Falsification of Temporal Properties of Hybrid Systems Using the Cross-Entropy Method.

Sriram Sankaranarayanan
University of Colorado, Boulder, CO, USA.
srirams@colorado.edu

Georgios Fainekos
Arizona State University, Tempe, AZ, USA
fainekos@asu.edu

ABSTRACT

Randomized testing is a popular approach for checking properties of large embedded system designs. It is well known that a uniform random choice of test inputs is often sub-optimal. Ideally, the choice of inputs has to be guided by choosing the right input distributions in order to expose corner-case violations. However, this is also known to be a hard problem, in practice. In this paper, we present an application of the cross-entropy method for adaptively choosing input distributions for falsifying temporal logic properties of hybrid systems. We present various choices for representing input distribution families for the cross-entropy method, ranging from a complete partitioning of the input space into cells to a factored distribution of the input using graphical models.

Finally, we experimentally compare the falsification approach using the cross-entropy method to other stochastic and heuristic optimization techniques implemented inside the tool S-Taliro over a set of benchmark systems. The performance of the cross entropy method is quite promising. We find that sampling inputs using the cross-entropy method guided by trace robustness can discover violations faster, and more consistently than the other competing methods considered.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics—*Probabilistic algorithms (including Monte Carlo)*

General Terms

Verification

Keywords

Hybrid Systems, Testing, Robustness, Metric Temporal Logic, Monte-Carlo Simulation, Cross-Entropy Method.

1. INTRODUCTION

In this paper, we propose the use of the cross-entropy method [33, 32] for falsifying Metric Temporal Logic (MTL) properties complex hybrid systems. Testing through the random sampling of input vectors is a simple, yet popular approach for checking hybrid system models that are too complex to be verified using more rigorous formal verification techniques. However, the algorithm used for choosing input vectors is crucial for the success of randomized testing. Often, the choice of inputs needs to incorporate detailed knowledge about the system in the form of appropriate input distributions that are easy to sample from, while also providing a bias towards input choices exhibiting property violations. Such user guidance is often impractical, since it requires a detailed knowledge of the system's internals along with insights as to how this knowledge may be incorporated into an appropriate input distribution. Secondly, distributions that are ideal for exposing property violations tend to be quite hard to sample from, in practice.

In this paper, we employ a versatile approach to sampling known as the *Cross-Entropy Method* [6, 32, 33], guided by the notion of *robustness* of execution traces of continuous and hybrid systems w.r.t MTL properties [17, 18, 28], in order to solve the problem of generating test inputs that falsify a given set of MTL properties.

The robustness semantics of MTL associates a real value with each trajectory. Formally, this value denotes the radius of a cylinder around the trajectory (defined using an appropriate metric over states), such that all trajectories inside this cylinder have an identical outcome for the given property as the given reference trajectory (Cf. Figure 2). As a result, the robustness value of a trajectory provides a mathematically sound notion of distance that can be used to express how "far away" a given trace is from violating a property. The notion of robustness of MTL formulae can, in turn, be used to associate an ideal probability distribution that samples each input according to the robustness of the resulting trajectory. However, this distribution is often complex and not known in a closed form.

In this work, the *cross-entropy* (CE) method is used sample from this complex distribution. The CE method is, fundamentally, a technique for sampling from a complex probability distribution that is not necessarily known in a closed form. The applications of this method include rare-event simulation, variance reduction for estimation problems and stochastic optimization [33]. The CE method seeks to approximate the target distribution by choosing amongst a family of distributions such as piecewise uniform distributions or Gaussian distributions, that are "easy" to sample from [32, 6]. The technique iteratively searches for a specific distribution from this family that is "as close as possible" to the intended distribution. Here closeness of distributions is measured using the standard *Kullback-Liebler* (KL) divergence (also known as the cross-entropy distance) between the distributions. At each step, cross-entropy method generates samples according to a current candidate distribution from the family. Next, it uses these samples to *tilt* the current candidate distribution towards a new candidate that mini-

HSCC'12, April 17–19, 2012, Beijing, China.

mizes the empirically estimated KL divergence over the current set of samples between the new candidate and the target distribution. As a result of this iteration, the candidate distribution is seen to get closer in the sense of KL divergence to the target distribution.

Applying the cross-entropy method requires choosing a family of distributions that is easy to sample from, while at the same time able to approximate the complex distribution induced by the trajectory robustness values. We find that a natural approach is to use piecewise uniform distributions, whenever the space of inputs is bounded. Such distributions are defined by subdividing the space of input vectors into finitely many cells and associating a fixed probability of choosing an input from a given cell. However, the number of cells grows exponentially in the number of components in the input vector. Therefore, we consider the application of cross-entropy method on factored input representations, wherein the distribution is factored into various marginal probability distributions that relate a small set of input variables. In this paper, we derive the necessary rules for tilting the cross entropy method for factorings based on graphical models.

Finally, we present a prototype implementation of our approach on the S-Taliro tool for testing Simulink/Stateflow models [3] [1]. Our experimental evaluation compares the performance of the Cross-Entropy method against various other techniques for stochastic and heuristic global optimization including Monte-Carlo methods [28, 33], genetic algorithms and simple uniform random sampling.

Summary of Contributions: (a) We present the use of the CE method for falsifying MTL properties using the robustness metrics over hybrid trajectories. (b) We consider the problem of specifying a family of input distributions that can approximate complex distributions with small KL divergences, while at the same time, be parameterized by a small number of parameters. We explore the use of factored input distributions using graphical models. (c) We present an experimental evaluation of our approach over a set of benchmarks, using a prototype implementation in our tool S-Taliro, along with a comparison with other optimization approaches supported inside S-Taliro that are also guided by MTL robustness.

2. PRELIMINARIES

In this section, we present some background on system models, metric temporal logics (MTL) and roubstness of traces. More details on our approach are available from our previous work on using stochastic optimization techniques for temporal falsification [28].

2.1 Systems and Inputs

We assume a black-box model of deterministic systems including continuous, discrete, and hybrid systems that combine continuous-time dynamics with instantaneous discrete switches [1].

A system $\mathcal{S}$ maps the initial conditions $\vec{x}_0 \in \mathbb{R}^n$, and input signals $\vec{u} : [0,T] \mapsto \mathbb{R}^k$ to output values $\vec{y} : [0,T] \mapsto Y$, wherein $T > 0$ is assumed to be a large, but finite time limit[2]. We assume that the initial state $\vec{x}_0 \in X_0$ for some set $X_0 \subseteq \mathbb{R}^n$ and $\vec{u}(t) \in U \subseteq \mathbb{R}^k$ for all time $t \in [0,T]$. Furthermore, the sets X_0 and U are assumed to be *boxes*, which are Cartesian products of intervals of the form $[a_1,b_1] \times [a_2,b_2] \times \cdots \times [a_n,b_n]$, wherein $a_i, b_i \in \mathbb{R} \cup \pm\infty$. Since the system is assumed to be deterministic, its output $\vec{y} = \mathcal{S}(\vec{x}_0, \vec{u})$ can be written as a function of its initial state $\vec{x}_0$ and inputs $\vec{u}$.

[1]S-Taliro is available as a open source tool on-line at `http://sites.google.com/a/asu.edu/s-taliro`

[2]Since the goal in this paper is that of testing hybrid systems, it is not strictly necessary to define input and output maps that extend for all time.

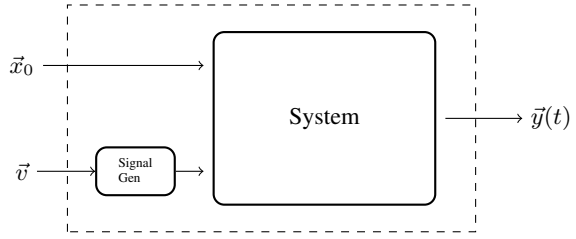


Figure 1: Block diagram of the system after parameterizing input signals.

Parameterizing Input Signals: Let $\mathcal{U}$ denote the space of all measurable functions $\vec{u} : [0,T] \mapsto \mathbb{R}^k$. The overall goal of randomized testing is to explore many points in the space the space $X_0 \times \mathcal{U}$ of inputs. However, arbitrary (measurable) input signals from a function space $\mathcal{U}$ are often hard to represent. Therefore, we restrict our attention to a class of signals from $\mathcal{U}$ that can be succinctly described in a finite dimensional space by a finite set of real valued parameters. Common examples of such families include:

- Piecewise constant (or linear) signals that are described by the values $\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_k$ at a fixed set of time points:

  $$0 = T_0 < T_1 < T_2 < \cdots < T_k < T_{k+1} = T\,,$$

  wherein $\vec{u}(t) = \vec{u}_j$ for all $t \in [T_j, T_{j+1})$ and $j = 0, \ldots, k$.
- Polynomial signals defined by parameters $\vec{a}_1, \ldots, \vec{a}_l$, wherein, $\vec{u}(t) = \vec{a}_0 + \vec{a}_1 t + \vec{a}_2 t^2 + \cdots + \vec{a}_l t^l$.
- Splines [15] can be specified piecewise by specifying a finite set of time points along with signal values and derivatives at these time points.

In practice, the family is chosen so that any signal of interest from $\mathcal{U}$ can be represented approximately with some small error. Thus, the chosen representation R for the signals ensures a finite set of parameters $\vec{v} \in V$ such that each $\vec{v}$ can be mapped onto a unique input $\mathbb{U}(\vec{v}) : [0,T] \to U$. The process of sampling described in this paper can therefore focus on sampling real numbers.

2.2 Metric Temporal Logic

Functional specifications for real-time embedded systems usually involve a number of critical properties such as timing requirements, stability and bounded response. Metric Temporal Logic (MTL) introduced by Koymans [24] is a popular formalism for expressing such properties. The problem of verifying MTL specifications is undecidable for hybrid systems. Consequently, the *bounded-time* verification or falsification of such properties has been studied [30, 31, 17].

Table 1 summarizes the syntax of MTL formulae. Let φ be an MTL formula, $t_0 \geq 0$ be a time instant and $\vec{y} : [0,T] \mapsto Y$ be an output trajectory. To define the semantics, we first define observation maps that provide meaning to the atomic propositions.

DEFINITION 2.1 (OBSERVATION MAP). *An* observation map $\mathcal{O} : AP \to \mathcal{P}(Y)$ *maps each proposition* $p \in AP$ *to a set* $\mathcal{O}(p) \subseteq Y$. *For simplicity, we assume that* $\mathcal{O}(p) \subseteq Y$ *is closed and compact for each* $p \in AP$.

We denote the satisfaction of the formula φ by the trajectory $\vec{y}$ starting from time $t = t_0$ by $(\vec{y}, t_0, \mathcal{O}) \models \varphi$. The semantics of MTL in terms of the $\models$ relation is also provided in Table 1.

Table 1: Metric Temporal Logic (MTL) Operators and their formal semantics at time $t = t_0$.

$\top$	*true*	Tautology
$p \in \text{AP}$	$\vec{y}(t_0) \in \mathcal{O}(p)$	Atomic Proposition holds.
$\varphi_1 \wedge \varphi_2$	$(\vec{y}, t_0, \mathcal{O}) \models \varphi_1 \ \wedge \ (\vec{y}, t_0, \mathcal{O}) \models \varphi_2$	Conjunction
$\varphi_1 \vee \varphi_2$	$(\vec{y}, t_0, \mathcal{O}) \models \varphi_1 \ \vee \ (\vec{y}, t_0, \mathcal{O}) \models \varphi_2$	Disjunction
$\neg \varphi$	$(\vec{y}, t_0, \mathcal{O}) \not\models \varphi$	Negation
$\Box_{\mathcal{I}} \varphi$	$(\forall t \in \mathcal{I})((t_0 + t < T) \ \Rightarrow \ (\vec{y}, t_0 + t, \mathcal{O}) \models \varphi)$	φ is Invariant in $\mathcal{I}$
$\Diamond_{\mathcal{I}} \varphi$	$(\exists t \in \mathcal{I})((t_0 + t < T) \wedge \ (\vec{y}, t_0 + t, \mathcal{O}) \models \varphi)$	φ eventually holds in $\mathcal{I}$
$\varphi_1 \mathcal{U}_{\mathcal{I}} \varphi_2$	$(\exists t \in \mathcal{I})((t_0 + t < T) \wedge \ (\vec{y}, t_0 + t, \mathcal{O}) \models \varphi_2 \ \wedge \ (\forall t' \in [0, t)) \ (\vec{y}, t_0 + t', \mathcal{O}) \models \varphi_1)$	φ_1 until φ_2

PROBLEM 2.1 (MTL FALSIFICATION). *For an MTL specification φ, the MTL falsification problem consists of finding valid initial state $\vec{x}_0$ and input signals $\vec{u} : [0, T] \to U$, such that the resulting output trajectory $\vec{y} : [0, T] \to Y$ falsifies the specification φ, i.e., $(\vec{y}, 0, \mathcal{O}) \not\models \varphi$.*

Robustness of Trajectories Our proposed solution for Problem 2.1 quantifies the *robustness of satisfaction* of an MTL formula over a system trajectory to guide the search for a falsifications [18].

We briefly present the robust interpretation (semantics) of MTL formulas. Details are available from our previous work [18, 28].

We provide semantics that maps an MTL formula φ and a trajectory $\vec{y}(t)$ to a value drawn from the linearly ordered set $\overline{\mathbb{R}}$. The semantics for the atomic propositions evaluated for $\vec{y}(t)$ consists of the distance between $\vec{y}(t)$ and the set $\mathcal{O}(p)$ labeling the atomic proposition p. Intuitively, this distance represents how robustly the point $\vec{y}(t)$ lies within (or outside) the set $\mathcal{O}(p)$.

First, let d be a distance metric on Y. For each point $\vec{y} \in Y$, we define the open ball $B_d(\vec{y}, \epsilon) = \{\vec{z} \mid d(\vec{y}, \vec{z}) < \epsilon\}$.

DEFINITION 2.2 (SIGNED DISTANCE). *Let $y \in Y$ be a point, $S \subseteq Y$ be a set and d be a distance metric on Y. We define the signed distance from y to S to be*

$$\mathbf{Dist}_d(y, S) := \begin{cases} -\inf\{d(y, y') \mid y' \in S\} & \text{if } y \notin S \\ \inf\{d(y, y') \mid y' \in Y \backslash S\} & \text{if } y \in S \end{cases}$$

If this distance is zero, then the smallest perturbation of the point y can affect the outcome of $y \in \mathcal{O}(p)$. We denote the robust valuation of the formula φ over the signal $\vec{y}$ at time t by $[\![\varphi, \mathcal{O}]\!]_d(\vec{y}, t)$. Formally, $[\![\cdot, \cdot]\!]_d : (MTL \times \mathcal{P}(Y)^{AP}) \to (Y^{[0,T]} \times [0, T] \to \overline{\mathbb{R}})$.

DEFINITION 2.3 (ROBUST SEMANTICS). *Let $\vec{y} \in Y^{[0,T]}$, $c \in \overline{\mathbb{R}}$ and $\mathcal{O} \in \mathcal{P}(Y)^{AP}$, then the robust semantics of any formula $\varphi \in MTL$ with respect to $\vec{y}$ is recursively defined as follows for $t \in [0, T]$:*

$$\begin{aligned}
[\![\top, \mathcal{O}]\!]_d(\vec{y}, t) &:= +\infty \\
[\![p, \mathcal{O}]\!]_d(\vec{y}, t) &:= \mathbf{Dist}_d(\vec{y}(t), \mathcal{O}(p)) \\
[\![\neg\varphi_1, \mathcal{O}]\!]_d(\vec{y}, t) &:= -[\![\varphi_1, \mathcal{O}]\!]_d(\vec{y}, t) \\
[\![\varphi_1 \vee \varphi_2, \mathcal{O}]\!]_d(\vec{y}, t) &:= \max([\![\varphi_1, \mathcal{O}]\!]_d(\vec{y}, t), [\![\varphi_2, \mathcal{O}]\!]_d(\vec{y}, t)) \\
[\![\varphi_1 \, \mathcal{U}_{\mathcal{I}} \varphi_2, \mathcal{O}]\!]_d(\vec{y}, t) &:= \\
&\sup_{t' \in (t +_{[0,T]} \mathcal{I})} \min \left([\![\varphi_2, \mathcal{O}]\!]_d(\vec{y}, t'), \inf_{t \geq t'' < t'} [\![\varphi_1, \mathcal{O}]\!]_d(\vec{y}, t'') \right)
\end{aligned}$$

where $t +_{[0,T]} \mathcal{I} = \{\tau \mid \exists \tau' \in \mathcal{I} \,.\, \tau = t + \tau'\} \cap [0, T]$.

It is easy to show that if the trajectory satisfies the property, then its robustness is non-negative and, similarly, it the trajectory does not satisfy the property, then its robustness is non-positive. The following result holds [18].

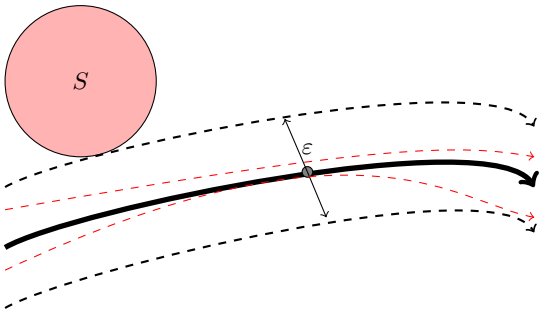


Figure 2: Illustration of robustness of trajectory shown in solid line. The property asserts the "unreachability" of the set S. The robustness value ε defines a cylinder around the trajectory, so that all trajectories that lie inside a cylinder of radius ε (dashed lines) also satisfy the property.

THEOREM 2.1. *Given a formula $\varphi \in MTL$, an observation map $\mathcal{O} \in \mathcal{P}(Y)^{AP}$ and a trajectory $\vec{y} \in Y^{[0,T]}$, the following hold:*

(1) If $(\vec{y}, t, \mathcal{O}) \models \varphi$, then $[\![\varphi, \mathcal{O}]\!]_d(\vec{y}, t) \geq 0$.

(2) Conversely, if $[\![\varphi, \mathcal{O}]\!]_d(\vec{y}, t) > 0$, then $(\vec{y}, t, \mathcal{O}) \models \varphi$.

(3) If for some $t \in \mathbb{R}^+$, $\varepsilon = [\![\varphi, \mathcal{O}]\!]_d(\vec{y}, t) \neq 0$, then for all $\vec{y}' \in B_d(\vec{y}, |\varepsilon|)$, we have $(\vec{y}, t, \mathcal{O}) \models \varphi$ if and only if $(\vec{y}', t, \mathcal{O}) \models \varphi$. I.e, ε defines a cylinder *around the trajectory such that trajectories lying entirely inside this cylinder also satisfy φ.*

In other words, if a trajectory $\vec{y}$ satisfies the formula φ at time instant $t \geq 0$ then its robustness value is non-negative.

Theorem 2.1 establishes the robust semantics of MTL as a natural measure of trajectory robustness. Namely, a trajectory is ε robust with respect to an MTL specification φ, if it can tolerate perturbations up to size ε and still maintain its current Boolean truth value. Alternatively, a trajectory with the opposite outcome for φ, if it exists, has a distance of at least ε away.

The efficient computation of MTL robustness over continuous and hybrid system trajectories has been investigated in [18, 17]. Implementations are available as part of the tool Taliro, which forms the core of our approach to temporal logic falsification [3].

2.3 Temporal Falsification as Optimization

Given a system $\mathcal{S} : X_0 \times V \mapsto Y$ (Cf. Figure 1), along with a MTL specification φ, we are interested in finding input signals $\vec{u}$ and initial conditions $\vec{x}_0$ such that the resulting output falsifies φ. More generally, we can search for inputs such that the robustness value of the resulting output trajectory w.r.t specification φ is the

least possible. Let $R(\vec{x}_0, \vec{v}; \varphi, \mathcal{S})$ denote the robustness value w.r.t φ for the trajectory $\vec{y}$ resulting from inputs $\vec{x}_0, \vec{u}$ to system $\mathcal{S}$

$$R(\vec{x}_0, \vec{v}; \varphi, \mathcal{S}) = [\![\varphi, \mathcal{O}]\!]_d(\vec{y}, 0), \text{ wherein } \vec{y} = \mathcal{S}(\vec{x}_0, \vec{v})\,.$$

Consider the optimization problem of finding inputs that yield the minimal robustness value possible:

$$(\vec{x}_0, \vec{v}) = \underset{\vec{x}_0 \in X_0, \vec{v} \in V}{\mathsf{argmin}} R(\vec{x}_0, \vec{v}; \mathcal{S}, \varphi)\,.$$

If the minimal robustness value is negative then the corresponding inputs yield a falsifying test case. Solving for $\vec{x}_0, \vec{v}$ is hard, even for the simplest of systems. Our goal is therefore to search for trajectories $\vec{y}$ that have small robustness values by sampling from the space of initial conditions X_0 and input signals V. If we discover a violation in the process, we can report such a violation to the user. Failing this, our search simply presents the least robust trajectory discovered thus far.

The strategy used for finding a trajectory with as small a robustness value of sampling is to draw samples according to a probability distribution. Let p be a probability density function over a set of support X. A sampling scheme produces a sequence of samples $x_1, \ldots, x_N \in X$, such that for any (measurable) subset $I \subseteq X$,

$$\lim_{N\to\infty} \sum_{i=1}^{N} \frac{\mathbb{1}(x_i \in I)}{N} = \int_I p(x)dx\,, \text{ where } \mathbb{1}(\varphi) = \begin{cases} 1 & \text{if } \varphi \\ 0 & \text{otherwise} \end{cases}.$$

In other words, as we draw a large number of samples, the empirical sample distribution converges in the limit to the distribution p.

Suppose we were able to draw numerous samples according to the probability distribution Ω over $X_0 \times V$, defined as

$$\Omega(\vec{x}_0, \vec{v}) = \frac{1}{W} e^{-K \cdot R(\vec{x}_0, \vec{v}; \mathcal{S}, \varphi)}\,,$$

wherein $K > 0$ is some chosen weighting factor and W is used to normalize the total mass of the distribution over $X_0 \times V$. Given the nature of the distribution, the probability of encountering inputs $\vec{x}_0, \vec{v}$ that yield negative robustness values (if such inputs exist) is exponentially larger than that of obtaining a positive robustness value. The precise ratio of these probabilities is controlled by K. Drawing samples according to Ω promises to be an effective way of searching for falsifications. However, there are two main problems: (a) Ω is not known in a closed form. To compute $\Omega(\vec{x}_0, \vec{v})$, we obtain $R(\vec{x}_0, \vec{v}; \mathcal{S}, \varphi)$ by simulating the system $\mathcal{S}$ over the inputs $(\vec{x}_0, \vec{v})$. However, the normalizing factor W is also unknown. (b) Techniques that attempt to sample inputs according to Ω often require a large number of simulations to converge [28].

An alternative approach is to start from a family of distributions $\mathcal{F}$ (eg., normal distributions) and attempt to find a distribution which is as close to Ω as possible. The tradeoff involved here is that we are sampling from a distribution family $\mathcal{F}$, which may not contain the distribution Ω. On the other hand, the distributions in $\mathcal{F}$ are chosen from known families such as normal or piecewise uniform families that are relatively easier to sample from. The cross-entropy method due to Rubinstein and Kroese attempts to solve this problem in a systematic manner [33, 32].

3. CROSS-ENTROPY METHOD

In this section, we present a brief overview of the cross entropy method which is a widely used rare-event simulation technique. Further details including a theoretical analysis of cross-entropy method are available elsewhere [32, 33, 6].

Our presentation will focus mostly on how cross-entropy method can be used to sample from a distribution Ω over the space of inputs $X_0 \times V$. As a first step, we fix a family of distributions p_θ, parameterized by a set of parameters $\theta \in P$.

- **Piecewise-Uniform Family:** The piecewise uniform family is useful when the input space $X_0 \times V$ is bounded. We partition the input space into a set of mutually disjoint measurable cells $C_1, \ldots, C_k$, wherein each C_i is bounded and has a finite volume. The family is parameterized by the individual cell sampling probabilities $\theta : (p_1, \ldots, p_k) \in [0,1]^k$, such that, $\sum_{i=1}^{k} p_i = 1$. Here p_k denotes the probability that an input from the cell C_i is chosen. In order to sample from a given distribution p_θ in the family, we choose a cell according C_i with probability p_i. Next we choose input $(\vec{x}_0, u) \in C_i$, uniformly at random.

  The piecewise uniform family can be made to approximate any distribution with arbitrary precision by (a) increasing the number of partitions and (b) by choosing the probabilities of each partition appropriately.

- **Gaussian Distribution:** A multivariate Gaussian distribution $N_{\vec{\mu},C}$ is parameterized by its mean $\vec{\mu}$ and its co-variance matrix C (a positive semi-definite matrix). These distributions (and other exponential distributions) are suitable when $X_0 \times V$ is unbounded. A simple extension to this model considers a mixture of Gaussian distributions, by averaging a fixed number of Gaussian distributions.

3.1 Overview of Cross-Entropy Method

Let Ω be a (complex) distribution over $I : X_0 \times V$ that we wish to sample from. We assume that $\Omega(\vec{x}, \vec{v}) \neq 0$ for all $(\vec{x}, \vec{v}) \in X_0 \times V$. In general, we may not know Ω as a closed form formula. However, for any two points $(\vec{x}_0, \vec{v}_0)$ and $(\vec{x}_1, \vec{v}_1)$ in the input space I, it is possible to compute the ratio $\frac{\Omega(\vec{x}_0, \vec{v}_0)}{\Omega(\vec{x}_1, \vec{v}_1)}$. Consequently, it is possible to compare the values of Ω at two or more points and rank these points according to the value of Ω.

Let p_θ be a family of distributions parameterized by $\theta \in P$. We assume that each p_θ has a set of support that contains $X_0 \times V$. In other words, $p_\theta(\vec{x}, \vec{v}) \neq 0$ for all $(\vec{x}, \vec{v}) \in X_0 \times V$. Our goal is to find a distribution p_θ from the family that is "as close" to Ω as possible. However, the notion of the "closeness" of two distributions needs to be formalized. We use the standard Kullback-Liebler (KL) divergence from information theory.

DEFINITION 3.1 (KULLBACK-LIEBLER DIVERGENCE). *Let us assume two distributions $p(\cdot)$ and $q(\cdot)$ over some set of support S, such that $\forall \vec{x} \in S, p(\vec{x}) \neq 0,\ q(\vec{x}) \neq 0$. The Kullback-Liebler (KL) divergence is defined as*

$$\mathcal{D}(p, q) = \int_S \log\left(\frac{p(x)}{q(x)}\right) p(x)dx = E_p\left[\log\left(\frac{p(x)}{q(x)}\right)\right],$$

wherein E_p denotes the expectation over the distribution p.

Note that the KL divergence is not a metric. In general, $\mathcal{D}(p, q) \neq \mathcal{D}(q, p)$. However, it can be shown that for all distributions p, q, $\mathcal{D}(p, q) \geq 0$. Furthermore, $\mathcal{D}(p, q) = 0$ iff $p = q$.

Our goal here is to choose a distribution p_θ from the chosen family that minimizes the KL divergence $\mathcal{D}(\Omega, p_\theta)$ [3]. Since Ω is not known in a closed form, it is not possible to evaluate $\mathcal{D}(\Omega, p_\theta)$ for a given θ. The idea is to adaptively search for suitable parameter values θ by performing the optimization over finitely many data points

[3]Note that this is not the same as minimizing $\mathcal{D}(p_\theta, \Omega)$. The choice of Ω as the first argument makes the minimization over samples possible without knowing Ω in a closed form.

obtained through sampling. Therefore, the cross-entropy method proceeds by approximating $\mathcal{D}(\Omega, p_\theta)$ empirically from samples and adaptively choosing values of θ. We start with some initial $\theta(0) \in P$ and iterate until some termination criterion.

(a) Draw a fixed number N_s according to p_θ using the current set of parameters $\theta = \theta(h)$.

(b) Let $\vec{x}^{(0)}, \ldots, \vec{x}^{(N_s)}$ be the samples sorted in descending order according to their Ω values.

(c) Choose the top m samples for some $m \ll N_s$.

(d) Obtain a new set of parameters $\theta(h+1)$ by *tilting*. The new parameter set $\theta(h+1)$ minimizes the empirically estimated KL divergence $\mathcal{D}(\Omega, p_\theta)$ over the sample points $\vec{x}^{(0)}, \ldots, \vec{x}^{(m-1)}$, for all $\theta \in P$.

$$\theta(h+1) = \underset{\theta \in P}{\mathsf{argmin}} \left(-\frac{1}{m} \sum_{i=0}^{m-1} \left(\frac{\log(p_\theta(\vec{x}^{(i)}))\Omega(\vec{x}^{(i)})}{p_{\theta(h)}(\vec{x}^{(i)})} \right) \right) .$$

The derivation of the formula above is shown elsewhere [32].

Termination is based upon some fixed convergence criterion. After termination, we sample extensively from the final distribution $\theta(h+1)$ to search for a possible violation.

Tilting: Given the distribution $\theta(h) \in P$ for the current iteration and the sample points $\vec{x}^{(0)}, \ldots, \vec{x}^{(m-1)}$, we seek to minimize the empirical KL divergence over the sample points

$$\theta(h+1) = \underset{\theta \in P}{\mathsf{argmin}} \left(-\frac{1}{m} \sum_{i=0}^{m-1} \left(\frac{\log(p_\theta(\vec{x}^{(i)}))\Omega(\vec{x}^{(i)})}{p_{\theta(h)}(\vec{x}^{(i)})} \right) \right) ,$$

to obtain the parameters for the subsequent iteration. Writing $\gamma_i = \frac{\Omega(\vec{x}^{(i)})}{p_{\theta(h)}(\vec{x}^{(i)})}$, we note that γ_i can be evaluated for each sample up to some fixed but unknown positive scaling factor W. However, this suffices to carry out the optimization for tilting. Simplifying, we obtain

$$\theta(h+1) = \underset{\theta \in P}{\mathsf{argmax}} \left(\sum_{i=0}^{m-1} \gamma_i \log(p_\theta(\vec{x}^{(i)})) \right) . \qquad (1)$$

The result depends, in general, on the distribution family chosen. It is relatively straightforward to solve for the optima above by computing partial derivatives w.r.t θ to obtain a closed form for standard families such as piecewise uniform and exponential distributions [32]. This yields an *updating rule* Θ_p for family p:

$$\theta(h+1) = \Theta_p(\theta(h),\ \vec{x}^{(0)}, \gamma_0,\ \ldots,\ \vec{x}^{(m-1)}, \gamma_{m-1}) . \qquad (2)$$

Updating Rules for Piecewise Uniform Distributions: Let us assume that the input space $X_0 \times V$ is partitioned into disjoint cells $C_1, \ldots, C_k$. Let $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}$ be the samples chosen for tilting. Our goal is to update the current values of the cell sampling probabilities $\vec{\theta}(h) : (\theta_{h,1}, \ldots, \theta_{h,k})$ to yield new set of parameters $\vec{\theta}(h+1)$ according to Eq. (2). This can be performed by setting the partial derivatives with respect to each unknown parameter $\theta_{h+1,j}$ to zero. The resulting update formula is given by

$$\theta_{h+1,j} = \frac{\sum_{i=0}^{m-1} \mathbb{1}(\vec{x}^{(i)} \in C_j)\gamma_i}{\sum_{i=0}^{m-1} \gamma_i} ,$$

wherein $\mathbb{1}(a \in S) = \begin{cases} 1 & \text{if } a \in S \\ 0 & \text{otherwise} \end{cases}$ In practice, the tilting is always performed *gradually* using a discount factor λ, by updating

$$\theta(h+1) = \lambda\theta(h) + (1-\lambda)\theta'(h) .$$

Updating rules for other families such as the Gaussian distributions and "natural exponential families" (NEF) are considered by Rubinstein and Kroese [32].

3.2 Illustrative Example

We now illustrate the operation of the cross-entropy method for finding an appropriate insulin infusion schedule for controlling the blood glucose level of a type II diabetic patient following the ingestion of a meal. The model and the parameters chosen have been inspired by the work of Fisher [19]. The dynamics of insulin and glucose in the patient are modeled by the ODE

$$\begin{array}{rcl} \frac{dG}{dt} & = & -p_1 G - X(G + G_b) + P(t) \\ \frac{dX}{dt} & = & -p_2 X + p_3 I \\ \frac{dI}{dt} & = & -n(I + I_b) + u(t)/V_I \end{array}$$

wherein state variable G refers to the level of glucose in the blood plasma above a fixed basal value G_b, I refers to the level of insulin above a fixed basal value I_b and X is a quantity that is proportional to the level of insulin that is effective in controlling blood glucose level. The function $P(t)$ refers to the addition of glucose in the blood after digestion. Following Fisher, we set $P(t) = ke^{-Bt}$ to model the characteristic peak and decay of the level of glucose added to the blood during the digestion process. The input $u(t)$ refers to the insulin infused directly by means of a direct infusion into the blood, wherein $0 \leq u(t) \leq 3$ for all $t \geq 0$.

For initial conditions are chosen from the intervals $G(0) \in [6, 10]$, $X(0) \in [0.05, 0.1]$ and $I(0) \in [-.1, .1]$. We wish to find an initial condition and a value of $u(t)$ that falsifies the MTL property

$$\varphi : \neg(\Box_{[0,20.0]}(G \in [-2, 10]) \ \wedge\ \Box_{[20,200.0]}(G \in [-1, 1])) .$$

Informally, φ specifies that the value of glucose should *not* be in the range $[-2, 10]$ during the first 20 minutes, or *fail to* remain the range $[-1, 1]$ over the next 180 minutes.

We ran a cross-entropy sampling guided by trace robustness to search for a suitable input. Each signal $u(t)$ is represented by a spline with 4 control points. This yields 7 input parameters. The range of permissible values for each input parameter is subdivided into 10 equally spaced subintervals. Figure 3 shows the final results of applying the cross-entropy method for 25 iterations lasting roughly 570 seconds. The technique does not find a falsifying input, it finds an input schedule that comes quite close yielding a low robustness value of 0.3. Figure 3 plots the minimal robustness trace. The run of cross-entropy method is illustrated by showing the tilted probability distributions at iteration numbers $1, 5, \ldots, 25$.

4. FACTORED INPUT DISTRIBUTIONS

In this section, we present some basic ideas on how families of input distributions may be formed and represented for applying the cross-entropy method for falsifying temporal properties. In general, there are two conflicting concerns that affect the choice of a family of distributions: (a) the ability to represent arbitrary input distributions to a good degree of approximation and (b) keeping the number of parameters that describe the family small.

We focus on piecewise uniform distributions obtained by subdividing the input space into many disjoint cells $C_1, \ldots, C_K$. Let us represent the vector of inputs in $X_0 \times V \subseteq \mathbb{R}^k$ by $(z_1, \ldots, z_k) \in \mathbb{R}^k$. We have assumed that the set of legal input values form a bounded box $[a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_k, b_k]$. We partition the set of possible values for each input variable $z_i \in [a_i, b_i]$ into a fixed number $n > 0$ different disjoint sub-intervals $U_{i,1}, \ldots, U_{i,n}$ [4].

[4]For simplicity, we assume that the number of subdivisions along each dimension is the same.

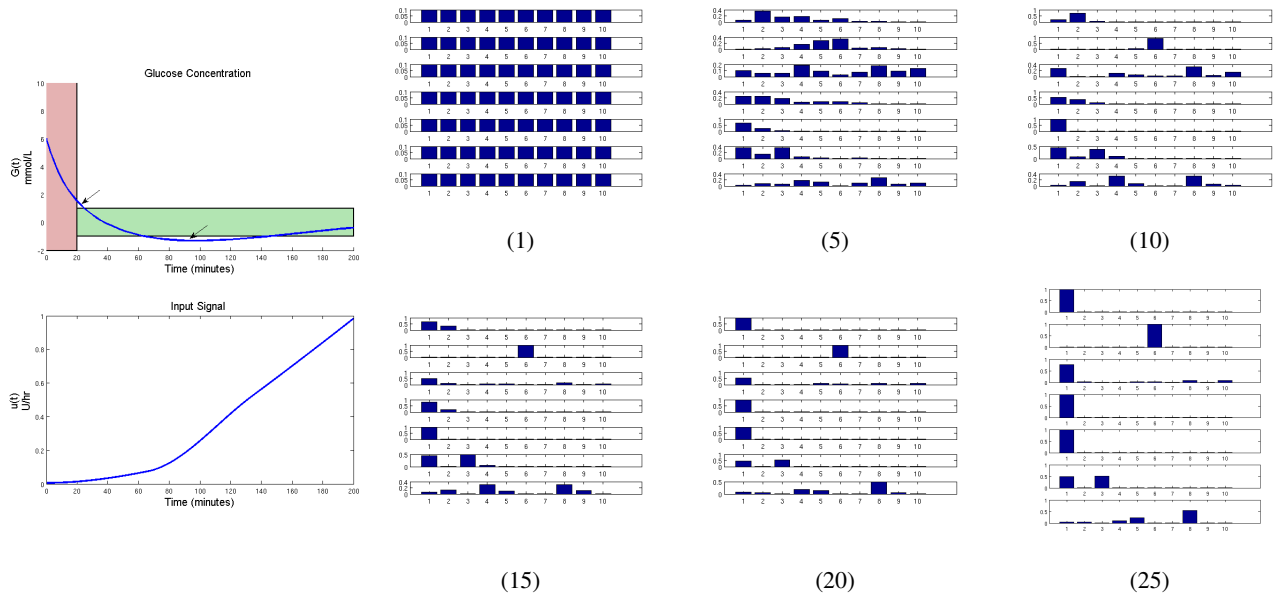


Figure 3: Results of cross-entropy method for the insulin glucose model. (Left) plot showing the least robust simulation result. (Right) Probability distributions over the various input subintervals at rounds $1, 5, \ldots, 25$.

This represents a partitioning of input space into n^k rectangular cells, wherein each cell $C_{j_1,\ldots,j_k}$ is a product of intervals $U_{1,j_1} \times U_{2,j_2} \times \cdots \times U_{k,j_k}$. As a result, representing the piecewise uniform distribution requires n^k parameters, which is exponential in the dimensionality of the input vector. Therefore, we consider tradeoffs in representing the family by means of factored distributions.

Fully Factored Distribution: A simple scheme for representing probability distributions over the cells is to associate a uniform probability value $P_{i,j}$ for each cell $U_{i,j}$ for input z_i and interval $U_{i,j}$, such that

$$\forall\, i \in [1,k] \sum_{j=1}^{n} P_{i,j} = 1\,.$$

The family of fully factored distributions are parameterized by the values $P_{i,j}$ for $i \in [1,k]$ and $j \in [1,n]$. The probability of choosing a cell $C_{j_1,\ldots,j_k}$ is assumed to be given by the product

$$\mathsf{Pr}(C_{j_1,\ldots,j_k}) = \Pi_{i=1}^{k} P_{i,j_i}\,.$$

Once a cell is chosen using the discrete distribution defined above, a point in the cell is chosen uniformly at random. In other words, the choice of a cell is obtained by independently choosing intervals for each input z_i.

We will now derive the update rule for tilting using families of fully factored piecewise uniform distributions by solving the optimization involved in Equation (1). Let $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}$ be the m samples used to compute the tilting. Furthermore, for each sample, we obtain the value $\gamma_l = \frac{W\Omega(\vec{x}^{(l)})}{p_{\theta(h)}(\vec{x}^{(l)})}$. Our goal is to compute optimal parameters $\theta^* = (P_{1,1}, \ldots, P_{k,n})$ such that

$$\theta^* = \mathsf{argmax} \left(\sum_{l=0}^{m-1} \gamma_l \log(p_\theta(\vec{x}^{(l)})) \right) \tag{3}$$

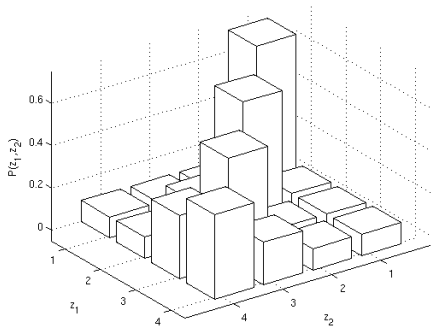


Figure 4: A distribution that cannot be factored.

We may write $\log(p_\theta(\vec{x}^{(l)}))$ as

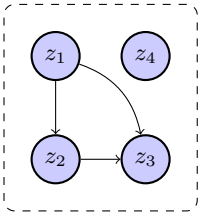
$$\log(p_\theta(\vec{x}^{(l)})) = \sum_{i=1}^{k}\sum_{j=1}^{n} \log(P_{i,j}) \mathbb{1}(\vec{x}_i^{(l)} \in U_{i,j})\,.$$

Lemma 4.1. *The optimal value of parameters $P_{i,j}$ that maximizes the objective in Equation* (3) *are given by*

$$P_{i,j} = \frac{\sum_{l=0}^{m-1} \gamma_l \mathbb{1}(\vec{x}_i^{(l)} \in U_{i,j})}{\sum_{l=1}^{m} \gamma_l}\,.$$

PROOF. This lemma is a special case of the more general Lemma 4.2 for graphical models that will be presented subsequently. □

One of the key advantages of a fully factored form of the input distribution is that the number of parameters representing the family is simply $k \times n$ as opposed to n^k. However, the assumption of independent choice of intervals along each dimension diminishes the ability to represent arbitrary probability distributions.



Node	Cond. Table Type
z_1	$P(z_1 \in U_{1,i})$
z_2	$P(z_2 \in U_{2,i} \mid z_1 \in U_{1,i_1})$
z_3	$P\left(z_3 \in U_{3,i} \;\middle|\; \begin{array}{l} z_1 \in U_{1,i_1}, \\ z_2 \in U_{2,i_2} \end{array}\right)$
z_4	$P(z_4 \in U_{4,i})$

Figure 5: Example graphical model over 4 variables $z_1, \ldots, z_4$ along with the types of tables associated with each node.

Example 4.1. *Figure 4 shows a distribution over two inputs z_1, z_2 wherein there is a significant degree of correlation between the choices of particular intervals for z_1 and for z_2. A fully factored representation loses this information to produce a poor approximation of this distribution. This situation presents interesting parallels to the general problem of abstracting sets of states that is considered in techniques such as symbolic model checking and abstract interpretation of systems.*

Graphical Model Factoring: An alternative to a fully factored representation consists of maintaining correlations between some of the input variables. In the setting of this paper, two inputs are correlated if, for the purposes of finding a falsifying input, the value chosen of one variable will affect the choice of a value for the other. Often, it is natural to consider correlations between certain classes of inputs during falsification. For instance, the value of a control input u at two adjacent time intervals can often be regarded as correlated. Depending on how input signals are parameterized, the choice of an input for the next time step may need to take the current choice of input into consideration.

In this section, we consider generic graphical models that can represent a factored probability distribution. Once again, we assume that the input space for input variable z_i has been partitioned into pairwise disjoint sub-intervals $U_{i,1}, \ldots, U_{i,n}$ for $i \in [1, k]$.

A graphical model G is a directed acyclic graph (DAG) with nodes $N = \{z_1, \ldots, z_k\}$, one node for each variable and a set of directed edges $E \subseteq N \times N$. A graphical model represents a factored distribution. For each node z_j, let $\{e_1 : z_{j_1} \to z_j, \ldots, z_{j_l} \to z_j\}$ represent the set of all incoming edges into z_j. We associate a conditional probability table $\mathbb{T}_j$ that has entries of the form

$$P(z_j \in U_{j,i} \mid z_{j_1} \in U_{j_1,i_1} \wedge \cdots z_{j_l} \in U_{j_l,i_l}), \quad \text{for } i, i_1, \ldots, i_l \in [1, n] \tag{4}$$

Thus, each variable is associated with a table of conditional probabilities for the variable belonging to a sub-interval in its domain, given a combination of choices of sub-intervals for the predecessors of the variable in the graph. The overall distribution is written as a product of its factors:

$$\mathsf{Pr}\left(\begin{array}{c} z_1 \in U_{1,l_1} \\ \ldots \\ z_k \in U_{k,l_k} \end{array}\right) = \prod_{i=1}^{k} P\left(z_i \in U_{i,l_i} \;\middle|\; \bigwedge_{z_j \to z_i \in E} z_j \in U_{j,l_j}\right).$$

A fully factored distribution is simply a graphical model G with the empty set of edges.

Example 4.2. *Figure 5 shows an example of a graphical model along with the type of conditional probability table for each node. The overall probability of drawing a sample from a cell in the input space is written out as a product of individual probabilities from each of the tables in the model according to Equation* (4)*.*

As a result, each graphical model G represents a family of piecewise uniform distributions parameterized by the entries in the tables associated with each node in the graph. The number of such parameters is bounded by $O(kn^{\Delta+1})$ wherein Δ is the maximum in-degree of any node in the graph. Consider an entry in a table of the form shown in Eq. (4). The *event associated* with the entry is denoted by the predicate:

$$z_j \in U_{j,i} \;\wedge\; z_{j_1} \in U_{j_1,i_1} \wedge \cdots \wedge\; z_{j_l} \in U_{j_l,i_l}\,.$$

Once again, we consider update rules for factored distributions for solving the optimization involved in Equation 1. Let $\vec{x}^{(1)}, \ldots, \vec{x}^{(m)}$ be the m samples used to compute the tilting with associated weights $\gamma_1, \ldots, \gamma_m$. Our goal is to compute optimal parameters $\theta^* = (P_1, \ldots, P_N)$ where each entry P_i stands for some unknown table entry representing some conditional property in the graphical model. Our goal once again is to optimize

$$\theta^* = \mathsf{argmax}\left(\sum_{l=0}^{m-1} \gamma_l \log(p_\theta(\vec{x}^{(l)}))\right) \tag{5}$$

The update rule for graphical models considers entries in a given table $\mathbb{T}_i$ associated with a node z_i, for $i \in [1, k]$. For parameter $P_{i,j}$, let $\varphi_{i,j}$ denote the associated event. Furthermore, let $\mathbb{1}(\vec{x}^{(l)} \models \varphi_{i,j})$ denote the condition that the l^{th} sample satisfies the event associated with table entry $P_{i,j}$.

Lemma 4.2. *The optimal parameter value for parameter $P_{i,j}$ in table $\mathbb{T}_i$ that maximizes the objective in Equation* (5) *is given by*

$$P_{i,j} = \frac{\sum_{l=0}^{m-1} \gamma_l \mathbb{1}(\vec{x}^{(l)} \models \varphi_{i,j})}{\sum_{l=0}^{m-1} \sum_{r \in \mathsf{Entries}(\mathbb{T}_i)} \gamma_l \mathbb{1}(\vec{x}^{(l)} \models \varphi_{i,r})}\,.$$

PROOF. A proof of this theorem is given in the appendix. □

The lemma above shows that the updating rule for factored distributions is quite simple given the samples $\vec{x}^{(l)}$ and weights γ_l.

We now briefly comment on the choice of an appropriate graphical model for factoring the input. Often, the fully factored representation is the easiest to implement. As mentioned earlier, this representation can be improved by tracking the joint distribution between the value of parameters that pertain to an input signal at the current time step to the input at the next time step. However, decisions on other inputs may need to be considered jointly for successful falsification, in practice.

Currently, it is unclear as to how such inputs may be identified. If, for instance, the function of the inputs to the system are well understood, it may sometimes be possible to classify sets as inputs as tightly coupled or otherwise. However, this requires detailed knowledge of the system's inner workings. In this regard, the problem of automatically identifying an ideal factoring of the input distribution for testing remains an open challenge.

5. IMPLEMENTATION & EXPERIMENTAL EVALUATION

We have implemented a prototype version of the techniques described thus far inside the S-Taliro framework for falsification of MTL properties. S-Taliro is implemented as a Matlab toolbox and supports the specification of a variety of system models including Simulink/Stateflow diagrams, Matlab functions and C programs interfaced with Matlab. The latest version of the tool supports various core primitives such as the specification of MTL formulas through a simple, user-friendly interface, various utilities to simulate models and visualize trajectories, support for input parameterization,

ranging from piecewise constant inputs to splines obtained by specifying control points, and support for robustness metrics over both continuous and hybrid traces. S-Taliro includes implementations of various search heuristics for optimization including UR: uniform random sampling and MC: Monte-Carlo sampling with simulated annealing. A detailed description of the framework is available elsewhere [3]. Furthermore, the latest version of the tool (along with the benchmarks used here) are available on-line as a open-source tool [5]

The implementation of the cross entropy method inside S-Taliro directly uses the key primitives implemented inside S-Taliro. Currently, our implementation supports piecewise uniform probability distributions in a fully factored form. Support for graphical models is currently being implemented.

Experimental Evaluation: Table 2 briefly describes the benchmarks used in our evaluation along with the properties checked for these benchmarks. These benchmarks along with a detailed explanation of the properties are available as part of the S-Taliro distribution. Furthermore, more detailed and up-to-date data comparing the various solvers on a larger set of benchmarks will be made available on our tool website.

Experimental Results: Table 3 shows the experimental comparisons over the set of benchmarks described in Table 2. We ran a fixed number of repetitions for each benchmark and property. Each run had a limit on the total number of simulations permitted (1000). The cross entropy method was applied for a maximum of 10 iterations with at most 100 simulations per iteration. Each technique terminates upon encountering a falsification. Due to the sheer size and number of these experiments, we ran them on a cluster with many different machines of roughly similar specifications — 64 bit Intel machines running Ubuntu 11.04 Linux with $6-12$ cores and $6-32$ GB of RAM. To facilitate comparison, we attempted all instances of a benchmark on the same machine (different cores).

Table 3 reports for each benchmark instance, the number of repetitions that resulted in a falsification. the average, minimum and maximum times. We notice from this comparison that the cross entropy method performed quite well on the IG and Mod benchmarks, resulting in the most amount of falsifications and was competitive on the remaining AT and PT benchmarks, wherein there were no clear winners between the three techniques compared. The cross entropy seems to outperform Monte Carlo simulations both in terms of time and number of falsifications on all but a few of the benchmarks, and is often competitive or better than uniform random testing which has negligible overhead.

6. RELATED WORK

It is well known that falsification of temporal logic properties for hybrid systems is a hard problem [1]. As a result, testing is a natural approach to the verification of continuous and hybrid systems [23]. The question of how to guide the choice of test cases better is an active area of research.

In this regard, Monte-Carlo techniques have been explored quite extensively. The use of Monte Carlo techniques for model checking was considered previously by Grosu and Smolka [21] in the form random walks over the state space of the system and by our previous work in the form of input sampling using Markov-Chain Monte-Carlo (MCMC) techniques [34, 28]. The techniques presented in the latter work were implemented as part of the S-Taliro framework [3]. The two approaches are quite distinct from each other. In practice, the rate convergence of random walks on the state space depends critically on the topology of the state transition graph. On the other hand, techniques that walk the state space can be extended readily to the case of systems with control inputs without requiring a finite parameterization of the control. The problem of integrating the two approaches remains a challenge.

In practice, the use of MCMC techniques has proven problematic for certain benchmarks due to the slow rate of convergence of MCMC techniques and their susceptibility to local minima in the search space. This has been instrumental in our quest for efficient stochastic search techniques that can exhibit faster convergence to the desired underlying distribution. We have also explored the use of other optimization techniques including ant-colony optimization (ACO) and genetic algorithms (GA) in conjunction with robustness metrics in the S-Taliro framework [2].

Other approaches to testing hybrid systems have focused on the use of state-space exploration techniques such as Rapidly exploring Random Trees (RRTs) [16, 4, 5, 27, 29] as well as notions of robustness over simulation trajectories [14, 20, 22, 25]. The work of Dang et al. attempts to bridge these approaches [13].

On the research front of falsification/verification of temporal logic properties through testing, the results are limited [30, 31, 17]. The work that is the closest to ours appears in [31]. The authors of that work develop a different notion of robustness for temporal logic specifications, which is also used as a fitness function for optimization problems. Besides the differences in the application domain, i.e., [31] focuses on parameter estimation for biological systems, whereas our paper deals with the falsification of hybrid systems. Furthermore, we have extended robustness metrics from purely continuous to hybrid trajectories, wherein we define robustness of trajectories using quasi-metrics instead of metrics [28].

Younes and Simmons, and more recently, Clarke et al. have proposed the technique of *Statistical Model Checking* (SMC) [36, 10], which generates uniform random inputs to a system subject to some constraints, thus converting a given system into a stochastic system. A probabilistic model checker can be used to prove assertions on the probability that the system satisfies a given temporal property φ within some given confidence interval. Statistical model checking, like our technique, requires a simulator to be available for the system but not a transition relation representation. In contrast to SMC, our approach is guided by a robustness metric towards less robust trajectories. On the other hand, the complex nature of the system and the robustness metrics imply that we cannot yet provide guarantees on the probability of satisfaction of the formula. Recent observations by Clarke and Zuliani have noted the need for importance sampling and rare-event simulation techniques for Statistical model checking [11]. Some of the ideas from this work on the use of factored input distributions and graphical models can also benefit statistical model checkers.

The work of Chockler et al. explores the use of the cross-entropy method for finding bugs in concurrent programs [7] and more recently for reconstructing concurrent executions for program replay [8]. Additionally, the use of cross-entropy method as a general combinatorial state-space search technique has been well-studied [6].

7. CONCLUSION

In conclusion, we have presented a framework for falsification of temporal logic properties using the Cross-Entropy method guided by a notion of robustness of trajectories w.r.t MTL formulae. We have also presented some ideas behind using factored probability distributions in the cross-entropy method and extended our notions

[5]Cf. `https://sites.google.com/a/asu.edu/s-taliro`.

[6]Cf. Rubinstein et al. [32] and the web site `http://www.cemethod.org` for a description.

Table 2: Benchmark systems and properties used in our evaluation.

Name	Description	Model Type	Property Type
Mod1-3	Third order Delta-Sigma Modulator [12] with varying initial conditions	S/S diagram	$\Box a$
IG1-3	Insulin Glucose control (Cf. Section 3.2) with varying initial conditions	ODE (matlab function)	$\Box_{[0,20.0]} p \wedge \Box_{[20,200.0]} q$
AT1 AT2 AT3-5	Auto Transmission Simulink demo [37] different predicates $q_{i,j}$	S/S diagram	$\neg(\Diamond p_1 \wedge \Diamond_{[0,10]} p_3)$ $\neg(\Diamond(p_1 \wedge \Diamond_{[0,7.5]} p_3))$ $\neg(\Diamond q_{1,i} \wedge \Diamond q_{2,i} \wedge \Diamond q_{3,i})$
PT1 PT2	Power train model [9]	Checkmate model [35]	$\neg\Diamond(g_2 \wedge \Diamond(g_1 \wedge \Diamond g_2))$ $\Box((\neg g_1 \wedge X g_1) \Rightarrow \Box_{[0,2.5]} \neg g_2)$
AIR1 AIR2 AIR3 AIR4 AIR5 AIR6	Aircraft model [26]	ODE (matlab function)	$\neg(\Box_{[.5,1.5]} a \wedge \Diamond_{[3,4]} b)$ $\neg(\Box_{[0,4]} a \wedge \Diamond_{[3.5,4]} d)$ $\neg\Diamond_{[1,3]} e$ $\neg(\Diamond_{[.5,1]} a \wedge \Box_{[3,4]} g)$ $\neg\Box_{[0,.5]} h$ $\neg\Box_{[2,2.5]} i$

to handle distributions factored using graphical models. Experimental results seem quite promising. In the future, we wish to run further experiments to quantify the effect of factoring on the overall performance. Furthermore, the problem of automatically identifying correlated input variables that can jointly influence the falsification remains to be investigated. Finally, we also wish to consider the application of our ideas to more general classes of distributions.

8. ACKNOWLEDGEMENTS

This work was supported, in part, by NSF awards CNS-1016994, CPS-1035845 and CNS-1017074.

9. REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] Y. S. R. Annapureddy and G. E. Fainekos. Ant colonies for temporal logic falsification of hybrid systems. In *Proceedings of the 36th Annual Conference of IEEE Industrial Electronics*, pages 91 – 96, 2010.

[3] Y. S. R. Annapureddy, C. Liu, G. E. Fainekos, and S. Sankaranarayanan. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, pages 254–257. Springer, 2011.

[4] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *HSCC*, volume 2993 of *LNCS*, pages 142–156. Springer, 2004.

[5] M. Branicky, M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control and verification of hybrid systems. *IEE Proc.-Control Theory Appl.*, 153(5):575–590, 2006.

[6] J. A. Bucklew. *Introduction to Rare-Event Simulations*. Springer, 2004.

[7] H. Chockler, E. Farchi, B. Godlin, and S. Novikov. Cross-entropy based testing. In *FMCAD*, pages 101–108. IEEE Computer Society, 2007.

[8] H. Chockler, E. Farchi, B. Godlin, and S. Novikov. Cross-entropy-based replay of concurrent programs. In *Fundamental Approaches to Software Engineering*, volume 5503 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2009.

[9] A. Chutinan and K. R. Butts. Dynamic analysis of hybrid system models for design validation. Technical report, Ford Motor Company, 2002.

[10] E. Clarke, A. Donze, and A. Legay. Statistical model checking of analog mixed-signal circuits with an application to a third order $\delta - \sigma$ modulator. In *Hardware and Software: Verification and Testing*, volume 5394/2009 of *LNCS*, pages 149–163, 2009.

[11] E. M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. In *Automated Technology for Verification and Analysis*, volume 6996 of *LNCS*, pages 1–12. Springer, 2011.

[12] T. Dang, A. Donzé, and O. Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In *ICFEM*, volume 3142 of *LNCS*, pages 97–109. Springer, 2004.

[13] T. Dang, A. Donze, O. Maler, and N. Shalev. Sensitive state-space exploration. In *Proc. of the 47th IEEE CDC*, pages 4049–4054, Dec. 2008.

[14] A. Donzé and O. Maler. Systematic simulation using sensitivity analysis. In *HSCC*, volume 4416 of *LNCS*, pages 174–189. Springer, 2007.

[15] M. Egerstedt and C. Martin. *Control Theoretic Splines: Optimal Control, Statistics, and Path Planning*. Princeton University Press, 2009.

[16] J. M. Esposito, J. Kim, and V. Kumar. Adaptive RRTs for validating hybrid robotic control systems. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, 2004.

[17] G. Fainekos, A. Girard, and G. J. Pappas. Temporal logic verification using simulation. In *FORMATS*, volume 4202 of *LNCS*, pages 171–186. Springer, 2006.

[18] G. Fainekos and G. Pappas. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42):4262–4291, 2009.

[19] M. E. Fisher. A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. *IEEE transactions on bio-medical engineering*, 38(1):57–61, 1991.

Table 3: Experimental comparison of the cross-entropy method with other optimization engines available in S-Taliro. Experiments were run on a variety of machines in a cluster. To enable comparison, each row is executed on different cores of the same machine. All timings are in seconds, rounded to the nearest integer. Note: MC run on PT2 did not finish after many days (`dnf`). Legend: #F: number of falsifying runs, av: average, lb: min, ub: max, #Rnds: number of iterations for CE.

Bench.	# Runs	Cross Entropy (CE)			Monte Carlo (MC)		Unif. Rand. (UR)	
		#F	Time (av,lb,ub)	#Rnds (lb,ub)	#F	Time (av,lb,ub)	#F	Time (av,lb,ub)
IG1	25	**25**	47,[3,91]	[1,4]	2	299,[3,443]	20	103,[1,275]
IG2	25	**25**	65,[20,124]	[1,6]	0	257,[209,317]	17	156,[25,279]
IG3	25	**25**	141,[81,286]	[3,7]	0	270,[184,324]	4	256,[33,298]
Mod1	100	**97**	15, [1,43]	[1,10]	84	11,[0,92]	81	19,[0,43]
Mod2	100	**87**	25, [1,43]	[1,10]	58	16,[0,94]	40	28,[0,38]
Mod3	100	13	41, [0,50]	[1,10]	**21**	61,[6,94]	1	37,[5,44]
AT1	100	36	102,[43,116]	[5,10]	**51**	61,[7,94]	0	94,[93,99]
AT2	100	0	111, [97,114]	[10,10]	0	93,[92,93]	0	93,[92,93]
AT3	100	**99**	34,[0,114]	[1,10]	93	24,[0,138]	86	56,[0,139]
AT4	100	**97**	53,[1,116]	[1,10]	94	25,[0,128]	55	81,[1,127]
AT5	100	0	111,[98,121]	[10,10]	0	115,[110,139]	0	111,[109,115]
PT1	25	25	170,[7,886]	[1,2]	23	754,[14,7630]	25	99,[14,329]
PT2	25	24	1013, [32,6381]	[1,10]	`dnf`	`dnf`	**25**	689,[21,2392]
AIR1	100	**100**	**8,[0,29]**	[1,49]	100	43,[1,177]	100	10,[1,296]
AIR2	100	99	133,[2,433]	[6,1000]	**100**	68,[6,244]	69	345,[1,612]
AIR3	100	**97**	379,[129,645]	[215,1000]	78	511,[50,1290]	3	1030,[7,1244]
AIR4	100	0	568,[561,580]	[1000,1000]	0	1442,[1418,1465]	0	1354,[1346,1362]
AIR5	100	**100**	**2,[0,10]**	[1,20]	100	54,[0,233]	100	5,[0,77]
AIR6	100	**100**	**10,[0,53]**	[1,99]	100	77,[0,256]	100	12,[0,57]

[20] A. Girard and G. J. Pappas. Verification using simulation. In *HSCC*, volume 3927 of *LNCS*, pages 272 – 286. Springer, 2006.

[21] R. Grosu and S. Smolka. Monte carlo model checking. In *TACAS*, volume 3440 of *LNCS*, pages 271–286, 2005.

[22] A. A. Julius, G. E. Fainekos, M. Anand, I. Lee, and G. J. Pappas. Robust test generation and coverage for hybrid systems. In *HSCC*, number 4416 in LNCS, pages 329–342. Springer, 2007.

[23] J. Kapinski, B. H. Krogh, O. Maler, and O. Stursberg. On systematic simulation of open continuous systems. In *HSCC*, volume 2623 of *LNCS*, pages 283–297. Springer, 2003.

[24] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, 1990.

[25] F. Lerda, J. Kapinski, E. M. Clarke, and B. H. Krogh. Verification of supervisory control software using state proximity and merging. In *HSCC*, volume 4981 of *LNCS*, pages 344–357. Springer, 2008.

[26] J. Lygeros. On reachability and minimum cost optimal control. *Automatica*, 40:917–927, 2004.

[27] T. Nahhal and T. Dang. Test coverage for continuous and hybrid systems. In *CAV*, volume 4590 of *LNCS*, pages 449–462. Springer, 2007.

[28] T. Nghiem, S. Sankaranarayanan, G. E. Fainekos, F. Ivančić, A. Gupta, and G. J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *Hybrid Systems: Computation and Control*, pages 211–220. ACM Press, 2010.

[29] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Hybrid systems: From verification to falsification. In *CAV*, volume 4590 of *LNCS*, pages 463–476. Springer, 2007.

[30] E. Plaku, L. E. Kavraki, and M. Y. Vardi. Falsification of LTL safety properties in hybrid systems. In *TACAS*, volume 5505 of *LNCS*, pages 368 – 382, 2009.

[31] A. Rizk, G. Batt, F. Fages, and S. Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *6th International Conference on Computational Methods in Systems Biology*, number 5307 in LNCS, pages 251–268. Springer, 2008.

[32] R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: An unified approach to combinatorial optimization, Monte-Carlo Simulation and Machine Learning*. Springer–Verlag, 2004.

[33] R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. Wiley Series in Probability and Mathematical Statistics, 2008.

[34] S. Sankaranarayanan, R. M. Chang, G. Jiang, and F. Ivančić. State space exploration using feedback constraint generation and Monte-Carlo sampling. In *ESEC/SIGSOFT FSE*, pages 321–330. ACM, 2007.

[35] B. I. Silva, K. Richeson, B. H. Krogh, and A. Chutinan. Modeling and verification of hybrid dynamical system using checkmate. In *ADPM 2000*, 2000.

[36] H. L. S. Younes and R. G. Simmons. Statistical probabilitistic model checking with a focus on time-bounded properties. *Information & Computation*, 204(9):1368–1409, 2006.

[37] Q. Zhao, B. H. Krogh, and P. Hubbard. Generating test inputs for embedded control systems. *IEEE Control Systems Magazine*, pages 49–57, August 2003.

APPENDIX

A. DERIVATION OF THE TILTING RULE FOR GRAPHICAL MODELS

Let V be a $k-$dimensional input space with variables $z_1, \ldots, z_k$ that form the input vector. Furthermore, the range of posible values for each z_i is subdivided into $n > 0$ subintervals $U_{i,1}, \ldots, U_{i,n}$. As a result, the overall input space has been partitioned into n^k cells $C_1, \ldots, C_K$ wherein each cell is of the form

$$C_j : z_1 \in U_{1,i_1}, \times \cdots \times z_k \in U_{k,i_k} .$$

Let G be a graphical model with nodes $N_G = \{z_1, \ldots, z_k\}$ and edge set $E \subseteq N_G \times N_G$. We assume that G is a directed acyclic graph. With each node in $z_j \in N_G$, let $\mathsf{Preds}(z_j) = \{z_l \mid z_l \rightarrow z_j \in E\}$ be the possibly empty set of nodes that are the predecessors of z_j in the DAG G. We associate a table $\mathbb{T}_j$ with node z_j of size $n^{1+|\mathsf{Preds}(z_j)|}$ where each entry is of the form

$$\mathsf{Pr}\left(z_j \in U_{j,l} \mid \bigwedge_{z_l \in \mathsf{Preds}(z_j)} z_l \in U_{l,p} \right) .$$

Note that the event predicate corresponding to this entry is defined as:

$$z_j \in U_{j,l} \ \wedge \bigwedge_{z_l \in \mathsf{Preds}(z_j)} z_l \in U_{l,p} .$$

The family of factored probability distribution is defined by parameters $(P_1, \ldots, P_M) \in [0,1]^M$ wherein each table entry for the tables $\mathbb{T}_1, \ldots, \mathbb{T}_k$ has a parameter $P_j \in [0,1]$. We denote the set of parameters associated with each table $\mathbb{T}_j$ by $\mathsf{Entries}(\mathbb{T}_j)$. Furthermore, for each table $\mathbb{T}_j$, the sum of its entries equal 1.

$$\sum_{P_i \in \mathsf{Entries}(\mathbb{T}_j)} P_i = 1 .$$

We let $\mathsf{Event}(P_i)$ for each entry P_i be the event predicate associated with the entry.

We now consider the problem of updating entries to the table by tilting. Let $P_{\theta(h)}$ be the current distribution defined by parameters $\theta(h)$. Let $\vec{x}^{(0)}, \ldots, \vec{x}^{(m-1)}$ be $m > 0$ samples that we use to tilt. We associate with each sample $\vec{x}^{(i)}$ a *weight* $\gamma_i = \frac{\Omega(\vec{x}^{(i)})}{P_{\theta(h)}(\vec{x}^{(i)})}$. Our goal is to compute an optimal set of parameters $\theta(h+1) \in P$ based on the current samples $\vec{x}^{(l)}$ and current candidate $\theta(h)$.

$$\theta(h+1) = \underset{\theta \in P}{\mathsf{argmax}} \left(\sum_{l=0}^{m-1} \gamma_l \log(p_\theta(\vec{x}^{(l)})) \right) . \tag{6}$$

We will now derive a closed form expression for each parameter in $\theta(h+1)$ for given samples with weights.

We first note that for each sample $\vec{x}^{(l)}$, we may write

$$p_\theta(\vec{x}^{(l)}) = \Pi_{j=1}^{k} \, \Pi_{P_i \in \mathsf{Entries}(\mathbb{T}_j)} \, P_i \mathbb{1}(\vec{x}^{(l)} \models \mathsf{Event}(P_i)) .$$

As a result, we may rewrite the objective function of the optimization in Eq. (6) as

$$\sum_{l=0}^{m-1} \gamma_l \left(\sum_{j=1}^{k} \left(\sum_{P_i \in \mathsf{Entries}(\mathbb{T}_j)} \log(P_i) \mathbb{1}(\vec{x}^{(l)} \models \mathsf{Event}(P_i)) \right) \right)$$

We may rearrange this summation as

$$\sum_{j=1}^{k} \sum_{P_i \in \mathsf{Entries}(\mathbb{T}_j)} \log(P_i) \left(\sum_{l=0}^{m-1} \gamma_l \mathbb{1}(\vec{x}^{(l)} \models \mathsf{Event}(P_i)) \right) .$$

Writing $c_i = \left(\sum_{l=0}^{m-1} \gamma_l \mathbb{1}(\vec{x}^{(l)} \models \mathsf{Event}(P_i)) \right)$, the overall optimization simplifies to

$$\begin{array}{ll} \max & \sum_{i=1}^{M} c_i \log(P_l) \\ \mathsf{s.t} & \sum_{P_i \in \mathsf{Entries}(\mathbb{T}_j)} P_i = 1 \quad j = 1, \ldots, k \end{array}$$

We solve this optimization by the Lagrange method of multipliers. We first form the Lagrangian:

$$L : \sum_{l=1}^{M} c_i \log(P_i) - \sum_{j=1}^{k} \lambda_j \left(\sum_{P_i \in \mathsf{Entries}(\mathbb{T}_j)} P_i \right) .$$

Next, we equate the partial derivative w.r.t each of the decision variables P_i to zero.

$$\frac{\partial \, L}{\partial \, P_i} = \frac{c_i}{P_i} - \lambda_j = 0 .$$

where $j \in [1,k]$ is the unique index such that $P_i \in \mathsf{Entries}(\mathbb{T}_j)$. Therefore, for each table $\mathbb{T}_j$ we obtain the following constraints

$$\lambda_j = \frac{c_{i1}}{P_{i1}} = \cdots = \frac{c_{iJ}}{P_{iJ}}, \text{ where } \mathsf{Entries}(\mathbb{T}_j) = \{P_{i1}, \ldots, P_{iJ}\} .$$

This yields the optimal solution for each $P_i \in \mathsf{Entries}(\mathbb{T}_j)$ as

$$\begin{array}{rcl} P_i & = & \frac{c_i}{\sum_{P_q \in \mathsf{Entries}(\mathbb{T}_j)} c_q} \\ & = & \frac{\sum_{l=0}^{m-1} \gamma_l \mathbb{1}(\vec{x}^{(l)} \models \mathsf{Event}(P_i))}{\sum_{P_{iJ} \in \mathsf{Entries}(\mathbb{T}_j)} \sum_{l=0}^{m-1} \gamma_l \mathbb{1}(\vec{x}^{(l)} \models \mathsf{Event}(P_{iJ}))} \end{array}$$