Test Generation for Hybrid Systems using Clustering and Learning Techniques

Sudhi Proch and Prabhat Mishra
University of Florida, Gainesville, Florida, USA
{sproch, prabhat}@ufl.edu

***Abstract*—In this paper, we propose a test generation method that employs clustering and learning techniques to reduce test generation time in hybrid systems. While learning-oriented test generation is a well-studied problem for digital systems, there are limited efforts for utilizing learning during generation of directed tests for hybrid systems. This paper makes two important contributions: i) it develops an efficient technique to cluster a set of functional scenarios that are expected to have similar test generation trajectory, and ii) it employs efficient learning mechanism such that beneficial information is shared during test generation of similar functional scenarios in a cluster. Our experimental results using two popular hybrid systems demonstrate that our approach can significantly (up to 3.8 times, 2.9 times on average) reduce the overall test generation time.**

I. Learning-oriented Test Generation

Validation of hybrid systems needs to consider the complexity of both continuous and discrete dynamics. Rapidly Exploring Random Tree (RRT) algorithm has been used by various researchers for test generation of hybrid systems [2], [3]. These methods are based on random exploration of possible system trajectories in the forward [2] and reverse direction [3]. Complex hybrid systems are likely to have numerous functional scenarios that must be satisfied for complete system validation. While researchers have demonstrated the utility of learning during test generation for digital systems, there are limited efforts to utilize learning across test generation instances in hybrid systems. It would be beneficial to learn from similar functional scenarios in hybrid systems. This paper makes two important contributions: i) it develops a clustering technique based on key parameters of the functional scenarios, and ii) it proposes a learning-based directed test generation method using reverse RRT algorithm.

The overview of our approach is shown in Fig. 1. It highlights two important steps of our methodology: clustering of functional scenarios and learning-based test generation. When considering multiple test generation instances, it is not possible to learn from each other unless their test trajectories have considerable similarity or overlap. Unless test generation is done, we do not know their exact trajectory or overlap. Therefore, it is a major challenge to cluster a set of similar functional scenarios before actual test generation. Once clustering is done, it is also equally critical to develop efficient methods to exploit learning during test generation of a cluster of similar scenarios to significantly reduce the overall test generation time.

This work was partially supported by National Science Foundation (NSF) grant CCF-1218629 and an IBM Faculty Award.

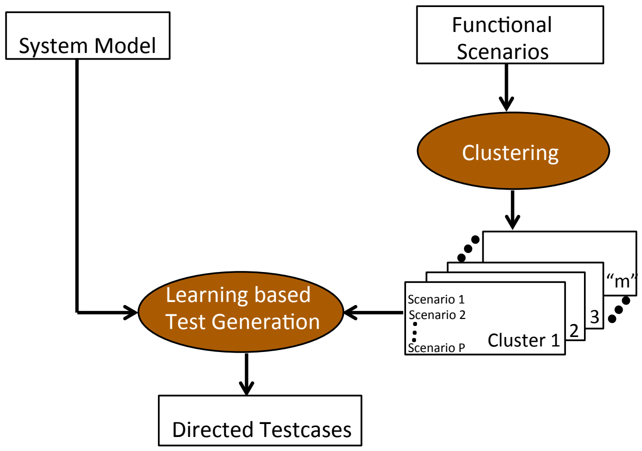


Fig. 1: Test generation using clustering and learning

A. Test Scenario Clustering

Test scenario clustering is challenging since the testcase trajectories are not known a priori. Neighboring functional scenarios of a state space region that also share the same initial region are likely to exhibit similar test trajectories. Hence, the similarity measure of our clustering algorithm is based on the Euclidean distance of functional scenario and initial region of the testcase instances. Fig. 2 demonstrates the concept of this clustering scheme. Each point in the state space of the system, with state variables X_1 and X_2, denotes a target functional scenario that must be activated by a testcase. Functional scenario $1, 2$ and 3 have a mutual Euclidean distance that is less than the cutoff threshold C_1. Hence all of them are considered part of the same cluster, A. Cluster size can be increased by increasing the threshold. This would cause some nearby functional scenarios to also become part of the same cluster. In Fig. 2, increasing the cluster cutoff threshold to C_2 results in functional scenario 4 and 5 to become part of cluster A'. Bigger clusters provide increased opportunity for learning. However, large cluster size may include scenarios that are not closely related, thus reducing the potential of trajectory match between cluster members.

B. Test Generation using Learning Techniques

Once clusters are formed, learning can be utilized between the test generation scenarios in a cluster. For the first member of a cluster (cannot learn from anyone), our method searches the state space using exhaustive search algorithm. This method is based on existing test generation technique

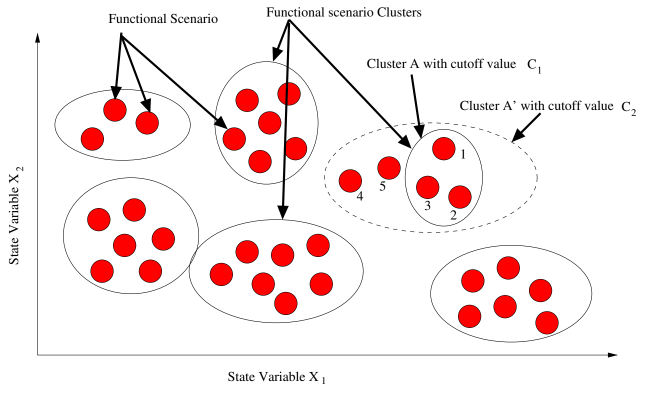

Functional Scenario
Functional scenario Clusters
Cluster A with cutoff value C_1
Cluster A' with cutoff value C_2
1
2
3
4
5
State Variable X_2
State Variable X_1

Fig. 2: Clustering based on similarity of functional scenarios

[2], [3]. However, for the remaining scenarios in the cluster, our method exploits learning and searches the state space of the previously generated tests of that cluster. Our approach has three important steps: region creation, random goal generation, and goal adjustment. The first step partitions the state space of the system in multiple smaller regions. The space can be partitioned in the direction of multiple system variables and with an adjustable granularity. Fig. 3 shows an example of a partitioned state space in the direction of system variable v_1. The next step selects a random node from a dynamically selected region of a previously generated testcase of the cluster. The dynamic selection of region influences the direction of goal generation leading to efficient tree growth. Fig. 3 shows an example tree growth using a learning-based goal generation. The goal sample (G_{rand}), for a particular iteration of tree growth, is selected from one of the nodes of a previous testcase named TC_1 that lies in region 3 of the state space. As a result, the testcase that is being generated (TC_2) grows towards the selected goal region of previous testcase (TC_1) and follows its trajectory. This learning results in significant reduction in test generation time for the new testcase.

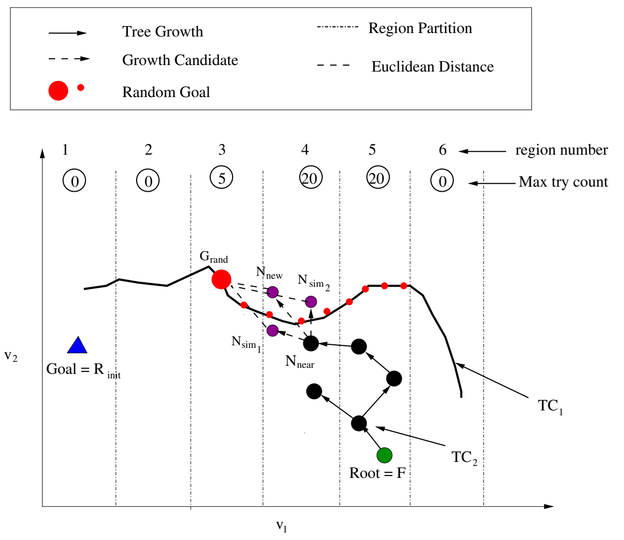
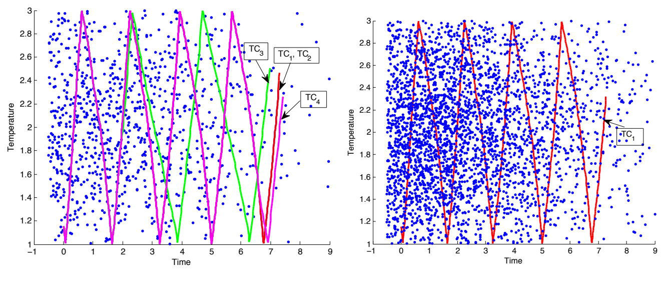


Fig. 3: Bias adjustment and goal generation for test generation

II. Case Studies

We demonstrate the applicability of our approach using two hybrid systems: thermostat and forced pendulum. *Thermostat* model has two discrete states namely "on" and "off". System variables are the system temperature and the total amount of time system is in operation. The system is designed to repeatedly transition between the "on" and "off" states in order to regulate the temperature between an upper bound and a lower bound. *Forced pendulum* is a single discrete state system whose behavior is described by its oscillation angle and its angular velocity. Motion of the pendulum depends on it's mass and length of swinging arm. A motor at the tip of the pendulum provides discrete torque values at random time intervals.



(a) With learning (b) Without learning [3]

Fig. 4: Number of dots indicate average random exploration of state space per test generation instance for *thermostat*.

Fig. 4a shows a cluster of related testcases (TC_{1-4}). For comparison, Fig. 4b shows one of the testcases of this cluster (TC_1) generated without learning using [3]. Average exploration of the state space per test generation instance is represented by the number of dots in both cases. While the number of random explorations remains the same for TC_1 in Fig. 4a, the other tests of the cluster (TC_{2-4}) use significantly less random explorations. As a result, the average random exploration of the state space per testcase reduces significantly (approx. 4 times reduction).

TABLE I: Test generation time comparison (in seconds)

System	Cluster Size	TG time (s) our method	TG time (s) no learning [3]	Improv. (times)
Thermostat	2	597.98	1158.4	1.94
	4	974.57	3055.9	3.14
	3	597.43	1681.5	2.81
	4	460.01	1728.1	3.76
	Average			2.91
Forced Pendulum	4	101.66	359.61	3.54
	2	89.07	169.77	1.91
	2	134.69	269.24	1.99
	2	115.55	227.99	1.97
	Average			2.35

Table I shows the improvement in test generation time compared to reverse RRT without learning [3] using two case studies. In each case study, we randomly selected four clusters. Our results indicate that for a cluster size of four we can reduce the test generation time up to 3.8 times (average 2.9 times) using our clustering and learning techniques.

References

[1] T. Dang and T. Nahhal. Coverage-guided test generation for continuous and hybrid systems. *FMSD*, 2009.

[2] J. Kim and J. Esposito. Adaptive sample bias for rapidly-exploring random trees with applications to test generation. *ACC*, 2005.

[3] S. Proch and P. Mishra. Directed test generation for hybrid systems. *International Symposium on Quality Electronic Design (ISQED)*, 2014.