A CP approach of the variability testing of software product lines

Arnaud Gotlieb
Certus Centre
Simula Research Laboratory
Norway
The Certus Centre

Software Validation and Verification

Hosted by SIMULA

Established and awarded SFI in Oct. 2011

duration: 8 years

www.certus-sfi.no
The Validation of CISCO’s Video Conferencing Product Line

In a continuous integration cycle

1. Test case selection
2. Test suite optimization
3. Test execution scheduling

Test suite optimization

How to select a test set which cover all the features in an acceptable amount of time (i.e., cost-effective optimization)?
Agenda

I. Introduction

II. Optimal Test Suite Reduction

III. Multi-objectives Test Suite Reduction

IV. Industrial Application

V. Conclusions and Perspectives
Optimal Test Suite Reduction
Optimal TSR: the core problem

F_i: Features or requirements
TC_i: Test case or test script

Optimal TSR: find a minimal subset of TC such that each F is covered at least once (Practical importance but NP-hard problem!) – An instance of Minimum Set Cover
Constraint Programming

Declarative programming paradigm where relations are modeled with variables, finite and continuous domains, and constraints

- e.g., arithmetical, \( X \text{ in } 13..59, Y \text{ in } 4..9, Y > 6*Y - X \)

- e.g., symbolic (terms, string..) \( t(X, r(3, Y)) = t(p(4), Z) \)

- e.g., numerical \( X = \sin(Y), X + Y = 0.567898 \)

**Global constraint:** A constraint which captures a relation over a non-fixed number of variables and implements a dedicated filtering algorithm
The nvalue global constraint

\[ nvalue(n, v) \]

Where:

- \( n \) is an FD_variable
- \( v = (v_1, \ldots, v_k) \) is a vector of FD_variables

\[ nvalue(n, v) \text{ holds iff } n = card(\{v_i\}_{i \in 1..k}) \]

Introduced in [Pachet and Roy’99], first filtering algorithm in [Beldiceanu’01]
Solution existence for nvalue is NP-hard [Bessiere et al. ‘04]
Optimal TSR: CP model with nvalue (1)

\[ F_1 \text{ in } \{1, 2, 6\}, \ F_2 \text{ in } \{3, 4\}, \ F_3 \text{ in } \{2, 5\} \]
\[ nvalue(\text{MaxNvalue}, (F_1, F_2, F_3)) \]
\[ \text{label(minimize(MaxNvalue))} \]
The global_cardinality constraint

\[ gcc(t, d, v) \]

Where

\[ t = (t_1, \ldots, t_N) \] is a vector of \( N \) variables, each \( t_j \) in \( Min_j \ldots Max_j \)
\[ d = (d_1, \ldots, d_k) \] is a vector of \( k \) values
\[ v = (v_1, \ldots, v_k) \] is a vector of \( k \) variables, each \( v_i \) in \( Min_i \ldots Max_i \)

\[ gcc(t, d, v) \] holds iff \( \forall i \) in \( 1 \ldots k \),
\[ v_i = \text{card}( \{ t_j = d_i \}_{j \text{ in } 1 \ldots N} ) \]

Filtering algorithms for \( gcc \) are based on max flow computations in a network flow [Regin AAAI’96]
Example

\[ \text{gcc}( (F_1, F_2, F_3), (1,2,3,4,5,6), (V_1, V_2, V_3, V_4, V_5, V_6)) \]

means that:

In a solution of TSR
TC\(_1\) covers exactly \( V_1 \) requirements in \((F_1, F_2, F_3)\)
TC\(_2\) " \( V_2 \) "
TC\(_3\) " \( V_3 \) "
...

Where \( F_1, F_2, F_3, V_1, V_2, V_3, \ldots \) denote finite-domain variables

\( F_1 \) in \{1, 2, 6\}, \( F_2 \) in \{3, 4\}, \( F_3 \) in \{2, 5\}
\( V_1 \) in \{0, 1\}, \( V_2 \) in \{0, 2\}, \( V_3 \) in \{0, 1\}, \( V_4 \) in \{0, 1\}, \( V_5 \) in \{0, 1\}, \( V_6 \) in \{0, 1\}

Here, for example, \( V_1 = 1, V_2 = 2, V_3 = 1, V_4 = 0, V_5 = 0, V_6 = 0 \) is a feasible solution

But, not an optimal one!
Optimal TSR: CP model with two gcc


F₁ in {1, 2, 6}, F₂ in {3, 4}, F₃ in {2, 5}
gcc( (F₁, F₂, F₃), (1, 2, 3, 4, 5, 6), (V₁, V₂, V₃, V₄, V₅, V₆) ),
gcc((V₁, V₂, V₃, V₄, V₅, V₆), (0-_), (Max0Req-_ )),
label(maximize(Max0Req))

/* search heuristics by enumerating the Vi first */
3. Optimal TSR: CP model Mixt (3)


\[
F_1 \text{ in } \{1, 2, 6\}, \quad F_2 \text{ in } \{3, 4\}, \quad F_3 \text{ in } \{2, 5\}
gcc( (F_1, F_2, F_3), (1,2,3,4,5,6), (V_1, V_2, V_3, V_4, V_5, V_6) ),
nvalue(\text{MaxNvalue}, (F_1, F_2, F_3),
label(\text{minimize(\text{MaxNvalue}))}
\]

/* + presolve + labelling heuristics based on max */
Model comparison on random instances
(Reduced Test Suite percentage in 30sec of search)
Model comparison on random instances (CPU time to find a global optimum)

<table>
<thead>
<tr>
<th>Requirements</th>
<th>TD1</th>
<th>TD2</th>
<th>TD3</th>
<th>TD4</th>
<th>TD5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test cases</td>
<td>20</td>
<td>90</td>
<td>60</td>
<td>60</td>
<td>30</td>
</tr>
<tr>
<td>Density</td>
<td>70</td>
<td>100</td>
<td>100</td>
<td>200</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>8</td>
</tr>
</tbody>
</table>
Optimal TSR: existing approaches

- Exact method: ILP formulation [Hsu Orso ICSE 2009] –
  MINTS/CPLEX, MINTS/MiniSAT

Minimize \( \sum_{i=1..6} x_i \) (minimize the number of test cases)

subject to

\[
\begin{align*}
  x_1 + x_2 + x_6 & \geq 1 \\
  x_3 + x_4 & \geq 1 \\
  x_2 + x_5 & \geq 1
\end{align*}
\]

(cover every req. at least once)

- Approximation algorithms (greedy) –

R = Set of reqs, Current = Ø
while( Current \( \notin \) R)
  Select a test case that covers the most uncovered reqs ;
  Add covered reqs to Current ;
return Current
Comparison with other approaches
(Reduced Test Suite percentage in 60 sec)

<table>
<thead>
<tr>
<th>Requirements</th>
<th>TD1</th>
<th>TD2</th>
<th>TD3</th>
<th>TD4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test cases</td>
<td>1000</td>
<td>1000</td>
<td>1000</td>
<td>2000</td>
</tr>
<tr>
<td>Density</td>
<td>5000</td>
<td>5000</td>
<td>5000</td>
<td>5000</td>
</tr>
<tr>
<td>Density</td>
<td>7</td>
<td>7</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>
Introducing model presolve

$F_1$ in $\{1, 2, 6\} \rightarrow F_1 = 2$ as $\text{cov}(TC_1) = \text{cov}(TC_6) \subseteq \text{cov}(TC_2)$
withdraw $TC_1$ and $TC_6$

$F_3$ is covered $\rightarrow$ withdraw $TC_5$

$F_2$ in $\{3, 4\} \rightarrow$ e.g., $F_2 = 3$, withdraw $TC_4$

We proposed an iterative algorithm to apply these preprocessing rules to simplify the problem
Presolve: Experimental results (1)
Presolve: Experimental results (2)

Presolve removes 380 test cases for both MINTS/CPLEX and Flower.
Multi-objectives Test Suite Reduction
Optimal TSR: the core problem

Requirements coverage is always a prerequisite but other criteria than the size of the test suite are also sought:

- TC1: 1 min
- TC2: 5 min
- TC3: 3 min
- TC4: 3 min
- TC5: 1 min
- TC6: 1 min

Optimal TSR

Execution time!
Optimal TSR: the core problem

Requirements coverage is always a prerequisite but other criteria than the size of the test suite are also sought:

Fault revealing capabilities!
Proposed approaches

1. Actual multi-objectives optimization with search-based algorithms (Pareto Front)

   Aggregated cost function using RW-algo, URW-algo, and many others
   Based on computed values

   **No constraint model!**

2. Cost-based single-objective constrained optimization
   Based on a CP model with global constraints

   **Constrained optimization model!**
Flower/C: An extension of Flower with costs

\[ R_1, \ldots, R_n: \text{Requirements} \]
\[ t_1, \ldots, t_m: \text{Test cases} \quad \text{- Each test case } t_i \text{ is associated a unitary cost } c_i \]
\[ O_1, \ldots, O_m: \text{Occurrences variables} \]

Minimize TotalCost

s.t

\[ \text{gcc}( (R_1, \ldots, R_n), (t_1, \ldots, t_m), (O_1, \ldots, O_m) ) \]
\[ \text{for } i=1 \text{ to } m \text{ do } B_i = (O_i > 0) \]
\[ \text{scalar_product}((B_1, \ldots, B_m), (c_1, \ldots, c_m), \text{TotalCost}) \]

where scalar_product encodes \( B_1 c_1 + \ldots + B_m c_m = \text{TotalCost} \)
Industrial Application
Variability model to describe a software product line

Unoptimized test suite

Diagnostic views, feature coverage

Optimized (reduced/prioritized) test suite
TITAN

- Reusing Pure::Variants plug-in for feature modelling and editing
- Desktop version + web-based service
- Patent under advisement in the US
- Deployed at Cisco Systems
- Commercial development (funded under the RCN’s FORNY program)
Conclusions

• Global constraints and CP can efficiently and effectively tackle difficult software validation problems – experimental results and initial industrial case studies

• So far, the links between feature modelling and software product line engineering and software validation has been little studied

• There is room for Research and Innovation in that area!