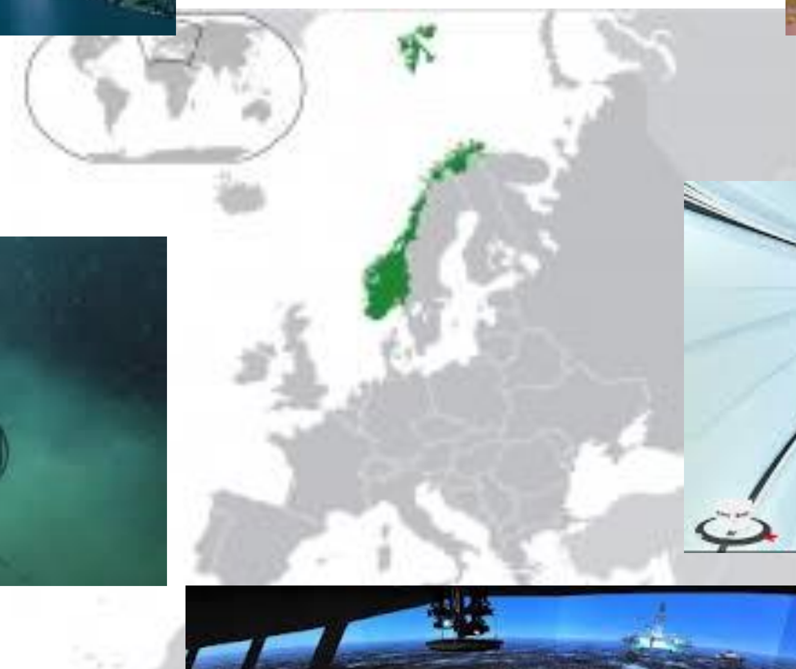
A CP approach of the variability testing of software product lines

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

Cisco Systems Norway

Norwegian Custom and excise

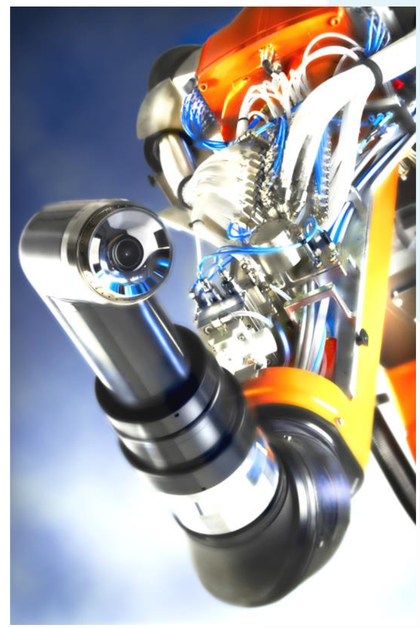


ABB Robotics Stavanger

Kongsberg Maritime

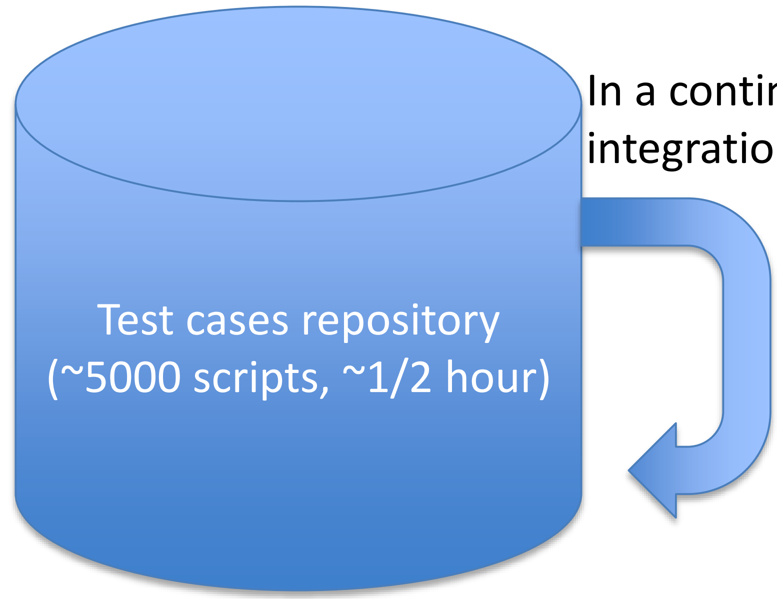
[simula . research laboratory]
- by thinking constantly about it

esito

Certus

sfi Centre for Research-based Innovation

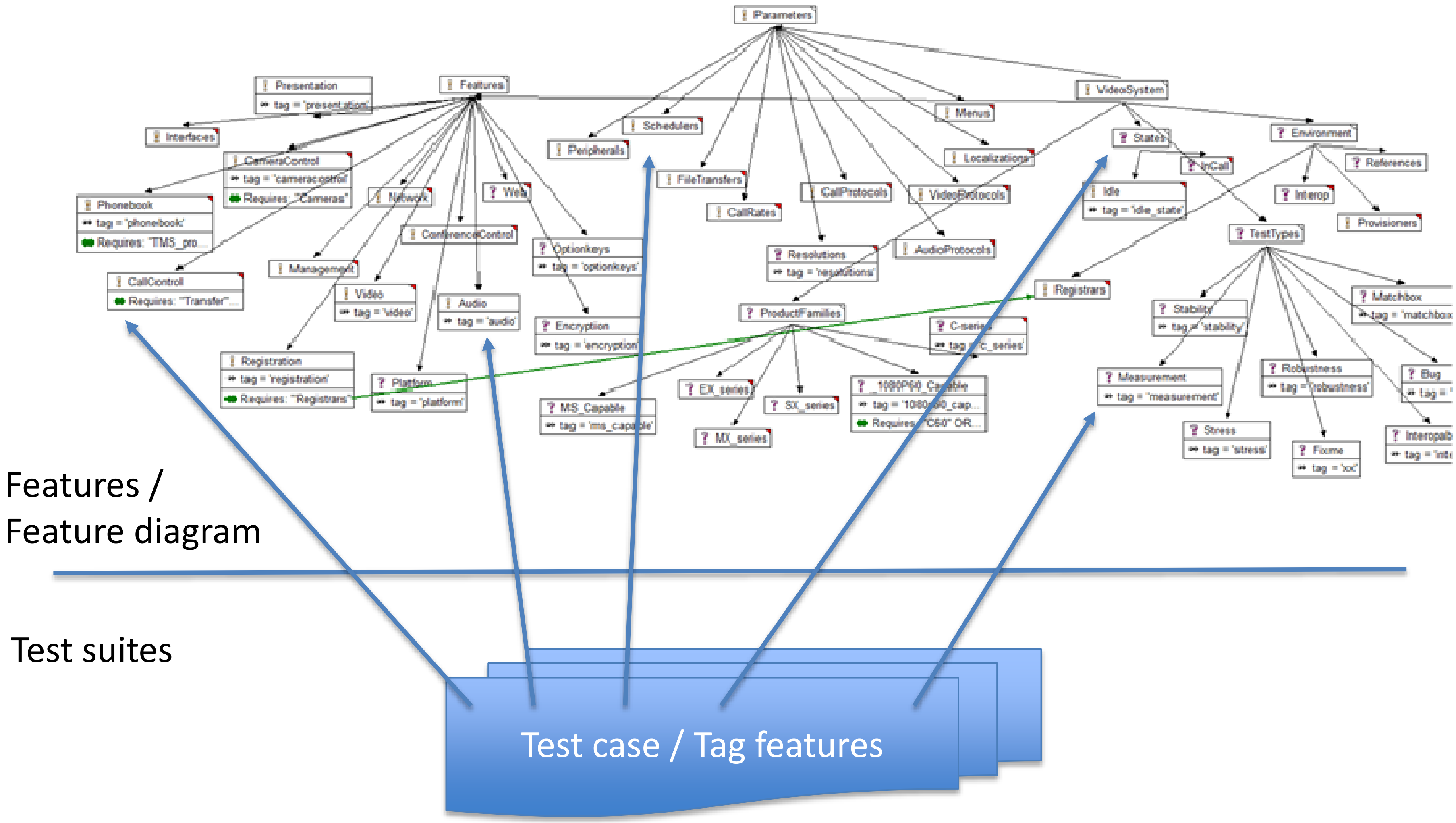
The Validation of CISCO's Video Conferencing Product Line



In a continuous integration cycle

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

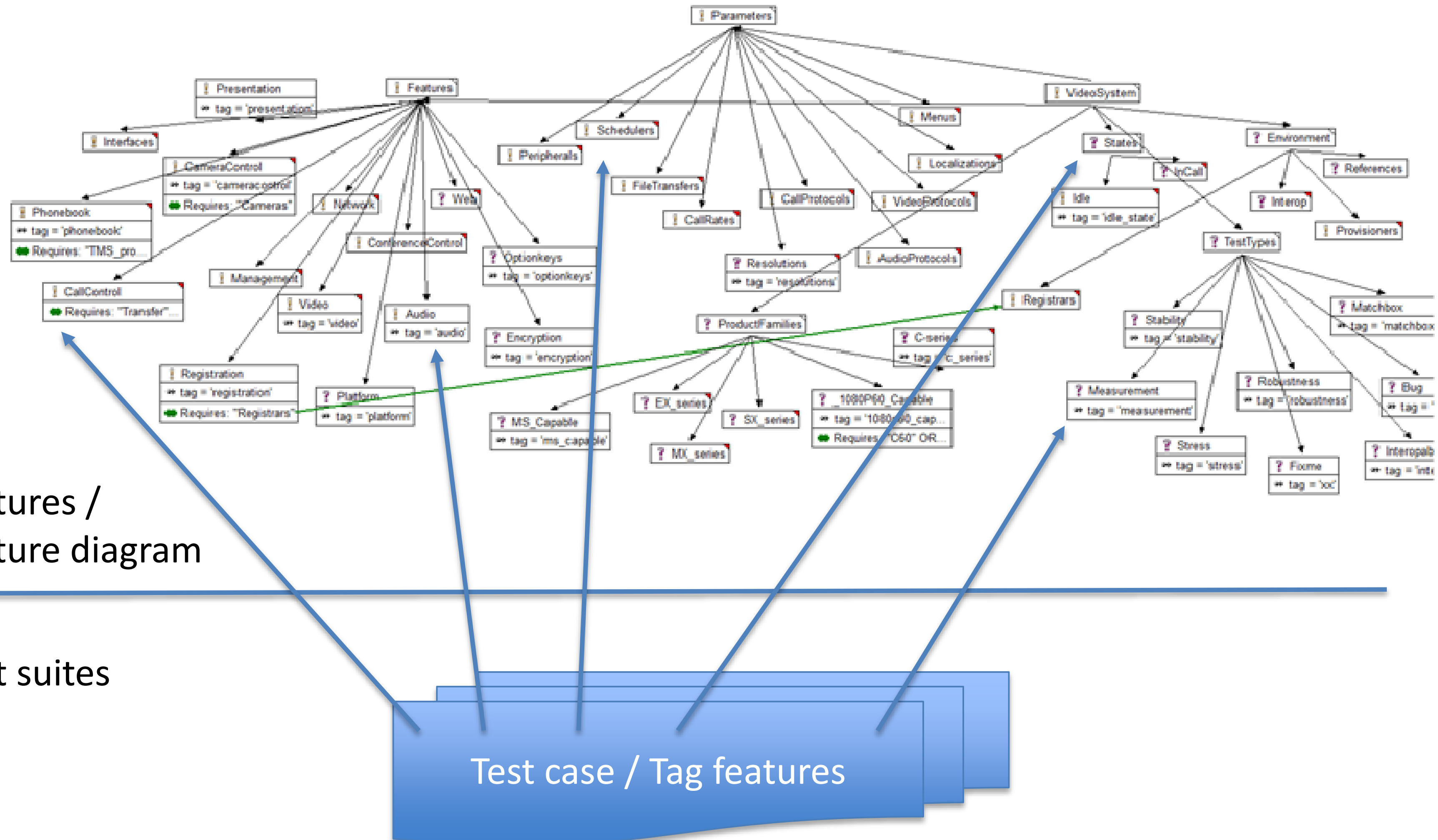
Test case selection based on feature modelling



S. Wang, S. Ali, A. Gotlieb, and M. Liaaen. A systematic test case selection methodology for product lines: results and insights from an industrial case study. *Empirical Software Engineering*, pages 1–37, 2014.

S. Wang, A. Gotlieb, S. Ali, and M. Liaeen. Automated test case selection using feature model: An industrial case study. In *ACM/IEEE 16th Int. Conf. on Model Driven Eng. Languages and Systems (MODELS'13), Awarded best application paper*, Miami, FL, USA, Sep. 2013.

Test suite optimization



Features /
Feature diagram

Test suites

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

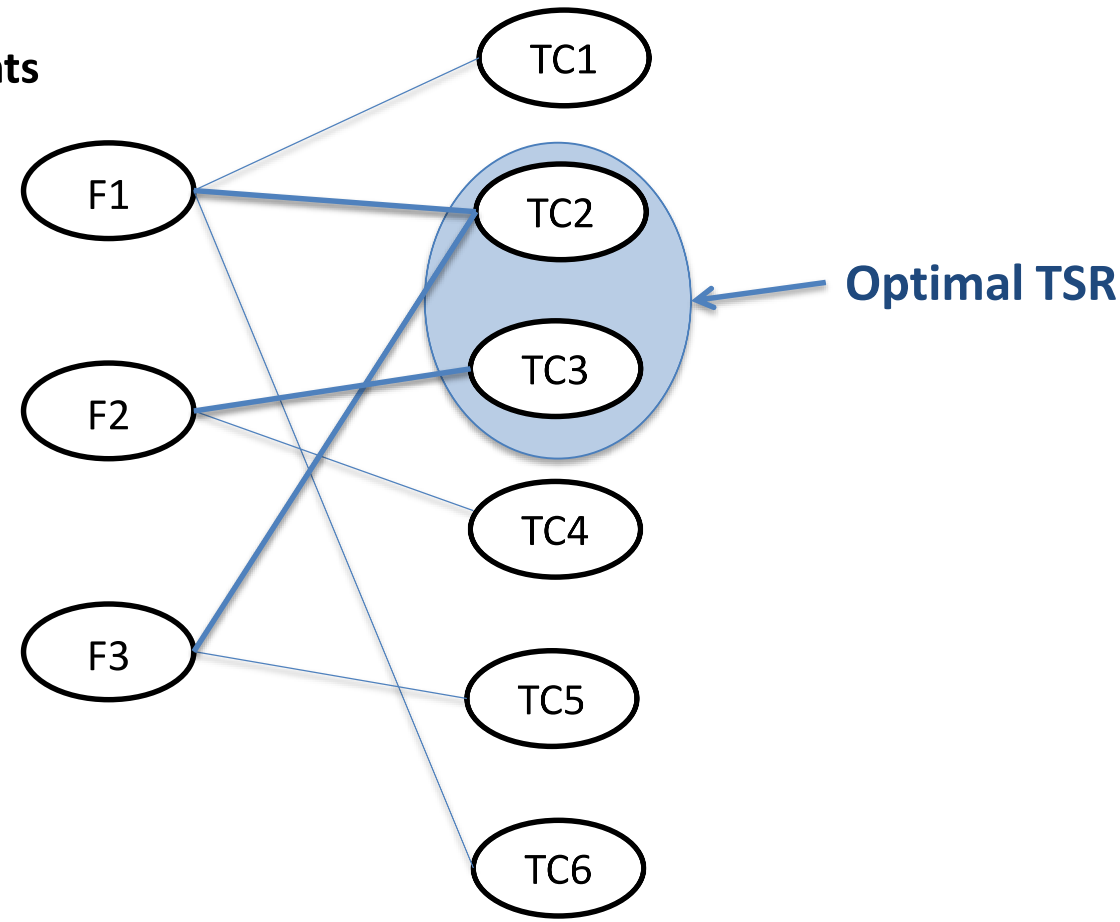
IV. Industrial Application

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 **in** 13..59, Y **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:

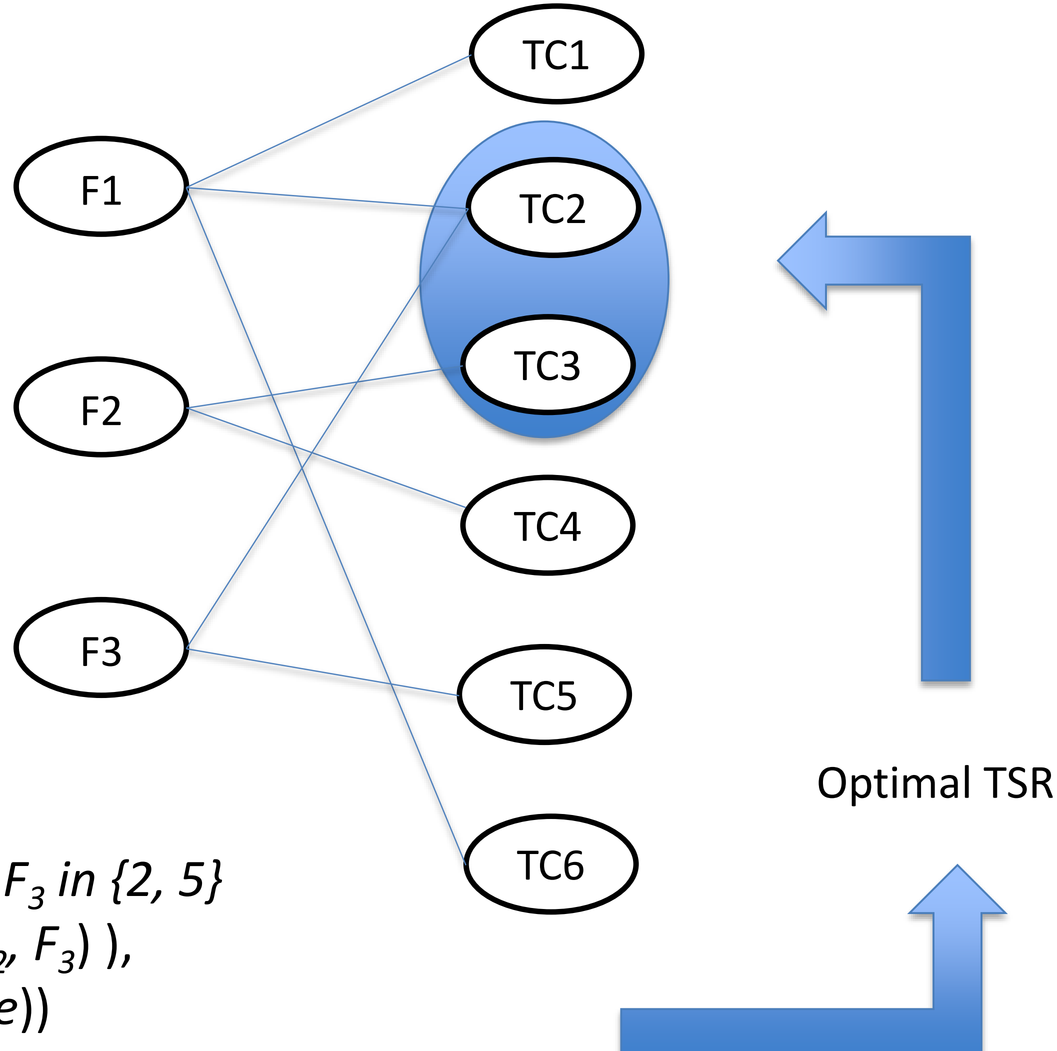
n is an FD_variable

$v = (v_1, \ldots, \; v_k)$ is a vector of FD_variables

$nvalue(n, v)$ 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 *in {1, 2, 6},* F_2 *in {3, 4},* F_3 *in {2, 5}*
nvalue(*MaxNvalue, (*F_1*,* F_2*,* F_3)),
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 $Min_j .. Max_j$

$d = (d_1, \ldots, d_k)$ is a vector of k values

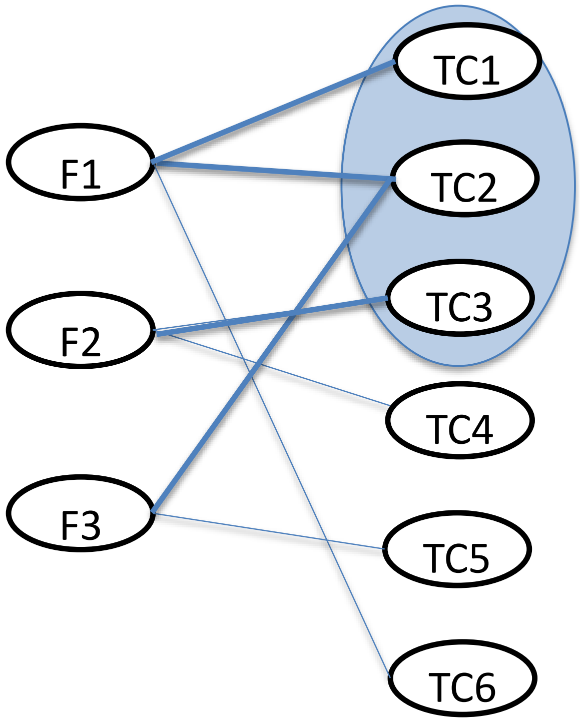
$v = (v_1, \ldots, v_k)$ is a vector of k variables, each v_i in $Min_i..Max_i$

$gcc(t, d, v)$ holds iff

$$\forall i \ in \ 1..k,$$
$$v_i = card(\ \{t_j = di\}_{j \ in \ 1..N})$$

Filtering algorithms for *gcc* are based on max flow computations in a network flow [Regin AAAI'96]

Example



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 $F_1, F_2, F_3, V_1, V_2, V_3$, ... denote finite-domain variables

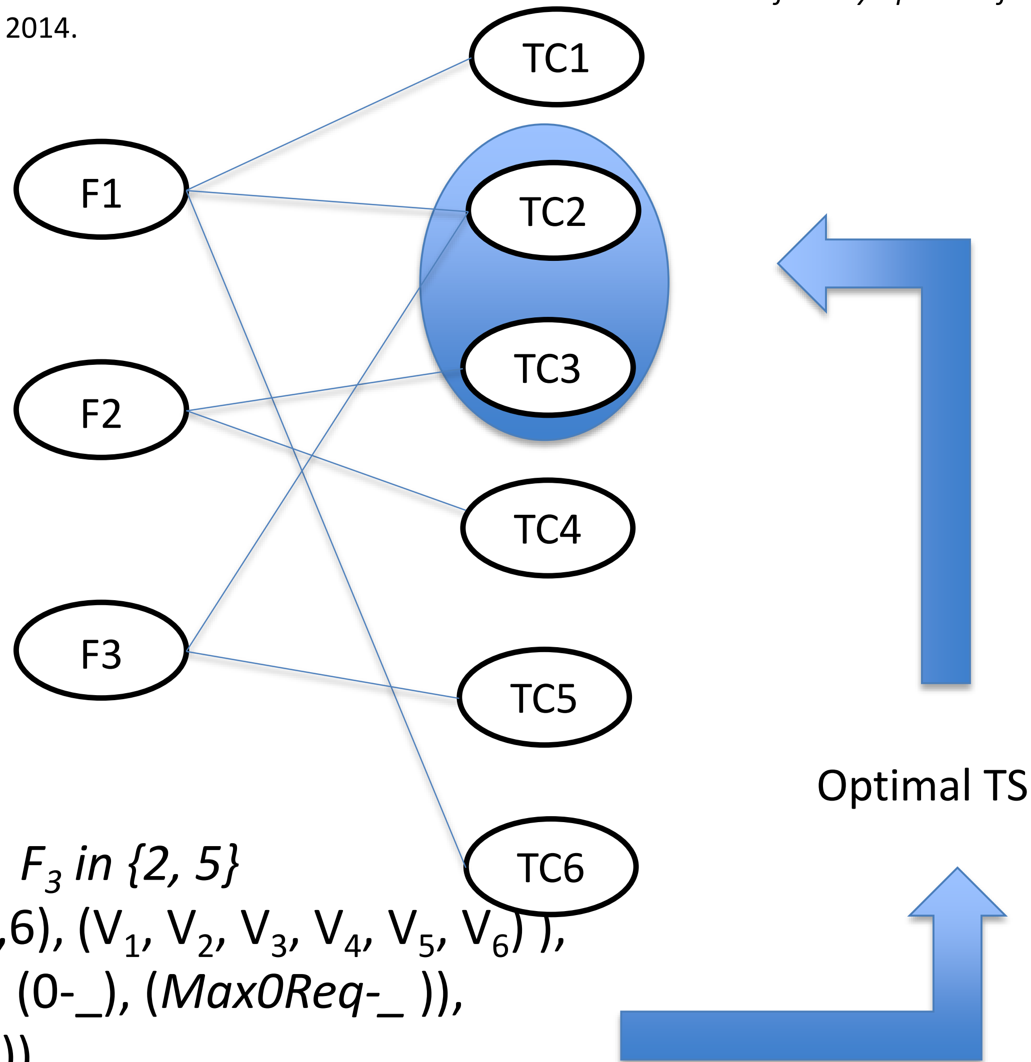
F_1 in {1, 2, 6}, F_2 in {3, 4}, F_3 in {2, 5}
V_1 in {0, 1}, V_2 in {0, 2}, V_3 in {0, 1}, V_4 in {0, 1}, V_5 in {0, 1}, V_6 in {0, 1}

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 (2)

A. Gotlieb and D. Marijan. Flower: Optimal test suite reduction as a network maximum flow. In *Proc. of Int. Symp. on Soft. Testing and Analysis (ISSTA'14)*, San José, CA, USA, Jul. 2014.

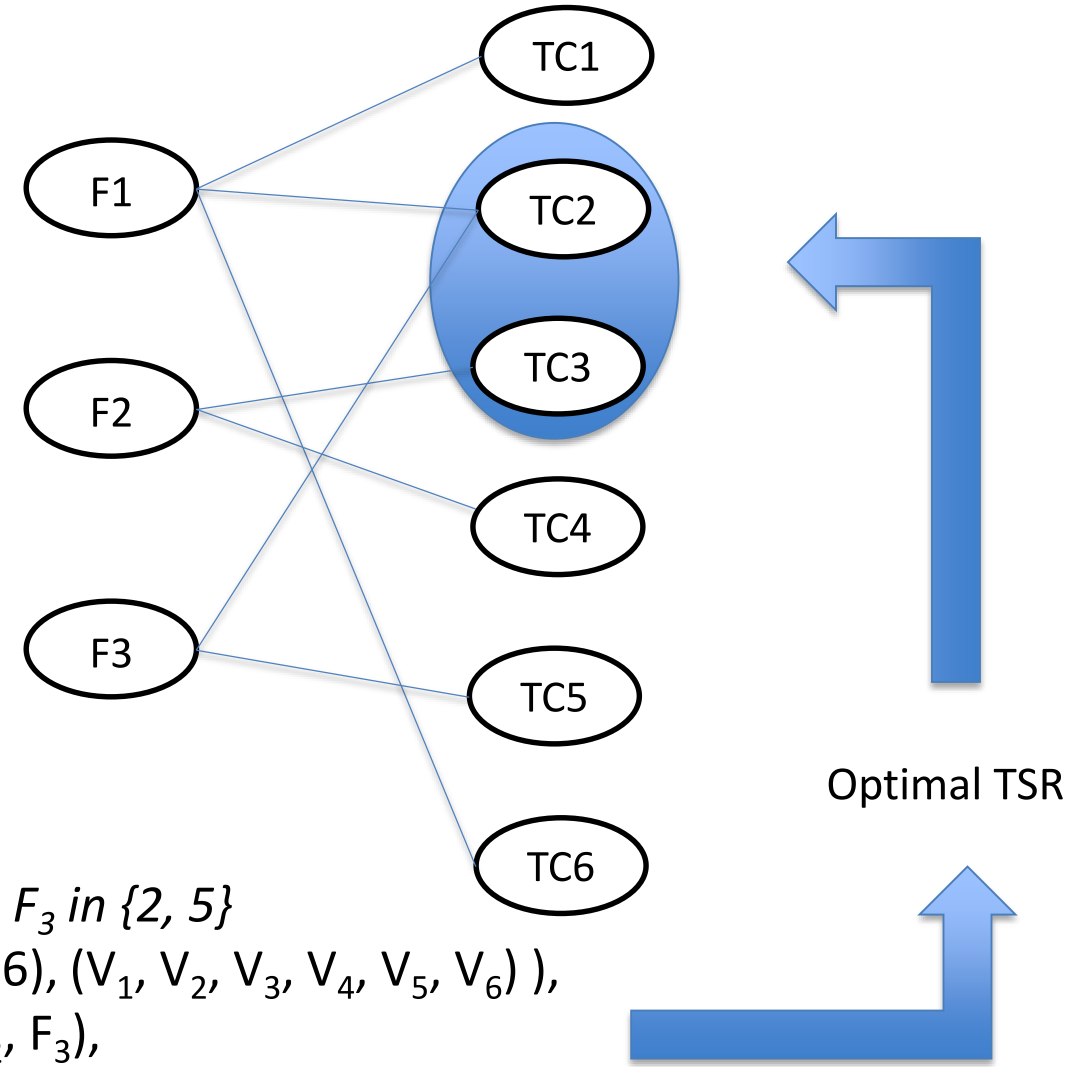


F_1 *in {1, 2, 6}*, F_2 *in {3, 4}*, F_3 *in {2, 5}*
gcc((F_1, F_2, F_3), (1,2,3,4,5,6), (V_1, V_2, V_3, V_4, V_5, V_6)),
gcc((V_1, V_2, V_3, V_4, V_5, V_6), (0-_), (*MaxOReq*-_)),
label(maximize(*MaxOReq*))

/* search heuristics by enumerating the Vi first */

3. Optimal TSR: CP model Mixt (3)

A. Gotlieb M. Carlsson M. Liaeen D. Marijan A. Petillon. *Automated Regression Testing using CP*. Under submission 2015



F_1 in {1, 2, 6}, F_2 in {3, 4}, F_3 in {2, 5}

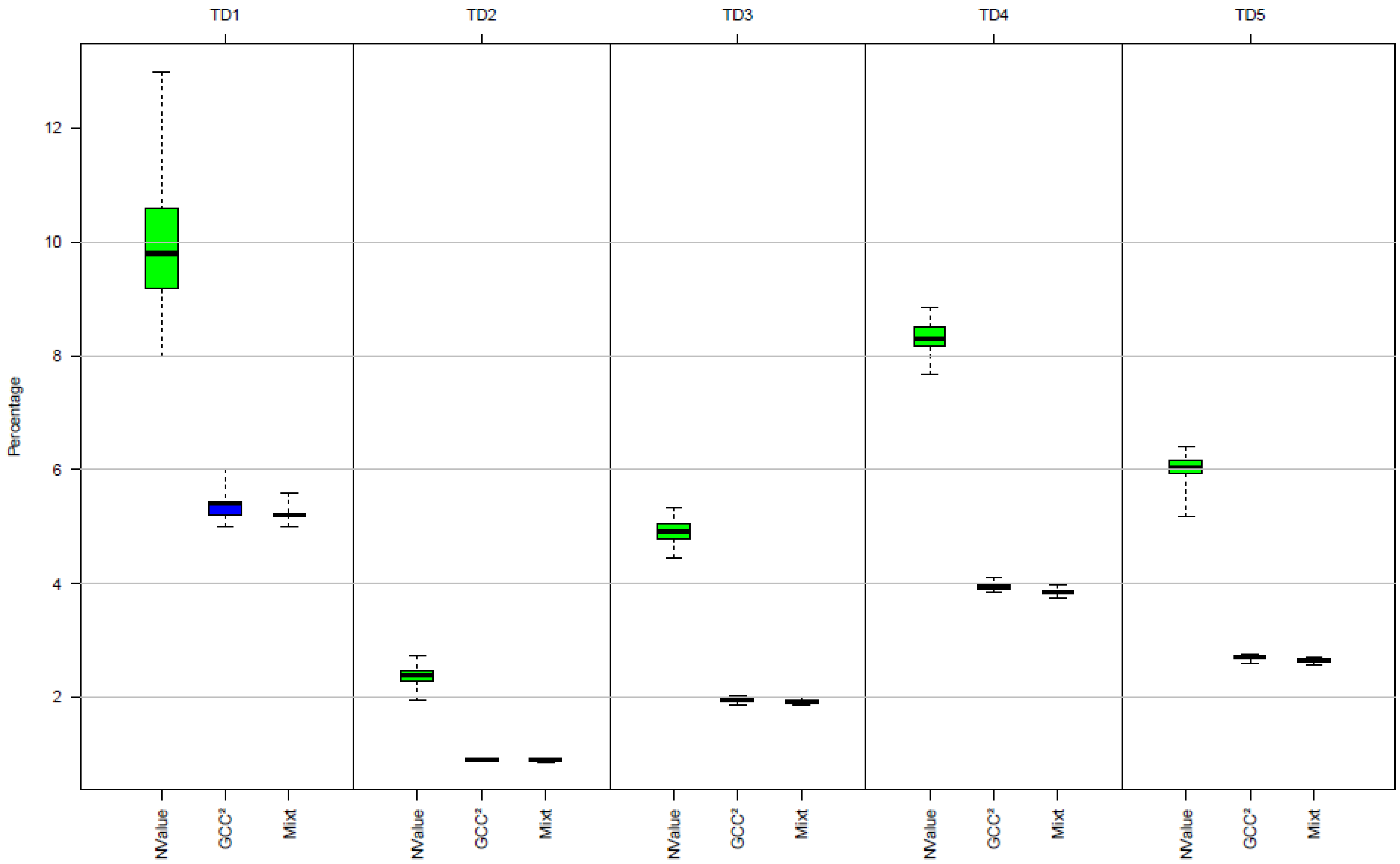
gcc((F_1, F_2, F_3), (1,2,3,4,5,6), (V_1, V_2, V_3, V_4, V_5, V_6)),

nvalue(MaxNvalue, (F_1, F_2, F_3),

label(minimize(MaxNvalue))

/* + presolve + labelling heuristics based on max */

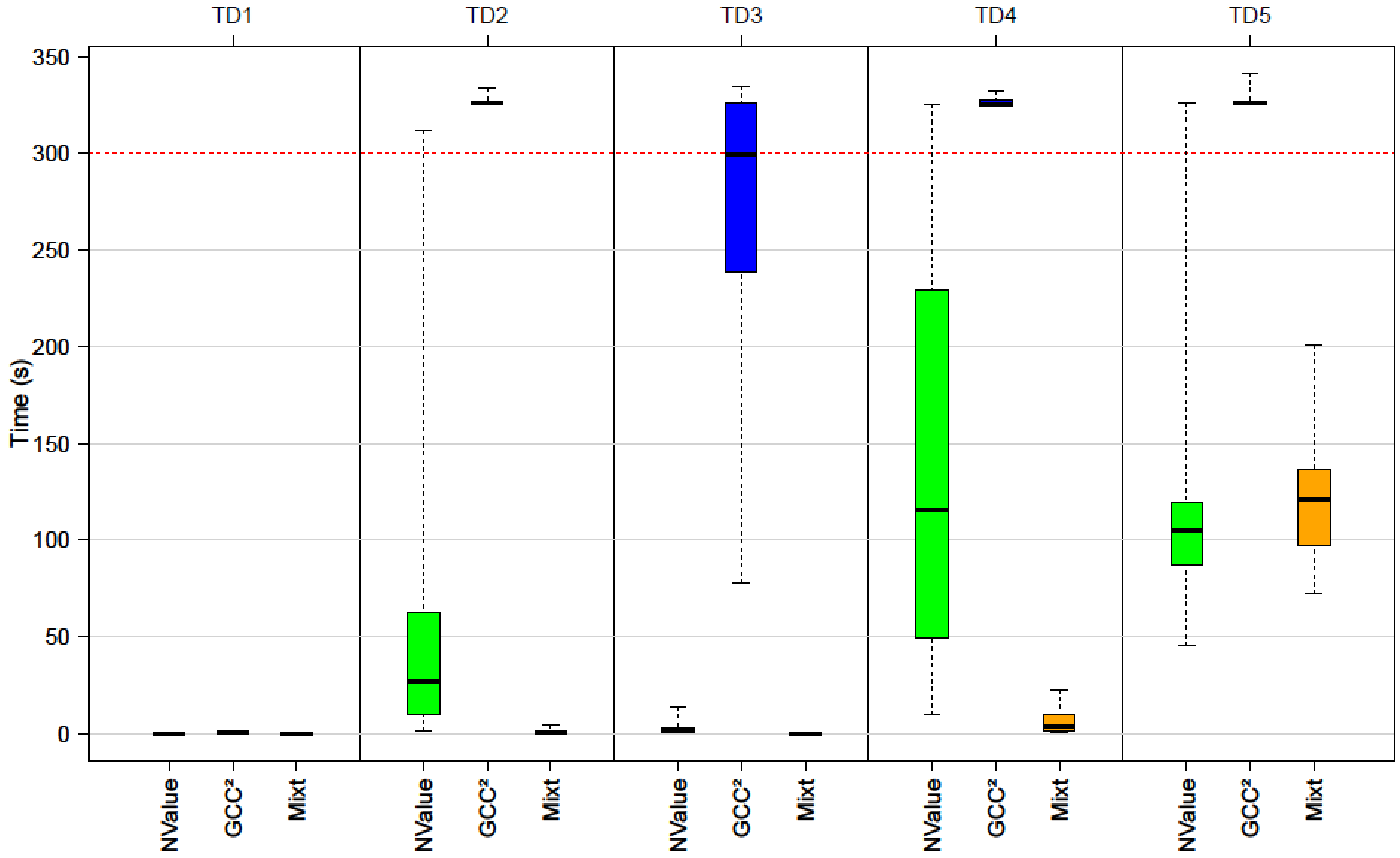
Model comparison on random instances
(Reduced Test Suite percentage in 30sec of search)



	TD1	TD2	TD3	TD4	TD5
Requirements	250	500	1000	1000	1000
Test cases	500	5000	5000	5000	7000
Density	20	20	20	8	8

Model comparison on random instances

(CPU time to find a global optimum)



	TD1	TD2	TD3	TD4	TD5
Requirements	20	90	60	60	30
Test cases	70	100	100	200	500
Density	8	20	20	20	8

Optimal TSR: existing approaches

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

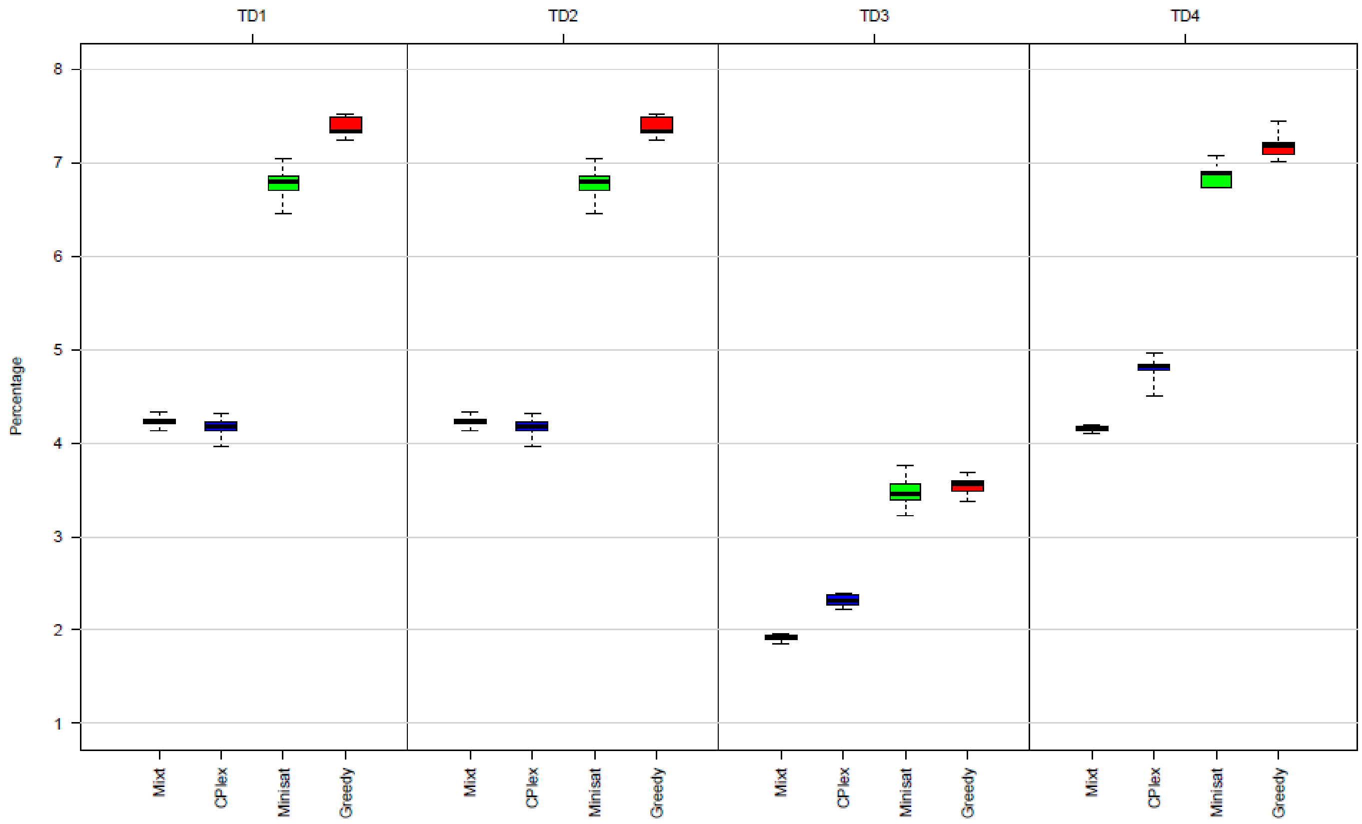
Minimize $\sum_{i=1..6} xi$ (minimize the number of test cases)

subject to $\begin{Bmatrix} x1 + x2 + x6 \geq 1 \\ x3 + x4 \geq 1 \\ x2 + x5 \geq 1 \end{Bmatrix}$ (cover every req. at least once)

- Approximation algorithms (greedy) –

```
R = Set of reqs, Current = Ø
while( Current ⊈ 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)



	TD1	TD2	TD3	TD4
Requirements	1000	1000	1000	2000
Test cases	5000	5000	5000	5000
Density	7	7	20	20

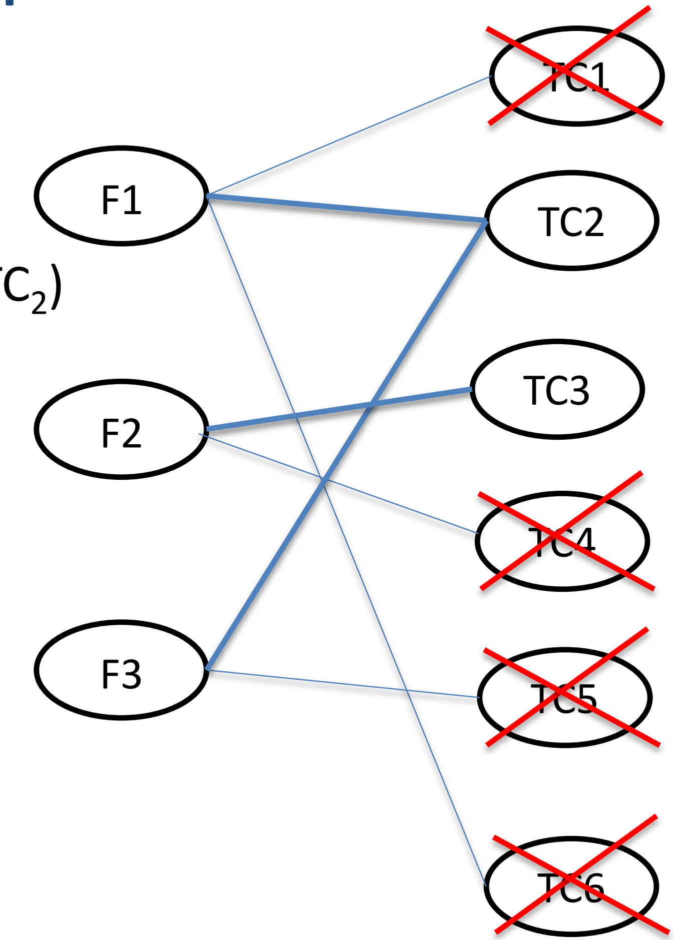
Introducing model presolve

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

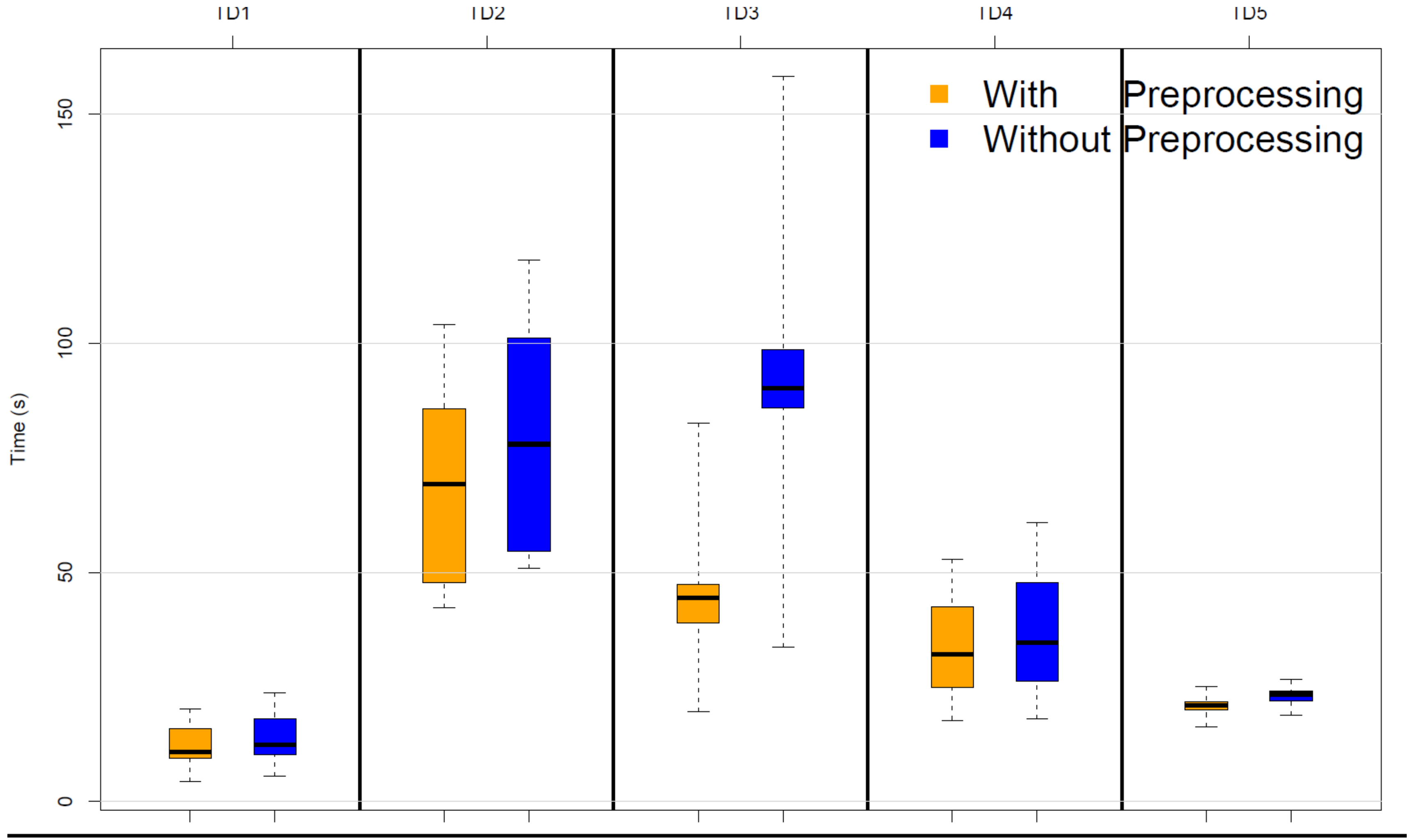
F_3 is covered → withdraw TC_5

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

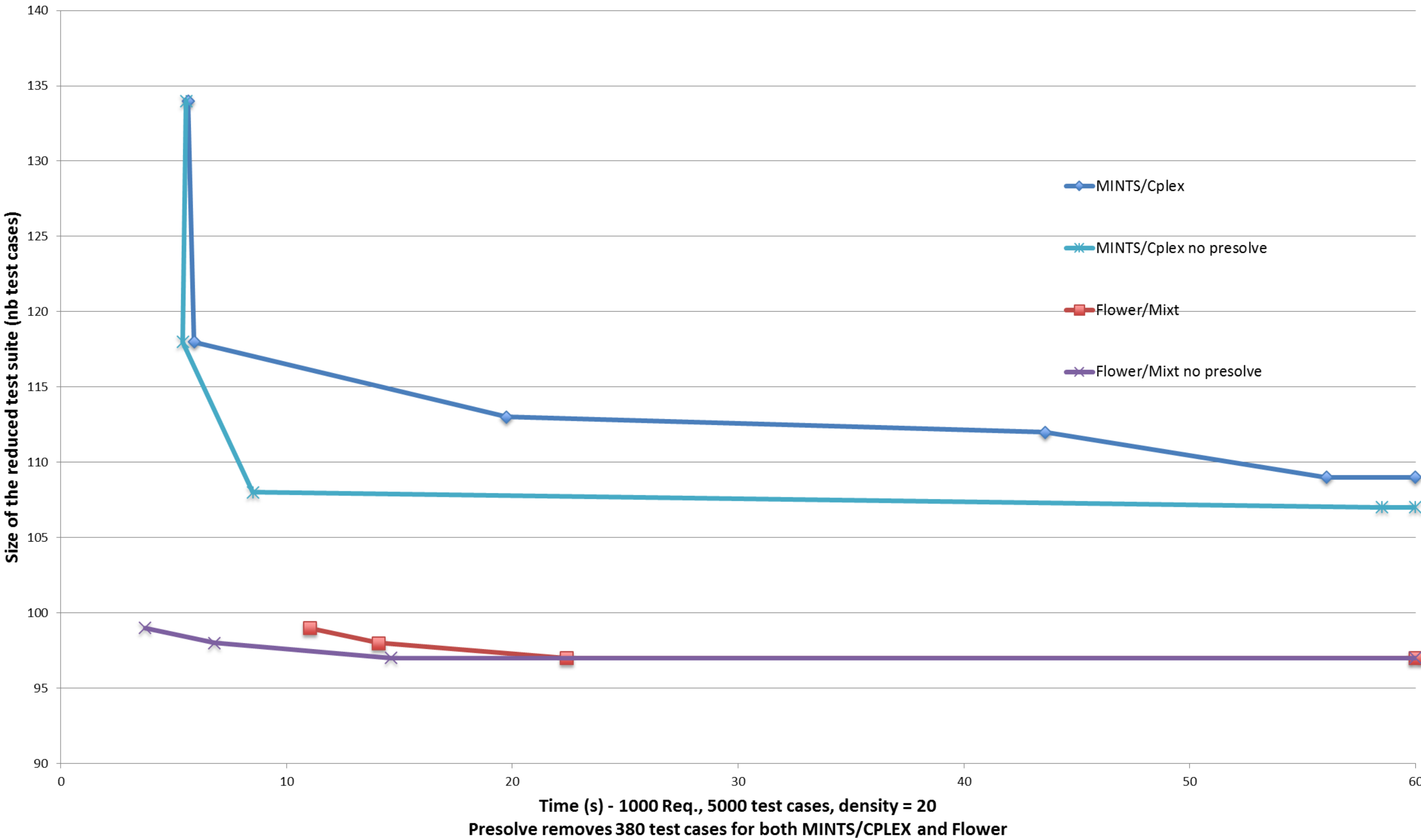
We proposed an iterative algorithm to apply these preprocessing rules to simplify the problem



Presolve: Experimental results (1)



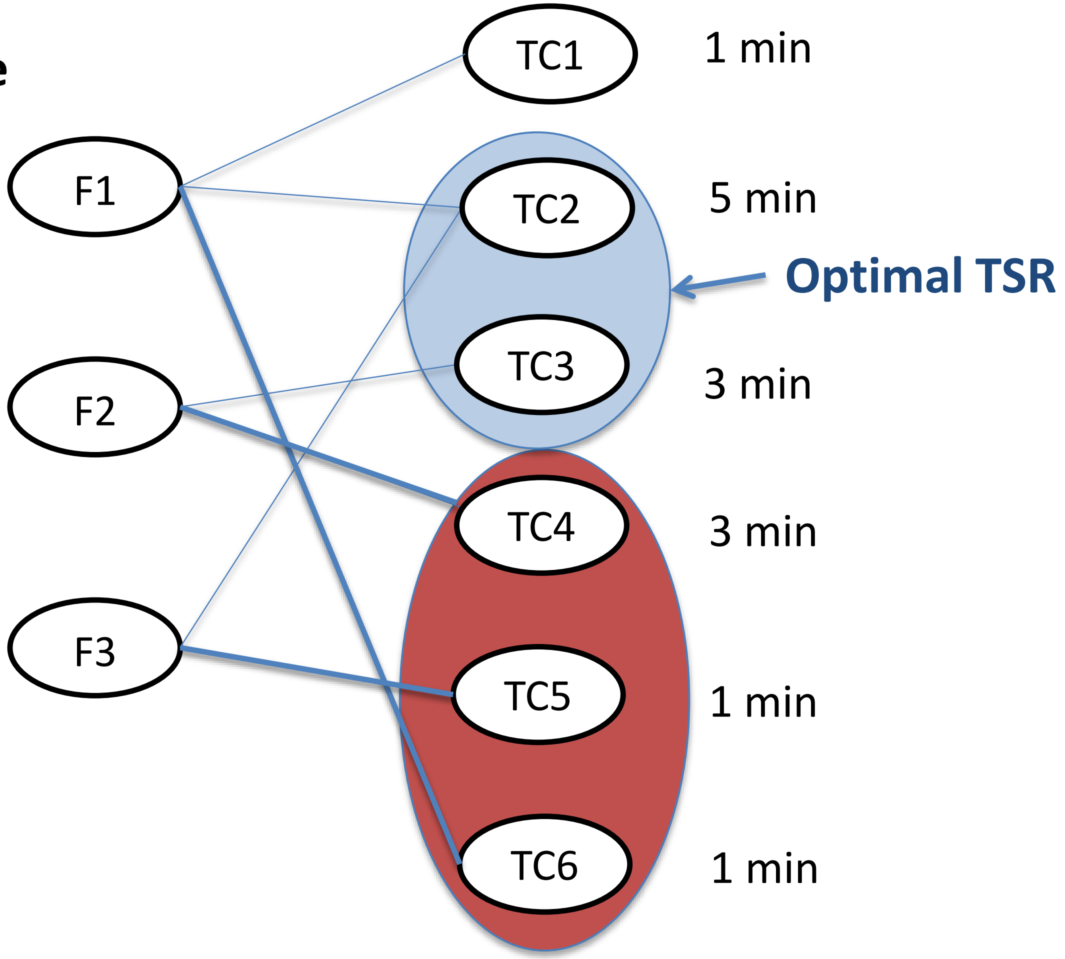
Presolve: Experimental results (2)



Multi-objectives Test Suite Reduction

Optimal TSR: the core problem

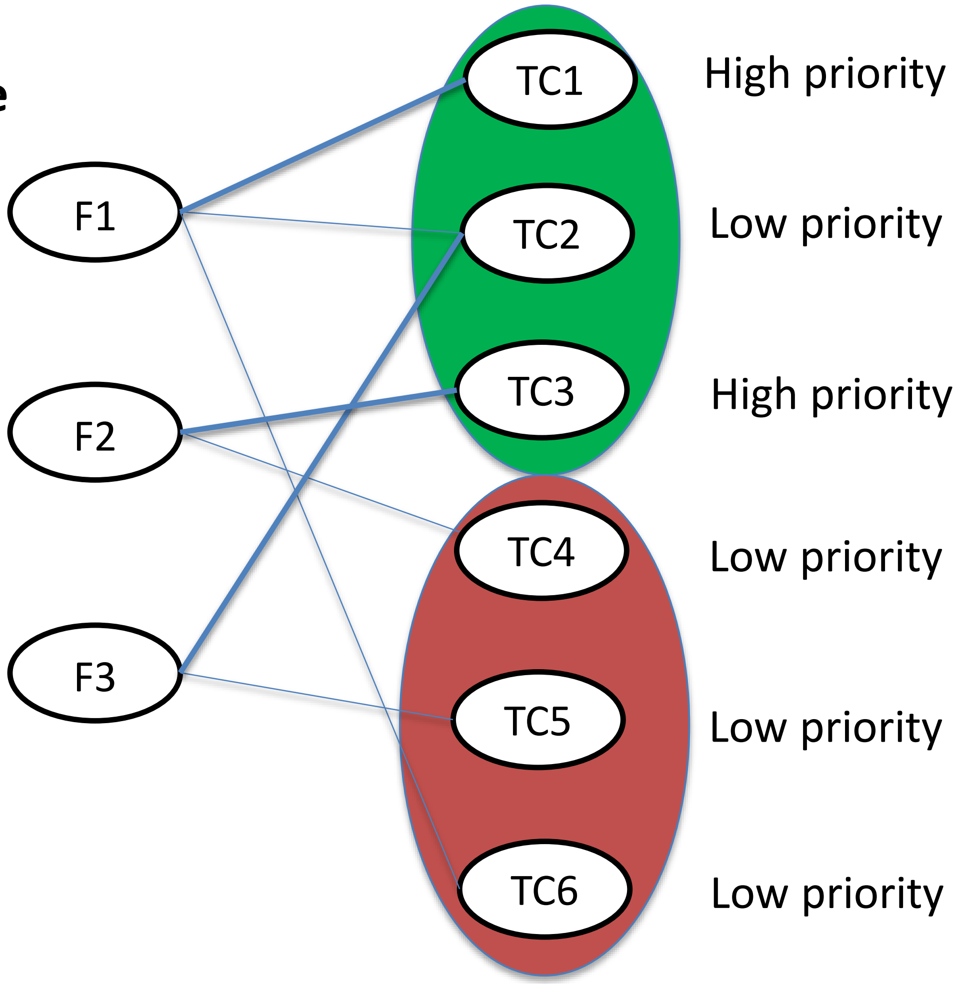
Requirements coverage is always a prerequiste but other criteria than the size of the test suite are also sought:



Execution time!

Optimal TSR: the core problem

Requirements coverage is always a prerequiste 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)
   S. Wang, D. Buchmann, S. Ali, A. Gotlieb, D. Pradhan, and M. Liaaen. Multi-objective test prioritization in software product line testing: An industrial case study. In *Software Product Line Conference (SPLC'14)*, Florence, Italy, 2014.

   Aggregated cost function using RW-algo, URW-algo, and many others
   Based on computed values
   S. Wang, S. Ali, and A. Gotlieb. Random-weighted search-based multi-objective optimization revisited. In *Int. Symp. on Search-Based Software Engineering (SSBSE'14)*, Fortaleza, Brazil, 2014.

   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

$R_1,..,R_n$: Requirements
$t_1,..,t_m$: Test cases - Each test case t_i is associated a unitary cost c_i
$O_1,..,O_m$: Occurrences variables

Minimize TotalCost

s.t

gcc($(R_1, ..., R_n), (t_1, ..., t_m), (O_1, ..., O_m)$)
for i=1 to m do $B_i = (O_i > 0)$
scalar_product($(B_1, ..., B_m), (c_1, ..., c_m)$, TotalCost)

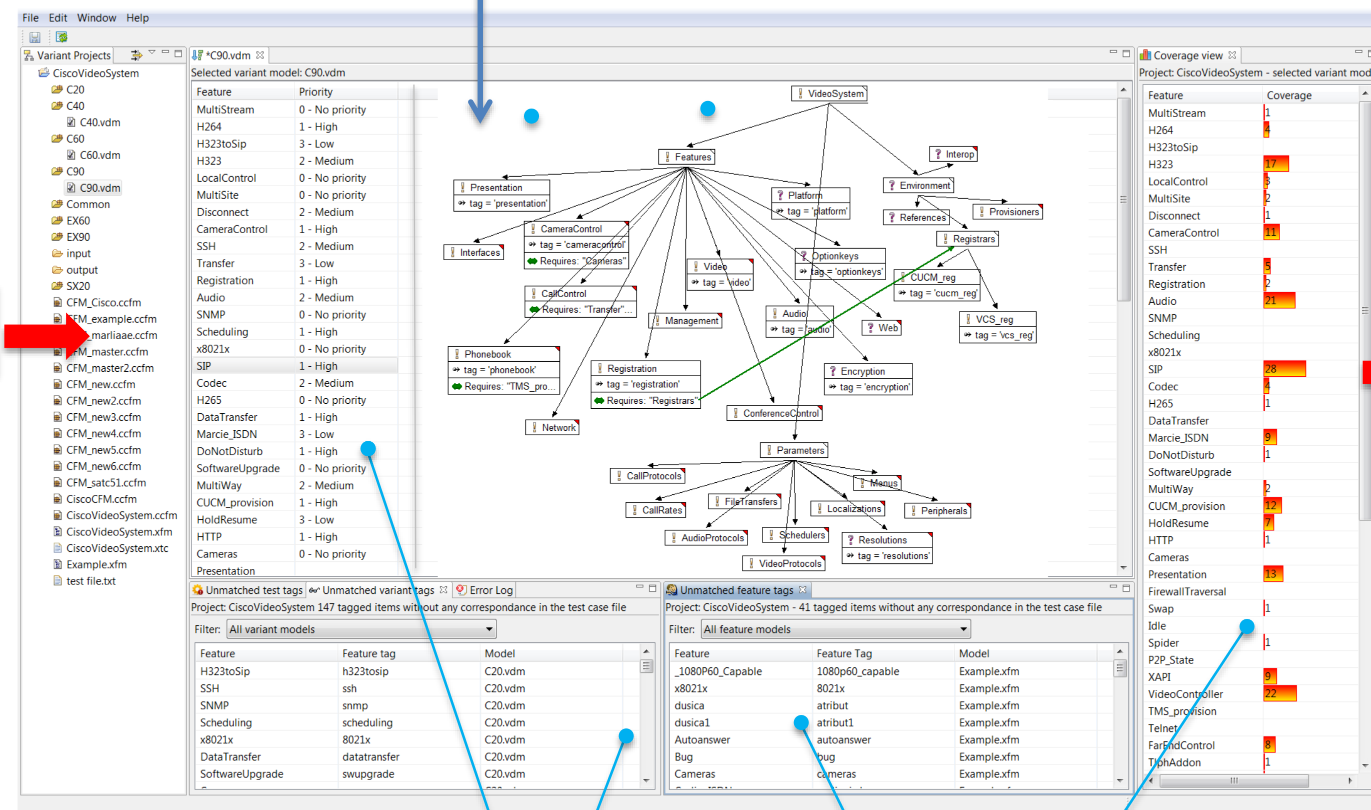
where scalar_product encodes $B_1 * c_1 + .. + B_m * c_m = TotalCost$

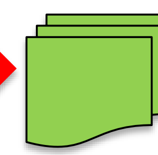
Industrial Application

TITAN

Variability model to describe a software product line

Unoptimized test suite





Optimized (reduced/ prioritized) test suite

Diagnostic views, feature coverage

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!