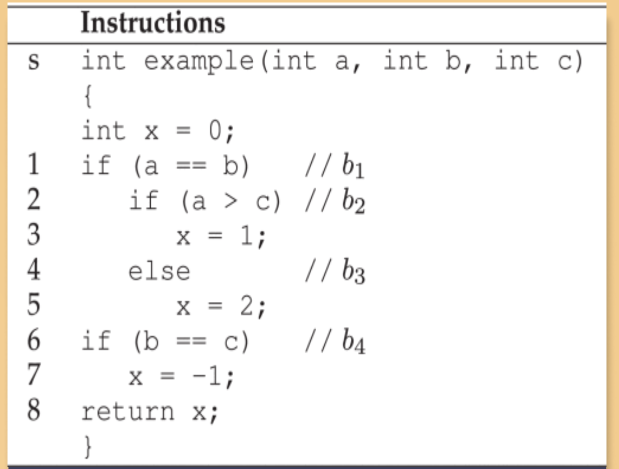
Diego Viero
d.viero@student.tudelft.nl

Assistant professor: Annibale Panichella
Supervisors: Mitchell Olsthoorn, Dimitri Stallenberg

Is PSO a valid option for search-based test case generation in the context of dynamically-typed languages?

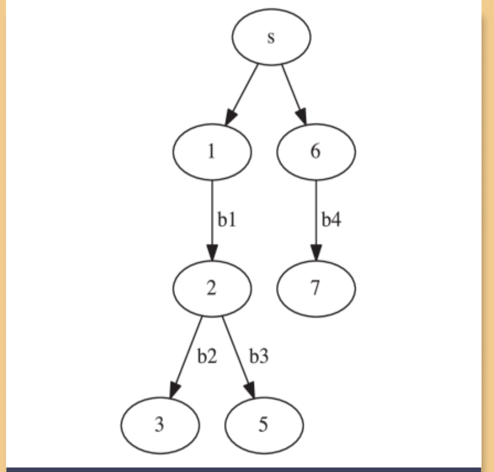
1) Introduction

- Writing meaningful and efficient tests often represents a blockage for many developers who would prefer spending their time implementing new features.
- Search based test generation uses evolutionary and meta-heuristics algorithms to evolve a set of randomly generated solutions in order to obtain a test-suite for the input program.
- NSGA-II is a multi-objective genetic algorithm. It uses non-dominated solutions and crowding distance as parameters to select the parents for the next generation.
- DynaMOSA is an extension of NSGA-II developed specifically for the automated test-case generation domain. It uses 2 heuristics to evolve the test-cases:
  - Dependency between conditional branches.
  - Preference criterion based on branch coverage and approach level.

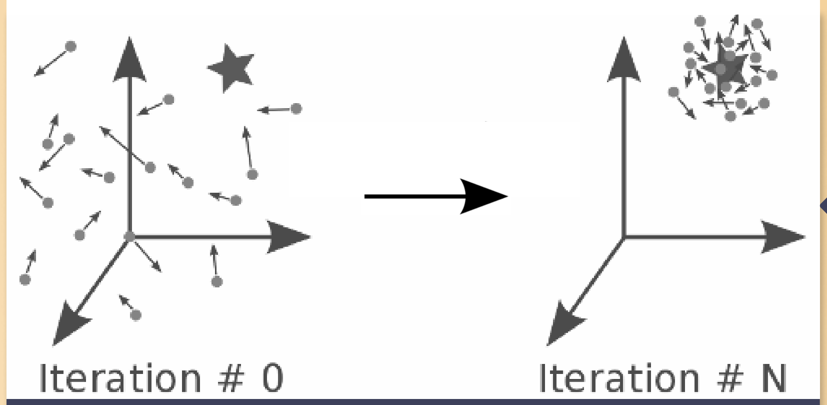


```
   Instructions
s  int example(int a, int b, int c)
   {
   int x = 0;
1  if (a == b)    // b1
2     if (a > c) // b2
3        x = 1;
4     else       // b3
5        x = 2;
6  if (b == c)    // b4
7     x = -1;
8  return x;
   }
```

Example program [1]



Control dependency graph [1]



Particles moving through the search space [2]

2) Research Question

This research focuses on evaluating two adaptations of PSO developed specifically for search-based test generation.

- PSO: Adaptation without DynaMOSA features.
- DynaPSO: Adaptation with DynaMOSA features.

RQ1: **How does DynaPSO perform compared to the default PSO implementation?**

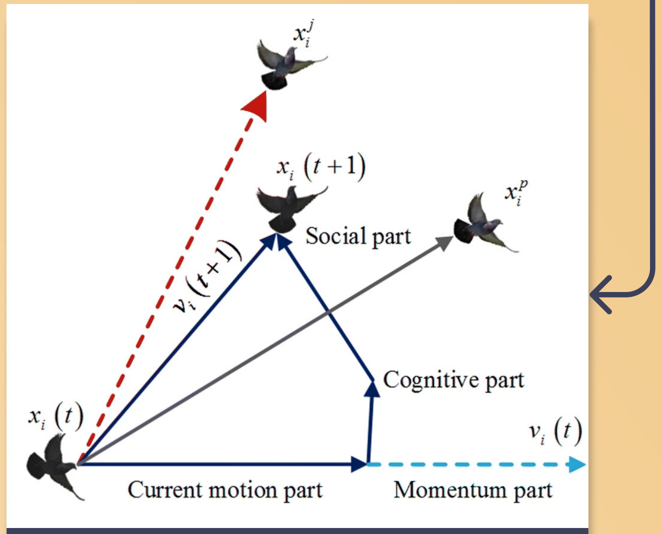
RQ2: **How does DynaPSO perform compared to the original DynaMOSA algorithm?**

3) Particle Swarm Optimization

Particle Swarm Optimization is an evolutionary algorithm inspired by the social behavior of bird flocks, which allows it to efficiently explore the search space and find optimal solutions for complex optimization problems.

The procedure behind PSO is the following:

- Randomly generate N particles over the search space, each particle represents a candidate solution for the problem.
- Keep track of the best solution over all particles and the best individual solution for each particle.
- Update each position iteratively until convergence.



Movement of a particle [3]

4) Method

Challenges with adapting PSO for this research:

- Represent branch coverage as optimization problem
  ↳Adapt algorithm to multi-objective optimization
- Solutions are now tests instead of numbers
  ↳Adapt particles position in the search-space
  ↳Adapt movement of particles
- Adapt algorithm with DynaMOSA features
  ↳Include preference selection in main routine

5) Results

The results were gathered using a benchmark consisting of 27 different files from 4 popular JavaScript projects. The set of files served as a proper representation of different code styles and syntax of the JavaScript language.

Branch coverage was used as the main metric to compare the different algorithms.

	# Lose	# No Diff.	# Win
PSO vs DynaPSO	6	20	1
DynaMOSA vs DynaPSO	0	23	4

The average coverage achieved by each algorithm is:

- PSO: 53.89%
- DynaPSO: 54.87%
- DynaMOSA: 55.24%

References

- [1] Annibale Panichella, Fitsum Meshesha Kifetew, andPaolo Tonella. Automated test case generation as a many-objective optimisation problem with dynamic selection ofthe targets. IEEE Transactions on Software Engineering,44(2):122–158, 2018.
- [2] Particle Swarm Optimization (PSO) — pagmo 2.19.0 documentation. (n.d.). https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html
- [3]: Elsheikh, A.H., Abd Elaziz, M. Review on applications of particle swarm optimization in solar energy systems. Int. J. Environ. Sci. Technol. 16, 1159–1170 (2019). https://doi.org/10.1007/s13762-018-1970-x