrm{post}}(\mathbf{h}_i).$$

Both $\mathrm{MLP}_{\mathrm{pre}}$ and $\mathrm{MLP}_{\mathrm{post}}$ have two layers with 256 hidden units and ReLU activations. $\mathrm{MLP}_{\mathrm{pre}}$ has 256 output units, and the message-passing layer uses $\mathbf{W} \in \mathbb{R}^{256\times256}$, so the concatenated features have 512 units. $\mathrm{MLP}_{\mathrm{post}}$ has d output units and no final activation. In the relative branch, the message function ψ has one hidden layer of 256 ReLU units and d linear output units. The final MLP f has two layers with 256 hidden ReLU units and d linear output units. The mean over an empty neighborhood is zero, and no batch normalization or dropout is used. All GNCA are trained with Adam (Kingma & Ba, 2015). Because the training sets are too large to hold in memory at once, the trajectories are split into chunks, and the GNCA is trained on one chunk at a time until its validation loss stops improving, carrying over the weights and learning rate to the next chunk. Within each chunk, the learning rate is reduced by a factor of 10 when the validation loss plateaus. Table 3 lists the settings of each model, where the batch size counts transitions and each transition contains all $N = 100$ agents.

Table 3: Training settings of the four GNCA. Patience values are in epochs.

	Fixed 2D	Fixed 3D	Dynamic 2D	Dynamic 3D
Initial learning rate	10^{-3}	10^{-3}, then 10^{-4}	10^{-3}	10^{-3}
Minimum learning rate	0	10^{-6}	10^{-6}	10^{-6}
Batch size	256	256	512	1,024
Learning-rate patience	10	3	3	3
Early-stopping patience	20	5	10	10
Trajectories per chunk	200	400	800	800

The 3D fixed-waypoint GNCA starts the later curriculum stages (Appendix A.3) at a learning rate of 10^{-4}.

A.2 Goal-Conditioned Boids Parameters

Table 4 lists the parameters of the goal-conditioned Boids expert (Algorithm 1) used to generate all training trajectories and to run the matched Boids rollouts during evaluation. The function $\mathrm{clamp}(\cdot)$ rescales a steering term to at most the maximum steering force, and $\mathrm{limit}(\cdot)$ rescales the velocity into the allowed speed range and caps the change in heading per step at the maximum turn angle.

Table 4: Parameters of the goal-conditioned Boids expert.

Parameter	Symbol	Value
Number of agents	N	100
Perception radius	r	0.1
Crowding radius	r_c	0.02
Separation, alignment, cohesion, and goal weights	$\beta_s, \beta_a, \beta_c, \beta_g$	0.35, 0.35, 0.001, 0.01
Maximum steering force (clamp)		0.1
Minimum and maximum speed (limit)		0.0001, 0.01
Maximum turn per step (limit)		5° in 2D, 15° in 3D
Time step		1
Spatial domain		$[-5, 5]^d$, positions clipped at the boundary
Start center	$\mathbf{c}$	uniform within a quadrant (2D) or octant (3D) of the domain
Initial positions	$\mathbf{p}_i$	$\mathbf{c} + 0.325\,\mathbf{u}_i$, with $\mathbf{u}_i \sim U[0, 1]^d$
Initial velocity	$\mathbf{v}_0$	maximum speed along the x-axis, $(0.01, 0)$ in 2D and $(0.01, 0, 0)$ in 3D
Arrival radius	ε	0.5 (0.25 for fixed-waypoint training data)
Fixed waypoints, 2D		$(3, -3) \to (3, 3) \to (0, 0)$
Fixed waypoints, 3D		$(-4, -4, -4) \to (4, 4, 4)$
Dynamic waypoints		uniform in $[-4.5, 4.5]^d$, minimum separation 2.5
Dynamic waypoints per trajectory		2 in training, 5 in evaluation

A.3 3D Fixed-Waypoint Curriculum

The 3D GNCA was initially trained jointly across four lower-hemisphere octants (Octants 4–7, 1,600 trajectories each), followed by sequential curriculum fine-tuning on upper-hemisphere octants (Octants 0–3, 1,600 trajectories each). During the final two curriculum stages, fine-tuning on Octant 1 and Octant 0 jointly incorporated 1,600 experience replay trajectories alongside the 1,600 new octant trajectories. These replay trajectories were balanced across previously learned regions (6 octants for Octant 1, 7 octants for Octant 0). This staged curriculum and replay scheme was determined empirically to generalize across octants while preventing catastrophic forgetting.

B Additional Results

B.1 Fixed-Waypoint Navigation

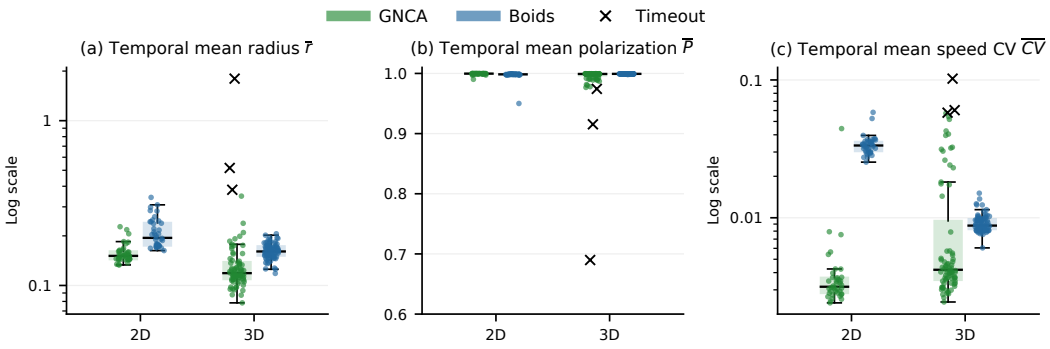

Figure 4: Distributions of per-rollout temporal mean metrics for the fixed-waypoint GNCA and matched Boids expert across 2D and 3D conditions: (a) flock compactness $\bar{r}$, (b) polarization P, and (c) speed coefficient of variation CV_v. Crosses denote timeout trajectories.

B.2 Dynamic-Waypoint Navigation

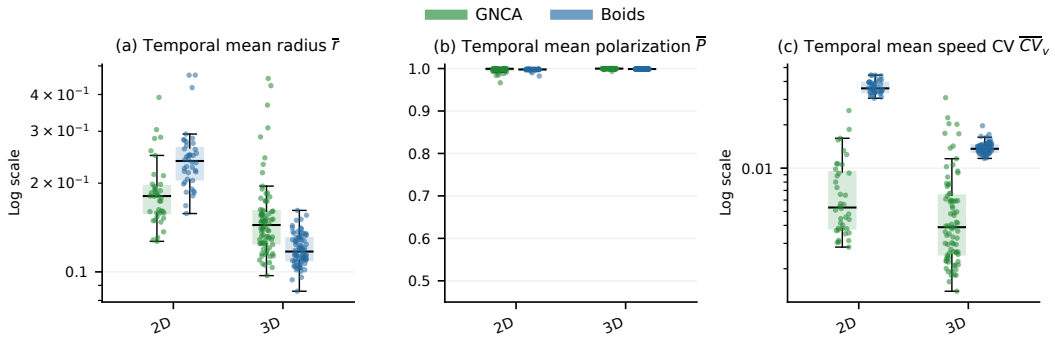

Figure 5: Distributions of per-rollout temporal mean metrics for dynamic-waypoint rollouts comparing the GNCA models against their matched Boids baselines. (a) Flock compactness $\bar{r}$, (b) polarization P, and (c) speed coefficient of variation CV_v across 2D and 3D conditions.

B.3 Visual Examples of Failed Rollouts

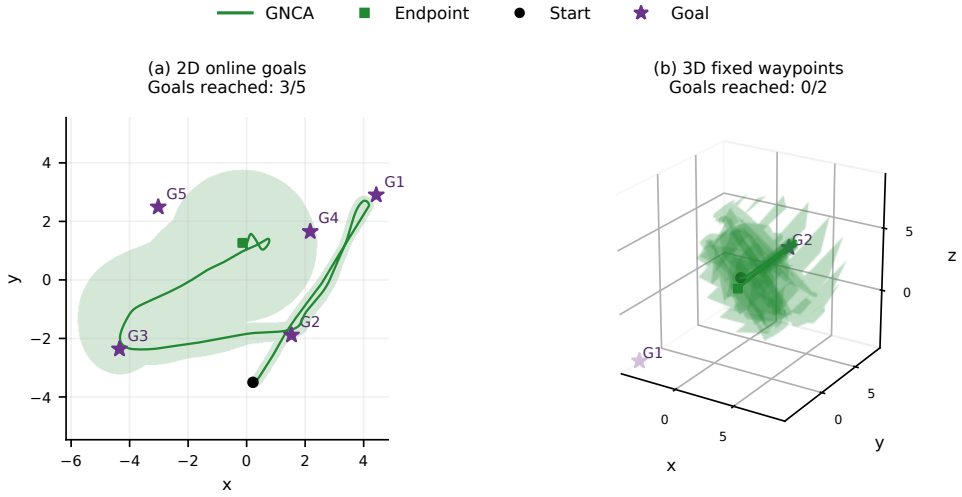

Figure 6: Visual examples of what a GNCA timeout looks like in (a) 2D dynamic-waypoint and (b) 3D fixed-waypoint navigation. Because the 2D dynamic-waypoint GNCA in Table 1 completes every rollout, panel (a) shows a timeout of a separate GNCA, which is included only for illustration. As the flock spreads out, its centroid fails to come within the goal completion radius before the 5,000-step limit. Solid lines represent the centroids of trajectories. Shaded/tubular regions depict the 95th-percentile agent radial score around each centroid path.

B.4 3D Dynamic-Waypoint Training Data Ablation

Table 5 compares 3D dynamic-waypoint GNCA trained on unperturbed trajectories with the GNCA reported in Section 4.2, which adds 4,000 perturbed trajectories to 8,000 unperturbed ones. All models are evaluated with the protocol of Section 4.1, and the matched Boids expert completes all 80 rollouts in each case. Failures are classified as numerical divergence or timeout.

Table 5: 3D dynamic-waypoint GNCA trained with and without perturbed trajectories.

Training data	Completed	Divergence	Timeout
8,000 unperturbed	13/80	67	0
16,000 unperturbed	49/80	28	3
8,000 unperturbed + 4,000 perturbed	**80/80**	0	0

B.5 Agent Removal

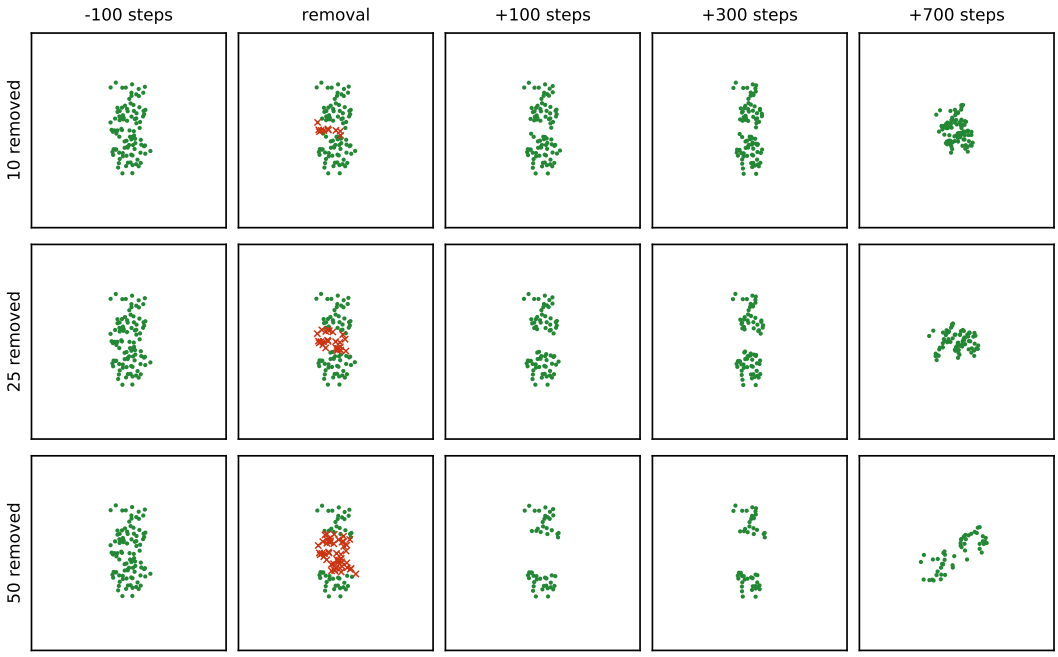

Figure 7: Slab removal in a 2D fixed-waypoint GNCA rollout. Each row removes a different number of agents (red crosses) from the same rollout, and each column shows the flock at a time relative to the removal step. The view follows the flock centroid. The removal splits the flock into two groups, which the GNCA merges back into a single flock.

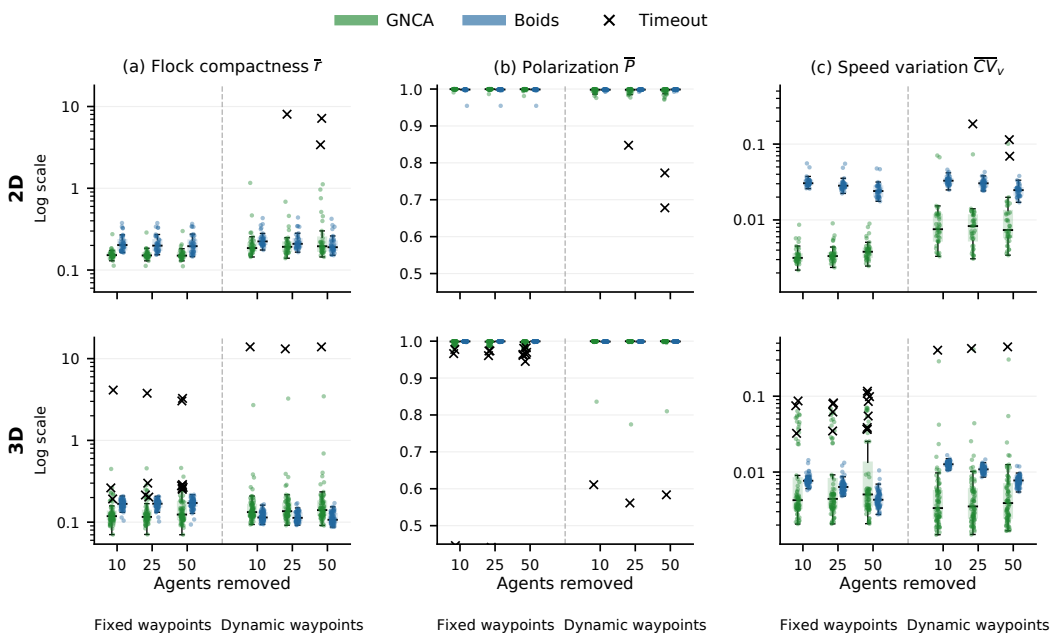

Figure 8: Distributions of per-rollout temporal mean metrics under slab removal comparing the GNCA models against their matched Boids baselines. (a) Flock compactness $\bar{r}$, (b) polarization P, and (c) speed coefficient of variation CV_v across fixed- and dynamic-waypoint tasks in 2D and 3D conditions.