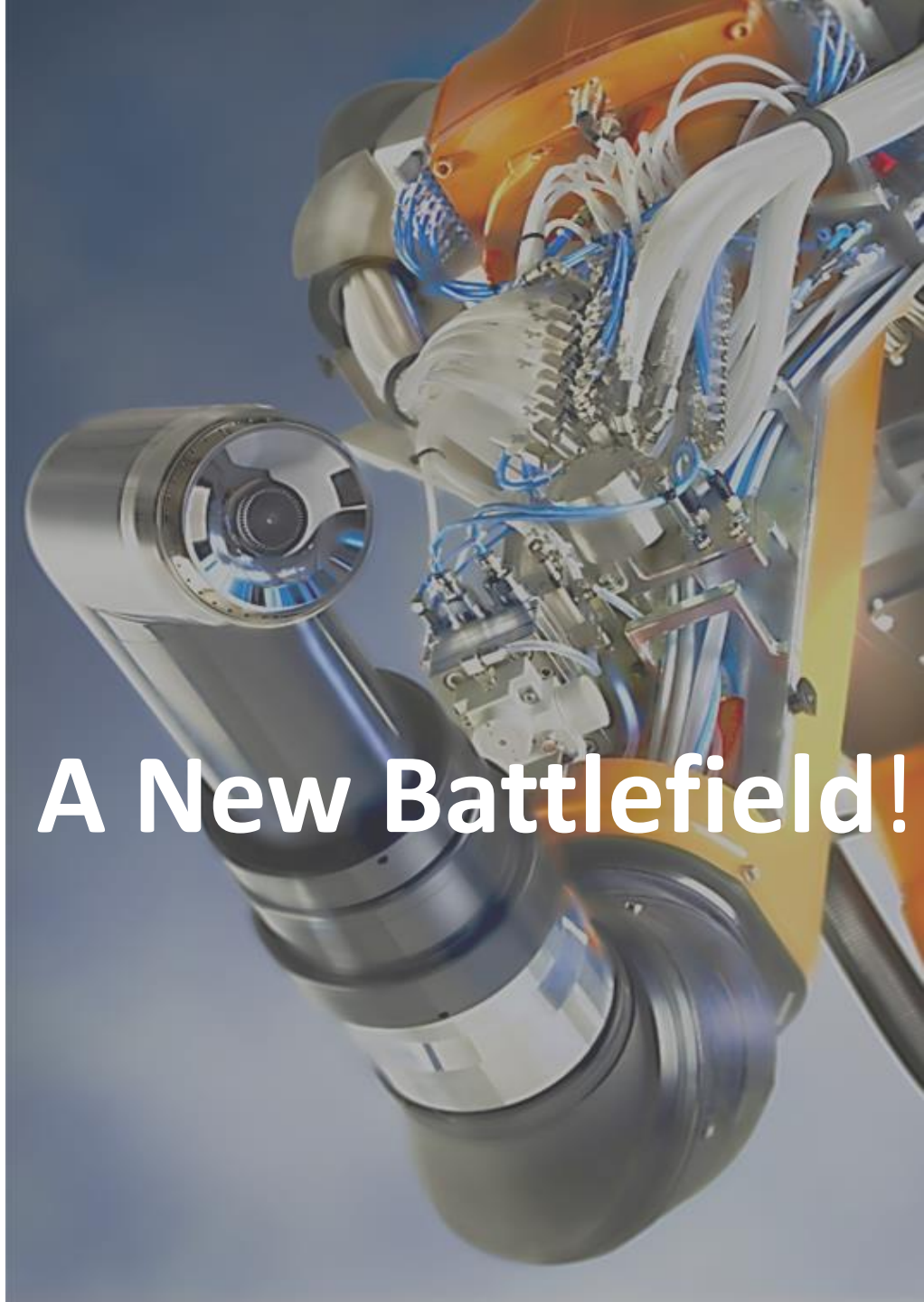
simula

Testing Robotic Systems:



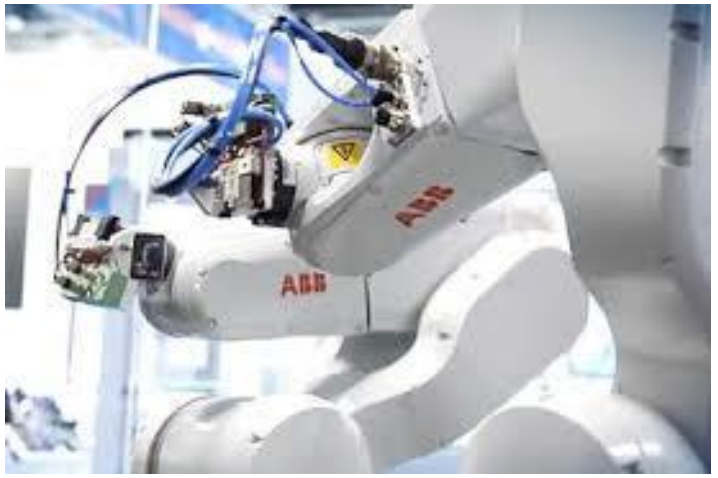
A New Battlefield!

Arnaud Gotlieb
Simula Research Laboratory
Norway

RoboSoft – Wed, 13-14 November 2019 –
Royal Academy of Engineering

Industrial Robotics Evolves Very Fast!

Industrial robots are now complex cyber-physical systems (motion control and perception systems, multi-robots sync., remote control, Inter-connected for predictive maintenance, …)



They are used to perform safety-critical tasks in complete autonomy (high-voltage component, on-demand painting with color/brush change, ..)

And they collaborate with human co-workers

Testing Robotic Systems is Crucial and Challenging

- The validation of industrial robots still involve too much human labour
- "*Hurry-up, the robots are uncaged!*": Failures are not anymore handled using fences
- Robot behaviour evolves with changing requirements
- Today, industrial robots can be taught by-imitation. Tomorrow, they will learn by themselves

More automation in testing

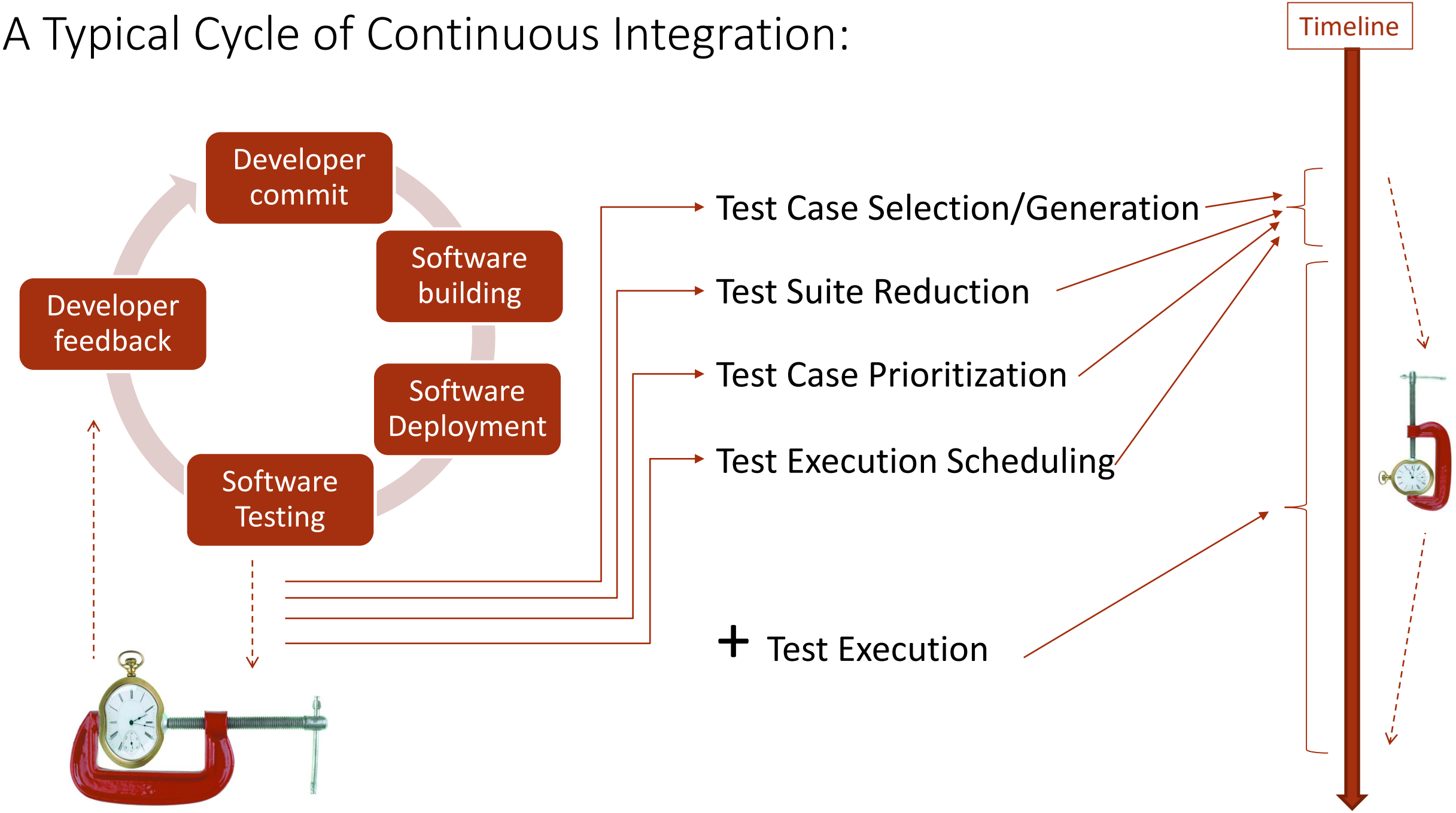
More diversity in testing

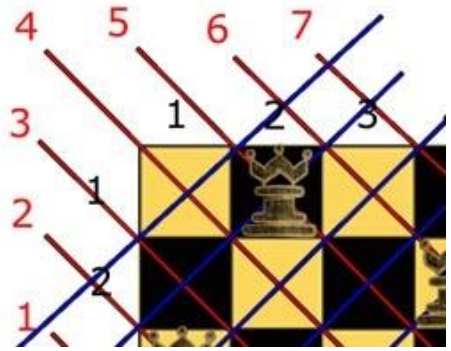
More efficiency in testing

How Software Development of Industrial Robots Has Evolved...

From….	To…
Single-core, single application system	Multi-core, complex distributed system
All source code maintained by a small team located at the same place	Subsystems developed by distinct teams located at distinct places in the world
Manual system testing only handled in a single place, on actual robots	Automated software testing handled in continuous integration, on virtual controllers

A Typical Cycle of Continuous Integration:



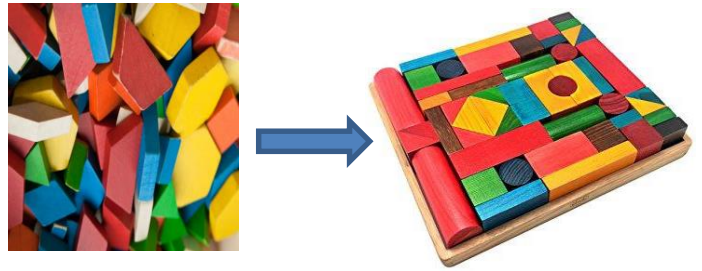


Constraint Programming

1. Test Suite Reduction

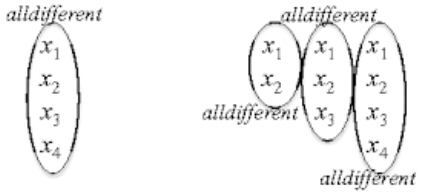
2. Test Execution Scheduling

3. Test of Intelligent Systems

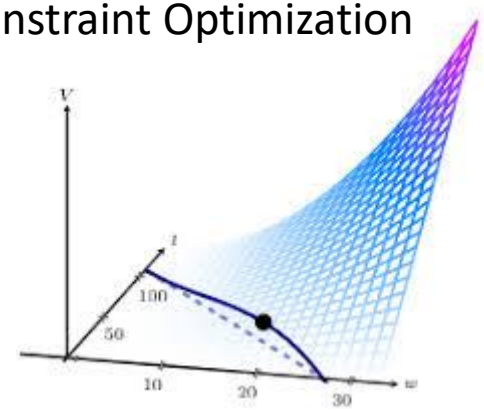


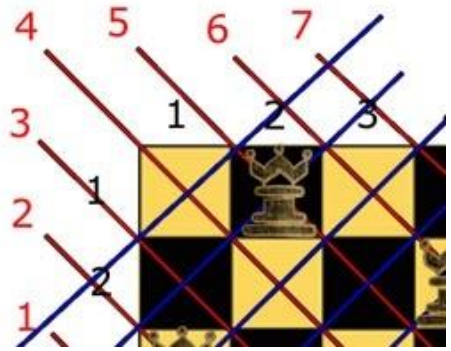
Constraint-based Scheduling

Global Constraints



Constraint Optimization



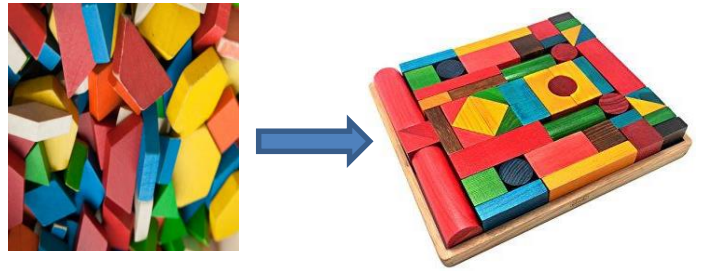


Constraint Programming

1. Test Suite Reduction

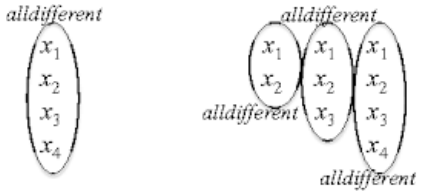
2. Test Execution Scheduling

3. Test of Intelligent Systems

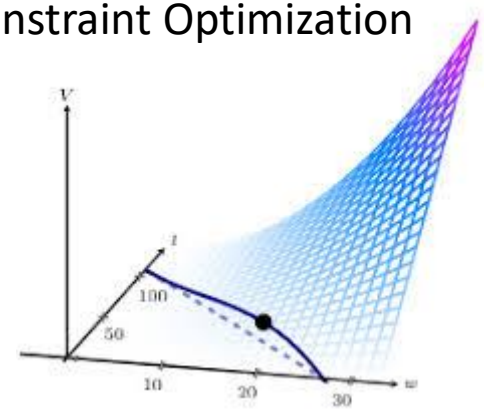


Constraint-based Scheduling

Global Constraints



Constraint Optimization



Test Selection and Test Suite Reduction

PRODUCT	BASIC SPECIFICATIONS	
IRB 14000 YuMi® 	Load (kg)	0.50
	Reach (m)	0.559
	Protection	Std:IP30; Clean room ISO 5
	Mounting	Bench, table
	Safety	PL b Cat B
IRB 14050 Single Arm YuMi 	Load (kg)	0.50
	Reach (m)	0.559
	Protection	Std:IP30; Clean room ISO 5
	Mounting	Any angle - table, wall, ceiling
	Safety	PL d Cat 3, PL b Cat b, SafeMove Pro option
IRB 1100 	Load (kg)	4.00 4.00
	Reach (m)	0.475 0.58
	Armload (kg)	0.50 0.50
	Protection	Std: IP40
	Mounting	Any angle
IRB 120 and IRB 120T 	Load (kg)	3.00
	Reach (m)	0.58
	Protection	Std: IP30 Option: Cleanroom class 5, certified by IPA
	Mounting	Floor, wall, inverted, and tilted angles

PRODUCT	BASIC SPECIFICATIONS	
IRB 1200 	Load (kg)	5.00 7.00
	Reach (m)	0.90 0.70
	Protection	Std: IP40 Option: IP67, Clean room ISO 4, food grade lubricant
	Mounting	Any angle
IRB 140 and IRB 140T 	Load (kg)	6.00
	Reach (m)	0.81
	Protection	Std: IP67 Option: Cleanroom class 6, Foundry Plus
	Mounting	Floor, wall, inverted, and tilted angles
IRB 1600 	Load (kg)	6.00 6.00 10.0 10.0
	Reach (m)	1.20 1.45 1.20 1.45
	Protection	Std: IP54 Option: IP67 with foundry plus 2
	Mounting	Floor, wall, inverted, tilted angles, and shelf
IRB 1660ID 	Load (kg)	4.00 6.00
	Reach (m)	1.55 1.55
	Protection	Std: IP40 (wrist IP67)
	Mounting	Floor, wall, inverted, and tilted angles

From a concrete set up:

Test Case Repository:

~10,000 Test Cases (TC)

~25 distinct Test Robots

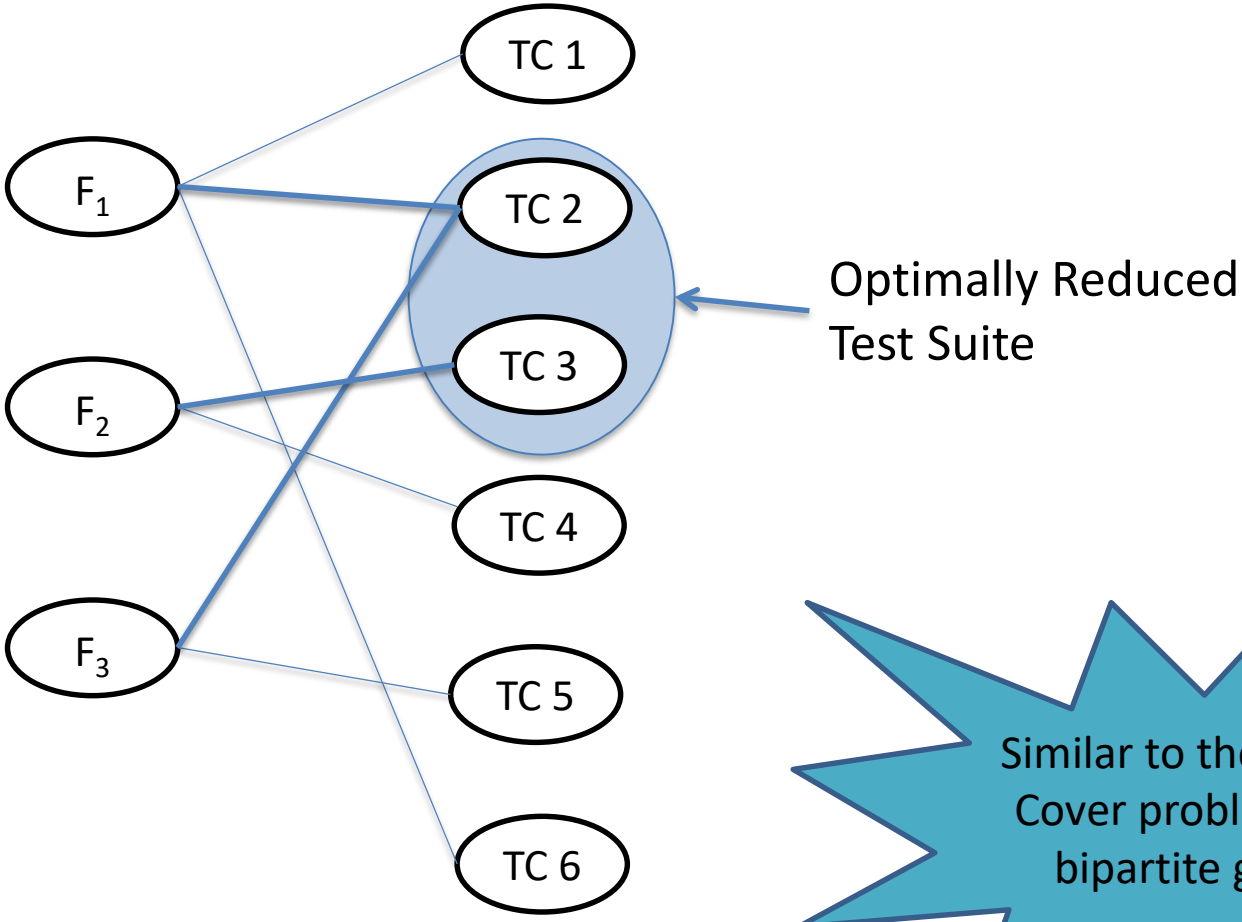
~500 distinct features

10..30 code changes per day

→ Select, schedule and execute about 150 TC per CI cycle

Optimal Test Suite Reduction

F_i: Requirements
TC: Test Cases

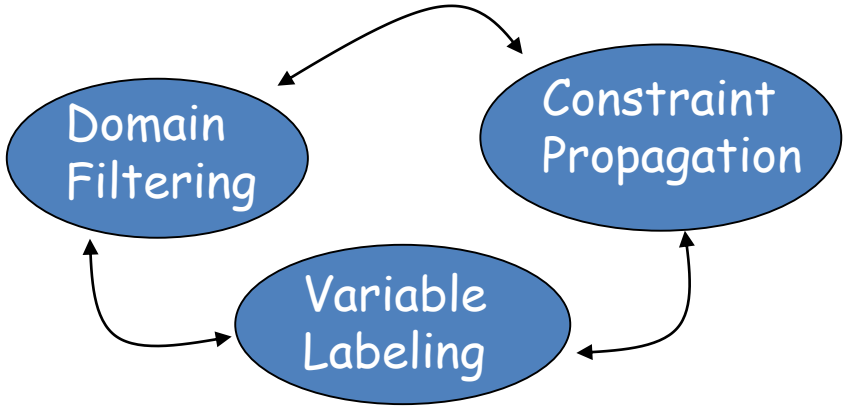


NP-hard problem!

Similar to the Vertex Cover problem in a bipartite graph

Constraint Programming (CP)

- Routinely used in Validation & Verification, **CP** handles efficiently hundreds of thousands of constraints and variables



- CP is versatile: user-defined constraints, dedicated solvers, programming search heuristics **but it is not a silver bullet** (developing efficient CP models and heuristics requires expertise)

→ **Global constraints**: relations over a non-fixed number of variables, implementing dedicated filtering algorithms

The **nvalue** global constraint

[Pachet Roy 1999, Beldiceanu 01]

nvalue(N, V)

Where:

N is a finite-domain variable

$V = [V_1, \ldots, V_k]$ is a vector of variables

nvalue(N, V) holds iff $N = card(\{V_i\}_{i\ in\ 1\ldots k})$

nvalue(N, [3, 1, 3]) entails N = 2

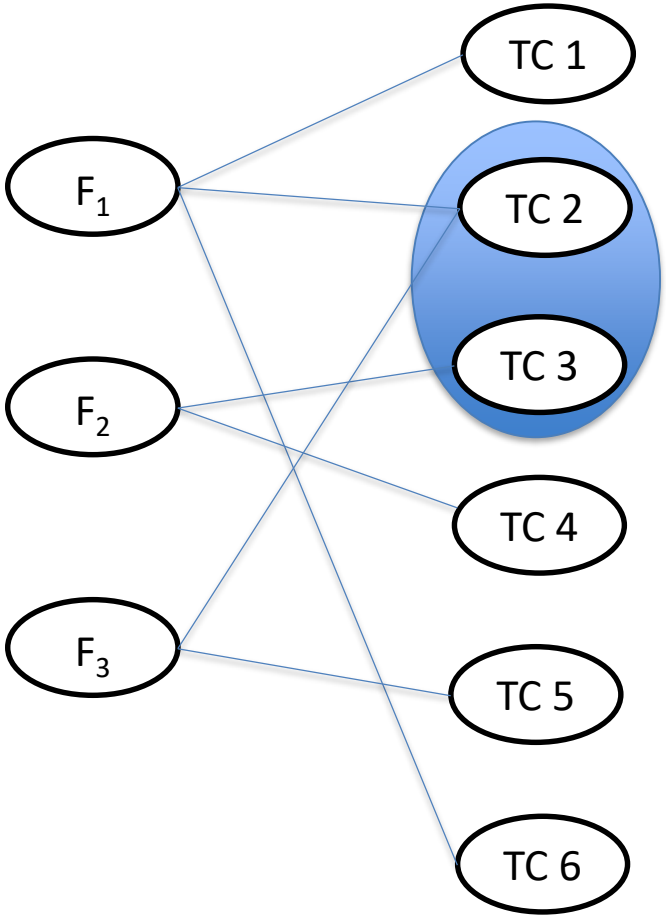
nvalue(3, [X_1, X_2]) fails

nvalue(1, [X_1, X_2, X_3]) entails $X_1 = X_2 = X_3$

N in 1..2**, nvalue**(N, [4, 7, X_3]) entails X_3 in {4,7}, N=2

Optimal Test Suite Reduction with **nvalue**

However,
only F_1, F_2, F_3
are available
for labeling!



Sol: $F_1 = 2$, $F_2 = 3$, $F_3 = 2$
Optimally Reduced Test Suite

F_1 in {1, 2, 6}, F_2 in {3, 4}, F_3 in {2, 5}
nvalue(*MaxNvalue*, *[F_1, F_2, F_3]*)
Minimize(*MaxNvalue*)

The global_cardinality constraint (**gcc**)

[Regin AAAI'96]

gcc(T, d, V)

Where

T = $[T_1, \ldots, T_N]$ is a vector of N variables

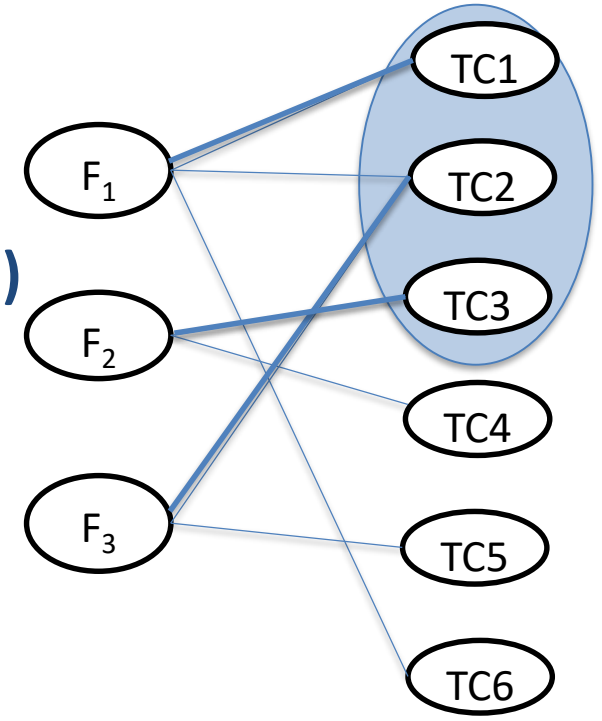
d = $[d_1, \ldots, d_k]$ is a vector of k values

V = $[V_1, \ldots, V_k]$ is a vector of k variables

gcc(T, d, V) holds iff $\forall i \; in \; 1..k,$ V_i= card({j | T_j=di})

Filtering algorithms for **gcc** are based on max-flow computations

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:

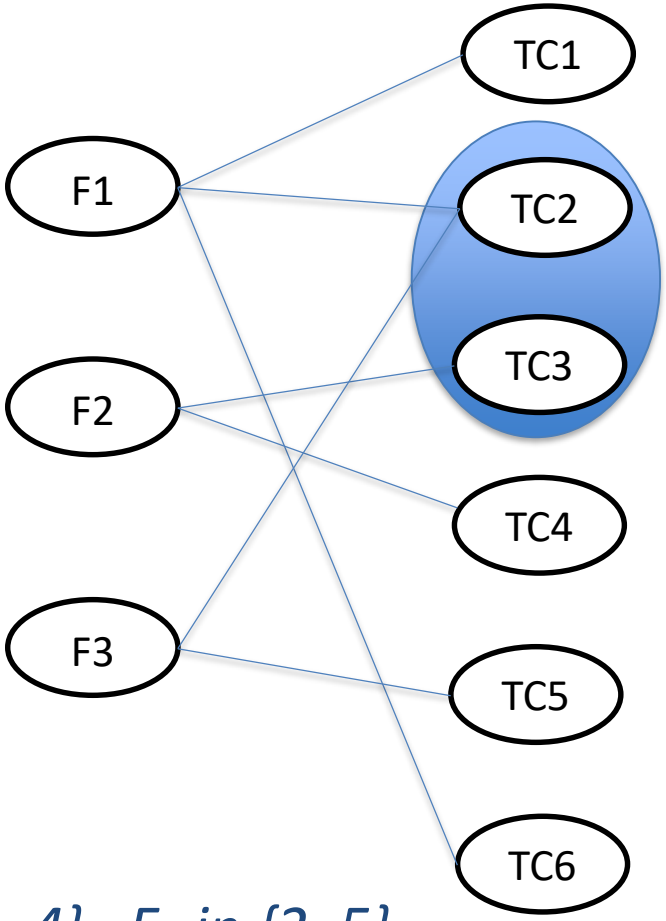
TC1 covers exactly V_1 features in [F_1, F_2, F_3]
TC2 " V_2 "
TC3 " V_3 "
...

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, 1, 2}, V_3 in {0, 1}, V_4 in {0, 1}, V_5 in {0, 1}, V_6 in {0, 1}

Here, **V_1=1, V_2=1, V_3=1, V_4=0, V_5=0, V_6=0** is a feasible solution

But, not an optimal solution!

Mixt model using **gcc** and **nvalue**



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])

Minimize(MaxNvalue)

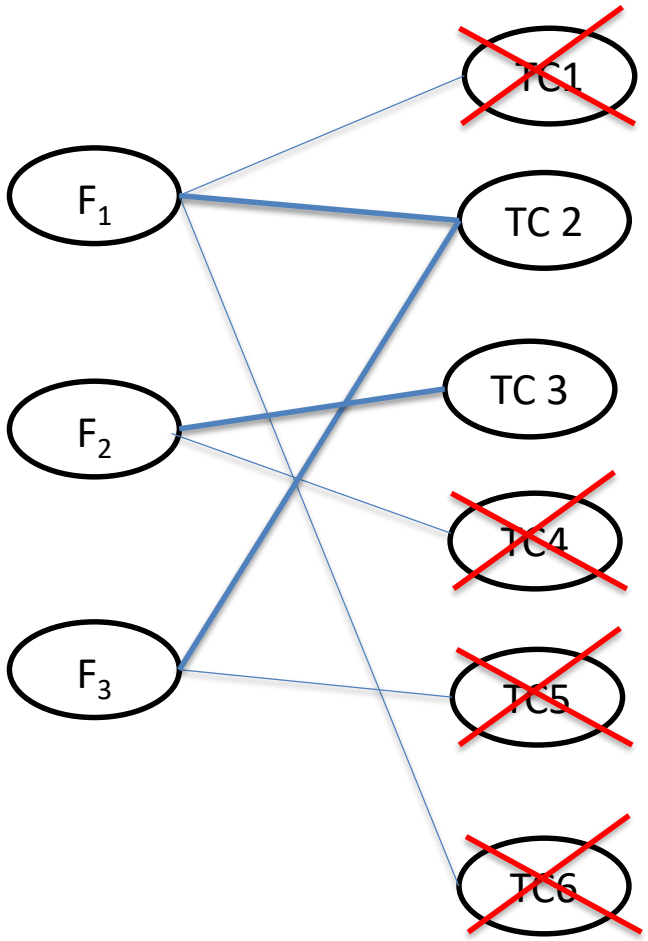
Model pre-processing

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

F_3 is covered → withdraw TC_5

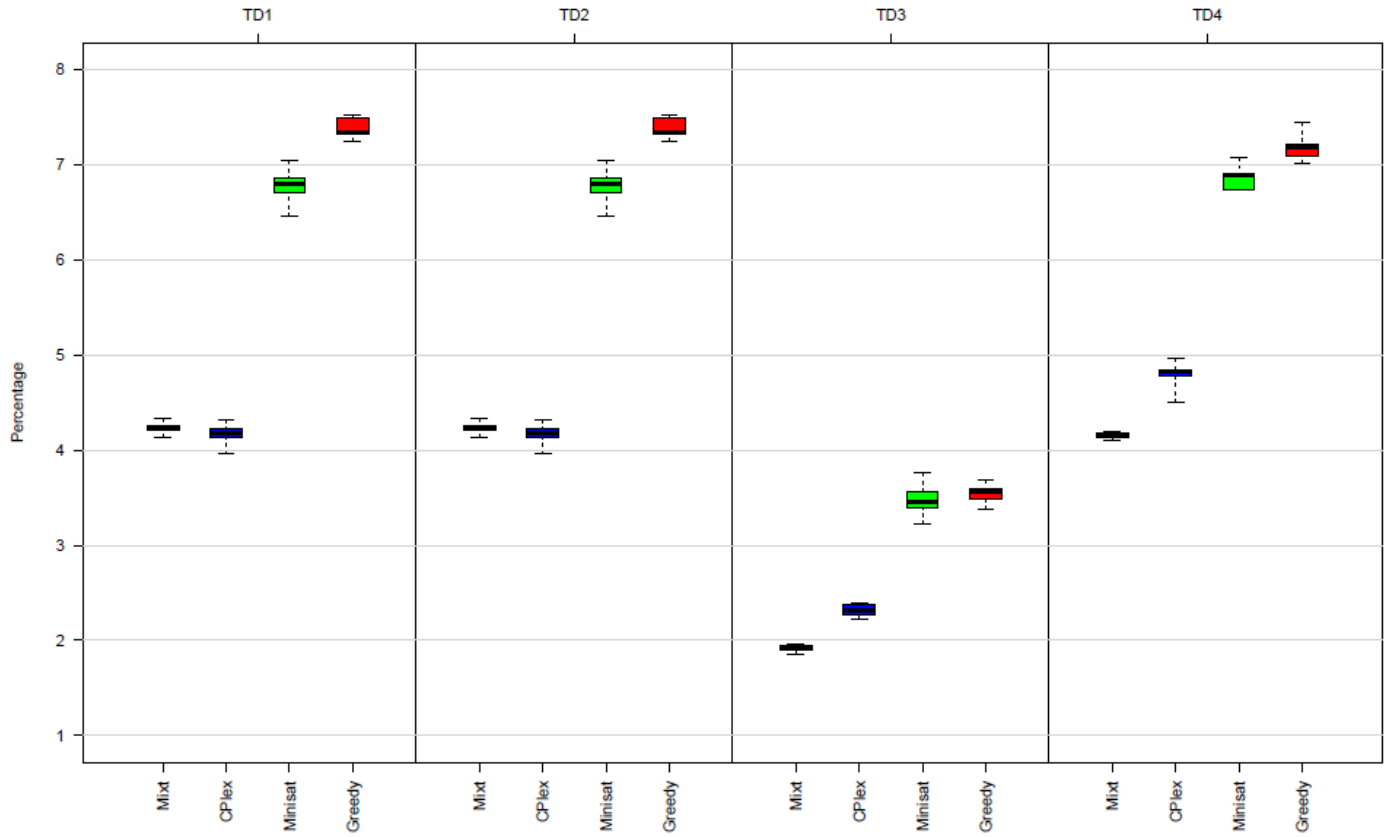
F_2 in {3,4} → e.g., F_2 = 3, withdraw TC_4

Pre-processing rules can be expressed once and then applied iteratively



Comparison with CPLEX, MiniSAT, Greedy (uniform costs)

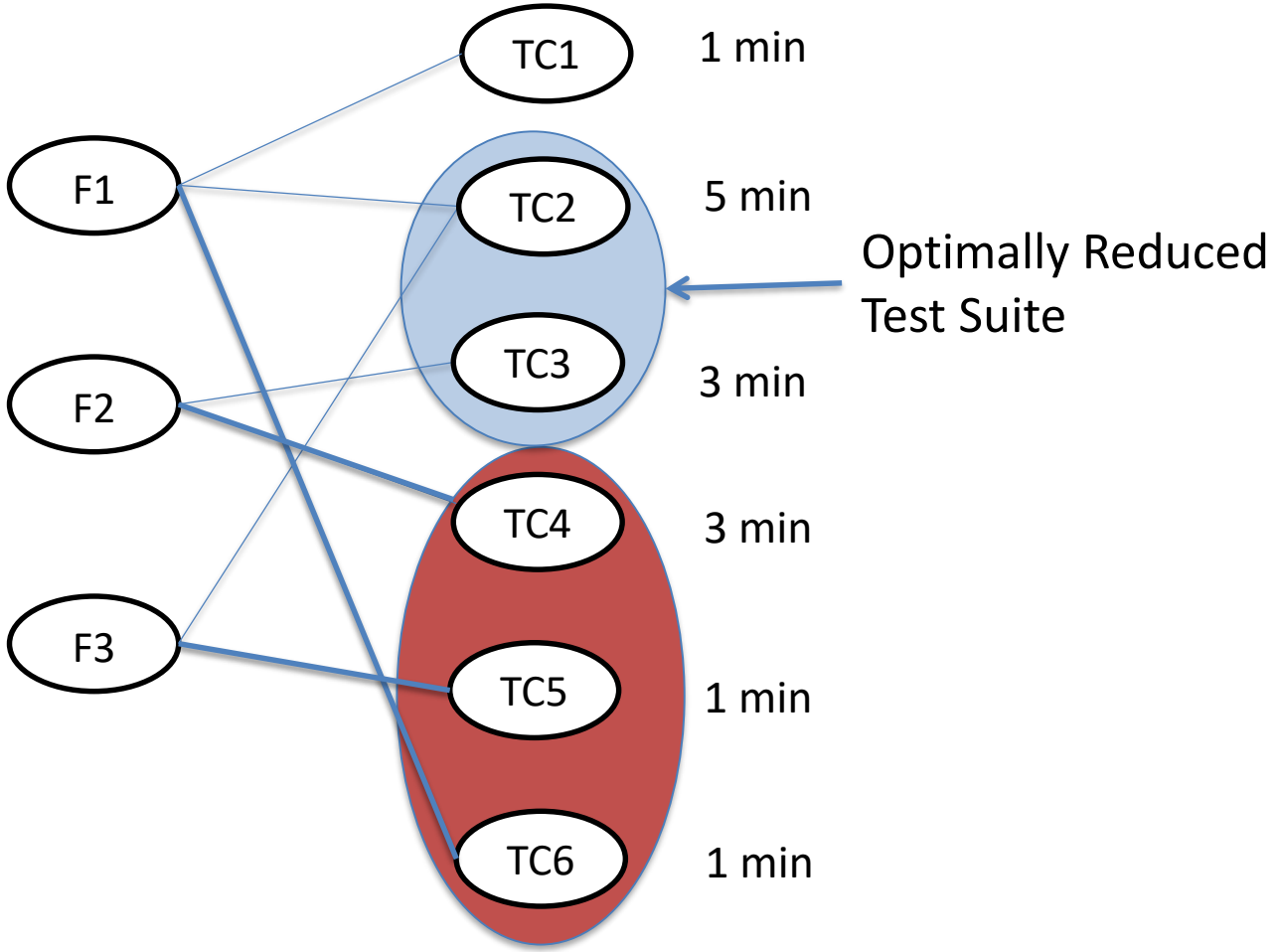
(Reduced Test Suite percentage in 60 sec)



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

Other criteria to minimize

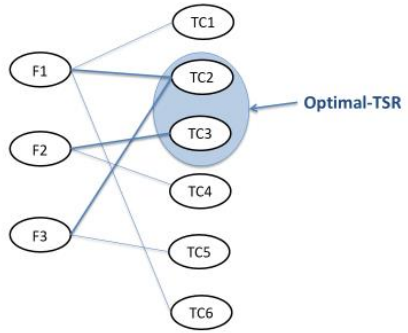
Requirement coverage
is always a prerequiste



Execution time!

Optimal Test Suite Reduction with Constraint Programming (CP)

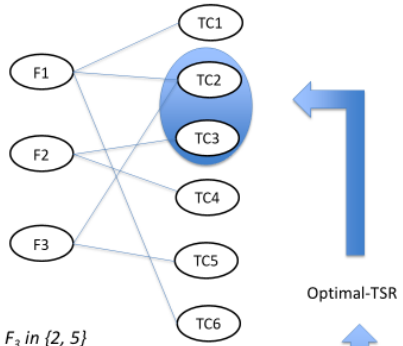
Optimal TSR: the core problem



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*

NP-hard problem!

Our best CP model: Mixt



Optimal-TSR

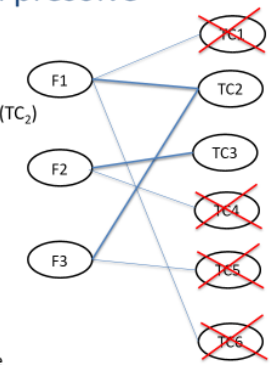
F_1 *in {1, 2, 6}, F_2 in {3, 4}, F_3 in {2, 5}*
gcc((F_1, F_2, F_3), (1,2,3,4,5,6), (V_1, V_2, V_3, V_4, V_5, V_6))
nvalue(MaxNvalue, (F_1, F_2, F_3)
label(minimize(MaxNvalue))

Introducing model presolve

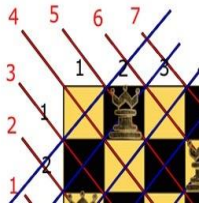
F_1 in {1, 2, 6} → F_1 = 2 as cov(TC_1) = cov(TC_6) ⊂ cov(TC_2)
withdraw TC_1 and TC_6

F_3 is covered → withdraw TC_5

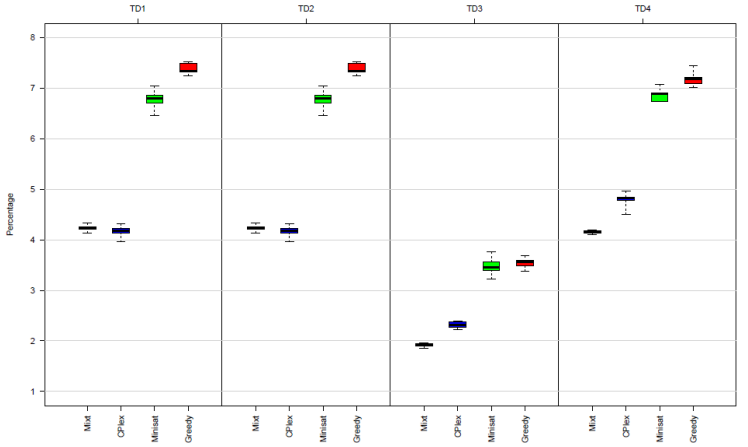
F_2 in {3,4} → e.g., F_2 = 3, withdraw TC_4



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



CP with **global constraints** (nvalue, gcc) and search heuristic and **presolve**
Time-contract solving of the **multi-criteria optimisation** problem



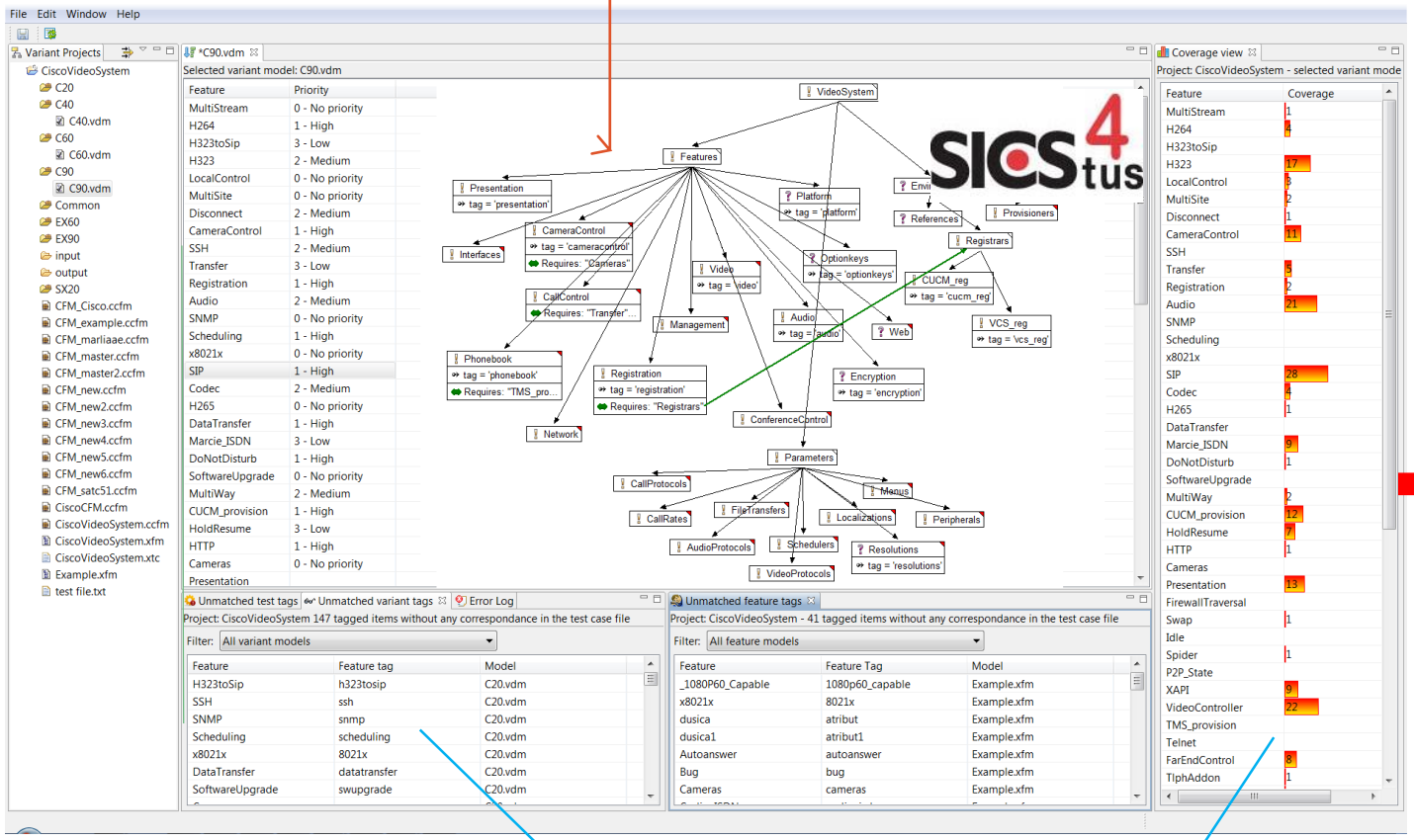
	TD1	TD2	TD3	TD4
Requirements	1000	1000	1000	2000
Test cases	5000	5000	5000	5000
Density	7	7	20	20

*A. Gotlieb and D. Marijan - **FLOWER: Optimal Test Suite Reduction As a Network Maximum Flow** – ACM Int. Symp. on Soft. Testing and Analysis (ISSTA'14), San José, CA, Jul. 2014.*

*M. Mossige, A. Gotlieb and H. Meling - **Generating Tests for Robotized Painting Using Constraint Programming** - In Int. Joint Conf. on Artificial Intelligence (IJCAI-16) - Sister Conference Best Paper Track. New York City, 2016.*

A. Gotlieb and D. Marijan - **Using Global Constraints to Automate Regression Testing -** *AI Magazine* 38, no. Spring (2017).

Variability model to describe a product line



Unoptimized test suite

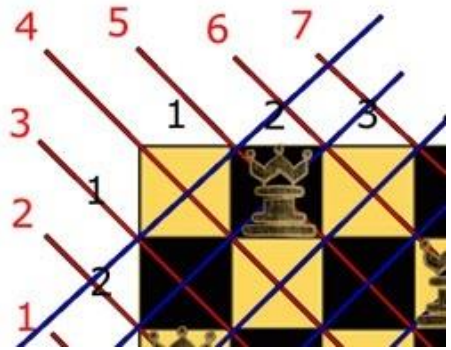
Optimized (reduced) test suite

Diagnostic views, feature coverage

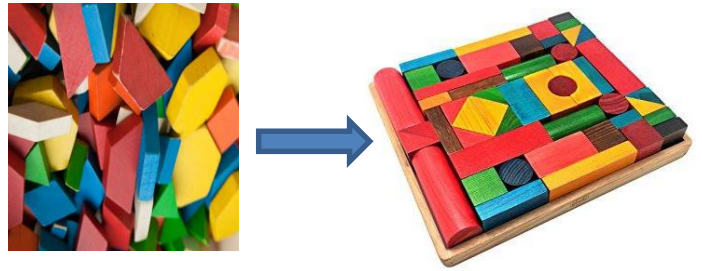
*S. Wang, S. Ali and A. Gotlieb - **Cost-Effective Test Suite Minimization in Product Lines Using Search Techniques** -Journal of Systems and Software 103 (2015): 370-391.*

*A. Gotlieb, M. Carlsson, D. Marijan and A. Petillon - **A New Approach to Feature-based Test Suite Reduction in Software Product Line Testing** - In ICSOFT-EA 2016, 11th Int. Conf. on Sof. Eng. and Applications, Lisbon, July 2016, **Best Paper Award**. INSTICC Press, 2016.*

*D. Marijan, A. Gotlieb, M. Liaaen, S. Sen and C. Ieva **- TITAN: Test Suite Optimization for Highly Configurable Software** - In International Conference on Software Testing, Verification and Validation (ICST 2017) . IEEE, 2017.*



Constraint Programming



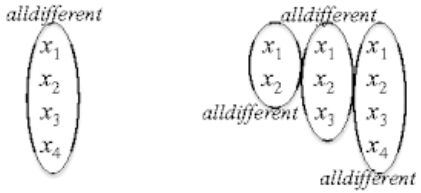
Constraint-based Scheduling

1. Test Suite Reduction

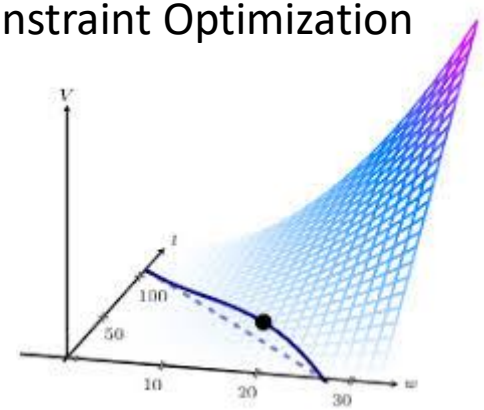
2. Test Execution Scheduling

3. Test of Intelligent Systems

Global Constraints



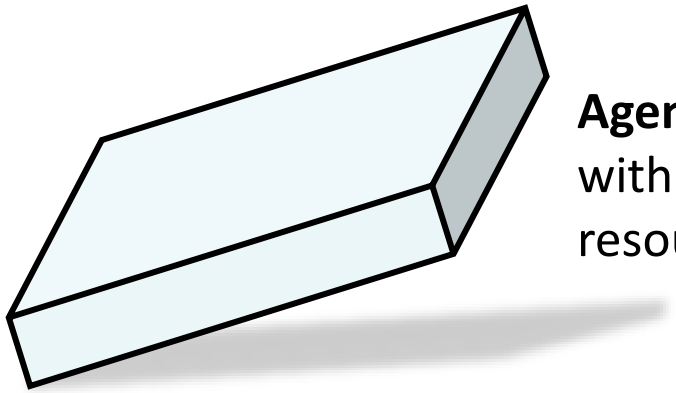
Constraint Optimization



Constraint-Based Scheduling

Tasks
with distinct characteristics

Schedule



Agents
with limited time or resources capacity

Assignment of **Tasks to Agents** such that:

1. Task execution is not interrupted or paused;
2. Agents are well occupied;
3. Tasks sharing a global resource are not executed at the same time;
4. Diversity of assignment of tasks to agents is ensured;

Goal:
Schedule as much tasks as possible on available agents such that the overall execution time is minimized

Test Case Execution Scheduling

(T, M, G, d, g, f)

***T**: a set of Test Cases*
***M**: a set of Machines, e.g., robots*
***G**: a set of (non-shareable) resources*

d: T → N estimated duration
g: T → 2^G usage of global resources
f: T → 2^M possible machines

Disjunctive scheduling,
non-preemptive,
non-shareable resources,
machine-independant
execution time

Function to optimize:
TimeSpan: the overall duration of test execution T_E
(in order to minimize the round-trip time)

In practice, global optimality is desired but not mandatory, it's more important to control T_S w.r.t T_E
→ Time-contract global optimization

A simple example

Test	Duration (d)	Executable on (f)	Use of global resource (g)
$t1$	2	$m1, m2, m3$	-
$t2$	4	$m1, m2, m3$	$r1$
$t3$	3	$m1, m2, m3$	$r1$
$t4$	4	$m1, m2, m3$	$r1$
$t5$	3	$m1, m2, m3$	-
$t6$	2	$m1, m2, m3$	-
$t7$	1	$m1$	-
$t8$	2	$m2$	-
$t9$	3	$m3$	-
$t10$	5	$m1, m3$	-

Test Cases: *t1, t2, t3, t4, t5, t6, t7, t8, t9, t9, t10*

r1

m3

m2

m1

The **CUMULATIVE** global constraint [Aggoun & Beldiceanu AAAI'93]

CUMULATIVE(t, d, r, m)

Where

$t = (t_1, \ldots, t_N)$ is a vector of tasks, each t_i in S_i .. E_i

$d = (d_1, \ldots, d_N)$ is a vector of task duration

$r = (r_1, \ldots, r_N)$ is a vector of resource consumption rates

m is a scalar

CUMULATIVE ***(t, d, r, m)*** holds iff

$$\sum_{i=1}^{N} r_i \leq m$$

$$t_i \leq t \leq t_i + d_i$$

Using the global constraint **CUMULATIVE**

CUMULATIVE$((t_1,..,t_{10}), (d_1,..,d_{10}), (1, ..,1), 3)$,
$M_1,..,M_6$ *in* $1..3$,
$M_7 = 1, M_8 = 2, M_9 = 3, M_{10}$ *in* $\{1,3\}$,
$(E_2 \leq S_3$ *or* $E_3 \leq S_2)$, $(E_2 \leq S_4$ *or* $E_4 \leq S_2)$,
$(E_3 \leq S_4$ *or* $E_4 \leq S_3)$,
MAX$(MaxSpan, (E_1, \ldots, E_{10}))$,
LABEL(**MINIMIZE**$(MaxSpan), (S_1,..,S_{10}), (M_1,..,M_{10}))$

Test	Duration	Executable on	Use of global resource
$t1$	2	$m1, m2, m3$	-
$t2$	4	$m1, m2, m3$	$r1$
$t3$	3	$m1, m2, m3$	$r1$
$t4$	4	$m1, m2, m3$	$r1$
$t5$	3	$m1, m2, m3$	-
$t6$	2	$m1, m2, m3$	-
$t7$	1	$m1$	-
$t8$	2	$m2$	-
$t9$	3	$m3$	-
$t10$	5	$m1, m3$	-

An optimal solution:
$S_1 = 0, S_2 = 4, S_3 = 8, S_4 = 0, S_5 = 4, S_6 = 7, S_7 = 2, S_8 = 9,$
$S_{10} = 3,$
$M_1 = 1, M_2 = 1, M_3 = 1, M_4 = 2, M_5 = 2, M_6 = 2, M_7 = 1,$
$M_8 = 2, M_9 = 3, M_{10} = 3$
$MaxSpan = 11$

Limitations of this model

- Static model – In practice, robots and test cases are not necessarily available at each CI cycle → Need a more dynamic model!
- Historical data about test case success/failure is not taken into consideration!
- Diversity in scheduling among CI cycles is not handled

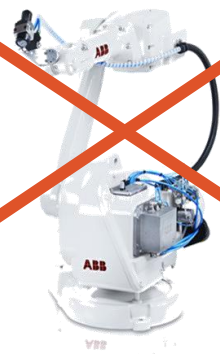
A New Approach Based on Priority and Affinity

A. Test results from n previous runs (Pass/Fail)
B. Developer priority
C. Test duration
D. Time since last execution

- Modeled as a Multi-Cycles Assignment Problem
- Computing priorities based on A, B, C (Priority)
- Combined with D (Affinity) with several heuristics
- Incremental solving from CI cycle to CI cycle

T2, T5, T34

T3, T6, T45, T78

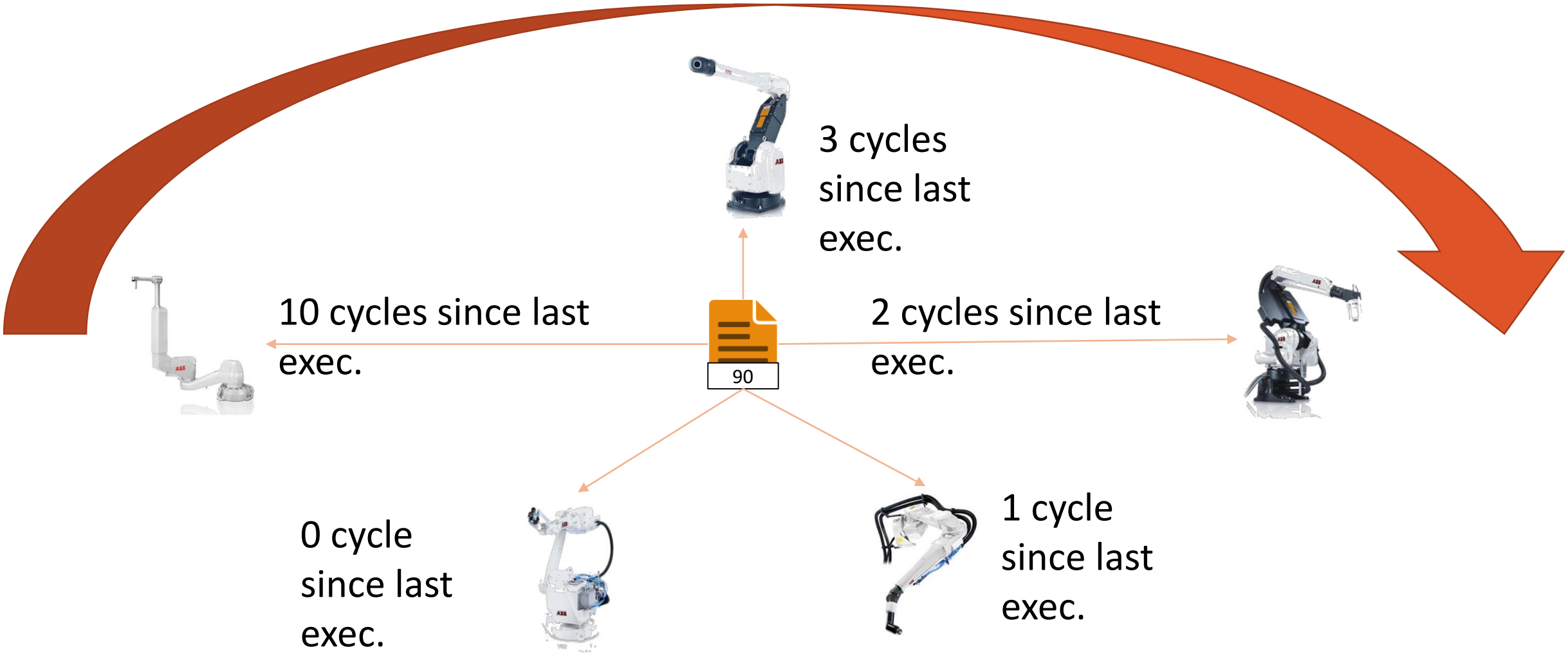


T7, T23

T56, T67

T45, T55

Affinity: more diversity in the test execution process



Rotational Diversity

Priority only (FOP) $v_{ij} \triangleq p_{ij}$

Affinity only (FOA) $v_{ij} \triangleq a_{ij}$

Definition 1. Multi-Cycle General Assignment Problem

$$\text{Maximize} \sum_{i \in \mathcal{A}^k} \sum_{j \in \mathcal{T}^k} x_{ij} v_{ij} \tag{1}$$

$$\text{subject to} \sum_{j \in \mathcal{T}^k} x_{ij} w_{ij} \leq b_i, \qquad \forall\, i \in \mathcal{A}^k \tag{2}$$

$$\sum_{i \in \mathcal{A}^k} x_{ij} \leq 1, \qquad \forall\, j \in \mathcal{T}^k \tag{3}$$

with

k : Index of the current cycle

$\mathcal{A}^k$: A set of integers i labeling m agents

$\mathcal{T}^k$: A set of integers j labeling n tasks

b_i : Capacity of agent i

v_{ij} : Value of task j when assigned to agent i (4)

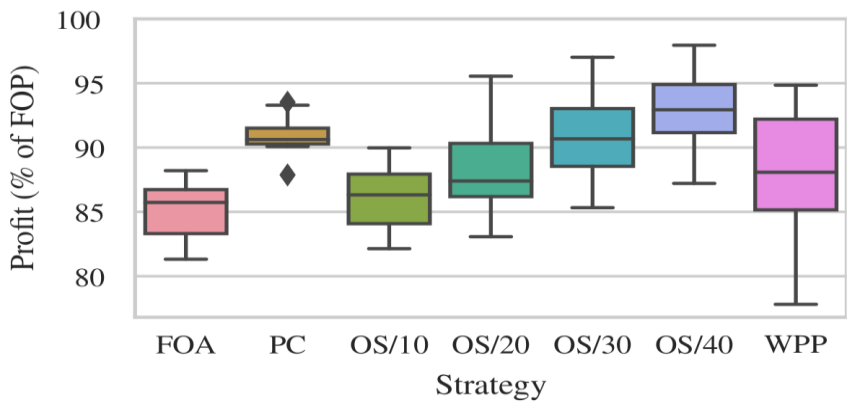
w_{ij} : Weight of task j on agent i

$$x_{ij} : \begin{cases} 1 & \text{Task } j \text{ is assigned to agent } i \wedge i \in \mathcal{C}_j^k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Objective Switch (OS) $v_{ij} \triangleq \begin{cases} p_{ij} & \text{if } \gamma > \max_{j \in \mathcal{T}^k} \mathrm{AP}_j^k \\ a_{ij} & \text{otherwise} \end{cases}$

Product Combination (PC) $v_{ij} \triangleq p_{ij}^{\alpha} \cdot a_{ij}^{\beta}$

Weighted Partial Profits (WPP) $v_{ij} \triangleq \lambda_j^k \cdot \dfrac{p_{ij}}{\max\limits_{i \in \mathcal{A}^k} \max\limits_{j \in \mathcal{T}^k} p_{ij}} + (1 - \lambda_j^k) \cdot \dfrac{a_{ij}}{\max\limits_{i \in \mathcal{A}^k} \max\limits_{j \in \mathcal{T}^k} a_{ij}}$



Agents	20	20	20	30	
Tasks	750	1500	3000	3000	Total
FOA	15 (24.4)	6 (15.7)	3 (9.5)	3 (8.5)	27 (14.5)
OS/10	14 (22.2)	6 (15.5)	**3 (9.4)**	**3 (8.4)**	26 (13.9)
OS/20	9 (18.6)	6 (15.3)	3 (9.2)	3 (8.3)	21 (12.9)
OS/30	7 (16.9)	5 (14.3)	3 (9.1)	3 (8.1)	18 (12.1)
OS/40	7 (16.2)	4 (13.1)	3 (8.9)	3 (7.9)	17 (11.5)
PC	**15 (24.0)**	**7 (14.4)**	3 (8.3)	3 (7.5)	**28 (13.6)**
WPP	14 (24.1)	7 (14.2)	3 (7.3)	3 (7.0)	27 (13.2)
FOP	3 (15.7)	0 (10.8)	0 (7.1)	0 (4.6)	3 (9.6)

(b) Diversity: Full rotations of all tasks (Avg. rotations per task)

SWMOD: Deployment of Test Case Execution Scheduling at ABB Robotics

- ~1500 lines of SICStus Prolog Code with CP(FD)
- Fully integrated into the MS-TFS Continuous Integration
- Using the global constraint binpacking + Rotational Diversity
- Deployed at ABB since Feb. 2019

Microsoft Visual Studio Team Foundation Server

python™

SICStus 4

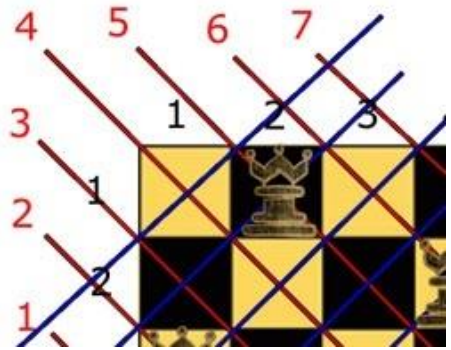
ABB

"SWMOD deployed at ABB Robotics and used every day to schedule tests throughout several ABB centers in the world (Norway, Sweden, India, China)"

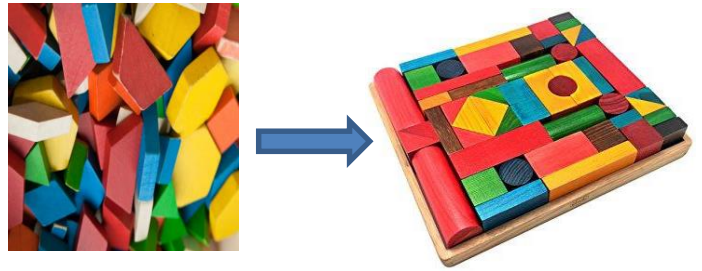
*M. Mossige, A. Gotlieb and H. Meling - **Generating Tests for Robotized Painting Using Constraint Programming** In Int. Joint Conf. on Artificial Intelligence (IJCAI-16) – Sister Conference **Best Paper Track**. New York City, 2016.*

*H. Spieker, A. Gotlieb, D. Marijan and M. Mossige - **Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration** - In Proc. of the 26th ACM Int. Symp. on Software Testing and Analysis (ISSTA'17). New York, NY, USA: ACM, 2017.*

*H. Spieker, A. Gotlieb and M. Mossige - **Rotational Diversity in Multi-Cycle Assignment Problems** - In Proc. of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19). Feb. 2019.*



Constraint Programming



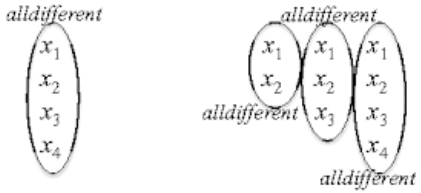
Constraint-based Scheduling

1. Test Suite Reduction

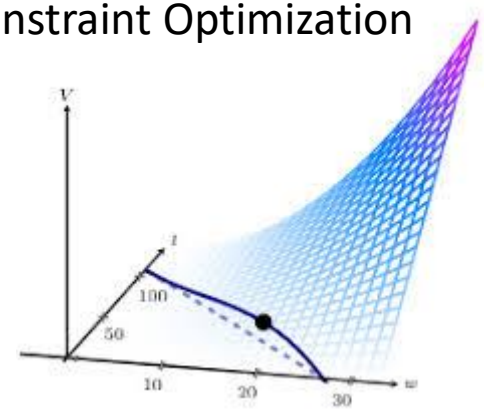
2. Test Execution Scheduling

3. Test of Intelligent Systems

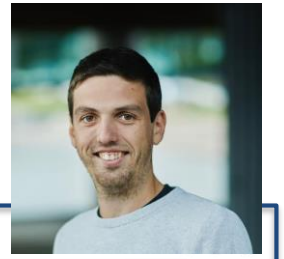
Global Constraints



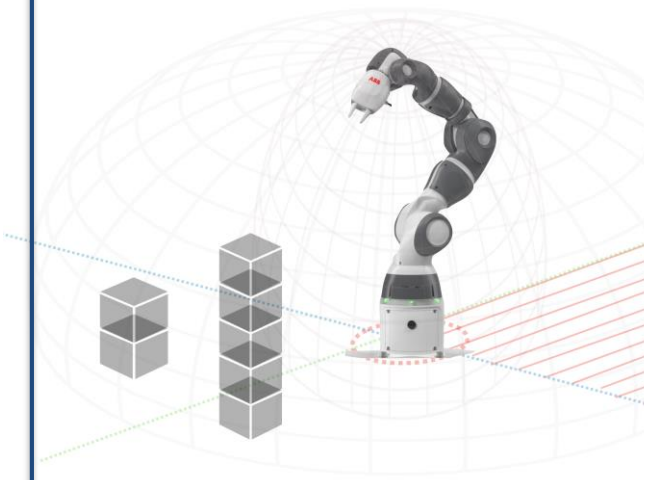
Constraint Optimization

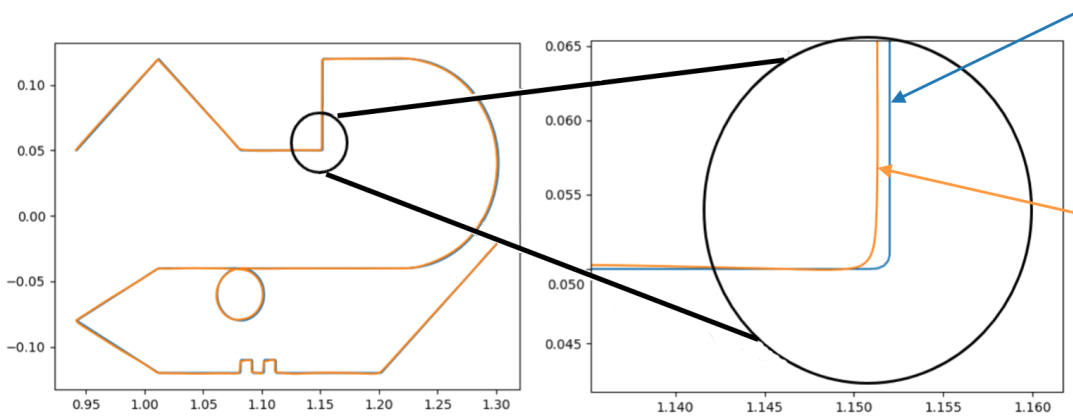


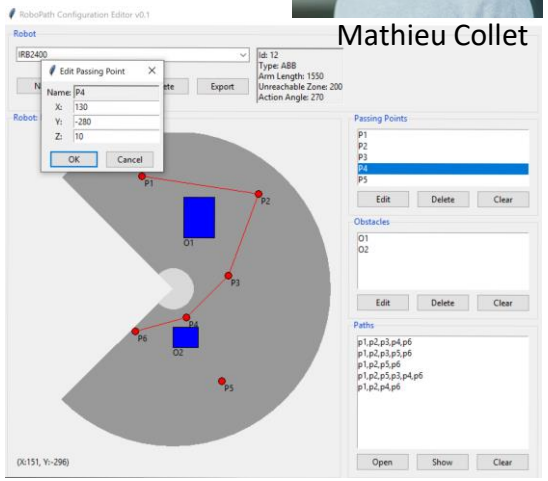
Optimal Stress Test Trajectories for Robots with CP



Mathieu Collet







Generate (near-optimal) stress test trajectories for detecting deviations in 3D workspace with obstacles

Robtest: using global constraints (**table**, **subcircuit**) and **dedicated search heuristics** (max-costs, max-regrets)

Results

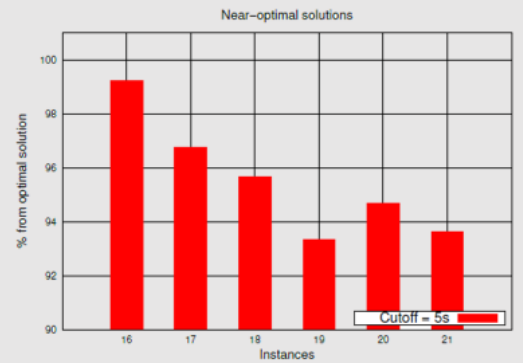
Near-Optimal solution

Computer configuration:
- Core i-7, 2,6 GHz
- 8 GB RAM

Parameter configuration:
- $C_{min} = 1$

#Points	CR	Rho	TG
10	9.32	0.53	0.06
11	10.88	0.50	0.09
12	15.29	0.49	0.22
13	17.53	0.44	0.52
14	20.02	0.48	0.54
15	20.97	0.51	1.52
16	22.65	0.52	5.92
17	26.38	0.53	79.46
18	30.51	0.51	80.46
19	37.02	0.47	132.04
20	37.48	0.53	TO (2 runs)
21	41.35	0.53	TO (6 runs)

TO: a timeout of 1800s



simula

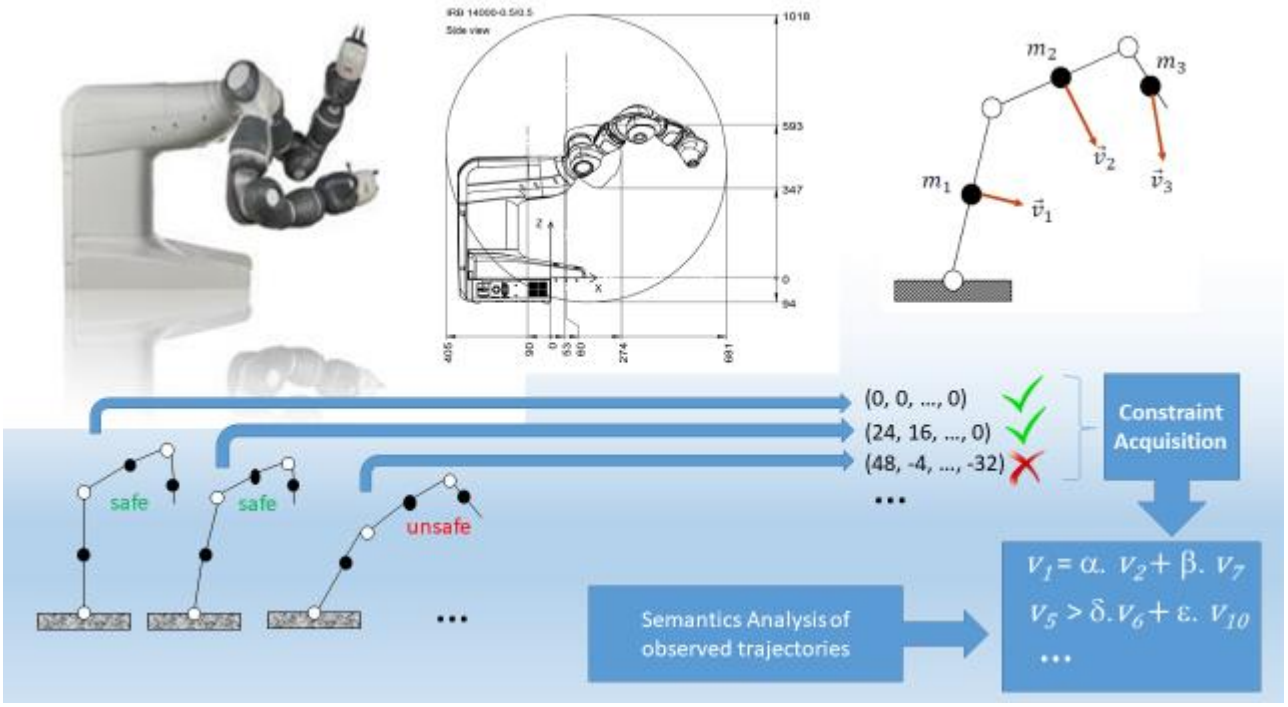
Certus ABB

M. Collet, A. Gotlieb, N. Lazaar and M. Mossige - **Stress Testing of Single-Arm Robots Through Constraint-Based Generation of Continuous Trajectories -** In Proc. of the 1st IEEE Artificial Intelligence Testing Conference (AI Test 2019). San Francisco, CA, USA. Apr. 2019.

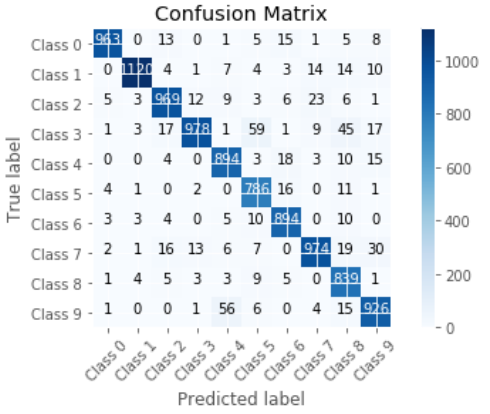
Test of Learning Robots

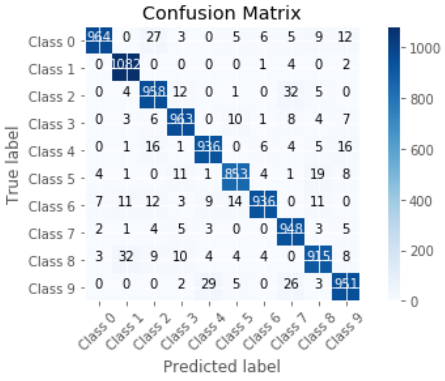
Mohit K. Ahuja

Using **constraint acquisition** to a CP model, enabling the testing of learning collaborative robots



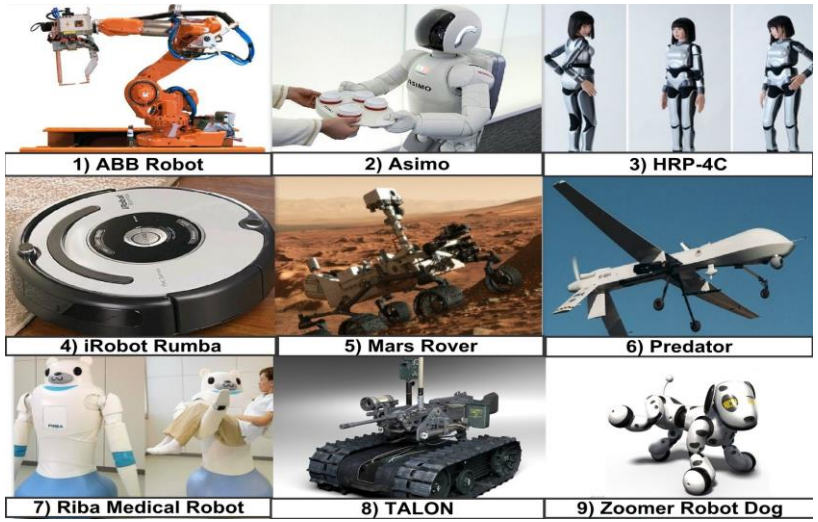
DeepRegression: exploiting regression testing to reduce training datasets





M. Ahuja, A. Gotlieb, D. Marijan and H. Spieker – **DeepRegression: Regression Testing of Deep Learning Systems using Reduced Training Dataset –** In writing

Adaptive Metamorphic Testing



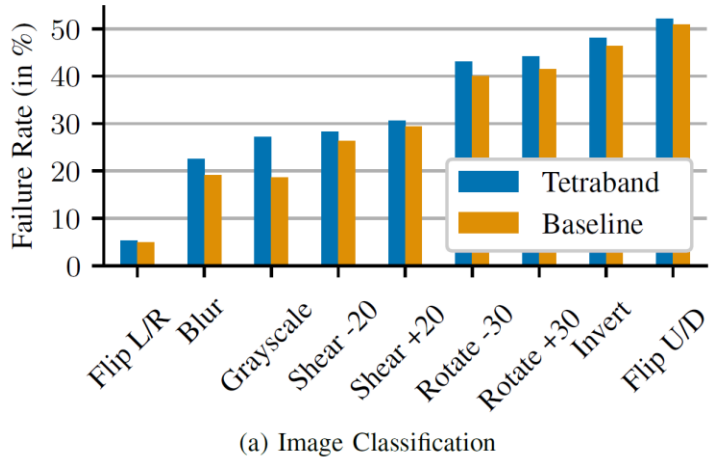
TensorFlow.org - Image classification – dataset of 10,000 images

Object Detection case study – MS COCO dataset of 5,000 images

Helge Spieker

Motivation: Deep Learning based vision systems are hard to test – Metamorphic Testing is the State-of-the-Art method

Adaptive Metamorphic Testing: using contextual bandits to select the Metamorphic Relation which works best



(a) Image Classification

	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Avg.
Blur	10.60	11.40	13.10	9.81	7.30	13.50	17.70	9.00	6.00	6.20	10.46
Flip L/R	2.90	1.00	4.10	6.71	2.20	6.80	1.30	2.40	0.90	2.40	3.07
Flip U/D	14.90	74.60	37.80	33.13	59.10	53.90	29.30	92.40	72.20	43.30	51.06
Grayscale	4.70	5.40	28.10	7.91	18.10	26.00	14.30	6.70	4.80	5.30	12.13
Invert	16.50	29.40	29.50	33.13	41.40	70.30	41.80	38.30	27.30	35.70	36.33
Rotation	25.49	37.09	35.43	17.70	69.00	46.10	20.63	60.44	42.44	50.01	40.43
Shear	11.22	4.99	26.69	35.79	45.45	51.97	15.63	40.24	19.78	55.24	30.70
Avg.	12.33	23.41	24.96	20.60	34.65	38.37	20.10	35.64	24.77	28.31	26.31

Table 1: CIFAR-10 dataset: Effects of MRs by the true class of the image. Each cell value shows the percentage of images in the class, which are wrongly classified after applying the MR. Every class contains 1000 images. Rotation and Shear are parameterized by 30 degrees.

Take Away Message

- *Testing industrial robots* brings new interesting challenges for software V&V research

- **Constraint Programming (CP) and global constraints** are successful in test case generation, test suite reduction and test execution scheduling

- Testing learning capabilities of collaborative robots ischallenging *as:*
  - **Expected behaviours** cannot be specified in advance
  - **Interactions with humans** involve more safety issues

We are eager **to collaborate with experts in Robotics**, to find new methods for testing learning robots

Thank You