Some Application of Optimisation to Software Engineering Problems

Shin Yoo / CREST

4th COW 26/02/2010

Outline

- Motivation
- Application Areas
  - Case Study 1. Requirement Engineering
  - Case Study 2. Regression Testing
- Optimisation Techniques
- Future Directions

Motivation: why optimise?

- Easier than building a perfect solution
- Computational power: fast, scalable
- Data-driven, quantitative
- Insightful; allows holistic observation of problem space

“The heavy use of computer analysis has pushed the game itself in new directions. The machine doesn't care about style or patterns or hundreds of years of established theory. It is entirely free of prejudice and doctrine and this has contributed to the development of players who are almost as free of dogma as the machines with which they train. (...) Although we still require a strong measure of intuition and logic to play well, humans today are starting to play more like computers.”

- Gary Kasparov, *“The Chess Master and the Computer”*

Application Areas

Requirement Analysis   Model Checking

Test Data Generation Regression Testing

Refactoring Software Design Tools

Program Comprehension Agent-based System

Automated Patch Generation Project Management

... still expanding with many more to come

Application Areas

Tier 1

Combinatorial problems
in SE context

- Requirement Analysis
- Regression Testing
- Project Management

Tier 2

Problems that are
specific to SE

- Test Data Generation
- Software Design Tools
- Model Checking
- Agent-based System
- Refactoring
- Program Comprehension
- Automated Patch Generation

Application Areas

Tier 1

Combinatorial problems in SE context

Tier 2

Problems that are specific to SE

Test Data Generation

Software Design Tools

Model Checking

Agent-based System

Refactoring

Program Comprehension

Automated Patch Generation

Application Areas

Tier 1

Combinatorial problems in SE context

- Set-cover
- Prioritisation
- Bin-packing

Tier 2

Problems that are specific to SE

- Test Data Generation
- Software Design Tools
- Model Checking
- Agent-based System
- Refactoring
- Program Comprehension
- Automated Patch Generation

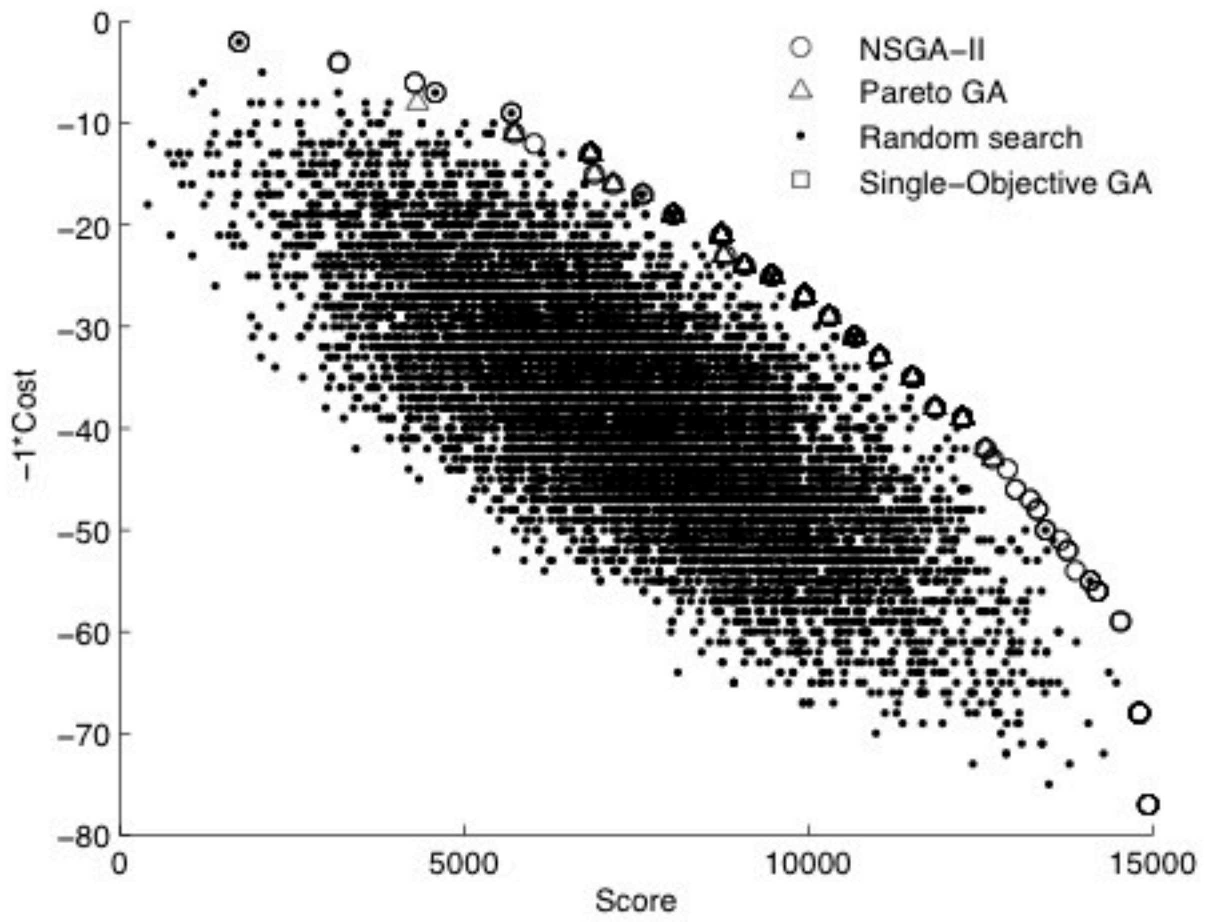
Case Study: Requirements

- “What is the most **cost-effective subset** of software requirements to be included in the next version?”
- “What is the most **efficient release schedule**?”
- “Are customers treated **fairly**?”

Requirements: selection

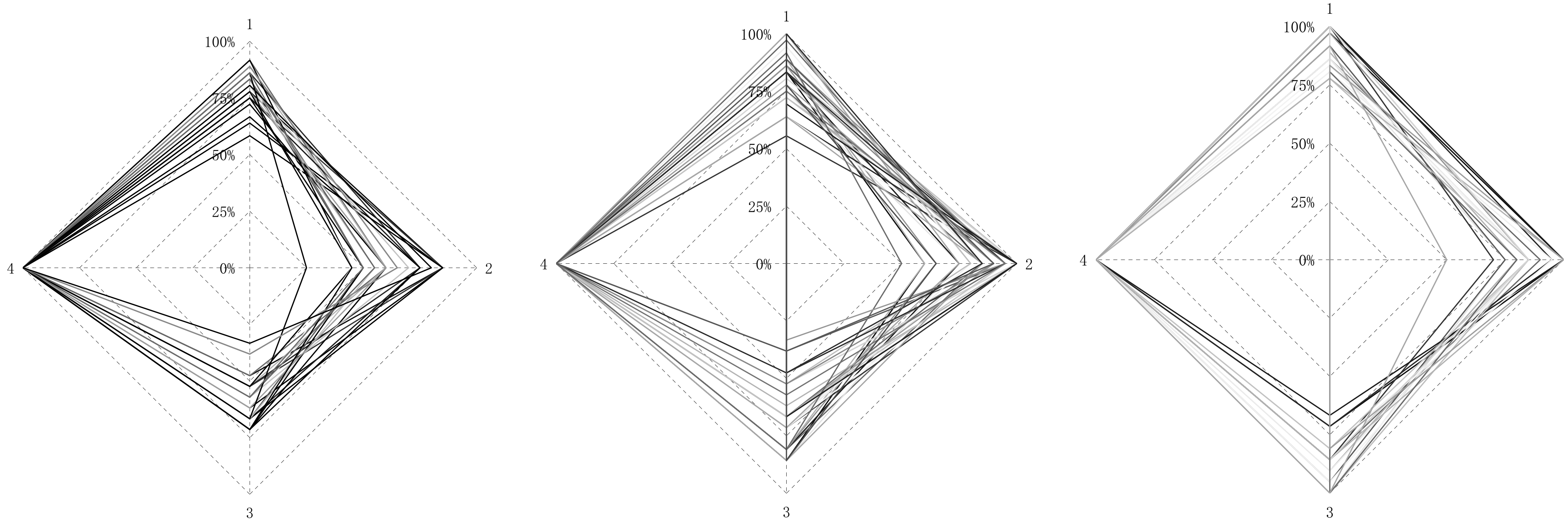
- Essential problem structure: **knapsack problem**
  - Requirements value: based on customer input, customer value, expected revenue, etc
  - Requirement cost: development cost, time, etc
- Goal: minimise cost, maximise value

Requirements: selection



(**d**) 100 customers; 20 requirements

Requirements: selection



(**a**) Motorola Data Set:
4 customers; 35 requirements
30% resource limitation

(**b**) Motorola Data Set:
4 customers; 35 requirements
50% resource limitation

(**c**) Motorola Data Set:
4 customers; 35 requirements
70% resource limitation

Case Study: Regression

- Regression testing: a test process that aims to gain confidence that **“existing”** functionality hasn’t been damaged by recent changes
- In order to test existing functionality, one has to execute old tests, of which there are **too many**

Case Study: Regression

- Regression testing: a test process that aims to gain confidence that "existing" functionality hasn't been damaged by recent changes

Software testing can only reveal faults, it cannot guarantee the lack of faults

- In order to test existing functionality, one has to execute old tests, of which there are **too many**

Case Study: Regression

- Regression testing: a test process that aims to gain confidence that **“existing”** functionality hasn’t been damaged by recent changes
- In order to test existing functionality, one has to execute old tests, of which there are **too many**

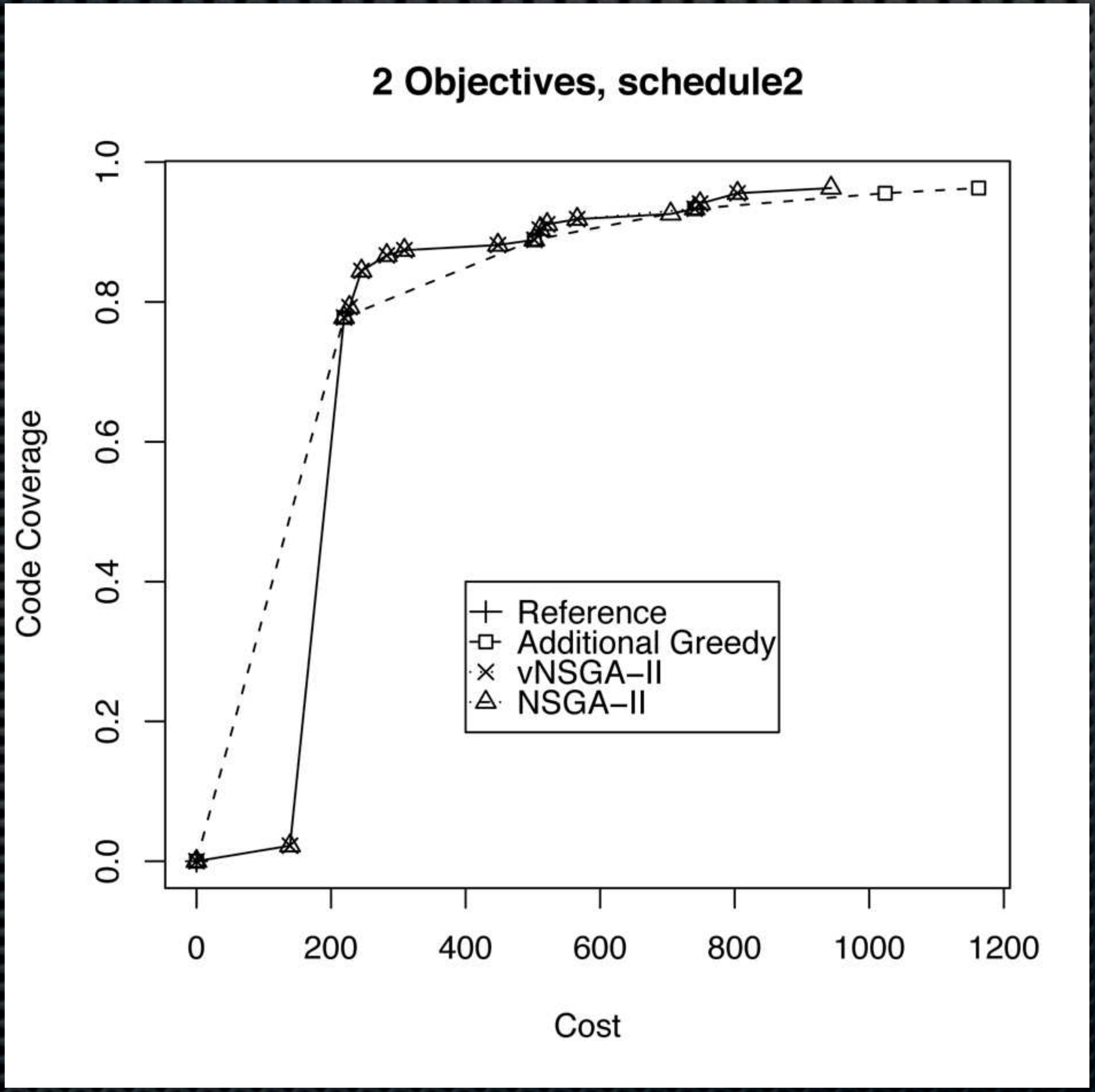
Case Study: Regression

- “What is the subset of tests that is **most likely to detect** the largest number of **faults**?”
- “Which test should I execute first in order to **detect faults as early as possible**?”

Regression: minimisation

- Essential problem structure: **set-cover problem**
  - Each test satisfies (or **covers**) different sets of test requirements; different coverage metrics have different correlation with fault-finding
  - Each test has associated cost
- Goal: to obtain the smallest subset that achieves the maximum test requirements

Regression: minimisation



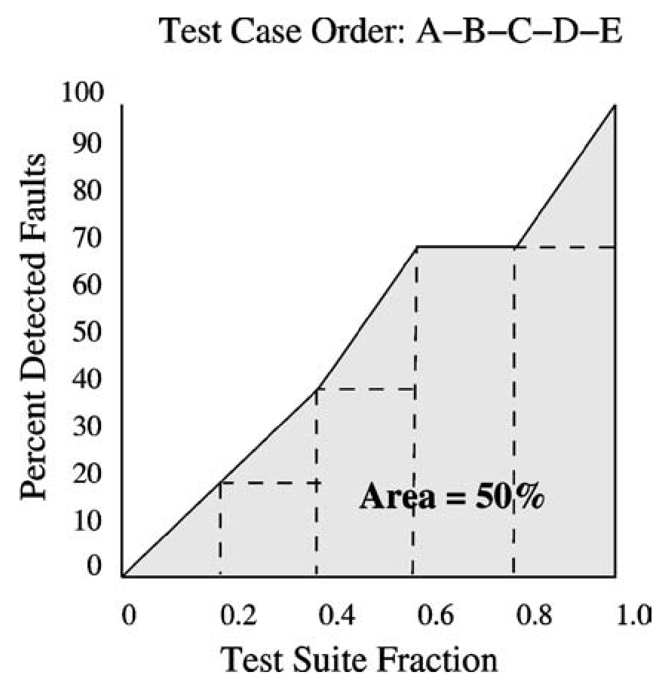
Regression: prioritisation

- Essential problem structure: **permutation**
  - Early maximisation of coverage - greedy algorithm is by definition very efficient but unable to deal with multiple criteria

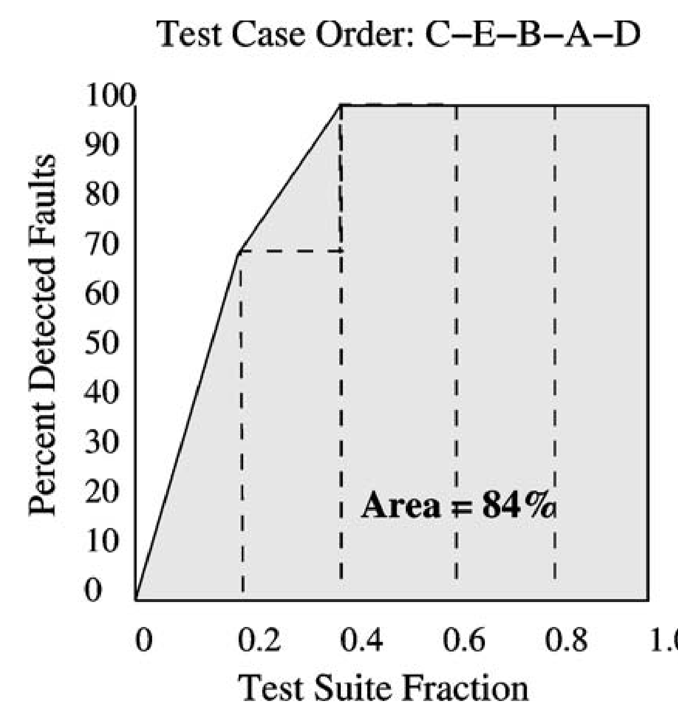
Regression: prioritisation

test	fault 1	2	3	4	5	6	7	8	9	10
A	x				x					
B	x				x	x	x			
C	x	x	x	x	x	x	x			
D					x					
E								x	x	x

(a)



(b)



(c)

Benefits of Abstraction

- Requirements
- Design
- Implementation
- Integration
- Testing
- Maintenance

Benefits of Abstraction

Requirements	subset selection	prioritisation
Design		
Implementation		
Integration		
Testing		
Maintenance		

Benefits of Abstraction

Requirements	subset selection	prioritisation
Design		
Implementation		
Integration		
Testing		
Maintenance	subset selection	prioritisation

Benefits of Abstraction

Requirements	subset selection	prioritisation
Design		
Implementation		
Integration		
Testing		
Maintenance	subset selection	prioritisation

Reformulating SE problems into optimisation problems reveals **hidden similarities**

Benefits of Abstraction

- Analytic Hierarchical Process: first used in Requirement Engineering, now also used for regression test prioritisation
- Average Percentage of Fault Detection: metric devised for regression test prioritisation, now being recast for prioritisation or requirements

Optimisation Techniques

- Genetic Algorithm: versatile, most popular (cool factor?)
- Hill climbing, Simulated Annealing: often as competitive as, or even better than, GA
- Exact methods: least widely used - scalable? flexible? multi-objectiveness?

Future Directions

- Multi-Objective Paradigm: already explored in testing and requirements, others to follow
  - Copes with complex constraints
  - Works well when there are multiple surrogate fitness

Future Directions

- Interactivity: relatively unexplored due to the high cost of human input
  - Eliciting human knowledge
  - Resolving ambiguities that are hard to quantise

Kasparov’s Advanced Chess

- Competition between teams consist of human + chess software
- It looks similar to our goal in a lot of ways...

Kasparov's Advanced Chess

- "..being able to access a database of a few million games meant that we didn't have to strain our memories nearly as much in the opening.."
- "Having a computer partner also meant never having to worry about making a tactical blunder."
- "Weak human + machine + better process was superior to a strong computer alone and, more remarkably, superior to a strong human + machine + inferior process."

Future Directions

- Our final goal is not to replace human decision making process; it is to **aid** the process with an **unbiased** alternative and an **insight** into the problem structure

References

- M. Harman, S. A. Mansouri, and Y. Zhang. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Technical Report TR-09-03, Department of Computer Science, King's College London, April 2009.
- Y. Zhang, M. Harman, and S. A. Mansouri. The Multi-Objective Next Release Problem. In GECCO '07: Proceedings of the 2007 Genetic and Evolutionary Computation Conference, pages 1129–1136. ACM Press, 2007.
- S. Yoo and M. Harman. Pareto efficient multi-objective test case selection. In Proceedings of International Symposium on Software Testing and Analysis (ISSTA 2007), pages 140–150. ACM Press, July 2007.
- S. Yoo, M. Harman, P. Tonella, and A. Susi. Clustering test cases to achieve effective & scalable prioritisation incorporating expert knowledge. In Proceedings of International Symposium on Software Testing and Analysis (ISSTA 2009), pages 201–211. ACM Press, July 2009.
- Gary Kasparov, "The Chess Master and the Computer", The New York Review of Books, http://www.nybooks.com/articles/23592