Table 1: Summary of notation

$x \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$	Design parameters (controlled by system designer)
$y \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$	Exogenous parameters (not controlled by designer)
$\xi \in \Xi \subseteq \mathbb{R}^{d_\xi}$	Behavior of a system (e.g. the simulation trace)
$S : \mathcal{X} \times \mathcal{Y} \mapsto \xi$	Simulator model of the system's behavior given design and exogenous parameters
$J : \Xi \mapsto R$	Cost function
$J_r : \mathcal{X} \times \mathcal{Y} \mapsto R$	Risk-adjusted cost function
$p_{x,0}(x), p_{y,0}(y)$	Prior probability distributions for design and exogenous parameters

Summary of notation

Table 1 provides a summary of notation used in this paper.

Sampling algorithms

Algorithm 1 relies on an MCMC subrouting for sampling from probability distributions given a non-normalized likelihood. Algorithms 2 and 3 provide examples of gradient-based (Metropolis-adjusted Langevin, or MALA) and gradient-free (random-walk Metropolis-Hastings, or RMH), respectively.

Algorithm 2: Metropolis-adjusted Langevin algorithm (MALA, [16, 21])

Input: Initial x_0, steps K, stepsize τ, density $p(x)$.
Output: A sample drawn from $p(x)$.
for $i = 1, \ldots, K$ **do**
  Sample $\eta \sim \mathcal{N}(0, 2\tau I)$ ▷ Gaussian noise
  $x_{i+1} \leftarrow x_i + \tau \nabla \log p(x_i) + \eta$ ▷ Propose next state
  $P_{accept} \leftarrow \frac{p(x_{i+1}) e^{-||x_i - x_{i+1} - \tau \nabla \log p(x_{i+1})||^2/(4\tau)}}{p(x_i) e^{-||x_{i+1} - x_i - \tau \nabla \log p(x_i)||^2/(4\tau)}}$
  With probability $1 - \min(1, P_{accept})$:
    $x_{i+1} \leftarrow x_i$ ▷ Accept/reject proposal
return x_K

Algorithm 3: Random-walk Metropolis-Hastings (RMH, [25])

Input: Initial x_0, steps K, stepsize τ, density $p(x)$.
Output: A sample drawn from $p(x)$.
for $i = 1, \ldots, K$ **do**
  Sample $\eta \sim \mathcal{N}(0, 2\tau I)$ ▷ Gaussian noise
  $x_{i+1} \leftarrow x_i + \eta$ ▷ Propose next state
  $P_{accept} \leftarrow \frac{p(x_{i+1}) e^{-||x_i - x_{i+1}||^2/(4\tau)}}{p(x_i) e^{-||x_{i+1} - x_i||^2/(4\tau)}}$
  With probability $1 - \min(1, P_{accept})$:
    $x_{i+1} \leftarrow x_i$ ▷ Accept/reject proposal
return x_K

AC Power Flow Problem Definition

The design parameters $x = (P_g, |V|_g, P_l, Q_l)$ include the real power injection P_g and AC voltage amplitude $|V|_g$ at each generator in the network and the real and reactive power draws at each load P_l, Q_l; all of these parameters are subject to minimum and maximum bounds that we model using a uniform prior distribution $p_{x,0}$. The exogenous parameters are the state $y_i \in \mathbb{R}$ of each transmission line in the network; the admittance of each line is given by $\sigma(y_i) Y_{i,nom}$ where σ is the sigmoid function and $Y_{i,nom}$ is the nominal admittance of the line. The prior distribution $p_{y,0}$ is an independent Gaussian for each line with a mean chosen so that $\int_{-\infty}^{0} p_{y_i,0}(y_i) dy_i$ is equal to the likelihood of any individual line failing (e.g. as specified by the manufacturer; we use 0.05 in our experiments). The simulator $\mathcal{S}$ solves the nonlinear AC power flow equations [26] to determine the state of the network, and the cost function combines the economic cost of generation c_g (a quadratic function of P_g, P_l, Q_l) with the total violation of constraints on generator capacities, load requirements, and voltage amplitudes:

$$
\begin{aligned}
J = & c_g + v(P_g, P_{g,min}, P_{g,max}) + v(Q_g, Q_{g,min}, Q_{g,max}) && (4)\\
& + v(P_l, P_{l,min}, P_{l,max}) + v(Q_l, Q_{l,min}, Q_{l,max}) && (5)\\
& + v(|V|, |V|_{min}, |V|_{max}) && (6)
\end{aligned}
$$

where $v(x, x_{min}, x_{max}) = L\left([x - x_{max}]_+ + [x_{min} - x]_+\right)$, L is a penalty coefficient ($L = 100$ in our experiments), and $[\circ]_+ = \max(\circ, 0)$ is a hinge loss.

Efficient solutions to SCOPF are the subject of active research [27] and an ongoing competition run by the U.S. Department of Energy [28]. In addition to its potential economic and environmental impact [26], SCOPF is also a useful benchmark problem for 3 reasons: 1) it is highly non-convex, 2) it has a large space of possible failures, and 3) it can be applied to networks of different sizes to test an algorithm's scalability. We conduct our studies on one network with 14 nodes and 20 transmission lines (32 design parameters and 20 exogenous parameters) and one with 57 nodes and 80 lines (98 design parameters, 80 exogenous parameters)

The simulator S solves the nonlinear AC power flow equations [5, 29] for the AC voltage amplitudes and phase angles $(|V|, \theta)$ and the net real and reactive power injections (P, Q) at each bus (the behavior ξ is the concatenation of these values). We follow the 2-step method described in [29] where we first solve for the voltage and voltage angles at all buses by solving a system of nonlinear equations and then compute the reactive power injection from each generator and the power injection from the slack bus (representing the connection to the rest of the grid). The cost function J is a combination of the generation cost implied by P_g and a hinge loss penalty for violating constraints on acceptable voltages at each bus or exceeding the power generation limits of any generator, as specified in Eq. 6. The data for each test case (minimum and maximum voltage and power limits, demand characteristics, generator costs, etc.) are loaded from the data files included in the MATPOWER software [30].

This experiment can be run with the `solve_scacopf.py` script in the `experiments/power_systems` directory.

Search-Evasion Problem Definition

This problem includes n_{seek} seeker robots and n_{hide} hider robots. Each robot is modeled using single-integrator dynamics and tracks a pre-planned trajectory using a proportional controller with saturation at a maximum speed chosen to match that of the Robotarium platform [24]. The trajectory $\mathbf{x}_i(t)$ for each robot is represented as a Bezier curve with 5 control points $\mathbf{x}_{i,j}$,

$$
\mathbf{x}_i(t) = \sum_{j=0}^{4} \binom{4}{j} (1-t)^{4-j} t^j \mathbf{x}_{i,j}
$$

The design parameters are the 2D position of the control points for the trajectories of the seeker robots, while the exogenous parameters are the control points for the hider robots. The prior distribution for each set of parameters is uniform over the width and height of the Robotarium arena ($3.2\,\mathrm{m} \times 2\,\mathrm{m}$).

We simulate the behavior of the robots tracking these trajectories for $100\,\mathrm{s}$ with a discrete time step of $0.1\,\mathrm{s}$ (including the effects of velocity saturation that are observed on the physical platform), and the cost function is

$$J = \sum_{i=1}^{n_{hide}} \left(\widetilde{\min_{t=t_0,\dots,t_n}} \left(\widetilde{\min_{j=1,\dots,n_{seek}}} \left\| \mathbf{p}_{hide,i}(t) - \mathbf{p}_{seek,j}(t) \right\| - r \right) \right)$$

where r is the sensing range of the seekers ($0.5\,\mathrm{m}$ for the $n_{seek} = 2$ case and $0.25\,\mathrm{m}$ for the $n_{seek} = 3$ case); $\widetilde{\min}(\circ) = -\frac{1}{b}\text{logsumexp}(-b\ \circ)$ is a smooth relaxation of the element-wise minimum function where b controls the degree of smoothing ($b = 100$ in our experiments); $t_0, \dots, t_n$ are the discrete time steps of the simulation; and $\mathbf{p}_{hide,i}(t)$ and $\mathbf{p}_{seek,j}(t)$ are the (x, y) position of the i-th hider and j-th seeker robot at time t, respectively. In plain language, this cost is equal to the sum of the minimum distance observed between each hider and the closest seeker over the course of the simulation, adjusted for each seeker's search radius.

This experiment can be run with the `solve_hide_and_seek.py` script in the `experiments/hide_and_seek` directory.

Formation Control Problem Definition

This problem includes n drones modeled using double-integrator dynamics, each tracking a pre-planned path using a proportional-derivative controller. The path for each drone is represented as a Bezier, as in the pursuit-evasion problem.

The design parameters are the 2D position of the control points for the trajectories, while the exogenous parameters include the parameters of a wind field and connection strengths between each pair of drones. The wind field is modeled using a 3-layer fully-connected neural network with $\tanh$ saturation at a maximum speed that induces $0.5\,\mathrm{N}$ of drag force on each drone.

We simulate the behavior of the robots tracking these trajectories for $30\,\mathrm{s}$ with a discrete time step of $0.05\,\mathrm{s}$, and the cost function is

$$J = 10||COM_T - COM_{goal}|| + \max_t \frac{1}{\lambda_2(q_t) + 10^{-2}}$$

where COM indicates the center of mass of the formation and $\lambda_2(q_t)$ is the second eigenvalue of the Laplacian of the drone network in configuration q_t. The Laplacian $L = D - A$ is defined in terms of the adjacency matrix $A = \{a_{ij}\}$, where $a_{ij} = s_{ij}\sigma\left(20(R^2 - d_{ij}^2)\right)$, d_ij is the distance between drones i and j, R is the communication radius, and s_{ij} is the connection strength (an exogenous parameter) between the two drones. The degree matrix D is a diagonal matrix where each entry is the sum of the corresponding row of A.

This experiment can be run with the `solve.py` script in the `experiments/formation2d` directory.

Hyperparameters

Table 2 includes the hyperparameters used for each environment.

Table 2: Hyperparameters used for each environment.

Environment	n_x	n_y	τ	K	M	Quench rounds
Formation (5 agents)	5	5	10^{-3}	50	5	5
Formation (10 agents)	5	5	10^{-3}	50	5	5
Search-evasion (6 seekers, 10 hiders)	10	10	10^{-2}	100	10	25
Search-evasion (12 seekers, 20 hiders)	10	10	10^{-2}	100	10	25
Power grid (14-bus)	10	10	10^{-6} for x 10^{-2} for y	100	10	10
Power grid (57-bus)	10	10	10^{-6} for x 10^{-2} for y	100	10	10