Sensitivity Analysis for Requirements Using Exact Optimization

Francisco Palomo Lozano

francisco.palomo@uca.es

UCASE Software Engineering Research Group

(Talk at CREST – University College London, May 2011)

UCA
Universidad de Cádiz

The Next Release Problem

1. Companies constantly release new products or versions
2. Requirements for a release can be of different kinds
   - Mandatory requirements vs. optional requirements
   - Commercial success of releases depends on both kinds
   - Decision-makers only control optional requirements
3. Requirements are subject to constrains
   - Policy
   - Time-to-market
   - Dependencies
   - Budget
4. Requirements have expected estimations
   - Costs
   - Revenues
5. Which optional requirements to implement?

Sensitivity analysis

1. Estimation is not an exact science
2. Requirements are affected by estimation errors
   - Costs are invariably underestimated
   - Revenues are usually more predictable
3. Budgets are subjected to cuts (now more true than ever)
4. Complex products go through different budgeting scenarios
5. Sensitivity analysis helps to identify the hot-spots
   - Efficiency issues
   - One-at-a-time approach
   - Exact versus approximate

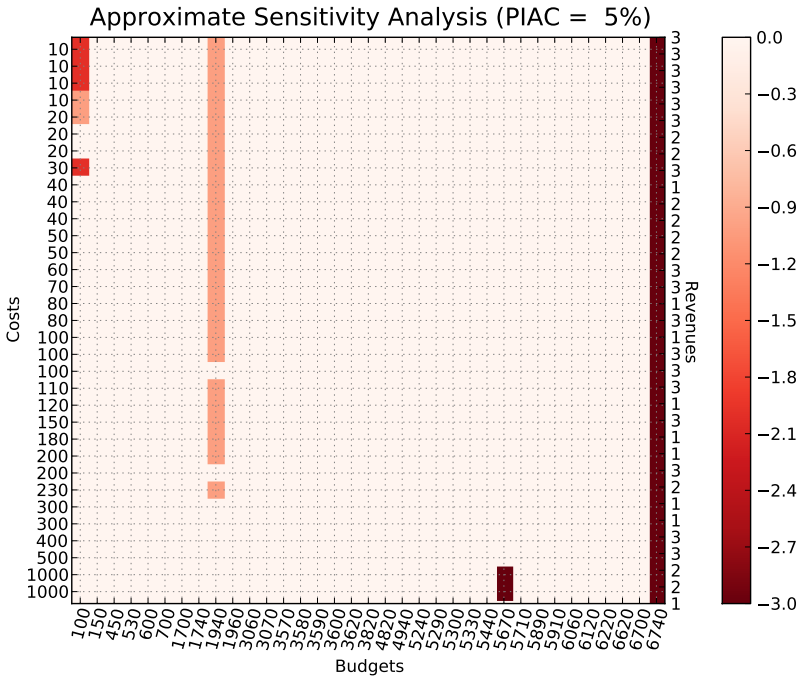
   How can we be sure that the observed variation stems from inherent sensitivity and not from the nature of the optimisation algorithm?

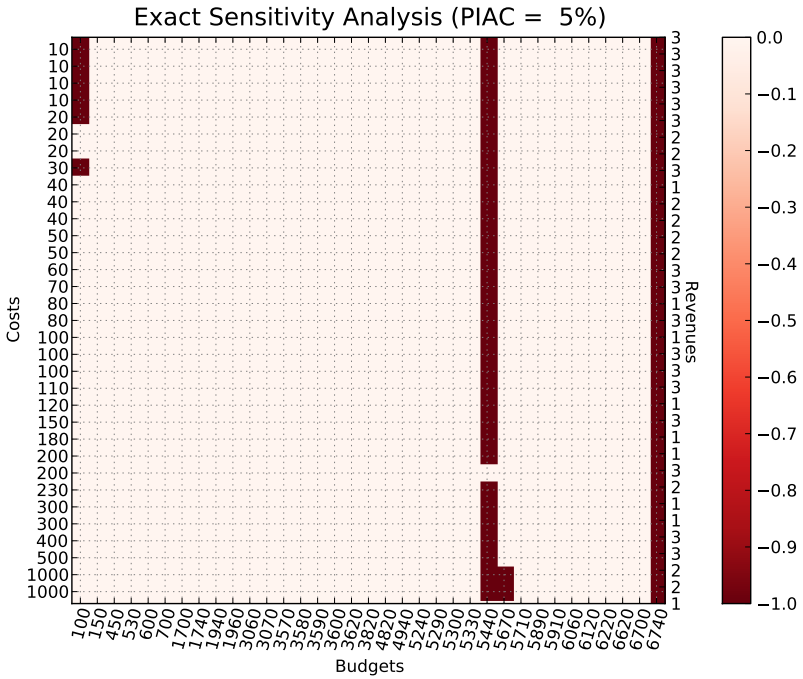
Industrial example: Motorola

- Motorola identified 40 requirements for a new product
- A small number of requirements were identified as mandatory
  - Any release of the product must include at least these
  - A fix amount of budget is devoted to this task
- Requirements can become mandatory by different reasons
  - Core product functionality
  - Dependence to other requirements
- Decision-makers faced 35 independent requirements
  - Costs were estimated
  - Revenues were based in customer assessments
- There are 2^{35} possible choices
  - $>$ 34000 million possibilities
  - $>$ 1000 million possibilities with half the total cost as budget

Sensitivity analysis of the Motorola data

1. 21 perturbations (from -50% to +50% with 5% steps)
2. Positive perturbations show the effect of underestimations





Knapsack formulation

The Next Release Problem (NRP)

- Three variants defined by Bagnall et al.
- Simplest NRP is just a classical knapsack problem in disguise
- Maximising revenues under budget constrains
  - Budget is known a priori
  - Optimal solutions exist and may not be unique

The Knapsack Problem (KP)

- First studied in the 1950s
- Basic variants are well understood
  - Most variants are *NP*-hard problems
  - Exact and approximation algorithms have been devised
  - We are lucky: KP is weakly *NP*-hard
  - KP admits pseudopolynomial time exact algorithms

Approximation algorithms

Guarantees

1. Different guarantees for errors depending on the algorithm
   - No guarantee: unbounded error algorithms
   - Relative error: $(1-\epsilon)$-approximation algorithms
   - Absolute error: absolute performance approx. algorithms
2. Two advanced schemes
   - Polynomial-time approximation schemes (PTAS)
   - Fully polynomial-time approximation schemes (FPTAS)

Facts

1. Guaranteeing absolute error for KP is as hard as exactness
2. Lucky again that KP is weakly *NP*-hard
3. FPTAS cannot exist for strongly *NP*-hard problems (if $P \neq NP$)

Greedy algorithms

Strategies

ICO Increasing Cost Order

DRO Decreasing Revenue Order

DRD Decreasing Revenue Density

- DRO and DRD can suffer from unbounded error
- DRD with postprocessing is an $\frac{1}{2}$-approximation algorithm

Advantages

1. Very fast and easy to implement
2. Do not impose artificial restrictions to their inputs
3. A natural basis for better algorithms, for example local search

Exact algorithms

Facts

1. Many algorithms have been devised in the last 50 years
2. Advanced algorithms are difficult to implement and test
3. For many algorithms space can be as critical as time

Approach

1. Focus in “simple” algorithms and data structures
2. Algorithms have to be efficient just for the application domain

No matter how good our exact algorithms are, they will always behave bad for an infinite number of instances, as long as $P \neq NP$

Dynamic programming algorithms

Bellman's equation for KP

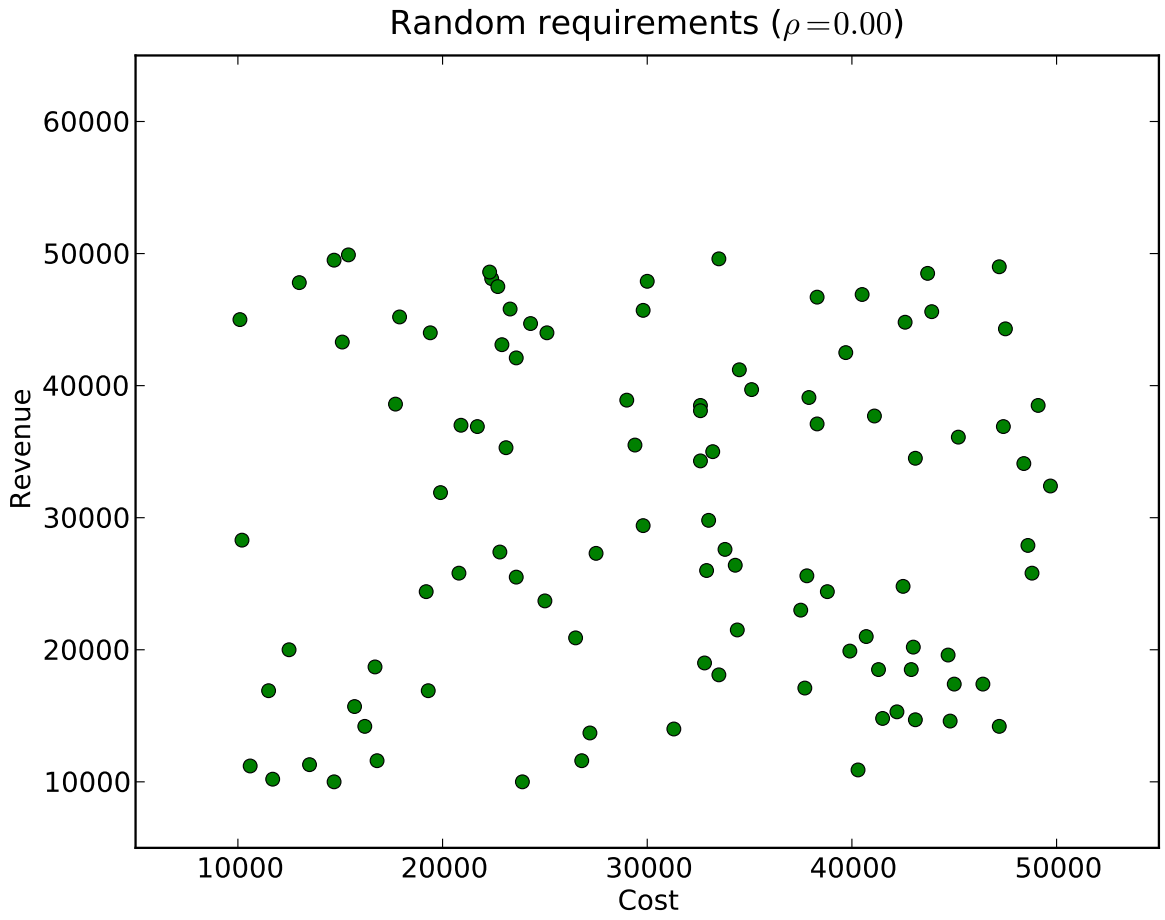
$$z(n,B) = \begin{cases} 0 & \text{if } n = 1 \wedge c_1 > B \\ r_1 & \text{if } n = 1 \wedge c_1 \leq B \\ z(n-1,B) & \text{if } n > 1 \wedge c_n > B \\ \max\{z(n-1,B), z(n-1,B-c_n) + r_n\} & \text{if } n > 1 \wedge c_n \leq B \end{cases}$$

Nemhauser-Ullmann's algorithm (NU)

1. As values of z are computed, they are saved for later reuse
2. Solution is recovered from the values of z previously saved
3. In 2003, Beier and Vöcking proved very interesting properties
   - NU solves a random KP in expected polynomial time
   - This is so under quite general conditions
   - Strongly correlated instances are provably harder for NU

Correlation

1. Highly-correlated instances are reported hard in the literature
2. We control Pearson's correlation during instance generation



Scalability experiments

1. 50 different datasets with 315 NRP random instances each
   - 15 problem sizes from 100 to 1500, with steps of 100
   - 21 correlation degrees from 0% to 100%, every 5%
2. Significant # of experiments: 15 750 instances solved
3. Experiments performed in just one core of a 1000 £ machine
   - Intel Core i7 2.67 GHz CPU with 12 GiB RAM
   - C++ on GNU/Linux
4. Results show that the model

$$t(n,\rho) = an^2 \exp \rho + bn^2 + cn \log n$$

fits quite well to our observed times

- NU behaves polynomially in the number of requirements
- Except when costs and revenues are very highly correlated

Conclusions

1. Exact algorithms enable precise sensitivity analysis (PSA)
   - Necessary to isolate estimation from approximation errors
   - Approximation errors may trick the decision-maker
2. Scalability is achievable, at least for reasonable cases
3. However, PSA is still expensive
   - 21 perturbations (from -50% to +50% with 5% steps)
   - $21 \cdot n$ instances for a single budget

Still, we can complete a PSA of a 500 requirements project for one budget proposal in less than one day

Ongoing work

Multi-objective NRP

- Maximising revenues while minimising costs
- A Holy Grail
  - Costs and revenues are conflicting objectives
  - Budget may be known a priori or a posteriori
- Pareto-optimal solutions instead of absolute, optimal solutions
- Different approaches
  - Use multi-objective KP as a model to find the Pareto frontier
  - Produce the Pareto frontier as a byproduct of a KP algorithm

Parallelisation

- Different budgets can be analysed at the same time
- As well as different cost perturbations when a budget is fixed

Open questions

1. How can we compare SBSE methods objectively?
2. How can we make testing these algorithms easier?
3. Are there any other areas where exact methods are key?
4. Is SBSE just about heuristics or also about exact methods?

References

- A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley.
  The next release problem.
  Information and Software Technology, 43(14):883–890, 2001.
- R. Beier and B. Vöcking.
  Random knapsack in expected polynomial time.
  In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, 2003.
- V. Boyer, D. El Baz, and M. Elkihel.
  Solving knapsack problems on GPU.
  Computers & Operations Research, to appear.
- H. Kellerer, U. Pferschy, and D. Pisinger.
  Knapsack Problems.
  Springer, 2004.