Adaptive Path Selection

Matthieu Petit
IRISA – Université de Rennes I
Campus Beaulieu
35042 Rennes cedex, FRANCE
Matthieu.Petit@irisa.fr

Phil McMinn
University of Sheffield
Regent Court, 211 Portobello St
Sheffield, S1 4DP, UK
p.mcminn@dcs.shef.ac.uk

1 Motivations

Random testing is a simple and well-known technique [3] which can be effective at finding software bugs. The family of algorithms around the work of Chen et al. on Adaptive Random Testing [1] attempt to maintain the benefit of random testing while increasing its efficiency. This method generates candidate inputs randomly, and at every step selects from them the best one. The best candidate is the test datum that is as far as possible than the already selected test data given some metric (Euclidean measure [1] for example). However, adaptive random testing may have some difficulties in providing high code coverage. For instance, the probability to select a candidate that activates the true branch of the conditional statement `if (x==0)` is equal to $\frac{1}{2^{32}}$ if x is a 32-bit integer program input.

This paper presents preliminary work on adaptive path selection. Our approach aims at guiding the test data selection such that a new test datum activates a path that is "as far as possible" from already tested paths. Test datum selection is performed in three steps : 1) building the set of candidate paths, 2) selection of the best candidate given a set of already tested paths and 3) generation of a test datum that activates the selected path. In this paper, we present a metric, called the path selector distance, that allows us to achieve the second step. The path selector distance aims at diversifying the set of tested path during the test data selection. For each step of the test datum selection, this distance permits us to select a path that activates some part of the source code as different as possible than the already tested path. In our future work, the combination of search-based [5] and constraint-based [2] approach will be used to achieve the first and the third step of the test data selection.

Section 2 introduces the adaptive selection of paths on an example. Section 3 introduces the definition of the path selector distance. Section 4 concludes the paper and presents our future work.

2 Illustrating example

Consider a program **foo** represented by its control flow graph in FIG. 1. The test data selection begins by uniform selection of a program input. Suppose that the path $1-3-5-7-9-10$ is activated and consider that the set of candidates is composed of the 9 program paths. The path selector distance aims at selecting an untested path and a path that has the most important number of different transfers of flow with $1-3-5-7-9-10$. The second test datum activates the path $1-2-10$. This path is an untested path and the number of different transfer of flows between $1-3-5-7-9-10$ and the different candidate is:

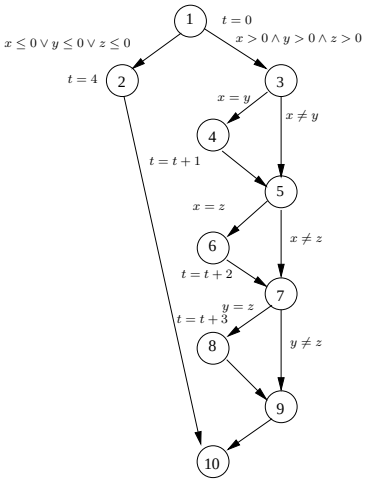

Figure 1. Control flow graph of foo

Path	No. of transfers of flow
$1-2-10$	4
$1-3-4-5-6-7-8-9-10$	3
$1-3-4-5-6-7-9-10$	2
$1-3-4-5-7-8-9-10$	2
...	...

The third test datum activates the path $1-3-4-5-6-7-8-9-10$. This path is an untested path and the number of different transfers of flow between $1-2-10$ and $1-3-4-5-6-7-8-9-10$ is 4 and between

$1-3-5-7-9-10$ and $1-3-4-5-6-7-8-9-10$ is 3. After eight selected inputs, suppose that all the paths have been selected except the path $1-3-5-6-7-8-9-10$. The path selector distance guides the test data selection such that the last path is selected. Note that this path is an unfeasible path. Constraints $x > 0 \wedge y > 0 \wedge z > 0 \wedge x \neq y \wedge x = z \wedge y = z$ on the input domain are not satisfiable. Our approach aims at using an uniform path selector of candidate path that can be updated by the detection of unfeasible paths [7]. When all the paths program are activated, the path selector distance aids the selection of a path that is the least activated by already selected test data as best candidate.

3 Path selector distance

The path selector distance is based on the combination of two distances extracted from information theory: the Shannon entropy and the Levenshtein distance. In this section, $Paths$ and $Candidates$ denote respectively the set of the already tested paths and the set of candidate paths.

3.1 Shannon index

The Shannon entropy has proved to be useful to measure the species diversity of a population. We tune this measure to diversify the set of tested paths. This new measure, called Shannon index, aims at selecting the "best" path of $Candidates$. We define the best candidate as an untested path or the least tested path between the set of the already tested paths.

Definition 1 (Shannon index)**.** Let $Paths$ be a set of paths, $(n_1, \ldots, n_k)$ be occurrence of $Path_1$ activation, $Path_2$ activation, ... and $N \in \mathbb{N}, \sum_{i=1}^{k} n_k = N$, then Shannon index, noted $Ent_{(n_1,\ldots,n_k)}$, is defined as follows:

$$Ent_{(n_1,\ldots,n_k)} = -\sum_{j=1}^{k} \frac{n_j}{N} \cdot \log\left(\frac{n_j}{N}\right).$$

Our approach aims at computing the Shannon index when a path of $Candidates$ is added in $Paths$. In this case, the Shannon index has two interesting properties: 1) the Shannon index is maximized when the path of $Candidates$ does not belong to $Paths$ and 2) if all the path of $Candidate$ belong to $Paths$, the Shannon index is maximized for the path with the minimal number of activations. For instance, suppose that all the paths of the program **foo** have been selected except the path $1-3-5-6-7-8-9-10$. The Shannon index is equal to 3.17 when this last path is selected as candidate whereas the Shannon index is equal to 2.95 when an other path is selected.

3.2 Levenshtein metric

In information theory, the Levenshtein distance is a metric for measuring the difference between two sequences. A path of a control flow graph can be defined by a sequence of critical edges. A critical edge represents a transfer flow during path execution. For example, the sequence $1-3\ 3-5\ 5-7\ 7-9$ represents the path $1-3-5-7-9-10$ of the program **foo**. The Levenshtein distance is adapted for our path selection problem. In this context, the Levenshtein distance between two paths represents the number of different transfers of control flow between these paths. The new metric, called the Levenshtein metric, computes the number of the different transfers of control flow between a path of $Candidates$ and each path of $Paths$.

Definition 2 (Levenshtein metric)**.** Let $Paths$ be a set of paths, $Path$ be a path and $\mathrm{levenshtein\_distance}$ be a function that computes the Levenshtein distance between two paths, then the Levenshtein metric, noted $\mathrm{Leven\_metric}$, is defined as follows:

$$\mathrm{Leven\_metric}(Path, Paths) = \sum_{Path_i \in Paths} \mathrm{levenshtein\_distance}(Path, Path_i).$$

The Levenshtein metric allows to choose a path over a subset of the set of candidate paths with the maximal Shannon index. The best candidate is the path that maximizes the Levenshtein metric.

4 Future Work

In this paper, we present ongoing work on adaptive selection of paths. A metric, named the path selector distance, allows us to select a path as different as possible than the already tested path. In a recent work, we conduct some experiments to validate this distance. In our future work, we want to deal with the problem of building of the set of path candidates. This is a challenging problem. Ideally, the computation of the set of path candidates will be performed by a uniform selection of paths. However in presence of loop, the number of paths is infinite. An other problem is the presence of unfeasible paths in a control flow control (3 over 9 for the program **foo**). Using of a uniform selector of feasible paths [7] seems to be useful to address this problem. A path oriented generation of test datum [4, 6] will be used to obtain a complete method of test data generation.

References

[1] T.Y. Chen, H. Leung, and I.K. Mak. Adaptive random testing. In *Proc. of ASIAN'04*, pages 320–329, Chiang Mai, Thailand, 2004. LNCS.

[2] R.A. DeMillo and J.A. Offutt. Constraint-based automatic test data generation. *IEEE TSE*, 17(9):900–910, Sept. 1991.

[3] J.W. Duran and S. Ntafos. An evaluation of random testing. *IEEE TSE*, 10(4):438–444, July 1984.

[4] A. Gotlieb and M. Petit. Constraint reasoning in path-oriented random testing. In *Proc. of COMPSAC'08*, Turku, Finland, July 2008.

[5] P. McMinn. Search–based software test data generation: A survey. *Software Testing, Verification and Reliability*, 14(2):105–156, June 2004.

[6] P. McMinn, M. Harman, D. Binkley, and P. Tonella. The species per path approach to search-based test data generation. In *Proc. of ISSTA'06*, ACM, pages 13–24, Portland, USA, 2006.

[7] M. Petit and A. Gotlieb. Uniform selection of feasible paths as a stochastic constraint problem. In *Proc. of QSIC'07*, IEEE, pages 280–285, Portland, USA, 2007.