Search-Based Software Engineering

• Application of all the optimisation techniques we have seen so far, to the various problems in software engineering.

• Not web search engines :(  

• Not code search :(
Capture Requirements
Generate Tests
Explore Designs
Maintain/Evolve
Regression Testing

# of test cases
Execution Time

Code Coverage
Fault Detection

Minimise
Maximise
Minimise

Maximise

- Capture Requirements
- Generate Tests
- Explore Designs
- Maintain/Evolve
- Regression Testing

# of Test Cases
Execution Time

Coverage
Fault Coverage
Good Starting Points


Cost Estimation

- Evolve mathematical functions (symbolic regression) that would predict the project development effort based on various input variables.

Project Planning

- Team allocation to project work packages, including the possibility of abandonment (i.e. work no longer needed/practical) and rework (i.e. additional work needed).

Next Release Problem

- Find the ideal set of requirements that balances customer requests, resource constraints, and interdependencies between requirements.


Optimising Source Code

• Random sampling of code transformation to find compiler optimisation


• Automated Parallelisation

Test Data Generation

• Many, many different approaches and ideas; too many to list all:

Regression Testing

- Pareto-efficient Test Suite Minimisation:

- Test Case Prioritisation:

- Multi-objective Prioritisation:

Figure 3: Boxplots of the APFDc metric across all studied subjects. MOEAs and their variants show higher median values and smaller variances.
Maintenance & Reverse Engineering

- Module Clustering: assign modules to clusters based on their relationships


Figure 3. A Module Dependency Graph and its Modularisation using Bunch, taken from [65]
Deep Parameter Optimisation

• Reveal a property hidden in software as a parameter for tuning.


Code Transplantation

![Diagram of code transplantation process]

Fig. 1: Transplant operation in Cflow donor transplant. Code snippet from the beginning of the graft. ⟷ means function inlining; `optArg` is mapped to `_host_input`; → means original statement replacement under α — renaming; grayed statements are deleted.

Monte Carlo Tree Search for Program Synthesis (instead of GP)

Move B

Reward (winning rate) = \( \frac{2}{3} \)

Fitness: \( \frac{2}{3} \)
Fitness: \( \frac{1}{3} \)
Fitness: 0

Reward = \( \frac{1}{3} \)
SBSE Repository

- Most of the papers published on SBSE, stored and categorised online:

- http://crestweb.cs.ucl.ac.uk/resources/sbse_repository/
Testing and Debugging: 52%
Others: 7%
Software/Program Verification: 3%
General Aspects: 5%
Requirements/Specifications: 5%
Distribution, Maintenance, and Enhancement: 9%
Design Tools and Techniques: 9%
Management: 10%

The ratio of SE research fields that involved in SBSE.
Hints for Project Ideas

- Your own experience and/or research
- Reading SBSE papers
- Reading SBSE Challenge Track from SSBSE Conference (see conference proceedings from 2013 and onwards)
- Major SE conferences (ICSE, ESEC-FSE, ASE) have sessions dedicated to SBSE
Project Pitch (29/31 October)

• 10 minute sales pitch on what you plan to do.

• **Explicitly** describe the following:
  
  • Problem

  • How to formulate as search/optimisation

  • How to evaluate, using which data (remember to allow **sufficient** time for data preparation)

• You are welcome to discuss the idea with me beforehand