Optimizing Stochastic Synthesis using Bayesian Inference

Hyunsu Kim
School of Computing
KAIST
Daejeon, South Korea
hyunsu.kim00@kaist.ac.kr

ABSTRACT

It has been widely known that randomized search for target program satisfying certain specification, or input of a program causing crash works well compared to strictly disciplined search strategy. However, there still remains an uncertainty of how to tame such randomness to find desired program or input effectively. Here, we propose the way of finding the optimal parameter that best controls the randomness achieving the goal. Given a grammar that is recursively used to generate a program, we use Bayesian inference to find the optimal weights for stochastic synthesis. Here, we typically use Metropolis-Hastings (MH) algorithm, which is one of the most well-known sampling methods to obtain unknown target distribution. We will validate the approach by comparing with naïve random search and genetic programming which are traditional approach to solve the problem.

CCS CONCEPTS

• Program Synthesis • Fuzzing • Stochastic Optimization

1 Motivation

It has been widely known that randomized search for target program satisfying certain specification, or input of a program causing crash works well compared to strictly disciplined search strategy. Known fuzzer like Csmith [1] is one of the known success. Still, how randomly generate the input, for instance, probability of choosing which grammar to apply at each step is under the hood. Moreover, we claim that such randomness has enough room to be adjusted according to the specification of a program or the target program that is being test.

2 Approach

We use Metropolis-Hastings (MH) algorithm, which is one of the most well-known sampling methods in the field of Bayesian machine learning. Our unknown target distribution is the probability of which grammar to apply next. For simplicity, we assume time homogeneous setting, where the distribution is fixed over iterations.

3 Expected Results

We will demonstrate our approach outperforms traditional approaches in various settings, simple to complicated specification of a desired program or input. For now, we are considering validating our approach against two traditional approaches, naïve random search with uniform distribution and genetic programming, [2] which is the bio-inspired way of program synthesis.

REFERENCES

[1] Xuejun Yang, Yang Chen, Eric Eide and John Regehr, 2011. Finding and Understanding Bugs in C Compilers. In *Proceedings of the 2011 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, San Jose, CA.

[2] Westley Weimer, Thanh Vu Nguyen, Claire Le Goues and Stephanie Forrest, 2009. Automatically Finding Patches using Genetic Programming. In *Proceedings of the 31st International Conference on Software Engineering (ICSE)*.