# Finding Unknown Unknowns using Cyber-Physical System Simulators (Extended Report)

Semaan Douglas Wehbe Stony Brook University Stony Brook, New York, USA swehbe@cs.stonybrook.edu Stanley Bak Stony Brook University Stony Brook, New York, USA stanley.bak@stonybrook.edu

## ABSTRACT

Simulation-based approaches are among the most practical means to search for safety violations, bugs, and other unexpected events in cyber-physical systems (CPS). Where existing approaches search for simulations violating a formal specification or maximizing a notion of coverage, in this work we propose a new goal for testing: to discover unknown rare behaviors by examining discrete mode sequences. We assume a CPS simulator outputs mode information, and strive to explore the sequences of modes produced by varying the initial state or time-varying uncertainties. We hypothesize that rare mode sequences are often the most interesting to a designer, and we develop two accelerated sampling algorithms that speed up the process of finding such sequences. We evaluate our approach on several benchmarks, ranging from synthetic examples to Simulink diagrams of a CPS, demonstrating in some cases a speedup of over 100x compared with a random sampling strategy.

### **KEYWORDS**

Cyber-physical Systems, Simulation, Testing

#### **ACM Reference Format:**

Semaan Douglas Wehbe and Stanley Bak. 2025. Finding Unknown Unknowns using Cyber-Physical System Simulators (Extended Report). In Proceedings of 7th International Workshop On Design Automation For CPS and IoT (DESTION '25). ACM, New York, NY, USA, 8 pages. https://doi.org/ 10.1145/nnnnnnnnn

## **1** INTRODUCTION

The systems engineering process [1] begins with a concept of operations, followed by requirements, design, implementation, integration and test, and fielding. During the course of the process, detecting misbehavior or errors earlier can result in substantial savings in time and cost. In this work we propose an additional use for integrated and CPS component simulators: finding previously unknown behaviors.

Specification-guided testing searches for *known* potential problems by looking for an unknown test input (known unknowns). In contrast, our method searches for *unknown* issues with unknown test inputs (unknown unknowns). Although not all unprecedented

DESTION '25, May 6, 2025, Irvine, CA

© 2025 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-x/YY/MM...\$15.00

https://doi.org/10.1145/nnnnnnnnnnnn

behaviors are necessarily problematic, we believe this process can often identify a small set of interesting candidate simulations that is feasible for manual review. In order for this idea to work, we must have a way to detect that a simulation exhibits a rare behavior.

The goal of this work is to analyze a cyber-physical system for which we have access to a simulator *S*. Simulators for CPS are often designed by engineers using languages like MATLAB Simulink [2], or via large and mostly opaque domain-specific simulators like CARLA [3]. Our accelerated testing approach does not require any visibility into the simulator's implementation, only its simulation outputs. We require *S* to be a deterministic function of *n* input variables, which may represent disturbances, control inputs, or initial state uncertainty. We refer to a valuation  $x \in \mathbb{R}^n$  of the input variables as an *input point*. For each system, we analyze a finite interval of input points  $\mathcal{I} = [l, u] =$  $\{x \in \mathbb{R}^n \mid l^{(d)} \leq x^{(d)} \leq u^{(d)}, d = 1, ..., n\}$ .

Tools like MATLAB Stateflow can output information about the discrete state of the simulation trace over time. Given the system's set of discrete states  $\Sigma$ , we define a simulation's *mode sequence* to be the sequence of discrete states visited during the simulation. We assume that the mode sequence y is the output of the simulator,  $S(x) = y \in \Sigma^*$ . We assume that each mode sequence represents a distinct behavior of the system, and that any two mode sequences can be compared for equality.

Let  $Y = \{y \in \Sigma^* \mid \exists x \in I, S(x) = y\}$  be the set of all mode sequences that can be produced by simulating the CPS of interest. For all  $y \in Y$ , we construct a mapping from the mode sequence y to its preimage, the set of input points that produce y upon simulation:  $\phi(y) = \{x \in I \mid S(x) = y\}$ . The input space I can be partitioned by mode sequence,  $I = \bigcup_{y \in Y} \phi(y)$ . Random simulations may waste much of the simulation budget by running simulations that repeat common or expected behaviors. We propose an accelerated testing method that finds rare mode sequences in fewer simulations than random sampling, allowing potentially unexpected or incorrect behaviors to be discovered earlier in the process. This work makes the following assumption about mode sequences and their corresponding input points:

Assumption 1 (Convex Mode Sequence Assumption).

$$\forall y \in Y, \phi(y) = CH(\phi(y))$$

where CH is the convex hull operator

$$CH(P) = \{\lambda p + (1 - \lambda)q \mid p, q \in P, \lambda \in [0, 1]\}$$
(1)

Intuitively, Assumption 1 makes sense in that nearby input points should behave similarly, up to some boundary where simulations switch from producing one mode sequence to another. Although Assumption 1 is unlikely to be strictly true in real CPS with many

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

DESTION '25, May 6, 2025, Irvine, CA

possible behaviors and large input spaces, in practice, it offers a useful criterion by which to select input points that efficiently discover rare mode sequences.

Of the first *i* input points, we refer to the subset that produces mode sequence y as  $\phi_i(y) = \{x_j \mid S(x_j) = y, j \in [1, i]\}$  We define  $Y_i$  to be the set of all mode sequences produced by at least one of the first *i* simulations. Assumption 1 gives us the following property:

$$\forall x \in CH(\phi_i(y)), \ S(x) = y \tag{2}$$

Therefore, there is no need to simulate any input point *x* that lies inside the convex hull of an existing set of simulated input points  $\phi_i(y)$ . We refer to  $R_y = CH(\phi_i(y))$  as a mode sequence region, or simply the region, corresponding to the mode sequence *y* after *i* simulations. Each region constitutes a convex subset of the input space whose elements behave the same as one another. We define  $\mathcal{R}_i = \{CH(\phi_i(y_j)) \mid j \in [1, |Y_i|]\}$  to be the set of all regions after *i* simulations.

Given a finite budget of  $\kappa$  simulations, our goal is to maximize the number of distinct mode sequences  $|Y_{\kappa}|$  produced by  $S(x_1), \ldots, S(x_{\kappa})$ . Our accelerated testing approach takes advantage of Assumption 1 to bypass simulations with familiar mode sequences. By leveraging our knowledge of prior simulation outcomes, we can focus our simulation effort on promising input points, allowing us to discover rare and potentially unexpected behaviors earlier in the testing process.

## 2 RELATED WORK

Many techniques have been proposed to analyze CPS models, ranging from verification to testing approaches. Formal methods such as reachability analysis [4, 5] compute sets of states that a hybrid system can enter in bounded time, providing strong guarantees about system behaviors. These methods usually require white-box information about symbolic differential equations, and are not applicable for most large CPS simulators that are given either in Simulink or as custom simulators in large code bases. Discrepancy function approaches [6, 7] can be used to provide probabilistic guarantees for systems by sampling trajectories from a black-box simulator. This process has been extended to gray-box simulators [8] where the discrete mode behavior of a simulator is known, as well as to control synthesis problems [9]. The assumptions for these approaches are stronger than our work, since they require knowledge of the mode transition graph and symbolic switching conditions, whereas we only require mode information as an output signal over time.

When specifications are given in a formal language like Signal Temporal Logic (STL) [10], falsification techniques can use general optimization algorithms to search for counterexamples. These methods convert a system trace to a scalar robustness score [11], seeking traces that minimize the robustness and violate the specification. Like our work, tools such as S-TaLiRo [12] and Breach [13] search over initial state and time-varying uncertainty. Falsification approaches can be effective, but they require the user to provide an STL specification, unlike our approach.

Several approaches use partial simulation segments to explore CPS behaviors. One line of work builds on the Rapidly-Exploring Random Tree algorithm [14] from robot motion planning in order to search for safety violations [15–17]. Multi-shooting methods [18], in contrast, use partial simulations to construct a discrete abstraction of a system and bridge gaps between partial simulations using abstraction refinement. To construct simulation segments, these approaches require simulators that are fully observable and can be started and stopped from arbitrary states, which may be inapplicable for many large domain-specific simulators such as CARLA.

Software testing schemes often strive to search for inputs that improve code coverage, with common metrics being line coverage, condition coverage, or Modified-Condition Decision Coverage (MCDC). For CPS control software, commercial tools like Reactis [19] can automatically run tests to improve MCDC, although these only analyze software. Another approach [20] adapts fuzz testing methods from software to CPS, by defining a notion of coverage related to the physical variables. This method is also specificationfree, but unlike our work, it ignores discrete mode information from the simulator and focuses only on coverage in a continuous space.

Rare event simulation methods [21] strive to accurately estimate the probability of rare events using the importance sampling technique from statistics. These methods have been used for high fidelity autonomous driving simulators [22], but they require a continuous measure of safety, similar to a robustness score in STL, to direct the search for known rare events.

## **3 ACCELERATED TESTING ALGORITHM**

In this section, we describe two strategies for accelerated testing: Convex Rejection Sampling (CRS) and Region Distance Maximization (RDM). Each method uses the assumption that regions are convex sets of input points with identical behavior. By leveraging this assumption, these methods avoid redundant simulations, focusing simulator effort on parts of the input space where it is possible to increase the number of distinct mode sequences  $|Y_K|$ .

## 3.1 Convex Rejection Sampling

The first proposed procedure, called *Convex Rejection Sampling*, selects new input points that lie outside all existing regions using rejection sampling. CRS begins by choosing a candidate input point  $x_i \in I$  at random. We then perform  $|\mathcal{R}_{i-1}|$  convex hull containment checks to see if  $x_i$  lies within any of the existing regions. Each containment check is formulated as a linear program (LP). Because LPs scale efficiently to high dimensions, our accelerated testing approach can be applied to CPS simulators with high-dimensional input spaces. Any candidate  $x_i$  that lies within a region is rejected, since we know by Assumption 1 that simulating  $x_i$  will produce the mode sequence corresponding to the region. The procedure ends once an  $x_i$  is selected that lies outside all existing regions. One drawback of CRS is that it may select an input point that is outside—but still very close to—an existing region. Such input points contribute little toward the total explored volume of I.

#### 3.2 **Region Distance Maximization**

We introduce a second input point selection strategy called *Region Distance Maximization*. The high-level idea of this strategy is to choose an input point  $x_i$  as far away as possible from all existing regions. We expect that such an  $x_i$  will either produce a new mode sequence (thus aiding in the main goal of maximizing  $|Y_{\kappa}|$ ), or else contribute significantly toward the volume of an existing region

Finding Unknown Unknowns using Cyber-Physical System Simulators (Extended Report)

#### Algorithm 1 Accelerated Testing

**Require:** CPS simulator  $S : \mathbb{R}^n \to \Sigma^*$ , random input point selection function  $g : [\mathbb{R}^n, \mathbb{R}^n] \to \mathbb{R}^n$ , input point selector NextPt (one of CRS or RDM), simulation budget  $\kappa$ 

**Ensure:** The set  $Y_{\kappa}$  of mode sequences produced by  $\kappa$  simulations.

1:  $x_1 \leftarrow g(I), y \leftarrow S(x_1), Y_1 \leftarrow \{y\}$ 2:  $\phi_1(y) \leftarrow \{x_1\}$ 3:  $\mathcal{R}_1 \leftarrow \{\phi_1(y)\}$ 4: **for** i = 2 **to**  $\kappa$  **do**  $x_i \leftarrow \mathsf{NextPt}(\mathcal{I}, \mathcal{R}_{i-1}), y \leftarrow S(x_i)$ 5: if  $y \notin Y_{i-1}$  then 6:  $Y_i \leftarrow Y_{i-1} \cup \{y\}$ 7:  $\phi_i(y) \leftarrow \{x_i\}$ 8:  $\mathcal{R}_i \leftarrow \mathcal{R}_{i-1} \cup \{\phi_i(y)\}$ 9: else 10:  $Y_i \leftarrow Y_{i-1}$ 11: 12:  $\phi_i(y) \leftarrow \phi_{i-1}(y) \cup \{x_i\}$  $\mathcal{R}_i \leftarrow (\mathcal{R}_{i-1} \setminus \{ \mathsf{CH}(\phi_{i-1}(y)) \}) \cup \{ \mathsf{CH}(\phi_i(y)) \}$ 13: end if 14: 15: end for 16: return  $Y_{\kappa}$ 

(and the total explored volume of I). To choose an input point using RDM, we require a notion of the distance from a point to the existing regions. In Appendix A, we discuss two metrics for calculating the distance between a candidate input point  $x_i$  and a region  $R_y$ .

# 3.3 Rare Mode Sequence Discovery with Accelerated Testing

Using one of the two input point selection strategies, we now give the overall algorithm for discovering rare mode sequences. This procedure is given by Algorithm 1. First, an initial input point from I is randomly selected and simulated. This initial point forms the first region. Note that on lines 3 and 9, the new region consists of a single point, so there is no need to take its convex hull. For each subsequent simulation, we use either CRS or RDM to generate an input point that is both within I and outside all existing regions. If simulating the point produces a new mode sequence, then we create a new region. Otherwise, we add the point to the existing region corresponding to its mode sequence. The addition of a new point expands the region's convex hull and contributes toward the total explored volume of I.

## **4 EVALUATION**

In this section, we evaluate the use of accelerated testing for finding rare mode sequences in CPS simulators. We compare accelerated testing against the baseline technique of random sampling on four CPS benchmarks, from a synthetic example to a Simulink diagram of a CPS. To perform the hull-wise and point-wise distance maximizations for RDM, we use the open source tool ZOOpt<sup>1</sup> [23].

Because random sampling and accelerated testing both select input points non-deterministically, we take the average over ten trials, unless noted otherwise. The curves in Figures 2, 4, 6, and 7 show



Figure 1: A 2-dimensional Voronoi diagram created using the given Gaussian distribution, representing the input space for the Voronoi system. A few large regions take up the majority of the input space, while many smaller regions occupy a small portion in the top-right.

the simulations required to discover a given number of distinct mode sequences. Lines become dashed to indicate that at least one of the ten trials has already terminated. Following the trial's termination, all subsequent mode sequences in that trial are considered to be discovered at  $\kappa$  simulations, for the purposes of computing an average. Note the logarithmic scaling of the simulations axis.

## 4.1 Scalable Convex Voronoi Regions

We first construct a scalable synthetic example system to evaluate the improvement in mode sequence discovery afforded by our approach. The *n*-dimensional input space consists of 100 convex regions, so |Y| = 100. The size and placement of these regions are determined by the Voronoi sites. Since we expect real systems to have mode sequence regions of different sizes, we place the Voronoi sites using a Gaussian, rather than uniform, distribution. 100 *n*-dimensional Voronoi sites *v* are generated using a truncated normal distribution, with  $v^{(d)} \in [0, 100] \forall d \in [1, n], \mu = 100$ , and  $\sigma = 10$ . Figure 1 shows the approximate sizes and locations of the Voronoi regions under this distribution for 2 dimensions. To perform a "simulation," we return the unique ID of the Voronoi site closest to the *n*-dimensional input. The Voronoi site ID serves as the mode sequence of a simulation.

The Voronoi system's input space has several properties that make our approach especially advantageous. First, each region is inherently convex. Having convex regions conforms to Assumption 1, and guarantees that no mode sequences are missed by skipping input points inside existing regions. Second, the specific distribution from which the Voronoi sites are selected creates a low number of large regions, and many small regions. Having a few large regions is advantageous for our approach because we are able to skip input points anywhere inside their large convex hulls. Compared to uniform random sampling, our approach spends more time sampling from the remaining unexplored portion, thus discovering more of the small regions. Figure 2 plots the mode sequences discovered by the four techniques in five dimensions, where RDM provided

<sup>&</sup>lt;sup>1</sup>https://github.com/polixir/ZOOpt



Figure 2: Results for the Voronoi system.

an average speedup factor of over 350x. Additional results for the Voronoi benchmark are given in Appendix B.1.

#### 4.2 Navigation

The navigation benchmark (NAV) models an object's movement through the plane [24]. The plane is divided into a grid of  $1 \times$ 1 cells with unique IDs, where each cell is encoded with one of eight possible desired velocities. While inside a cell, the object experiences linear dynamics that bring its velocity toward that cell's desired velocity. The 4-dimensional continuous state consists of the object's velocity and position in the plane. The object begins within a given interval  $\mathcal{I}$  of 4-dimensional initial states in the grid. The discrete state of the system is the ID of the current cell. Discrete state transitions occur when the object touches the boundary of a neighboring grid cell. Some cells are designated as *terminal cells*; simulation ends once the object transitions to a terminal cell. The mode sequence of a simulation is the ordered list of cells that the object visits. A cell may appear in the mode sequence more than once. Figure 3 shows one of the rare behaviors in the NAV 10 benchmark, where the object exits the initial mode to the right instead of the left.

Figure 4 shows mode sequence discovery in the NAV 10 benchmark. Accelerated testing discovered an average of over 136 mode sequences in 5000 simulations, while random testing failed to discover even half as many mode sequences in 100000 simulations. Additional results for other NAV benchmarks appear in Appendix B.2. Across the 16 NAV benchmarks analyzed, CRS provided an average speedup of 2.63x, and RDM provided an average speedup of 6.24x.

## 4.3 Gearbox Meshing

The gearbox meshing benchmark (Gearbox) models the meshing of a sleeve with a gear during automotive motor-transmission from first to second gear [25]. If the sleeve is properly aligned with the gear when they meet, then the meshing process will complete successfully. Otherwise, the sleeve will collide with a gear tooth, bounce away, and reattempt the meshing process. The sleeve is modeled by a point in the 4-dimensional plane, where the position and velocity of the sleeve relative to the gear comprise the continuous state. The discrete state is one of meshed or free. Discrete transitions occur whenever the sleeve collides with the upper or



Figure 3: A rare behavior in one of the navigation benchmarks, where the object transitions out of the initial cell to the right, then proceeds clockwise to the terminal cell.



Figure 4: Mode sequence discovery rates for the NAV 10 benchmark. RDM was able to discover more than twice as many distinct behaviors in 20x fewer simulations than random sampling.

lower gear tooth-altering the sleeve's velocity and accumulating impact impulse-as well as once the meshing process completes. The simulation ends once the meshed state is reached. The sequence of discrete transitions serves as the mode sequence. We refer to the transition free  $\rightarrow$  free as 1 when the sleeve bounces against the lower tooth, and 2 when the sleeve bounces against the upper tooth. We refer to the transition free  $\rightarrow$  meshed as 3. An example simulation and its corresponding mode sequence are shown in Figure 5. Compared to the original benchmark, we choose the expanded input interval  $p_x \in [-0.017, -0.016], p_u \in [-0.005, 0.005]$ proposed in [26] to allow for a larger number of distinct behaviors. Additionally, we allow the initial velocities,  $v_x$  and  $v_y$ , to take on a range of values,  $v_x \in [-0.2364, 0.2364], v_y \in [-0.1260, 0.1260].$ Figure 6 plots the number of mode sequences discovered by each of the input point selection strategies. RDM provided more than an 8x improvement, finding 30 mode sequences in 515.8 simulations compared to 4358.4 random simulations.



Figure 5: An example trajectory of the gearbox meshing benchmark. The sleeve bounces once against the upper tooth, then twice against the lower tooth, before successfully meshing with the gear; thus, the mode sequence of this simulation is 2,1,1,3.



Figure 6: Results for the 4D gearbox meshing benchmark. RDM testing on average found novel mode sequences that random simulations never produced.

#### 4.4 Automatic Transmission

The Automatic Transmission benchmark (AT) simulates a vehicle equipped with an automatic transmission system [27]. The system's continuous states are the engine speed  $\omega$  (RPM) and vehicle velocity v (mph). The discrete state of the system is the gear  $g_{\alpha}$  for  $\alpha \in [1, 4]$ . The system is *n*-dimensional and depends deterministically on two control inputs, throttle and brake, which are piecewise constant over (n/2) fixed intervals. In the experiments below, we select 5 pairs of throttle and brake inputs, applied every 6 seconds throughout the 30-second simulation. This amounts to an *n*-dimensional input space for the system, with n = 10.

We define the mode sequence for the AT benchmark using an *Extended Bit Vector* of 13 pass/fail bits. We consider 13 STL safety specifications based on the STL properties from [27]. The mode sequence of a simulation is the 13-bit vector consisting of 1s and 0s, determined by whether each safety specification was satisfied. Although this mode sequence definition uses information besides discrete modes, it only requires simulation outputs that would be



Figure 7: Results for the 10-dimensional automatic transmission benchmark. Using the *Extended Bit Vector* definition of mode sequence, RDM produced 17 distinct behaviors with less than half as many simulations as random.

available from a black-box simulator. The 13 STL specifications are:

$\Box_{[0, 10]} v < \bar{v},$	$\bar{v} \in \{80, 85, 90, 95\}$	(four bits)				
$\Box_{[0,8]} \omega < \bar{\omega},$	$\bar{\omega} \in \{4500, 4600, 4700\}$	(three bits)				
$\Box_{[0,30]} \omega < 2000 \rightarrow \Box_{[0,8]} v < \bar{v},$						
$\bar{v} \in \{$	(two bits)					
$\Box_{[0,30]} ((\neg g_{\alpha} \land \circ g_{\alpha}) \to \circ \Box_{[0,1]} g_{\alpha}),$						
$\alpha \in \{$	(four bits)					

where  $\circ \varphi \equiv \diamond_{[0.001, 0.1]} \varphi$ . For the *Extended Bit Vector* definition of mode sequence, RDM discovered an average of 22 mode sequences, whereas random sampling averaged 18.2 in the same number of simulations. All ten trials produced at least 17 mode sequences; however, accelerated testing with RDM required only 231.9 simulations, compared to 530.7 simulations with random sampling.

#### **5 DISCUSSION AND FUTURE WORK**

The results indicate that considering convex mode sequence regions constructed from simulated input points is a promising strategy for rapidly discovering new mode sequences (and thus, unprecedented behaviors) in CPS simulators. Many system characteristics can affect the performance of our approach. Depending on the inherent convexity of mode sequence regions in the input space, accelerated testing could sometimes miss interesting areas with many distinct mode sequences. We therefore expect accelerated testing to perform well on systems with inherently convex mode sequence regions, such as the Voronoi benchmark. In the future, we intend to devise a metric that estimates the inherent convexity of a hybrid system's regions before simulating. Occasionally sampling within existing regions could also act as a safeguard against non-convex regions.

Another factor that influences the performance of our approach is the relative sizes of the regions. Accelerated testing has the greatest impact when there are a few large regions, and many smaller regions. The early input points quickly approach the hulls of the large regions, preventing many redundant simulations. On the other hand, when all regions are of uniform size, random sampling will quickly uncover all the distinct mode sequences. Another metric we intend to develop will estimate a system's distribution of region sizes and predict the improvement afforded by accelerated testing. The dimension of a system also contributes to our approach's mode sequence discovery rate. In higher dimensions, the volume of the input space becomes exponentially larger. Furthermore, many more input points are needed in order to enclose the convex hull of a region. CRS provided almost no improvement in 10 dimensions or higher, since very few of the first several thousand input points were inside an existing region. The RDM approach quickly explores portions of the input space far from previous input points. It is therefore less reliant on existing regions with large volume, and tends to discover mode sequences than CRS, particularly in higher dimensions.

## 6 CONCLUSION

In this work, we proposed a new goal for simulation-based analysis of cyber-physical systems: finding rare behaviors by analyzing mode sequence outputs of a black-box CPS simulator. We hypothesize that rare behaviors often correspond with unknown unknowns—unanticipated problems that can manifest in a complex system. As an engineer's time is limited and expensive, our method identifies the most interesting situations for manual review. We proposed two algorithms, Convex Rejection Sampling and Region Distance Maximization, that accelerate the process of finding these rare behaviors, in some cases by over two orders of magnitude. As our approach does not require a specification be provided, we believe it can be a complementary tool in a comprehensive CPS testing framework, which includes other approaches like manual feature testing, regression testing, and falsification methods.

## ACKNOWLEDGMENTS

This material is based upon work supported by the Office of Naval Research under award number N00014-22-1-2156, and the National Science Foundation under Award No. 2237229. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Navy.

#### REFERENCES

- B. S. Blanchard, W. J. Fabrycky, and W. J. Fabrycky, Systems engineering and analysis. Prentice hall Englewood Cliffs, NJ, 1990, vol. 4.
- [2] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, "Powertrain control verification benchmark," in *Proceedings of the 17th international conference on Hybrid systems: computation and control.* ACM, 2014, pp. 253–262.
- [3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.
- [4] M. Althoff, G. Frehse, and A. Girard, "Set propagation techniques for reachability analysis," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 369–395, 2021.
- [5] X. Chen and S. Sankaranarayanan, "Reachability analysis for cyber-physical systems: Are we there yet?" in NASA Formal Methods Symposium. Springer, 2022, pp. 109–130.
- [6] C. Fan and S. Mitra, "Bounded verification with on-the-fly discrepancy computation," in *International Symposium on Automated Technology for Verification and Analysis.* Springer, 2015, pp. 446–463.
- [7] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: a verification tool for stateflow models," in *International Conference on Tools and Algorithms for* the Construction and Analysis of Systems. Springer, 2015, pp. 68–82.
- [8] C. Fan, B. Qi, S. Mitra, and M. Viswanathan, "DryVR: Data-driven verification and compositional reasoning for automotive systems," in *International Conference* on Computer Aided Verification, vol. 10426 LNCS. Springer, 2017, pp. 441–461.
- [9] B. Qi, C. Fan, M. Jiang, and S. Mitra, "Dryvr 2.0: a tool for verification and controller synthesis of black-box cyber-physical systems," in *Proceedings of the* 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week), 2018, pp. 269–270.

- [10] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.
- [11] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications for continuous-time signals," *Theoretical Computer Science*, vol. 410, no. 42, pp. 4262–4291, 2009.
- [12] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *International Conference* on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2011, pp. 254–257.
- [13] A. Donzé, "Breach, a toolbox for verification and parameter synthesis of hybrid systems," in *International Conference on Computer Aided Verification*. Springer, 2010, pp. 167–170.
- [14] S. M. LaValle, Planning algorithms. Cambridge university press, 2006.
- [15] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Hybrid systems: from verification to falsification by combining motion planning and discrete search," *Formal Methods* in System Design, vol. 34, no. 2, pp. 157–182, 2009.
- [16] J. Kim, J. M. Esposito, and V. Kumar, "An rrt-based algorithm for testing and validating multi-robot controllers." in *Robotics: Science and Systems*. Boston, MA, 2005, pp. 249–256.
- [17] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in NASA Formal Methods Symposium. Springer, 2015, pp. 127–142.
- [18] A. Zutshi, J. V. Deshmukh, S. Sankaranarayanan, and J. Kapinski, "Multiple shooting, cegar-based falsification for hybrid systems," in *Proceedings of the 14th International Conference on Embedded Software*, 2014, pp. 1–10.
- [19] Reactive Systems, Inc., "Reactis product description," http://www.reactivesystems.com/index.msp.
- [20] S. Sheikhi, E. Kim, P. S. Duggirala, and S. Bak, "Coverage-guided fuzz testing for cyber-physical systems," in 2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS). IEEE, 2022, pp. 24–33.
- [21] J. A. Bucklew and J. Bucklew, Introduction to rare event simulation. Springer, 2004, vol. 5.
- [22] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, "Scalable end-toend autonomous vehicle testing via rare-event simulation," Advances in neural information processing systems, vol. 31, 2018.
- [23] Y.-R. Liu, Y.-Q. Hu, H. Qian, C. Qian, and Y. Yu, "Zoopt: Toolbox for derivative-free optimization," arXiv preprint arXiv:1801.00329, 2017.
- [24] A. Fehnker and F. Ivančić, Benchmarks for Hybrid Systems Verification. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 326–341. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24743-2\_22
- [25] H. Chen, S. Mitra, and G. Tian, "Motor-transmission drive system: a benchmark example for safety verification." in ARCH@ CPSWeek, 2014, pp. 9–18.
- [26] P. S. Duggirala and S. Bak, "Aggregation strategies in reachable set computation of hybrid systems," in Special issue of ACM Transactions on Embedded Computing Systems (TECS) associated with 16th International Conference on Embedded Software, ser. EMSOFT, 2019.
- [27] B. Hoxha, H. Abbas, and G. Fainekos, "Benchmarks for temporal logic requirements for automotive systems." ARCH@ CPSWeek, vol. 34, pp. 25–30, 2014.

Finding Unknown Unknowns using Cyber-Physical System Simulators (Extended Report)

## A POINT-TO-REGION DISTANCE METRICS

The first metric, which we call the *hull-wise distance*, measures the distance from  $x_i$  to the convex hull of a region. In order to formulate the hull-wise distance calculation as an LP, we seek the smallest radius  $r \in \mathbb{R}$  such that an axis-aligned hypercube, with radius r and center  $x_i$ , intersects the region. Note that the intersection can be any convex combination of the previously simulated input points. We represent the axis-aligned hypercube C centered at  $p \in \mathbb{R}^n$  with radius  $r \in \mathbb{R}$  as

$$C(p, r) = \{ q \in \mathbb{R}^n \mid p - r \, \vec{1}^{(n)} \le q \le p + r \, \vec{1}^{(n)} \}$$
(3)

where  $\vec{1}^{(n)}$  is the *n*-dimensional vector of all ones;  $r \vec{1}^{(n)}$  represents scalar-vector multiplication; and  $a \leq b$  performs element-wise comparison, satisfied only if  $a^{(d)} \leq b^{(d)} \quad \forall d \in [1, n]$ . We calculate the hull-wise distance between an input point  $x_i$  and a region  $R_y$ :

HullDist
$$(x_i, R_y) = \min r$$
  
s.t.  $C(x_i, r) \cap R_y \neq \emptyset$  (4)

The second distance metric is *point-wise distance*. The point-wise distance between  $x_i$  and region  $R_y$  is the squared distance between  $x_i$  and the nearest previously simulated input point  $p \in \phi_{i-1}(y) \subseteq R_y$ . Assume that region  $R_y$  has access to the data structure containing the mapping from y to  $\phi_{i-1}(y)$ .

$$\text{PointDist}(x_i, R_y) = \min_{p \in \phi_{i-1}(y)} \sum_{d=1}^n \left( x_i^{(d)} - p^{(d)} \right)^2$$
(5)

We wish to optimize over the input space I in search of an input point  $x_i$  whose distance from the closest region is as far as possible. Formally, we seek the  $x_i$  returned by the following optimization:

$$f(I, \mathcal{R}_{i-1}) = \underset{x_i \in I}{\operatorname{argmax}} \min_{R_y \in \mathcal{R}_{i-1}} \operatorname{Dist}(x_i, R_y)$$
(6)

where Dist is one of HullDist or PointDist. To solve the maximization problem, we use zeroth-order optimization, also known as derivative-free or black-box optimization. Given a budget of  $\beta$  optimization iterations, the solution is the input point  $x_i$  with the largest distance to its closest region. We perform a convex hull inclusion check (an LP) per region for each intermediate solution, to ensure that the input point lies outside all existing regions.

A discussion of the trade-offs between the two distance metrics follows. One disadvantage of performing optimization using the hull-wise distance metric is that for every input point  $x_i$ , RDM must perform  $\beta$  LPs (one at each optimization iteration, to calculate the hypercube radius of each intermediate solution). However, hullwise distance has the benefit that with a large enough optimization budget  $\beta$ , the returned solution approximates the global optimum.

The point-wise distance metric offers the advantage of faster computation time by avoiding the need to solve an LP for each intermediate solution. Instead, the primary calculation required to find the point-wise squared distance is a simple dot product  $D \cdot D^T$ , where *D* represents the difference between the intermediate solution  $x_i$  and each of the previously simulated input points:

$$D = \begin{bmatrix} x_1^{(1)} - x_i^{(1)} & \dots & x_{i-1}^{(1)} - x_i^{(1)} \\ \vdots & \ddots & \vdots \\ x_1^{(n)} - x_i^{(n)} & \dots & x_{i-1}^{(n)} - x_i^{(n)} \end{bmatrix}$$
(7)

**Table 1: Voronoi Mode Sequence Discovery** 

	κ	$ Y_{\kappa} $	Random	Speedup
n = 2				
CRS	1000	95.2	16329.5	16.3
RDM (hull)	300	90.6	10830.8	36.1
RDM (point)	1200	99.7	30551.2	25.5
n = 3				
CRS	2000	75.1	22133.4	11.1
RDM (hull)	500	88.3	46719.5	93.4
RDM (point)	2500	95	84934.2	34.0
n = 5				
CRS	5000	27.8	9907.2	1.98
RDM (hull)	1000	69.8	354345.9	354.3
RDM (point)	10000	50.5	111991.1	11.2
<i>n</i> = 10				
CRS	1000	24.4	765.3	0.77
RDM (hull)	1000	51	369151.7	369.2
RDM (point)	10000	40.1	56578.3	5.7

The disadvantage of performing optimization with point-wise distance is that it might converge to an  $x_i$  that is inside an existing region. Take for example a region that is a large *n*-dimensional simplex. Using point-wise distance, RDM might return the center of this simplex as its optimal solution, because it is technically far from all previously simulated input points. To prevent this solution from being returned in future optimization attempts, when using point-wise distance, we insert every input point returned by the optimizer into the region, but we only simulate input points when they lie outside all existing regions.

## **B** ADDITIONAL EVALUATION

Tables 1 and 2 give the number of distinct mode sequences  $|Y_{\kappa}|$  found by accelerated testing within a simulation budget of  $\kappa$ . We compare this against the average number of random simulations required to find this number of mode sequences. In cases where accelerated testing averages to a non-integer number of mode sequences, we consider the number of random simulations required to find the floor,  $\lfloor |Y_{\kappa}| \rfloor$ . In these cases, the listed speedup factor is a conservative estimate, since the extra mode sequence discovered by some accelerated testing trials could be exceedingly rare.

The speedup factor is calculated as:

Speedup Factor = 
$$\frac{\text{Avg. Rand Sims}}{\kappa}$$
 (8)

Note that we are underapproximating the true speedup, as accelerated testing may have discovered the  $|Y_{\kappa}|^{\text{th}}$  mode sequence earlier than the  $\kappa^{\text{th}}$  simulation, whereas random simulations were halted immediately upon finding the  $|Y_{\kappa}|^{\text{th}}$  mode sequence.

#### **B.1 Scalable Convex Voronoi Regions**

Table 1 lists the full results for each input point selection strategy in the synthetic scalable Voronoi regions benchmark. Figure 8 plots the average number of mode sequences discovered. All four strategies struggled when searching for mode sequences in 10 dimensions, finding roughly half as many mode sequences (about 50) as in lower

**Table 2: Navigation Mode Sequence Discovery** 

	κ	$ I_{\kappa} $	Rand Sims	Speedup
NAV 10				
CRS	5000	54.5	56519.5	11.30
RDM	2000	57 <sup>2</sup>	72133.6	36.07
NAV 11				
CRS	5000	63	23494.9	4.70
RDM	5000	84.2	72779.3	14.56
NAV 16				
CRS	5000	88.8	7445.2	1.49
RDM	5000	95.7	19634.2	3.93
NAV 17				
CRS	5000	60.1	11837.1	2.37
RDM	5000	63.2	18061.1	3.61
NAV 18	0000	00.2	1000111	0101
CRS	5000	130.1	10262.9	2.05
RDM	5000	134.6	14113 5	2.82
NAV 20	3000	101.0	11115.5	2.02
CRS	2500	38.2	7093 4	284
RDM	2000	38.2	7093.4	3 55
NAV 21	2000	50.2	7075.4	5.55
CPS	5000	Q2 Q	14240	2 85
DDM	5000	03.0 80	14247	2.05
NAV 22	3000	09	20238.5	4.05
CDS	5000	147 2	156796	2 1 4
DDM	5000	147.5	100/0.0	5.14 0.94
KDM	5000	192.0	49203.1	9.84
NAV 23	5000	200 (		1 50
CRS	5000	399.6	7655.7	1.53
RDM	3500	386.7	/120.8	2.03
NAV 24			5/10.0	4 50
CRS	5000	857.1	7618.3	1.52
RDM	5000	901.2	8287.4	1.66
NAV 25				
CRS	1000	31.4	1844.3	1.84
RDM	5000	48	23869.6	4.77
NAV 26				
CRS	1000	54.4	1352.8	1.35
RDM	5000	92.1	18085.2	3.62
NAV 27				
CRS	1000	109.8	1168.5	1.17
RDM	5000	208.9	11778.8	2.36
NAV 28				
CRS	5000	116.5	8945	1.79
RDM	5000	126.3	17894.1	3.58
NAV 29				
CRS	1000	188.9	1020	1.02
RDM	10000	535.8	20295.6	2.03
NAV 30				
CRS	7500	790.9	8811.3	1.17
RDM	5000	733.6	7037.6	1.41



Figure 8: Additional results for the Voronoi system.

dimensions (about 90). However, the RDM method was still over 300 times faster in this case compared with random sampling. The reduced performance of all methods can be explained by the curse of dimensionality. For example, in 10 dimensions, we would need  $2^{10} = 1024$  input points to enclose the convex hull of a hypercube, whereas in 2 dimensions, we would only need  $2^2 = 4$  input points. The number of rejections experienced during CRS per successful simulation is nearly zero in 10 dimensions.

#### **B.2** Navigation

Table 2 provides details about the exact speedup factor for each NAV benchmark. Note that all RDM trials for the NAV benchmarks use the point-wise distance metric. In the evaluations, we omit NAV benchmarks for which we witnessed ten or fewer distinct mode sequences, since random testing tends to be sufficient for finding the behaviors in a low number of simulations.

 $<sup>^2</sup>$  Due to the amount of computation time required for random testing, we only consider the first 57 mode sequences discovered by RDM in NAV 10. To find the full 136 mode sequences that RDM discovered, a single random trial required multiple days and more than 800000 simulations. We were therefore unable to compute the precise speedup factor for the full 136 mode sequences, but we predict that it would be around 160 times.