Bayesian optimization of distributed neurodynamical controller models for spatial navigation

Armin Hadzic[a,*], Grace M. Hwang[a,b,1], Kechen Zhang[c], Kevin M. Schultz[a], Joseph D. Monaco[c,**]

[a]*The Johns Hopkins University Applied Physics Laboratory, Laurel, 20723, MD, USA*
[b]*Kavli Neuroscience Discovery Institute, Johns Hopkins University, Baltimore, 21218, VA, USA*
[c]*Department of Biomedical Engineering, Johns Hopkins University School of Medicine, Baltimore, 21205, MD, USA*

Abstract

Dynamical systems models for controlling multi-agent swarms have demonstrated advances toward resilient, decentralized navigation algorithms. We previously introduced the NeuroSwarms controller, in which agent-based interactions were modeled by analogy to neuronal network interactions, including attractor dynamics and phase synchrony, that have been theorized to operate within hippocampal place-cell circuits in navigating rodents. This complexity precludes linear analyses of stability, controllability, and performance typically used to study conventional swarm models. Further, tuning dynamical controllers by manual or grid-based search is often inadequate due to the complexity of objectives, dimensionality of model parameters, and computational costs of simulation-based sampling. Here, we present a framework for tuning dynamical controller models of autonomous multi-agent systems with Bayesian optimization. Our approach utilizes a task-dependent objective function to train Gaussian process surrogate models to achieve adaptive and efficient exploration of a dynamical controller model's parameter space. We demonstrate this approach by studying an objective function selecting for NeuroSwarms behaviors that cooperatively localize and capture spatially distributed rewards under time pressure. We generalized task performance across environments by combining scores for simulations in multiple mazes with distinct geometries. To validate search performance, we compared high-dimensional clustering for high- vs. low-likelihood parameter points by visualizing sample trajectories in 2-dimensional embeddings. Our findings show that adaptive, sample-efficient evaluation of the self-organizing behavioral capacities of complex systems, including dynamical swarm controllers, can accelerate the translation of neuroscientific theory to applied domains.

Keywords: Bayesian optimization, multi-agent control, swarming, dynamical systems models, spatial navigation, UMAP

1. Introduction

Collective biological behaviors of animal groups, including swarming, flocking, and schooling behaviors [1–6] have long inspired robotics and computer science research into problems of decentralized control and coordination for autonomous groups of artificial agents [7–12]. In particular, advancing the autonomous spatial capabilities of multi-agent swarm control has been a key objective of simulation studies and analyses of artificial swarms based on dynamical systems models [13]. Complementarily, the impressive recent progress of artificial intelligence based on deep learning [14] has demonstrated the importance of adopting key biological inspirations from neuroscience and the brain. However, it has been unclear how to integrate complex temporal features of brain dynamics thought to support crucial mechanisms of neural computation [15]. Thus, addressing critical questions in autonomous robotics and artificial intelligence may depend on efficient exploration and optimization of dynamical systems models with complex interactions among many units. In both domains, major gaps in state-of-the-art capabilities are highlighted by tasks involving autonomous spatial navigation and foraging [16–19] in complex, novel, or changing environments.

Bayesian optimization provides a probabilistic framework for adaptive, sample-efficient optimization of 'black box' models with moderate dimensionality (up to ~20 parameters) and expensive sample evaluations. In this framework, a task-dependent objective function signifies the output performance of the complex underlying model, and the optimizer traces parameter-space trajectories of candidate points from acquisition functions operating on a simpler surrogate model. The typical surrogate model is a Gaussian process that populates the parameter space of interest with multivariate normal distributions and which serves as a prior distribution for candidate-point updates [20, 21]. Bayesian optimization with Gaussian process surrogate models has enabled applications including the hyperparameter tuning and optimization of evolutionary algorithms, multi-modal functions, robotic controllers, and other complex systems [22–27].

The collective behavioral states of some swarming models are tractable to linear analysis of stability, density, and clustering properties [28–32]. However, for dynamical systems

*Corresponding author (arminhadzic@outlook.com)
**Alternate contact (joe@selfmotion.net)

[1]This material is based on work supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

that preclude such analysis due to nonlinearity, nonstationarity, stochasticity, or other complications, the computational budget for parameter exploration or optimization with simulation-based samples is a limiting factor for translation to engineered designs. Indeed, standard methods based on gradient descent have two main drawbacks in this context: they can discover local optima, but resist exploration of system behaviors for other purposes; and their basic operation is massively sample-inefficient, which can be prohibitive for expensive simulation-based sample evaluations. Moreover, emergent collective behaviors like swarming outstrip conventional agent-based learning methods based on the restrictive action and policy spaces of reinforcement learning, particularly for uncertain, changing, or open-ended tasks.

We previously introduced the *NeuroSwarms* framework for modeling emergent high-level navigation and foraging in a brain-inspired multi-agent metacontroller [33, 34]. NeuroSwarms addressed decentralized, distributed control by analogy to neural circuit dynamics, including oscillations [35–38] and attractors [39–41], and associative synaptic plasticity [42] related to rodent spatial cognition; the resulting collective behaviors of NeuroSwarms models included swarming, patroling, and goal-finding in simulated maze environments with complex, irregular, or fragmented geometry [34]. These behaviors enabled NeuroSwarms to complete cooperative multiple reward-capture tasks without pretraining across distinct environments [34]. However, the nonlinearities inherent in NeuroSwarms' oscillatory phase-coupled self-organization precluded analytic approaches to global identification, exploration, or optimization of system behaviors. Thus, this class of dynamical systems model can provide insights into key aspects of brain structure and function that may inspire theoretical advances as well as new directions for systems engineering designs. This insight depends crucially on devising a task-dependent objective function that can guide the efficient discovery of system behaviors and optimal performance. In this paper, we demonstrate that Bayesian optimization can utilize such an objective function to efficiently and usefully find paths through otherwise prohibitive model spaces. In particular, we show that a neurodynamical controller model with emergent properties can be characterized and tuned using Bayesian optimization with Gaussian process surrogate models.

2. Models and methods

2.1. NeuroSwarms model

Monaco *et al.* (2020) [34] introduced the NeuroSwarms framework and described a model implementation with 300 agents; baseline wall-avoiding, momentum-carrying motion-vector updates; maze environments whose geometry occluded agents' line-of-sight; interagent communication between mutually visible agents; cosine-coupling of internal phase variables driving interagent attraction and repulsion; and 9 key dynamical parameters (Table 1) that had required intensive manual fine-tuning to balance swarming and reward capture.

Name	Range	Description
σ	$[10^{-3}, 4]$	Normalized interagent spatial scale
κ	$[10^{-3}, 4]$	Normalized reward-approach spatial scale
η_s	$[10^{-3}, 4]$	Recurrent interagent learning rate
η_r	$[10^{-3}, 4]$	Feedforward reward-approach learning rate
ω_0	$[0, 1]$	Baseline agent oscillation frequency
ω_I	$[0, 1]$	Max. activation-based frequency increase
τ_q	$[0, 1]$	Recurrent interagent time-constant
τ_r	$[0, 1]$	Feedforward reward time-constant
τ_c	$[0, 1]$	Sensory input time-constant

Table 1. Tunable parameters that governed the spatiotemporal dynamics of the example NeuroSwarms model implementation [34]. 'Range' indicates the limits of the parameter subspace made available for Bayesian optimization. All other NeuroSwarms parameter values and constants were fixed at the defaults listed in Table 1 of Monaco *et al.* (2020) [34].

2.2. Bayesian optimization

Bayesian optimization constructs and performs sequential optimization on a surrogate model that represents the objective performance of a more complex model [43–45]. Learning surrogate models can be beneficial if directly optimizing a complex model is not computationally tractable given resource constraints. These surrogate models can then be deployed to predict the performance of the underlying model at untested parameter points without requiring a full model simulation of those parameter values (Fig. 1).

We implemented Bayesian optimization with surrogate models defined as Gaussian processes [20, 47, 48]. Gaussian processes are parametric models that iteratively learn a probabilistic mapping $f : \mathbb{X} \mapsto \mathbb{R}$ such that the density estimate $\mathrm{p}(y_i|\mathbf{x}_i) = f(\mathbf{x}_i, y_i)$, where $\mathbb{X} \subseteq \mathbb{R}^p$ is the bounded parameter subspace being optimized, $\mathbf{x}_i \in \mathbb{X}$ is a parameter point, and $y_i \in \mathbb{R}$ is an objective function output value [21, 49, 50]; e.g., $p = 9$ NeuroSwarms parameters in this paper. Thus, the underlying 'black box' objective function f_{true} is assumed to be distributed according to a Gaussian process,

$$f_{\text{true}} \sim \mathcal{GP}_{\mu,k}(X)\,,$$

where $\mu(\cdot)$ and $k(\cdot)$ are mean and covariance kernels applied to an input parameter set, $X \subset \mathbb{X}$. The posterior distribution of a q-sized batch of candidate points $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_q\}$ conditioned on the observed training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ takes the form of a p-dimensional multivariate normal distribution, i.e., $\mathrm{P}(\mathcal{GP}(X)|\mathcal{D}) \sim \mathcal{N}^p(\mu(X), k(X))$.

2.3. Acquisition functions

Bayesian optimization relies on acquisition functions to provide the candidate parameter points that navigate the underlying model space. Acquisition functions define a strategy to manage the trade-off between exploring the parameter space and exploiting regions that yielded improvement for previous samples [51]. An acquisition function can be evaluated on the Gaussian process posterior $\mathrm{P}(\mathcal{GP}(X)|\mathcal{D})$ by averaging a set of Monte Carlo (MC) samples, e.g.,

$$\hat{\alpha}_n(X; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} a\left(\mathcal{E}^i_{\mathcal{D}}(X)\right), \tag{1}$$

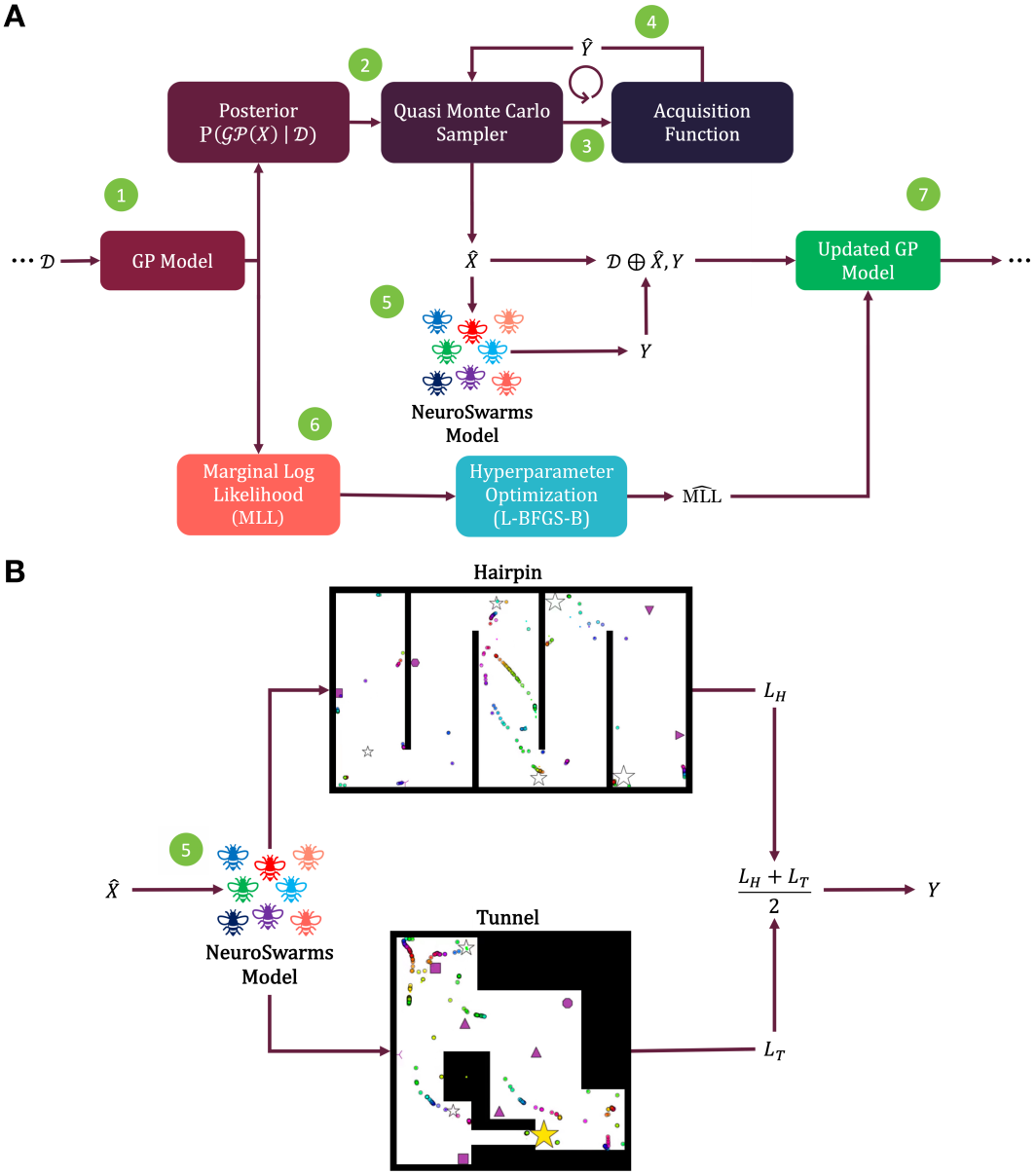

Fig. 1. Computation flow for optimization and simulation-based sampling. *A*, Step 1: The posterior distribution is computed from the Gaussian process surrogate model (GP Model) based on the training data $\mathcal{D}$. Step 2: The acquisition function's Quasi Monte Carlo sampling process uses the posterior distribution to select new candidate parameters $\hat{X}$ (Step 3) based on the acquisition function's estimated objective function value $\hat{Y}$ (Step 4). Step 5: The NeuroSwarms model [33, 34] is simulated with candidate parameter points $\hat{X}$ to generate the observed objective value Y (see *B*). Step 6: The initial Gaussian process model's marginal log-likelihood (MLL) is then calculated and used to optimize the Gaussian process using the L-BFGS-B algorithm [46]. Step 7: The resulting $\mathcal{D}$ (from Step 5) and MLL (from Step 6) update the Gaussian process model for the next iteration of the outer loop. *B*, Flow diagram of simulation-based candidate-point evaluation. For each sample (see Step 5 in *A*), the optimizer executes play-throughs in both the Hairpin (top) and Tunnel (bottom) maze environments. The sample's objective value Y is computed as the average of the respective loss values L_H and L_T (Eq. 3).

where n is the sample count and $a(\cdot)$ is the net utility function providing objective function output. Thus, $\hat{\alpha}_n$ is an expectation of posterior samples $\mathcal{E}_\mathcal{D} \sim \mathrm{P}(\mathcal{GP}(X)|\mathcal{D})$. We study a pair of MC-based acquisition functions: q-Expected Improvement (qEI) [52] and Noisy q-Expected Improvement (qNoisyEI) [53]. We compare qEI and qNoisyEI to random sampling of candidate parameters. First, similar to $\hat{\alpha}$ (Eq. 1), qEI calculates an expectation over posterior samples,

$$\mathrm{qEI}(X) \approx \frac{1}{n} \sum_{i=1}^{n} \max_{j=1}^{q} \left[\mathcal{E}_j^i - Y^*\right]_+ ,$$

where $[\cdot]_+$ indicates linear rectification and Y^* is the best observed objective function value. Thus, qEI estimates a noise-free expected improvement of the posterior with respect to the best value. Second, qNoisyEI approximates improvement relative to the expected best objective value conditioned on the observed MC sampling history $\mathcal{E}_\mathrm{obs}$ within each batch [54]; simplistically, the constrained batch-sampling performed by qNoisyEI [53, 55] approximates

$$\mathrm{qNoisyEI}(X; \mathcal{D}) \approx \frac{1}{n} \sum_{i=1}^{n} \max_{j=1}^{q} \left[\mathcal{E}_j^i - \max\, \mathcal{E}_\mathrm{obs}\right]_+ ,$$

but more detailed treatments of this complex optimization problem provide critical analyses and caveats [cf. 53–55].

Throughout our study, Bayesian optimization with any of the three acquisition functions employed 512 MC samples, 30 training epochs (with a batch size of 3), and 8 random training samples to initialize the Gaussian process surrogate model.

2.4. Objective function

We constructed an objective function to evaluate the performance of the example NeuroSwarms model [34] in a time-pressured cooperative foraging task. The objective function quantifies how quickly the swarm of agents collectively capture several spatially distributed rewards in a given maze. Let $n_{\text{cap}}(t)$ be the cumulative number of cooperatively captured rewards by time t. A reward is captured if, at any timestep, at least n_s/n_r agents were simultaneously colocated within a defined radius from the reward, where $n_s = 300$ agents and $n_r = 3$ and 5 rewards in the Tunnel and Hairpin mazes, respectively. For a given simulated play-through, this objective function can be expressed as a loss which is updated at every timestep until all rewards are captured,

$$L = -t/\left(n_t n_{\text{cap}}(t) + 1\right), \tag{2}$$

where n_t is the total number of time steps. The agent group's behavior is time-pressured by t growing continuously until all rewards are captured. If the swarm is not able to capture all the rewards in the environment, t will be set to the maximum number of timesteps allowed for the simulation n_t and the loss will reflect the number of missed rewards. Loss values range from $[-1, 0]$, with better task performance closer to zero.

To account for the generalizability of spatial task performance across distinct environmental geometries, each simulation-based sample constitutes play-throughs of both the Hairpin and Tunnel mazes, respectively providing loss values L_H and L_T as calculated in Eq. 2 (see Figure 1*B*). Thus, the generalized performance at a given parameter point $\mathbf{x}_i$ is indicated by the objective value Y, computed as the average

$$y_i(\mathbf{x}_i) \doteq Y = \frac{L_H + L_T}{2}. \tag{3}$$

2.5. Gaussian process training

The means and variances of the Gaussian process surrogate model are updated with each sample evaluation to reflect the expected values and uncertainty, respectively, of the underlying model's performance. We use the Bayesian optimization library BoTorch [50] to implement the outer loop of surrogate model training based on iteratively updating a Gaussian process following initialization with sample data $\mathcal{D}$. The posterior distribution $\mathrm{P}(\mathcal{GP}(X)|\mathcal{D})$ is then sampled from a batched MC sampling process using an acquisition function to determine the candidate parameter points $\hat{X}$ from the subspace bounded by the ranges listed in Table 1. The candidate points are selected based on predictive estimates of utility value $\hat{Y}$ (Fig. 1*A*) and evaluated by simulating the NeuroSwarms model to generate loss values (Eq. 2) and objective function output Y (Eq. 3) (Fig. 1*B*). Lastly, the resulting $(\hat{X}, Y)$ tuple is appended to training data $\mathcal{D}$ to update the Gaussian process for the next iteration.

The surrogate model hyperparameters were tuned by first computing the marginal log-likelihood (MLL) of the Gaussian process applied to observed parameters X and fitting hyperparameters with the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm with simple bounds (L-BFGS-B) [46]. The fitting process provides an updated MLL for the next optimization step.

2.5.1. Convergence metrics

This hyperparameter tuning process described above was repeated until convergence according to two metrics: maximum posterior variance and minimum candidate dissimilarity. First, maximum posterior variance for training epoch M was computed following

$$\max \mathrm{Var}\left(\mathrm{P}(\mathcal{GP}(x_M) \,|\, \mathcal{D}_M)\right)$$

to indicate whether the Gaussian process' posterior variance was no longer increasing and that training should cease. Second, minimum candidate dissimilarity measures the stabilization of candidate selection as an inverse cosine similarity; i.e., we calculated the metric following

$$\min_{i=1}^{M-1}\left[1 - \frac{x_i \cdot x_M}{\|x_i\| \cdot \|x_M\|}\right]$$

to confirm whether epoch M selected for similar neighborhoods of parameter points as in previous training epochs. These convergence metrics determined hyperparameter convergence and enabled the resulting Gaussian process surrogate model to efficiently adapt to the NeuroSwarms parameter space.

2.6. Parameter visualization

The low-dimensional representations produced by the uniform manifold approximation and projection (UMAP) [56] result from a locality-preserving embedding that serves to spatially cluster higher-dimensional vectors such as p-dimensional parameter points. A 2D UMAP projection allows these point clusters to be simply visualized as images or scatter plots, for which the x-axis and y-axis constitute an arbitrary coordinate frame. For UMAP scatter plots, as in Figure 3 and Figure 6, the marker for each point can be colored for convenient visual inspection of associated values, including vector elements or computed output. We use this visual clustering to qualitatively inspect the parameter-dependence and structure of the Gaussian process surrogate model by selecting a UMAP data point with, e.g., high performance indicated by its loss value y_i (Eq. 3), and assessing that point's other values in the context of its location and neighborhood relative to UMAP-based clusters.

3. Results and discussion

3.1. Overview

We demonstrate Bayesian optimization methods (see Section 2.2) for tuning the parameters of a neuroscience-inspired

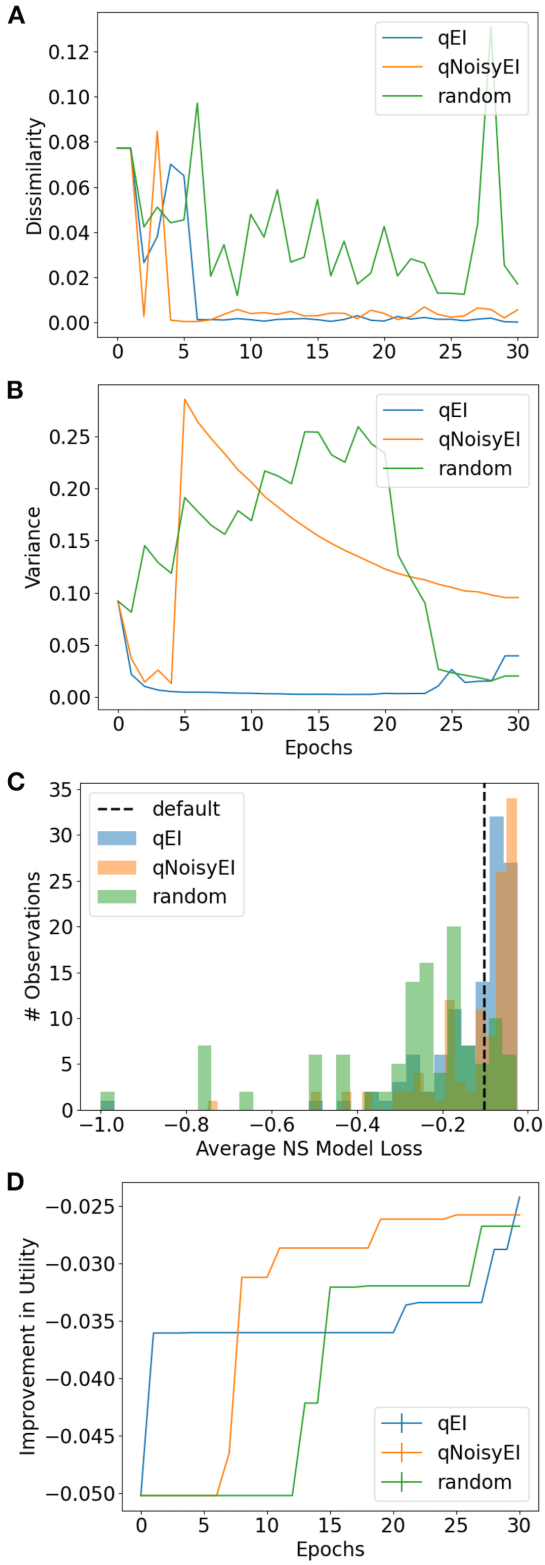

Fig. 2. Convergence metrics and objective function values for acquisition functions across training. *A*+*B*, Training convergence metrics: minimum candidate dissimilarity (*A*) and maximum posterior variance (*B*). *C*+*D*, Gaussian process models with corresponding acquisition functions model performances as shown by observed objective function histograms (*C*) and best observed values (*D*), where values closer to 0 indicate stronger performance.

swarming model, NeuroSwarms [33, 34, 38] (see Section 2.1), to find cooperative foraging behaviors for capturing multiple rewards in distinct maze environments under time pressure (see Section 2.4). We train Gaussian process surrogate models (see Section 2.5) to characterize the NeuroSwarms parameter space using noise-free (i.e., qEI) and observed sampling history-dependent (i.e., qNoisyEI) acquisition functions (see Section 2.3). Then we show how the locality-preserving dimensionality reduction provided by UMAP embeddings (see Section 2.6) can be used to evaluate surrogate model structure to identify critical system behaviors.

3.2. Training the surrogate model for swarming performance

Small variations in the $p = 9$ dynamical NeuroSwarms parameters (Table 1) can substantially impact collective behaviors. Optimal parameters that allow NeuroSwarms models to accomplish generalized cooperative foraging may not be limited to a single set of parameters due to the complexity and potential degeneracy of emergent collective behaviors in a distributed multi-agent system. Thus, we constructed a simple time-pressured objective function to measure the progress of reward-capture (Section 2.4) and guide Bayesian optimization using Gaussian process surrogate models (Fig. 1*A*). We utilized acquisition functions to sample candidate parameter points and optimize the Gaussian process' predictive performance compared to observed NeuroSwarms simulations (Section 2.5). We evaluated the surrogate models in two environments for each sample: a Hairpin maze and a Tunnel maze (Fig. 1*B*). By simultaneously assessing mazes with distinct geometries, the surrogate model optimization was allowed to find swarming and navigational dynamics resulting in time-efficient cooperative foraging that may generalize across environments.

We started training with an initial set of 24 randomly selected parameter points with corresponding simulation results. Each Gaussian process was trained by an acquisition function for selecting candidate points: q-batched Expected Improvement (qEI), q-batched Noisy Expected Improvement (qNoisyEI), or random parameter sampling (Section 2.3). Gaussian process modeling and training was implemented using BoTorch [50] and optimized with 512 MC samples over 30 training epochs (Section 2.5). We verified that the EI-based acquisition functions converged based on metrics of minimum candidate dissimilarity and maximum posterior variance (Section 2.5.1). The EI-based acquisition functions approached zero dissimilarity during training (Fig. 2*A*). Similarly, the maximum posterior variance for each surrogate model had converged by the end of training (Fig. 2*B*).

We evaluated how effective each acquisition function was at finding regions of the parameter space that optimize the NeuroSwarms objective function (Eqs. 2 and 3). Both qEI and qNoisyEI discovered more parameter points with high-performance values than random sampling (Fig. 2*C*). Both random sampling and the default parameters from Monaco *et al.* (2020) [34] were outperformed by the EI-based acquisition functions. Thus, qEI and qNoisyEI demonstrated the strongest utility improvement of best observed values during training as

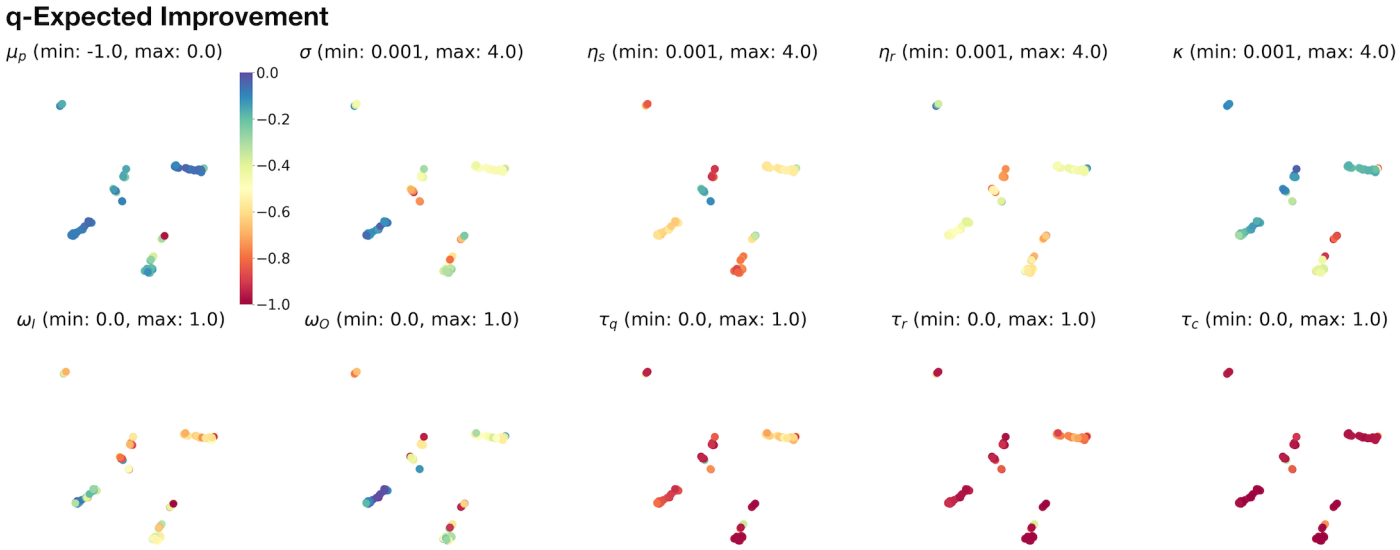

Fig. 3. UMAP-clustered parameter points selected by the noise-free qEI acquisition function.

the NeuroSwarms parameter space was learned by the corresponding surrogate models (Fig. 2*D*).

3.3. Evaluating UMAP-clustering of selected parameters

Understanding the results of the above Bayesian optimization process requires a visual representation of the parameter space, yet it can be challenging to represent data with > 3 dimensions. We considered that a visualizing parameter points in lower dimensions could facilitate the discovery of critical surrogate model structures, including clusters of high-performing parameters that potentially yield distinct behavioral solutions to the cooperative foraging task. Thus, we used UMAP (Section 2.6) to reduce sets of 9-dimensional NeuroSwarms parameters (Table 1) into locality-preserving 2D representations. For qEI-selected parameters, we assigned colors to the resulting 2D UMAP-clustered data points according to posterior mean estimates of objective values (top, left plot) or individual parameter values (Fig. 3). The resulting visual representation in Figure 3 shows where the highest utility (i.e., best posterior mean estimate of objective value) data points cluster into groups based on the values of NeuroSwarms parameters

Given that qEI demonstrated the largest utility improvement (Fig. 2*D*) and consistently identified high-performing parameters (Fig. 2*C*), we consider its UMAP representation for further analysis. The qEI-based parameter samples formed two clusters of data points with the highest utility (Fig. 3). In the (top, left) posterior mean plot, we selected one of these points from the lower, left cluster and matched it with the numerical values of its associated parameters, which we subsequently evaluated in NeuroSwarms simulations.

We simulated the qEI-optimized NeuroSwarms model on both the Hairpin and Tunnel mazes (see Figure 1*B*). Trajectory-trace plots for the Hairpin (Fig. 4, blue traces) depict the movement of each agent that contributed to reward capture throughout the simulation, up to the timestep at which cooperative capture of each reward goal was achieved. Likewise, trajectory traces in orange (Fig. 4) reflect the behavior of the reward-capturing agents after the reward had been captured. For example, the transition from swarming and goal-directed dynamics to post-capture exploration is depicted by the capture of Reward 3 (R3) in the third row of Figure 4, in which a subset of agents converged on and captured R3 and immediately dispersed, thus permitting the search for and capture of subsequent reward goals. Agents recommenced exploration following reward-capture because NeuroSwarms relies on local, line-of-sight communication between agents, meaning that agent motion may not be influenced by nearby rewards if they are occluded by walls of the maze. The qEI-tuned swarms were able to quickly capture all five rewards on the Hairpin environment (t = 25.38 s), as shown in Figure 4, whereas the original default parameters of NeuroSwarms—determined by hand-tuning as described in our previous work [34]—produced relatively slow reward capture (t = 41.02 s). Reward-capture speed using the default parameters was additionally exacerbated in the Tunnel maze (t = 175.42 s). In contrast, the qEI-tuned swarm captured all three rewards (Fig. 5) faster than the default swarm captured two rewards (t = 34.88 s). We attribute the worse performance of the hand-tuned default parameters to longer dynamical time-constants and thus slower behavioral responsivity. Thus, compared to manual parameter tuning for each maze environment, our Bayesian batch-optimization process (Section 2.3; Fig. 1*A*) with joint objective sampling (Section 2.4; Fig. 1*B*) was able to simultaneously, jointly, and efficiently discover distinct high-performing dynamical parameters for multiple mazes.

A key feature of our Bayesian optimizer is that the objective *indirectly* quantifies (i.e., as a 'black box' model) cooperative foraging without *directly* modifying NeuroSwarms' underlying mechanisms. In general, this feature allows a task-dependent objective to evaluate multi-agent performance in collective tasks involving, e.g., social coordination or distributed consensus. In contrast to the regular but fragmented geometry of the Hairpin maze (Fig. 4), the Tunnel maze required

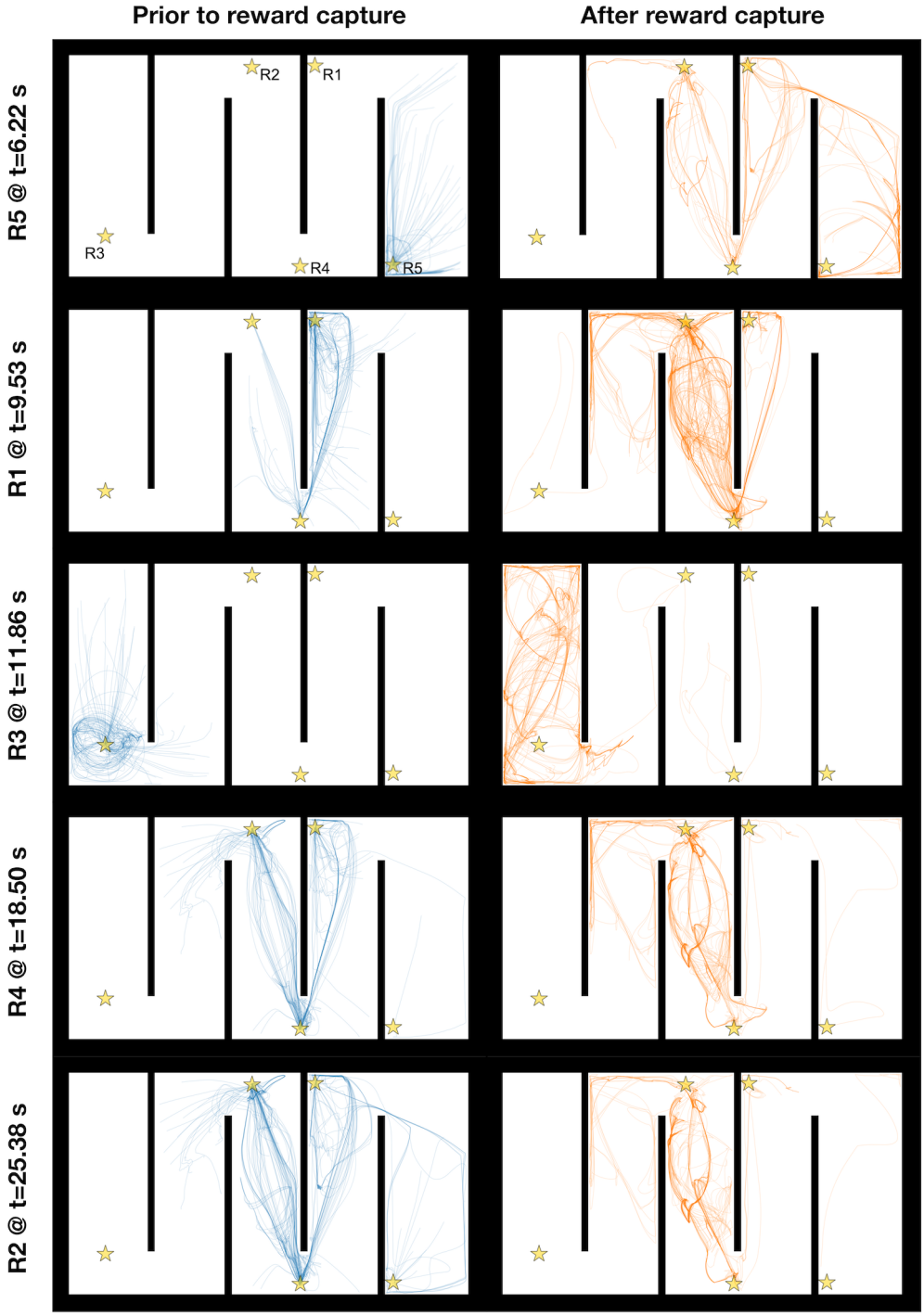

Fig. 4. NeuroSwarms trajectories depicting reward capture in the Hairpin maze. Simulated trajectory-trace plots in the Hairpin environment. The locations of the five reward goals (labeled R1–R5 in the top, left maze plot) are marked with gold stars.

the swarm to distribute through an irregular geometry to complete the foraging task (Fig. 5). Additionally, whereas agents were initialized at uniform random locations in the Hairpin maze, all agents in the Tunnel maze were initialized to points inside a small disc circumscribed within its Southwest quadrant. As a result, the agents rapidly capture R2 (Fig. 5, top row) and then split into subgroups to capture the remaining two rewards (Fig. 5, lower two rows). An additional challenge of the Tunnel maze is that R3 is initially visible to all agents and closer than R1, yet the tunnel constricts access to it. Conversely, R1 is initially visible and accessible, yet further away and partially occluded once agents have converged onto R2's location. The fast capture of R1 (t = 5.46 s) vs. R3 (t = 31.78 s) reflects the characteristic time-scale differences between coordinated reward-approach trajectories and exploratory swarming trajectories, respectively. Comparing the pre-capture (blue, left) and

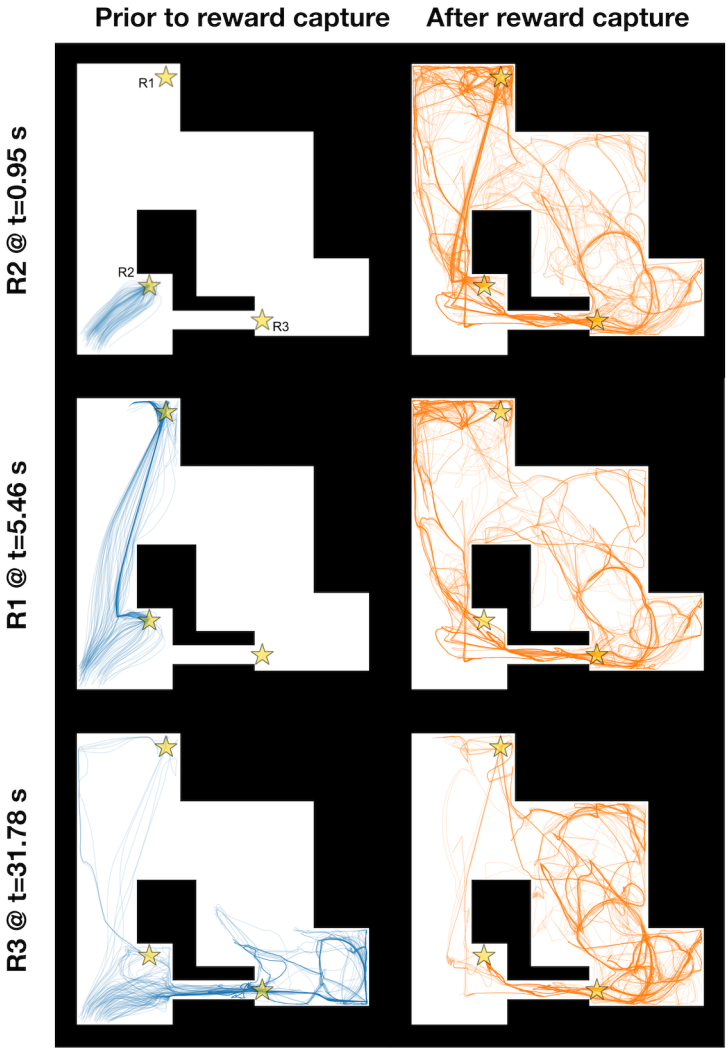

Fig. 5. NeuroSwarms trajectory-trace plots depicting reward capture in the Tunnel maze. As labeled in the top-left maze plot, the locations of reward goals R1–R3 are marked with gold stars in the Northwest, Southwest, and Southeast quadrants, respectively.

post-capture (orange, right) trajectories for each reward (Fig. 5), the agents began using the large opening in the center of the map only once R2 and R1 were both captured. This behavioral transition suggests that exploration traded off with goal-directed exploitation by adaptively forming and regrouping subgroups of agents. Thus, distinct challenges presented by the Tunnel maze, in concert with our optimizer's objective function definition (Section 2.4), may have induced collective behaviors that can flexibly adapt to diverse foraging problems.

3.4. Exploring the future parameter space

Trained acquisition functions can be used to predict the performance of unobserved regions of the parameter space. To test predictive selection, we generated 500 samples from the qEI acquisition function and the posterior distribution of its trained Gaussian process surrogate model. The qEI sample means from the posterior (Fig. 6, top-left plot) were similar across most data points because qEI had adapted to parameter regions with the highest likelihood of utility improvement. As in the previous section (3.3), we selected candidate points from these anticipated future qEI parameters to simulate in the Hairpin and Tunnel mazes, but we chose points that featured mid-range parameter values, i.e., whose vector elements were not at or near the range limits of the respective parameter (Table 1). In particular, we selected parameters where the time-constants were greater than the minimum of their ranges (1 ms), constituting a parameter regime that was distinct from clusters of qEI samples which minimized their respective time-constants in response to the time-pressure imposed by our objective function (Eq. 2). We chose these points, with corresponding simulations shown in Figure 7, to demonstrate the distinct behavioral solutions to the foraging task that can be discovered by the same acquisition function and associated surrogate model. Trajectory-trace plots of reward-capturing agents before and after rewards were cooperatively captured on the Hairpin and Tunnel mazes show that the selected parameters resulted in slower reward capture for the Hairpin (t = 47.44 s; Fig. 7*A*) and Tunnel (t = 66.96 s; Fig. 7*B*) mazes compared with the optimized parameters in Figure 4 (Hairpin, t = 25.38 s) and Figure 5 (Tunnel, t = 31.78 s). Additionally, the default parameters from Monaco *et al.* (2020) [34]

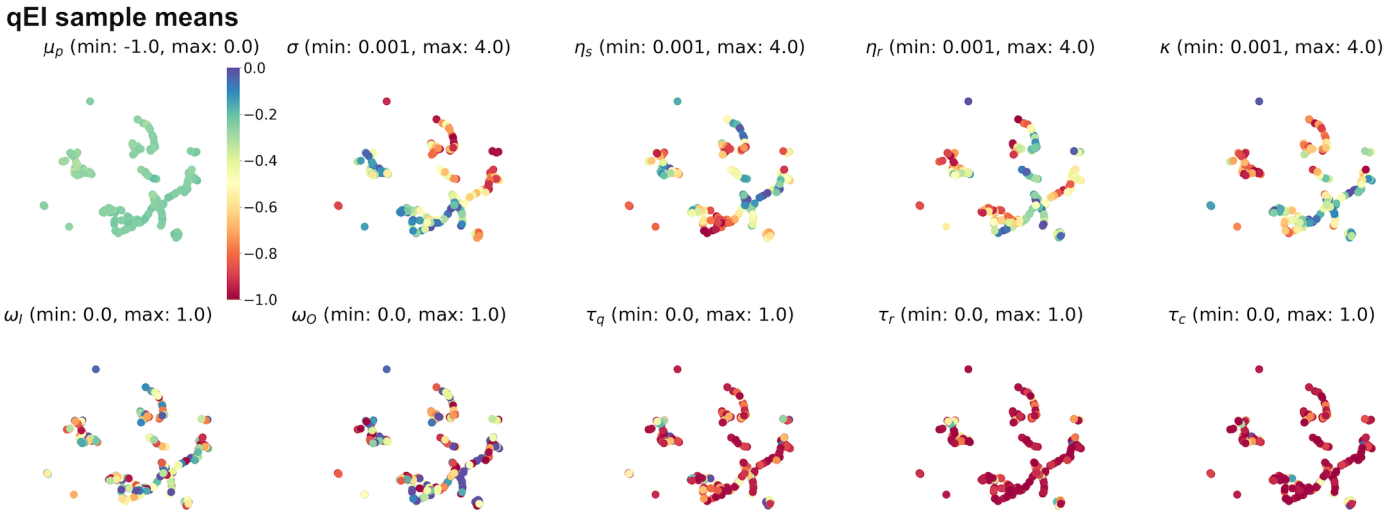

Fig. 6. Anticipated future qEI-sampled parameter points.

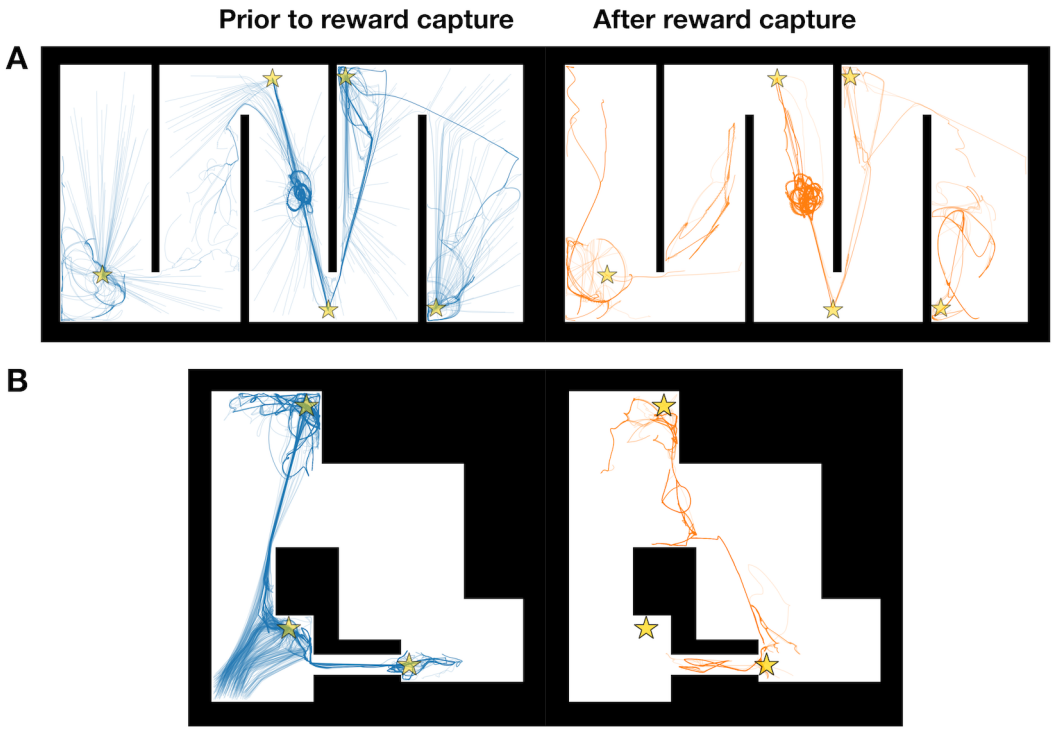

Fig. 7. Example reward-capture trajectories from selected future qEI-sampled NeuroSwarms parameters. Swarm trajectory-trace plots in the Hairpin (*A*) and Tunnel (*B*) mazes using parameters selected for mid-range values (i.e., away from parameter range limits) from predictive (anticipated future) samples generated by the trained qEI-based surrogate model.

entailed strong reward-approach exploitation (e.g., $\kappa = 6.6$), but weak swarming-based exploration (e.g., $\sigma = 2.0$). This combination of behavioral forces increased the time-to-capture for all five rewards. Thus, we attribute slow reward-capture to a combination of longer dynamical time-constant parameters and exploration–exploitation mismatches. Moreover, if the energy budget of agent locomotion (e.g., speed, turning, etc.) were to be taken into account by the objective function, a slower behavioral repertoire enabled by these parameter regimes could help to minimize energetic or inefficient navigational patterns.

4. Concluding remarks

Neuroscience-inspired learning and control methods have had increased interest in robotics, artificial intelligence, and multi-agent control. Here, we presented a demonstration of exploring and visualizing the parameter space of a multi-agent model with complex dynamical behaviors using sample-efficient Bayesian optimization with Gaussian process surrogate models. We introduced an objective function for a spatial cooperative foraging task in NeuroSwarms simulations [34] to predict reward-capture performance across two distinct maze

environments. Training the surrogate model was facilitated by the qEI and qNoisyEI acquisition functions. In particular, qEI was shown to guide optimizer trajectories towards parameter regions with high utility improvement, outperforming random sampling and manual tuning.

By learning UMAP embeddings [56], we demonstrated visualization of 9-dimensional parameter points to identify and select high performing clusters of parameters. We illustrated the identification of parameters that generalized across environments by jointly evaluating the NeuroSwarms metacontroller in two distinct maze environments. Overall, our study serves as an example application of Bayesian optimization of complex multi-agent models to explore and select for complex behaviors like goal-directed spatial navigation in a system with distributed neural control.

As parameter size grows, the computational cost of the matrix inversions required to calculate updated Gaussian process parameters increases exponentially and eventually outweighs the gains in adaptive search efficiency provided by computing the acquisition function over the surrogate model to advance the sample trajectory [20]. This limitation on model dimensionality does not, in general, prohibit analysis of complex dynamics, particularly in systems of homogeneous particles, but it would reasonably detract the feasibility of Bayesian optimization for modeling systems with nontrivial heterogeneity in agent/particle behaviors. Within that moderate limit on model complexity—e.g., for p up to ~20—Bayesian optimization may facilitate adaptive and efficient computational exploration of dynamical parameter spaces, resulting in the identification of distinct and complex system behaviors.

Future work is needed to develop new controller models and critical spatial tasks to explore the capabilities of multi-agent objective functions that adapt efficiently to the characteristics of diverse environments (e.g., geometry, distributions of rewards and sensory cues). We theorize that heterogeneous variation of swarm spatial structure and intertemporal coordination dynamics will be able to support a form of swarm metacognition that allows adjustment to the available goals in an environment, without initial knowledge of the goals or their locations. This approach could extend the flexibility of Bayesian optimization to operate in diverse environments and adapt efficiently to tasks with difficult or uncertain goals.

5. Acknowledgments

Funding for this work was provided by the National Science Foundation (NCS/FO Award No. 1835279 to GMH, KZ, KVS, and JDM), the NIH National Institute for Neurological Disorders and Stroke (NINDS R03NS109923 to KZ and JDM), and Johns Hopkins University Applied Physics Laboratory (JHUAPL) internal research and development programs (AH, GMH, and KVS). Additional support was provided to GMH by the Johns Hopkins University Kavli Neuroscience Discovery Institute and the JHUAPL Innovation and Collaboration Janney Program.

References

[1] K. M. Passino, Biomimicry for optimization, control, and automation, Springer Science & Business Media, 2005.
[2] T. D. Seeley, R. A. Morse, P. K. Visscher, The natural history of the flight of honey bee swarms, Psyche 86 (2-3) (1979) 103–113.
[3] S. Boinski, P. A. Garber, On the move: how and why animals travel in groups, University of Chicago Press, 2000.
[4] I. D. Couzin, Collective cognition in animal groups, Trends in Cognitive Sciences 13 (1) (2009) 36–43.
[5] D. J. Sumpter, Collective animal behavior, Princeton University Press, 2010.
[6] J. E. Herbert-Read, A. Perna, R. P. Mann, T. M. Schaerf, D. J. Sumpter, A. J. Ward, Inferring the rules of interaction of shoaling fish, Proceedings of the National Academy of Sciences 108 (46) (2011) 18726–18731.
[7] G. Beni, From swarm intelligence to swarm robotics, in: International Workshop on Swarm Robotics, Springer, 2004, pp. 1–9.
[8] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in: International workshop on swarm robotics, Springer, 2004, pp. 10–20.
[9] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (1) (2013) 1–41.
[10] L. Bayındır, A review of swarm robotics tasks, Neurocomputing 172 (2016) 292–321.
[11] K. Hasselmann, F. Robert, M. Birattari, Automatic design of communication-based behaviors for robot swarms, in: International Conference on Swarm Intelligence, Springer, 2018, pp. 16–29.
[12] D. S. Brown, R. Turner, O. Hennigh, S. Loscalzo, Discovery and exploration of novel swarm behaviors given limited robot capabilities, in: Distributed Autonomous Robotic Systems, Springer, 2018, pp. 447–460.
[13] M. Coppola, G. C. de Croon, Optimization of swarm behavior assisted by an automatic local proof for a pattern formation task, in: International Conference on Swarm Intelligence, Springer, 2018, pp. 123–134.
[14] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[15] J. D. Monaco, K. Rajan, G. M. Hwang, A brain basis of dynamical intelligence for AI and computational neuroscience (2021).
[16] I. C. Price, G. B. Lamont, Ga directed self-organized search and attack uav swarms, in: Proceedings of the 2006 Winter Simulation Conference, IEEE, 2006, pp. 1307–1315.
[17] N. Quijano, K. M. Passino, Honey bee social foraging algorithms for resource allocation: Theory and application, Engineering Applications of Artificial Intelligence 23 (6) (2010) 845–861.
[18] Q. Lu, J. P. Hecker, M. E. Moses, Multiple-place swarm foraging with dynamic depots, Autonomous Robots 42 (4) (2018) 909–926.
[19] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. Marshall, A. Reina, Sophisticated collective foraging with minimalist agents: a swarm robotics test, Swarm Intelligence 14 (1) (2020) 25–56.
[20] C. E. Rasmussen, Gaussian Processes in Machine Learning, in: Summer school on machine learning, Springer, 2003, pp. 63–71.
[21] J. Snoek, H. Larochelle, R. P. Adams, Practical Bayesian Optimization of Machine Learning Algorithms, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems 25, Curran Associates, Inc., 2012, pp. 2951–2959.
[22] I. Roman, J. Ceberio, A. Mendiburu, J. A. Lozano, Bayesian optimization for parameter tuning in evolutionary algorithms, in: 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 4839–4845.
[23] V. Nguyen, Bayesian optimization for accelerating hyper-parameter tuning, in: 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE), IEEE, 2019, pp. 302–305.
[24] I. Roman, A. Mendiburu, R. Santana, J. A. Lozano, Bayesian Optimization approaches for massively multi-modal problems, in: International Conference on Learning and Intelligent Optimization, Springer, 2019, pp. 383–397.
[25] E. Kieffer, M. Rosalie, G. Danoy, P. Bouvry, Bayesian Optimization to enhance coverage performance of a swarm of UAV with chaotic dynamics.
[26] A. Rai, R. Antonova, F. Meier, C. G. Atkeson, Using simulation to improve sample-efficiency of Bayesian optimization for bipedal robots, The Journal of Machine Learning Research 20 (1) (2019) 1844–1867.

[27] F. Berkenkamp, A. Krause, A. P. Schoellig, Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics, Machine Learning (2021) 1–35.
[28] M. Iwasa, K. Iida, D. Tanaka, Hierarchical cluster structures in a one-dimensional swarm oscillator model, Phys Rev E Stat Nonlin Soft Matter Phys 81 (4 Pt 2) (2010) 046220.
[29] M. Iwasa, D. Tanaka, Dimensionality of clusters in a swarm oscillator model, Phys Rev E Stat Nonlin Soft Matter Phys 81 (6 Pt 2) (2010) 066214.
[30] K. P. O'Keeffe, H. Hong, S. H. Strogatz, Oscillators that sync and swarm, Nature communications 8 (1) (2017) 1504.
[31] K. O'Keeffe, C. Bettstetter, A review of swarmalators and their potential in bio-inspired computing, in: Proc.SPIE, Vol. 10982, 2019.
[32] K. O'Keeffe, S. Ceron, K. Petersen, Collective behaviour of swarmalators on a 1D ring (2021).
[33] J. D. Monaco, G. M. Hwang, K. M. Schultz, K. Zhang, Cognitive swarming: an approach from the theoretical neuroscience of hippocampal function, in: Micro-and Nanotechnology Sensors, Systems, and Applications XI, Vol. 10982, International Society for Optics and Photonics, 2019, p. 109822D.
[34] J. D. Monaco, G. M. Hwang, K. M. Schultz, K. Zhang, Cognitive swarming in complex environments with attractor dynamics and oscillatory computing, Biological cybernetics 114 (2) (2020) 269–284.
[35] G. Buzsáki, Theta rhythm of navigation: link between path integration and landmark navigation, episodic and semantic memory, Hippocampus 15 (7) (2005) 827–840.
[36] J. D. Monaco, J. J. Knierim, K. Zhang, Sensory feedback, error correction, and remapping in a multiple oscillator model of place-cell activity, Frontiers in Computational Neuroscience 5 (2011) 39.
[37] H. T. Blair, A. Wu, J. Cong, Oscillatory neurocomputing with ring attractors: a network architecture for mapping locations in space onto patterns of neural synchrony, Philosophical Transactions of the Royal Society B: Biological Sciences 369 (1635) (2014) 20120526.
[38] J. D. Monaco, R. M. De Guzman, H. T. Blair, K. Zhang, Spatial synchronization codes from coupled rate-phase neurons, PLoS computational biology 15 (1) (2019) e1006741.
[39] A. Samsonovich, B. L. McNaughton, Path integration and cognitive mapping in a continuous attractor neural network model, J Neurosci 17 (15) (1997) 5900–5920.
[40] K. Zhang, Representation of spatial orientation by the intrinsic dynamics of the head-direction cell ensemble: a theory, Journal of Neuroscience 16 (6) (1996) 2112–2126.
[41] J. J. Knierim, K. Zhang, Attractor dynamics of spatially correlated neural activity in the limbic system, Annu Rev Neurosci 35 (2012) 267–85.
[42] A. Lansner, Associative memory models: from the cell-assembly theory to biophysically detailed cortex simulations, Trends in Neurosciences 32 (3) (2009) 178–186.
[43] A. O'Hagan, Curve fitting and optimal design for prediction, Journal of the Royal Statistical Society: Series B (Methodological) 40 (1) (1978) 1–24.
[44] D. R. Jones, M. Schonlau, W. J. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (4) (1998) 455–492.
[45] M. A. Osborne, Bayesian Gaussian processes for sequential prediction, optimisation and quadrature, Ph.D. thesis, Oxford University, UK (2010).
[46] C. Zhu, R. H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization, ACM Transactions on Mathematical Software (TOMS) 23 (4) (1997) 550–560.
[47] C. K. Williams, Prediction with Gaussian processes: From linear regression to linear prediction and beyond, in: Learning in graphical models, Springer, 1998, pp. 599–621.
[48] D. J. MacKay, Gaussian processes-a replacement for supervised neural networks?
[49] K. Krauth, E. V. Bonilla, K. Cutajar, M. Filippone, AutoGP: Exploring the capabilities and limitations of Gaussian process models, arXiv preprint arXiv:1610.05392.
[50] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, E. Bakshy, BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization, in: Advances in Neural Information Processing Systems 33, 2020.
[51] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. De Freitas, Taking the human out of the loop: A review of Bayesian optimization, Proceedings of the IEEE 104 (1) (2015) 148–175.
[52] J. T. Wilson, F. Hutter, M. P. Deisenroth, Maximizing acquisition functions for Bayesian optimization, arXiv preprint arXiv:1805.10196.
[53] B. Letham, B. Karrer, G. Ottoni, E. Bakshy, et al., Constrained Bayesian optimization with noisy experiments, Bayesian Analysis 14 (2) (2019) 495–519.
[54] W. Scott, P. Frazier, W. Powell, The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression, SIAM Journal on Optimization 21 (3) (2011) 996–1026.
[55] P. I. Frazier, A Tutorial on Bayesian Optimization, arXiv preprint arXiv:1807.02811.
[56] L. McInnes, J. Healy, J. Melville, UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, arXiv preprint arXiv:1802.03426.