A Bayesian approach to breaking things: efficiently predicting and repairing failure modes via sampling

Anonymous Author(s)
Affiliation
Address
email

Abstract: Before autonomous systems can be deployed in safety-critical applications, we must be able to understand and verify the safety of these systems. For cases where the risk or cost of real-world testing is prohibitive, we propose a simulation-based framework for a) predicting ways in which an autonomous system is likely to fail and b) automatically adjusting the system's design to preemptively mitigate those failures. We frame this problem through the lens of approximate Bayesian inference and use differentiable simulation for efficient failure case prediction and repair. We apply our approach on a range of robotics and control problems, including optimizing search patterns for robot swarms and reducing the severity of outages in power transmission networks. Compared to optimization-based falsification techniques, our method predicts a more diverse, representative set of failure modes, and we also find that our use of differentiable simulation yields solutions that have up to 10x lower cost and requires up to 2x fewer iterations to converge relative to gradient-free techniques.

Keywords: Automatic design tools, root-cause failure analysis, optimization-as-inference

1 Introduction

From power grids to transportation and logistics systems, autonomous systems play a central, and often safety-critical, role in modern life. Even as these systems grow more complex and ubiquitous, we have already observed failures in autonomous systems like autonomous vehicles and power networks resulting in the loss of human life [1]. Given this context, it is important that we be able to verify the safety of autonomous systems *prior* to deployment; for instance, by understanding the different ways in which a system might fail and proposing repair strategies.

Human designers often use their knowledge of likely failure modes to guide the design process; indeed, systematically assessing the risks of different failures and developing repair strategies is an important part of the systems engineering process [2]. However, as autonomous systems grow more complex, it becomes increasingly difficult for human engineers to manually predict likely failures.

In this paper, we propose an automated framework for predicting, and then repairing, failure modes in complex autonomous systems. Our effort builds on a large body of work on testing and verification of autonomous systems, many of which focus on identifying failure modes or adversarial examples [3, 4, 5, 6, 7, 8], but we identify two major gaps in the state of the art. First, many existing methods [4, 5, 9, 7] use techniques like gradient descent to search *locally* for failure modes; however, in practice we are more interested in characterizing the distribution of potential failures, which requires a global perspective. Some methods exist that address this issue by taking a probabilistic approach to sample from an (unknown) distribution of failure modes [6, 10]. However, these methods suffer from a second major drawback: although they can help the designer predict a range of

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

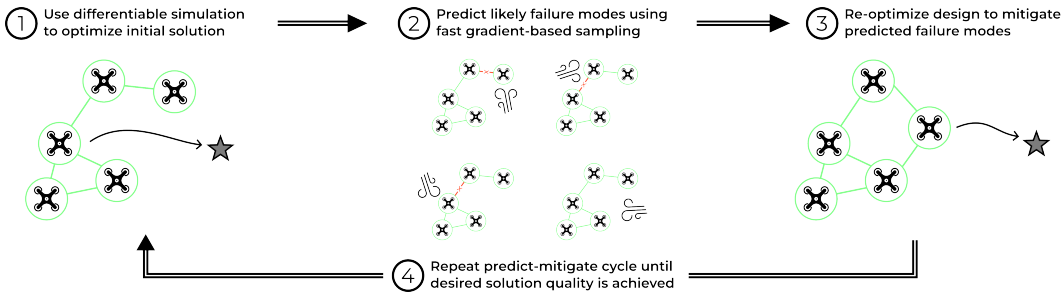


Figure 1: An overview of our method for predicting and repairing failure modes in autonomous systems, shown here handling connectivity failures in a drone swarm.

failure modes, they do not provide guidance on how those failure modes may be mitigated; they are also inefficient due to their use of gradient-free inference methods.

We address all of these drawbacks to develop a framework, shown in Fig. 1, for predicting and repairing failure modes in autonomous systems. Taking inspiration from inference-based methods [10, 6], we make three novel contributions:

1. We frame the failure prediction problem as a probabilistic sampling problem rather than deterministic optimization, allowing us to avoid local minima and quickly find high likelihood, high severity failure modes.
2. We exploit the duality between failure prediction and repair to not only predict likely failure modes but also suggest low-cost repair strategies.
3. We employ automatic differentiation to take advantage of fast gradient-based sampling algorithms, substantially improving performance relative to the state of the art.

We demonstrate our approach on several large-scale robotics and control problems: swarm formation control with up to 10 agents, multi-robot search with up to 32 agents, an electric power transmission network with up to 57 nodes and 80 transmission lines. We compare our approach with baselines for both failure mode prediction and repair, showing that our framework outperforms the state-of-the-art and scales well beyond the capabilities of existing tools, converging to solutions that are up to 10x lower cost while requiring less than half as many iterations. We also demonstrate that the repair strategies developed using our approach can be deployed on hardware for the multi-robot search example, and we include a software implementation in the supplementary materials.

2 Related Work

Model-based verification Early approaches to model-based verification and fault identification used symbolic logical models of the system under test to formally reason about failures using (computationally expensive) satisfiability (SAT) solvers or search [11, 12]. More recent approaches to model-based failure mode identification have used mathematical models of the system dynamics to frame the problem as a reachability [13] or optimal control [14] problem. The challenge in applying these methods is that it may be difficult or impossible to construct a symbolic model for the system under test. In this work, we seek to retain the interpretability of model-based techniques while eliminating the requirement for a fully symbolic model, using automatically differentiable computer programs instead. Such models are comparatively easy to construct [8] and can even include implicitly differentiable components such as the solutions to optimization problems [15].

Adversarial testing Verification using adversarial optimization has been applied in both model-based [7, 5, 9] and model-free [3] contexts. Generally speaking, model-based adversarial techniques use gradient-based optimization to locally search for adversarial examples that cause a system failure, then use gradient-based optimization to locally repair those failures [7, 5]. The drawback of

these methods is that they are inherently local and typically yield only a single adversarial counterexample. Model-free approaches [3] can avoid the issue of local minima by using zero-order black-box optimization techniques but incur additional computational cost as a result. In contrast, we take a probabilistic approach where sample-efficient gradient-based sampling algorithms can be used to escape local minima and efficiently generate multiple potential failure cases [16].

Inference Ours is not the first work to take a probabilistic approach to failure mode prediction. O'Kelly *et al.* develop an end-to-end verification pipeline for autonomous vehicles based on adaptive importance sampling [10], and Zhou *et al.* develop a failure mode prediction system based on gradient-free Markov Chain Monte Carlo (MCMC) [6]. We take inspiration from these works and make two key improvements. First, these existing works focus exclusively on predicting likely failure modes — they do not include a method for mitigating these failure modes once discovered — while we combine failure mode prediction with repair by recognizing the duality between these problems. Second, we use differentiable simulation to replace inefficient zero-order MCMC algorithms with fast gradient-based samplers, resulting in a substantial performance improvement.

There is also a complimentary body of work on algorithms for rare-event simulation [17, 18] that provide extensions to MCMC-based sampling algorithms that perform well even when we seek to simulate extremely rare failure cases. Our framework is completely compatible with rare-event simulation strategies commonly used in Sequential Monte Carlo (SMC), and we view the incorporation of these methods into our framework as a promising direction for future work.

3 Assumptions and Problem Statement

At the heart of our approach is a simulation model of the system under test, parameterized by two distinct sets of parameters. The *design parameters* $x \in \mathcal{X} \subseteq \mathbb{R}^n$ are those parameters that the system designer may vary, while the *exogenous parameters* $y \in \mathcal{Y} \subseteq \mathbb{R}^m$ are those that may vary uncontrollably (due to environmental variation, adversarial disturbance, the actions of other agents, etc.). Together, x and y define the *behavior* of the system $\xi \in \Xi$ (e.g. a trace of all relevant states and observables) through the *simulator function*, denoted $\xi = S(x, y)$. In addition, we assume that a *cost function* $J(\xi)$ is known. A summary of our notation is provided in Table 1 in the appendix; we will use "designs" and "failure modes" interchangeably with "design parameters" and "exogenous parameters", respectively. Given this context, we make the following assumptions.

Assumption 1. *S and J are programs that we can automatically differentiate almost everywhere.*

This setting is more general than the case when an explicit mathematical model is known, but less general than a black-box setting (although we note that many black-box systems in the robotics literature can be automatically differentiated [19]).

Assumption 2. *x and y are continuous random variables with known, automatically differentiable, and potentially unnormalized prior probability densities $p_{x,0}(x)$ and $p_{y,0}(y)$.*

It may be counter-intuitive to model the design parameters as random variables (since they are set by the designer), but this formulation allows us to capture the designer's degree of confidence in a choice of design parameters before the testing process begins. Parameters of which the designer is fairly certain will have densities concentrated about the prior value, while those which are still undetermined will have a broader prior distribution. We can also naturally impose constraints on the design parameters by assigning zero probability to parameters that violate these constraints. We restrict our focus to the continuous-parameter case, since that is the setting where the performance gains from gradient-based sampling are most pronounced. Our approach can be extended to handle mixed discrete parameters using block-resampling [20], but at the cost of increased computation.

In this context, *failure prediction* entails finding exogenous parameters y^* that, for some given x, lead to a high cost. To ensure that predicted failures are plausible, we must also find values for y^* with high prior likelihood. To achieve this balance, we define the metric of *risk-adjusted cost*

$$J_r(x, y) = J \circ S(x, y) + \log p_{y,0}(y) \tag{1}$$

where $\circ$ denotes function composition. Failure prediction is thus the problem of finding parameters y^* that lead to a high risk-adjusted cost; moreover, since it is likely that J_r will have multiple local minima with respect to y (i.e. multiple likely failure modes), we wish to sample a set $\left\{y_1^*, \ldots, y_{n_y}^*\right\}$ of such failures. To generate this set, we convert the deterministic optimization problem $y^* = \arg\min_y J_r(x, y)$ to the problem of sampling from the unnormalized probability distribution

$$y^* \sim p(y^*|x) \propto p_{y,0}(y^*) e^{J \circ S(x, y^*)} \tag{2}$$

Likewise, the *failure repair* problem seeks to find design parameters x^* that both have high prior likelihood (thus respecting the designer's prior beliefs about the design space) and result in a low cost across a range of anticipated failure modes; i.e. sampling from the unnormalized distribution

$$x^* \sim p(x^*|y_1^*, \ldots, y_{n_y}^*) \propto p_{x,0}(x^*) e^{-\sum_i J \circ S(x^*, y_i^*)/n_y} \tag{3}$$

The likelihoods (2) and (3) have similar structure to a Bayesian posterior, combining the prior likelihood with a cost-sensitive update term. As a result, we can view this as a reframing from an optimization problem to one of Bayesian inference.

4 Approach: Adversarial Inference

The primary challenge in sampling from these failure and repair distributions is that they will naturally shift as the design changes. Once the design has been updated to account for the current set of predicted failures, those failures will likely be out of date. To address this issue, we define a novel adversarial sampling method to alternate between sampling improved failure modes $\{y_1^*, \ldots, y_n^*\}$ and then sampling improved design parameters x^* to repair those failure modes, thus improving the robustness of the design while maintaining an up-to-date set of failure modes.

Our algorithm (detailed in Algorithm 1) proceeds in the style of a sequential Monte Carlo algorithm [18]. We begin by initializing n_y potential failure modes and n_x candidate designs sampled from their respective prior distributions. In each of K rounds, we first sample n_x new candidate designs from distribution (3) to repair the current set of predicted failure modes. We then select the design that performs best against all currently-predicted failures and sample n_y new sets of exogenous parameters (each representing a potential failure mode) from distribution (2). To sample from distributions (2) and (3), we use n_x and n_y parallel executions of a Markov chain Monte Carlo (MCMC) sampler. In order to handle potential multimodality in the design and failure space, we include optional tempering to interpolate between the prior and target distributions [18].

Our proposed adversarial inference algorithm can accept any MCMC sampling algorithm as a subroutine, either gradient-free or gradient-based. In our experiments, we compare the results of using both gradient-free (random-walk Metropolis-Hastings, or RMH) and gradient-based (Metropolis-adjusted Langevin algorithm, or MALA [21]); both of these are included in the appendix. Empirically, gradient-based samplers typically converge faster, particularly on high-dimensional problems, but in cases where a differentiable simulator is not available, a gradient-free sampler will suffice.

On a theoretical level, any MCMC sampler will be sound so long as the resulting Markov chain is ergodic and satisfies detailed balance [22]. Unfortunately, there can be a large practical gap between the asymptotic guarantees provided by these conditions and practically acceptable performance. First, if the target distribution is multimodal and the modes are well-separated, then MCMC algorithms may be slow to move between modes and will yield a biased sampling distribution with any finite number of steps. To mitigate this effect, we include a tempering schedule $0 \leq \lambda_1 \leq \ldots \leq \lambda_K \leq 1$ to interpolate between the prior and target distributions and run multiple MCMC instances in parallel from different initial conditions. Empirically, we find that tempering is not always needed, but we include it for completeness.

The second potential practical challenge arises from the continuity and differentiability (or lack thereof) of the simulator and cost function $J \circ S$. Although gradient-based MCMC samplers like MALA remain sound so long as the target distribution is continuously differentiable almost everywhere (i.e. discontinuous or non-differentiable on a set of measure zero), in practice performance

Algorithm 1: Failure prediction and repair using gradient-based sampling

Input: Design and failure population sizes n_x, n_y; rounds K; substeps M; stepsize τ; tempering schedule $\lambda_1, \ldots, \lambda_K$.

Output: Robust design x^* and a set of failures $\left\{y_1^*, \ldots, y_{n_y}^*\right\}$ with high risk-adjusted cost.

Initialize candidate designs $[x]_0 = \{x_1, \ldots, x_{n_x}\}_0$ sampled from $p_{x,0}(x)$

Initialize candidate failures $[y]_0 = \left\{y_1, \ldots, y_{n_y}\right\}_0$ sampled from $p_{y,0}(y)$

for $i = 1, \ldots, K$ **do**

  $p_{x,i}(x) := p_{x,0}(x) e^{-\lambda_k / n_y \sum_{y \in [y]_{i-1}} J \circ S(x,y)}$

  $[x]_i \leftarrow \text{Sample}([x]_{i-1}, M, \tau, p_{x,i})$ ▷ Update candidate designs using predicted failures

  $p_{y,i}(y) := p_{y,0}(y) e^{\lambda_k \min_{x \in [x]_{i-1}} J \circ S(x,y)}$ ▷ Update failure predictions for new best design

  $[y]_i \leftarrow \text{Sample}([y]_{i-1}, K, \tau, p_{y,i})$

$x^* \leftarrow \arg\max_{x \in [x]_N} p_{x,i}(x)$ ▷ Choose best design

return x^*, $[y]_K$

may suffer when the target distribution has large discontinuities. Because of these issues, we design our method to be compatible with either gradient-based or gradient-free sampling algorithms, and we compare the results of using both methods in Section 5.

The final practical consideration is that although the stochasticity in our sampling-based approach can help us explore the design and failure spaces, we incur a degree of sub-optimality as a result. When using gradient-based sampling, we have the option to reduce this sub-optimality by "quenching" the solution: switching to simple gradient descent (e.g. using MALA for the first 90 rounds and then gradient descent on the last 10 rounds). In practice, we find that quenching can noticeably improve the final cost without compromising the diversity of predicted failure modes.

5 Experimental Results

There are two questions that we must answer in this section: first, does reframing this problem from optimization to inference lead to better solutions (i.e. lower cost designs and predicted failures that accurately cover the range of possible failures)? Second, does gradient-based MCMC with differentiable-simulation offer any benefits over gradient-free MCMC when used in our approach?

We benchmark our algorithm on a range of robotics and industrial control problems. We compare against previously-published adversarial optimization methods [7, 5] and compare the results of using gradient-based and gradient-free MCMC subroutines in our approach. We then provide a demonstration using our method to solve a multi-robot planning problem in hardware. We include a copy of the software used for these experiments in the supplementary materials, and we plan to release an open-source implementation upon publication.

Baselines We compare with the following baselines. **DR**: solving the design optimization problem with domain randomization $\min_x \mathcal{E}_y[J_r(x,y)]$. **GD**: solving the adversarial optimization problem $\min_x \max_y J_r(x,y)$ by alternating between optimizing a population of n_x designs and n_y failure modes using local gradient descent, as in [7, 5, 23]. We also include two versions of our method, using both gradient-free (**RMH**) and gradient-based (**MALA**) MCMC subroutines. The gradient-free version of our approach implements quenching for the last few sampling rounds. The hyperparameters used for each experiment are included in the appendix.

Environments We use three environments for our simulation studies, which are shown in Fig. 2 and described more fully in the appendix. The number of agents in each environment can be varied to test performance on different problem dimensions. **Multi-robot search:** a set of seeker robots must plan trajectories that adequately cover a search region to ensure detection (within a fixed sensing radius) of a set of hider robots. The design and exogenous parameters define trajectories for

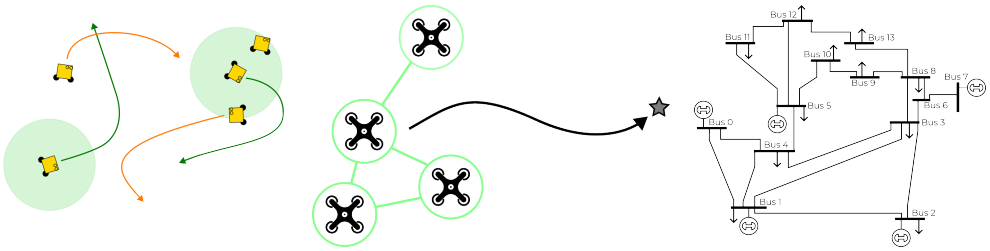


Figure 2: Three environments used in our experiments. (Left) Multi-agent hide and seek. (Middle) Formation control. (Right) Power dispatch on a 14-bus test network.

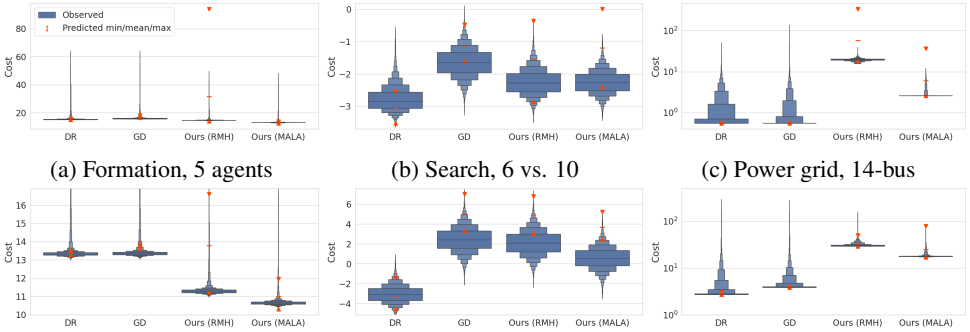
the seekers and hiders, respectively; failure occurs if any of the hiders escape detection. This environment has small (6 seeker vs. 10 hider, $\dim x = 60$, $\dim y = 100$) and large (12 seeker vs. 20 hider, $\dim x = 120$, $\dim y = 200$) versions. **Formation control:** a swarm of drones must fly to a goal configuration while maintaining full connectivity within a limited communication radius. The design parameters define trajectories for each robot in the swarm, while the exogenous parameters are the parameters of a neural network that defines an uncertain wind velocity field. The cost measures the second eigenvalue of the graph Laplacian of the swarm (which is positive if the network is fully connected); failures occur when this connectivity measure is too close to zero. This environment has small (5 agent, $\dim x = 30$, $\dim y = 1280$) and large (10 agent, $\dim x = 100$, $\dim y = 1280$) versions. **Power grid dispatch:** a fleet of electric generators must be scheduled to minimize economic cost of generation and ensure that the network satisfies voltage and maximum power constraints in the event of transmission line outages. The design parameters are generator setpoints and the exogenous parameters are line admittances; failures occur when any of the voltage or power constraints are violated. This environment has small (14-bus, $\dim x = 32$, $\dim y = 20$) and large (57-bus, $\dim x = 98$, $\dim y = 80$) versions. We implement our method and all baselines in Python using the JAX framework for automatic differentiation. Experiments were run on a laptop with 8 GB of RAM, an 8-core $1.8\,\text{GHz}$ CPU, and no GPU. All methods were run with the same population sizes and total sample budget.

Solution quality For each environment, we first solve for an optimized design and a set of predicted failure modes using each method. We then compare the performance of the optimal design on the predicted failure modes with the performance observed on a large test set of 10^5 randomly sampled exogenous parameters. The results of this experiment are shown in Fig. 3.

We find that both DR and GD often fail to predict failure modes that accurately cover the tail of worst-case behaviors: in the formation and power grid examples, both DR and GD falsely indicate that all predicted failures have been successfully repaired, despite a long tail of possible failures in both cases. In the search example, adversarial GD is able to predict a set of useful failure modes, but DR fails to do so. Only our method (with both gradient-free and gradient-based MCMC) accurately predicts the worst-case performance of the optimized design.

In addition to comparing the quality of the predicted failure modes, we can also compare the performance and robustness of the optimized design itself. On the search problem, our method finds designs with slightly improved performance relative to GD (but not relative to DR, since DR is not optimizing against a challenging set of predicted failure modes). On the formation problem, our method is able to find substantially higher-performance designs than either baseline method. On the power grid problem, our method finds designs that incur a higher best-case cost, since this problem includes a tradeoff between economic cost and robustness, but our method's designs are substantially more robust than the baselines, with much lighter tails in the cost distribution.

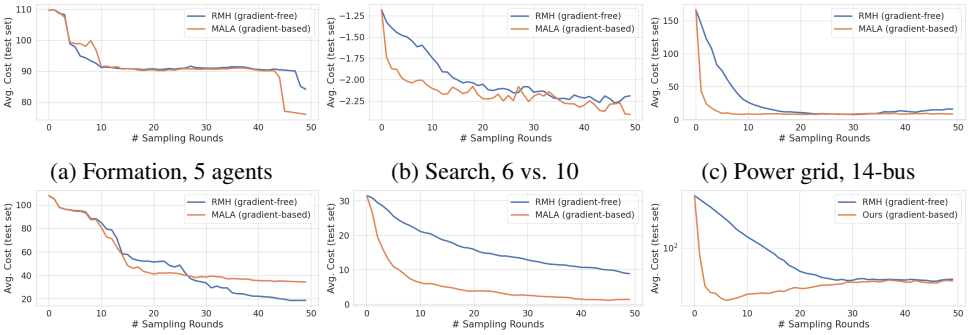
Benefits of differentiable simulation Although we have designed our method to be compatible with either gradient-based or gradient-free MCMC subroutines, we observe that gradient-based samplers tend to converge faster than their gradient-free counterparts. We measure the performance of



(a) Formation, 5 agents (b) Search, 6 vs. 10 (c) Power grid, 14-bus

(d) Formation, 10 agents (e) Search, 12 vs. 20 (f) Power grid, 57-bus

Figure 3: A comparison of the cost of the optimal design on the predicted failure modes (red) and 10^5 randomly sampled test cases (blue).



(a) Formation, 5 agents (b) Search, 6 vs. 10 (c) Power grid, 14-bus

(d) Formation, 10 agents (e) Search, 12 vs. 20 (f) Power grid, 57-bus

Figure 4: Convergence rates of gradient-based (orange) and gradient-free (blue) MCMC samplers when used as subroutines for Algorithm 1.

the best-performing candidate design at each round of Algorithm 1 against a static test set of 100 randomly sampled exogenous parameters. Fig. 4 plots the convergence of gradient-based and gradient-free methods across all environments. Although gradient-free and gradient-based MCMC perform similarly on the formation problem, we see a clear pattern in the search and power grid examples where gradient-based MCMC converges much faster. Moreover, the advantage for gradient-based methods becomes more significant for higher-dimensional problems (second row).

Hardware experiments We deploy the optimized hider and seeker trajectories in hardware using the Robotarium multi-robot platform [24] (we use 3 seekers and 5 hiders, since we had difficulty testing with more agents in the limited space). We first hold the search pattern (design parameters) constant and optimize evasion patterns against this fixed search pattern, yielding the results shown on the left in Fig. 5 where the hiders easily evade the seekers. We then optimize the search patterns using our approach, yielding the results on the left where the hiders are not able to evade the seekers. Videos comparing sub-optimal straight-line search patterns and optimized trajectories for both 2-hider, 2-seeker and 5-hider, 3-seeker games are provided in the supplementary materials.

6 Discussion and Conclusion

Before sending any autonomous system out into the real world, it is important to understand how it will behave in range of operational conditions, including during potential failures. In this paper, we have presented a tool to allow the designers of autonomous systems to not only predict the ways in which a system is likely to fail but also automatically adjust their designs to mitigate those failures.

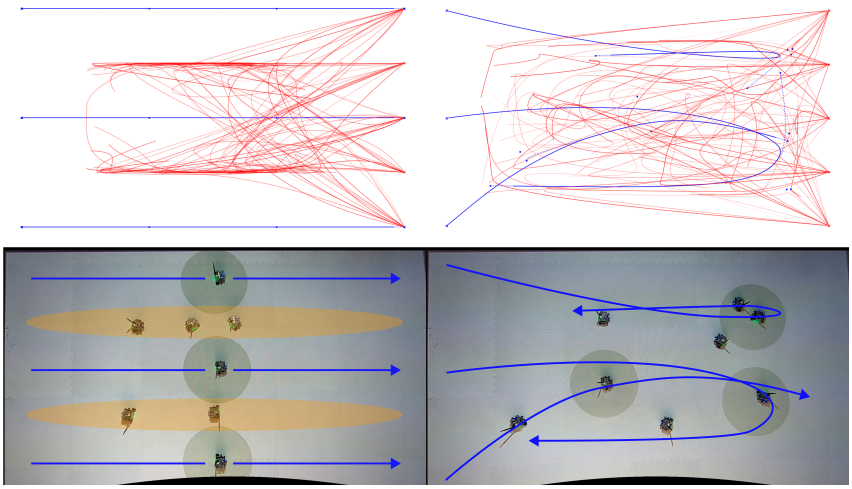


Figure 5: HW results for search-evasion with 5 hiders and 3 seekers. A basic search pattern (left, blue) leaves many places to hide (lower left, orange). The red trajectories show the distribution of predicted failures y^*, where darker trajectories better evade the seekers. Our method yields an optimized search pattern (right) that leaves fewer hiding places. A video is included in the supplementary materials.

We apply our framework in simulation studies to a range of robotics and industrial control problems, including multi-robot trajectory planning and power grid control. Our results show that, relative to existing adversarial optimization methods, our novel sampling-based approach yields better predictions of possible failure modes, which in turn lead to more robust optimized designs. We also show empirically that, when it is possible to define a differentiable simulator, gradient-based MCMC methods allow our method to converge more than twice as fast as gradient-free methods.

We hope that our framework will support the work of humans designing autonomous systems and build towards a more scalable design-analysis-optimization feedback loop.

6.1 Limitations

We have so far restricted our attention to problems with continuous design and exogenous parameters, since gradient-based inference methods realize the greatest benefit on problems with a continuous domain. It is possible to extend our method to problems with mixed continuous-discrete domains by using a gradient-based sampling algorithm for the continuous parameters and a gradient-free sampler for the discrete parameters; we hope to explore the performance implications of this combination in future work. In addition, although the gradient-based methods we use in this paper remain theoretically sound when the simulator and cost landscape are discontinuous (as is the case in robot manipulation problems, for example), it remains to be seen what practical effects discontinuity might have on convergence rate and solution quality. Finally, although we include details on how to use tempering with our approach, we found that tempering was not needed for convergence on any of the problems we studied; more work is needed to understand when tempering is necessary for convergence of MCMC-based algorithms on various robotics problems.

References

[1] University of Texas at Austin. The Timeline and Events of the February 2021 Texas Electric Grid Blackouts. Technical report, University of Texas at Austin, July 2021.

[2] R. Shishko. *NASA Systems Engineering Handbook*. Number 6105 in NASA SP. National Aeronautics and Space Administration, Washington, D.C.?, 1995.

[3] A. Corso, R. J. Moss, M. Koren, R. Lee, and M. J. Kochenderfer. A Survey of Algorithms for Black-Box Safety Validation of Cyber-Physical Systems. *Journal of Artificial Intelligence Research*, 72:377–428–377–428, Oct. 2021. ISSN 1076-9757. doi:10.1613/JAIR.1.12716.

[4] A. Corso and M. J. Kochenderfer. Interpretable Safety Validation for Autonomous Vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, Sept. 2020. doi:10.1109/ITSC45102.2020.9294490.

[5] P. Donti, A. Agarwal, N. V. Bedmutha, L. Pileggi, and J. Z. Kolter. Adversarially robust learning for security-constrained optimal power flow. In *Advances in Neural Information Processing Systems*, volume 34, pages 28677–28689. Curran Associates, Inc., 2021.

[6] Y. Zhou, S. Booth, N. Figueroa, and J. Shah. RoCUS: Robot Controller Understanding via Sampling. In *5th Annual Conference on Robot Learning*, Nov. 2021.

[7] C. Dawson and C. Fan. Certifiable robot design optimization using differentiable programming. In *International Conference on Intelligent Robots (IROS)*. IEEE/RSJ, Oct. 2022.

[8] C. Dawson and C. Fan. Robust counterexample-guided optimization for planning from differentiable temporal logic. In *Robotics: Science and Systems*, June 2022.

[9] S. Yaghoubi and G. Fainekos. Gray-box Adversarial Testing for Control Systems with Machine Learning Component. *HSCC 2019 - Proceedings of the 2019 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 179–184, Dec. 2018. doi:10.48550/arxiv.1812.11958.

[10] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi. Scalable End-to-End Autonomous Vehicle Testing via Rare-event Simulation. *Advances in Neural Information Processing Systems*, 31, 2018.

[11] J. de Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1): 97–130, Apr. 1987. ISSN 0004-3702. doi:10.1016/0004-3702(87)90063-4.

[12] D. Benard, G. A. Dorais, E. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Nayak, N. Rouquette, K. Rajan, and P. Norvig. Remote Agent Experiment. Jan. 2000.

[13] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan. S-TaLiRo: A tool for temporal logic falsification for hybrid systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6605 LNCS: 254–257, 2011. ISSN 03029743. doi:10.1007/978-3-642-19835-9_21/COVER.

[14] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay. Using control synthesis to generate corner cases: A case study on autonomous driving. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2906–2917, Nov. 2018. ISSN 02780070. doi:10.1109/TCAD.2018.2858464.

[15] B. Amos and J. Z. Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 136–145, Sydney, NSW, Australia, Aug. 2017. JMLR.org.

[16] Y. A. Ma, Y. Chen, C. Jin, N. Flammarion, and M. I. Jordan. Sampling can be faster than optimization. *Proceedings of the National Academy of Sciences of the United States of America*, 116(42):20881–20885, Oct. 2019. ISSN 10916490. doi:10.1073/PNAS.1820003116/-/DCSUPPLEMENTAL.

[17] G. Rubino and B. Tuffin. Introduction to Rare Event Simulation. *Rare Event Simulation using Monte Carlo Methods*, pages 1–13, Jan. 2009. doi:10.1002/9780470745403.CH1.

[18] N. Chopin and O. Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer International Publishing, Cham, 2020. ISBN 978-3-030-47844-5.

[19] T. Howell, S. Le Cleac'h, Z. Kolter, M. Schwager, and Z. Manchester. Dojo: A differentiable simulator for robotics. *arXiv preprint arXiv:2203.00806*, 2022.

[20] D. van Ravenzwaaij, P. Cassey, and S. D. Brown. A simple introduction to Markov Chain Monte–Carlo sampling. *Psychonomic Bulletin & Review*, 25(1):143–154, Feb. 2018. ISSN 1531-5320. doi:10.3758/s13423-016-1015-8.

[21] G. O. Roberts and O. Stramer. Langevin Diffusions and Metropolis-Hastings Algorithms. *Methodology And Computing In Applied Probability*, 4(4):337–357, Dec. 2002. ISSN 1573-7713. doi:10.1023/A:1023562417138.

[22] S. J. Russell and P. Norvig. *Artificial Intelligence: a modern approach*. Pearson, 3 edition, 2009.

[23] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain. Adversarial Attacks and Defenses in Images, Graphs and Text: A Review. *International Journal of Automation and Computing*, 17(2):151–178, Apr. 2020. ISSN 1751-8520. doi:10.1007/s11633-019-1211-x.

[24] S. Wilson, P. Glotfelter, L. Wang, S. Mayya, G. Notomista, M. Mote, and M. Egerstedt. The Robotarium: Globally Impactful Opportunities, Challenges, and Lessons Learned in Remote-Access, Distributed Control of Multirobot Systems. *IEEE Control Systems Magazine*, 40(1): 26–44, Feb. 2020. ISSN 1941-000X. doi:10.1109/MCS.2019.2949973.

[25] C. Geyer. *Introduction to Markov Chain Monte Carlo*. Chapman and Hall/CRC, May 2011. ISBN 978-0-429-13850-8. doi:10.1201/b10905-6.

[26] M. B. Cain, R. P. O'Neill, and A. Castillo. History of Optimal Power Flow and Formulations. Technical report, Federal Energy Regulatory Commission, 2012.

[27] F. Capitanescu, J. L. Martinez Ramos, P. Panciatici, D. Kirschen, A. Marano Marcolini, L. Platbrood, and L. Wehenkel. State-of-the-art, challenges, and future trends in security constrained optimal power flow. *Electric Power Systems Research*, 81(8):1731–1741, Aug. 2011. ISSN 0378-7796. doi:10.1016/j.epsr.2011.04.003.

[28] U. D. of Energy. Grid Optimization Competition. https://gocompetition.energy.gov/, 2022.

[29] P. L. Donti, D. Rolnick, and J. Z. Kolter. DC3: A learning method for optimization with hard constraints, Apr. 2021.

[30] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas. MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education. *IEEE Transactions on Power Systems*, 26(1):12–19, Feb. 2011. ISSN 1558-0679. doi:10.1109/TPWRS.2010.2051168.