TDSGen: An Environment Based on Hybrid Genetic Algorithms for Generation of Test Data

Luciano Petinati Ferreira and Silvia Regina Vergilio

Federal University of Parana (UFPR), CP: 19081,
CEP: 81531-970, Curitiba - Brazil
{petinati, silvia}@inf.ufpr.br

The software testing has gained importance and can be considered a fundamental activity to ensure the quality of the software being developed. In the literature, there are three groups of testing techniques proposed to reveal a great number of faults with minimal effort and costs: functional technique, structural technique and fault-based technique. These techniques are generally associated to testing criteria. A criterion is a predicate to be satisfied to consider the testing activity ended and the program tested enough [3]. The criterion generally requires the exercising of certain elements of the source code (decision statement) called required elements. To satisfy a testing criterion (coverage of 100%), it is necessary to provide input data to execute paths that exercise all the required elements. This is a very hard task because it is not possible its complete automatization due to several testing limitations.

The generation of test data has been addressed by many authors and different techniques, such as: random generation, generation based on symbolic execution and generation based on dynamic execution, have been applied with varying degrees of success, maybe by the testing limitations and the complexity inherent to the test generation task. Therefore some authors proposed the use of metaheuristics algorithms, such as Genetic Algorithm, originating a new research field named Evolutionary Testing [4].

Many works address GA for test generation but most of them do not offer a testing environment for supporting the organization and the complete application of a testing strategy including different testing criteria.

We explore the use of GA for test data generation and describe an environment, named TDSGen (**T**est **D**ata **S**et **Gen**erator), that evolves a population of test inputs to satisfy different testing criteria. The goal is to obtain a population that represents a set that better satisfies the chosen criterion. TDSGen uses the coverage measurement for the fitness evaluation and implements different mechanisms based on the strategies: elitism, sharing and tabu lists.

TDSGen integrates two different testing tools, Poketool [1] and Proteum [2], allowing to apply structural and fault-based criteria of C programs. These tools are responsible for the generation of the required elements and for the coverage evaluation, by producing the list of the covered elements. TDSGen presents other advantages: the tester provides a configuration file and initial functional test set, no analysis of the program is necessary, it implements a mechanism to encode the program input allowing the testing of different types of programs

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 1431–1432, 2004.

We have conducted some experiments using TDSGen with three strategies: a) random generation; b) GA based generation; c) HGA (Hybrid Genetic Algorithm) based generation, the uses the mechanisms implemented by TDSGen. These strategies were used for criteria: mutation analysis (MA), all-nodes (AN), all-edges (AE), all-potential-uses (PU), all-potential-uses/du (PUDU) and potential-du-paths (PDU). Table 1 presents the coverage obtained for each criterion and strategy.

Table 1. Coverage Obtained for Each Criterion and Strategy

Strategy	Size	AN	AE	PU	PDU	PUDU	MA
Random	10	68.18	7.34	7.34	7.34	8.67	36.27
	50	70.22	11.33	8.02	14.0	13.34	48.74
	200	75.68	26.67	28.00	31.33	32.00	63.50
GA	10	68.63	9.33	9.33	8.00	8.00	34.44
	50	69.31	10.67	12.00	11.33	13.33	48.75
	200	75.00	36.67	32.67	45.00	36.00	63.10
HGA	10	69.09	11.33	9.33	9.34	11.33	38.29
	50	74.54	32.00	30.67	40.0	40.00	52.72
	200	91.14	89.34	90.00	90.66	88.67	67.10

The results obtained in the experiments reveals an increase in the performance by using the mechanisms implemented by TDSGen, mainly the HGA strategy, without increase the costs and execution time. The mean runtime of the HGA strategy is lower than the GA strategy runtime.

The experiments showed yet that TDSGen is a promising tool for generation of test data sets. It has three important characteristics, that makes it different from the most works found in the literature: 1) uses a fitness function based on the coverage of the test case with the goal of satisfying a given testing criterion; 2) has mechanisms of memorization and hybridization to increase the performance of the GA; and 3) integrates two different testing tools.

An observed limitation is the difficulty to get a complete coverage because it is not always possible in due infeasible elements or equivalent mutants.

References

1. M.L. Chaim. *POKE-TOOL - Uma Ferramenta para Suporte ao Teste Estrutural de Programas Baseado em Análise de Fluxo de Dados.* Master Thesis, DCA/FEEC/Unicamp, Campinas - SP, Brazil, April 1991. (in Portuguese).
2. M. E. Delamaro and J.C. Maldonado. A tool for the assesment of test adequacy for c programs. In *Proceedings of the Conference on Performability in Computing Systems*, pages 79–95. East Brunswick, New Jersey, USA, July 1996.
3. S. Rapps and E.J. Weyuker. Selecting software test data using data flow information. *IEEE Trans. on Soft. Engineering*, SE-11(4):367–375, April 1985.
4. J. Wegener, A. Baresel, and H. Sthamer. Evolutionary test environment for automatic structural testing. *Information and Software Technology*, 43:841–854, 2001.