

Performance Analysis of Metaheuristic and Constraint Programming approaches to effectively generate Structural Test Cases

Neelesh Bhattacharya^{1,2}, Abdelilah Sakti^{2,3}, Giuliano Antoniol¹, Yann-Gaël Guéhéneuc², and Gilles Pesant³

¹SOCER Lab, DGIGL, École Polytechnique de Montréal, Canada

²Ptidej Team, DGIGL, École Polytechnique de Montréal, Canada

³Quosséca Lab, Department of Computer and Software Engineering

E-mail: {neesh.bhattacharya, abdelilah.sakti, giuliano.antonio, yann-gael.gueheneuc, gilles.pesant}@polymtl.ca



Introduction

- Structural test case generation has been carried out by various approaches in software testing.
- Metaheuristics and constraint programming approaches are two of the more important approaches used for generating structural test cases.
- Metaheuristics and constraint programming approaches have potential limitations.

Limitations of CP and Metaheuristic approaches

- Metaheuristics get stuck in the local optima and fail to prove that a test target is Infeasible.
- Constraint programming suffers in terms of execution time, when the input domain is large.

Goal

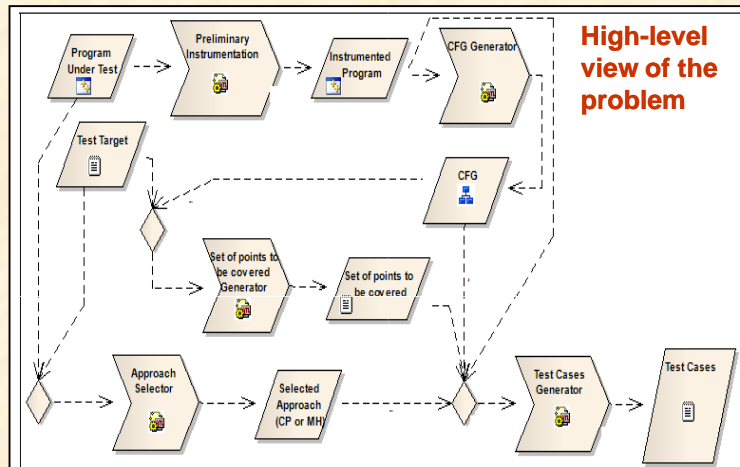
To overcome the limitations of constraint programming and metaheuristic approaches and get the best of both worlds, we want to propose a way to combine both approaches and the order in which they would be executed. Combining both of the approaches would allow their use in various applications.

Problem

For combining both metaheuristic and constraint programming approaches, we must have sufficient information about properties of both these approaches.

Solution

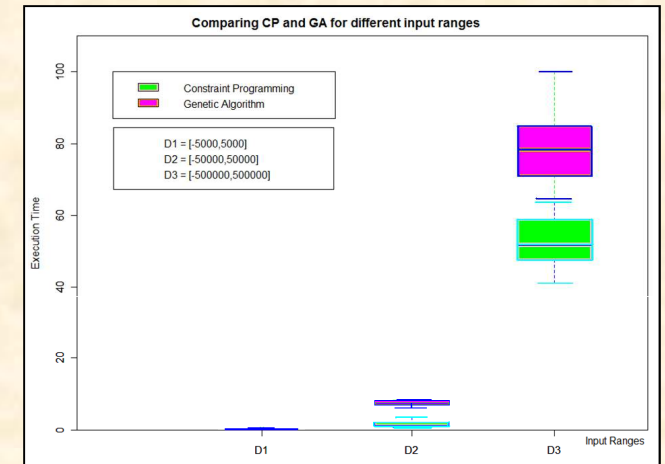
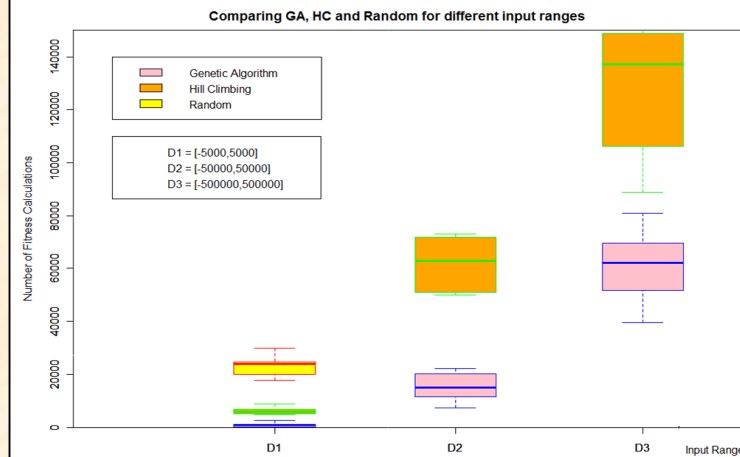
To compare these approaches in term of performance, we generate test cases to fire divide-by-zero exceptions in a sample code and compare the number of fitness calculations (fails) and execution time required by both of these approaches.



High-level view of the problem

Expérimentation and Subject program

- We characterize a PUT using five dimensions: the domain size (DS) of program input parameters, the program size (PS), the level of nested conditional statements (LNC), existence of pointers in the program and presence of function calls in the program.
- The current program under test (CPUT) is a function contains three statements that can fire divide-by-zero exceptions.
- The CPUT characteristics are as follows:-
 $500000 \leq DS \leq 500000$; $PS \leq 50$; no pointers; no function call; $LNC = 1$.



Empirical Study results

- Various input domain ranges are considered.
- The genetic algorithm is the best metaheuristic in term of fitness calculation required for generating test cases.
- Constraint programming outperform genetic algorithm for all the input domains.

Conclusion

When the test goal is to fire divide-by-zero exceptions constraint programming will be executed before metaheuristics, because constraint programming reaches a solution faster than any other metaheuristic approach.

Future Work

- In the future, we will conducting experiments with well known programs so as to generalize our conclusion.
- We are presently, working on combining both metaheuristic and constraint programming approaches in an effective way