simula

# Artificial Intelligence in Software Testing: An Overview

## Application to **Industrial Robotics**

Arnaud Gotlieb
Simula Research Laboratory
Norway



JFTL 2018 – 10 april 2018

# The Certus Centre

Software Validation and Verification


Cisco Systems Norway


Cancer Registry of Norway

[ simula ]

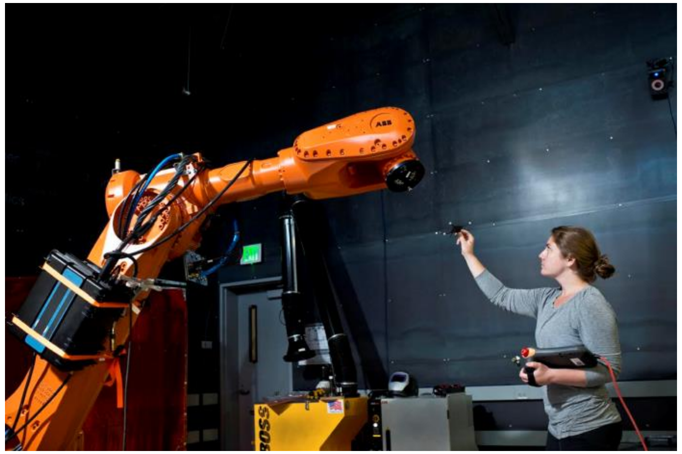


ABB Robotics

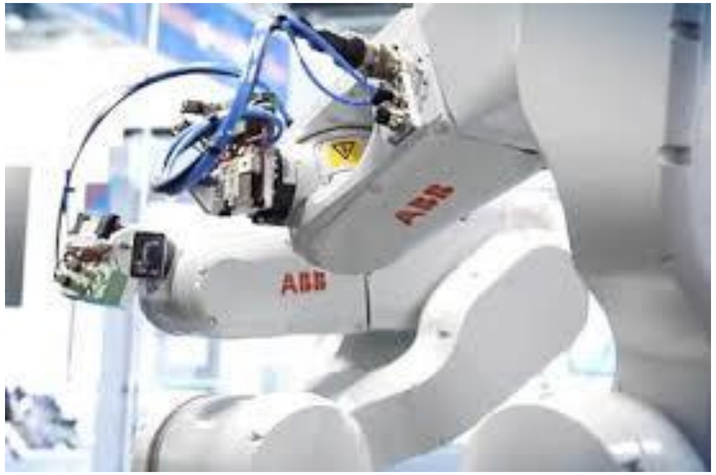

Kongsberg Maritime

www.certus-sfi.no

esito

# 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 to collaborate with human co-workers

# Testing Robotic Systems is Crucial and Challenging

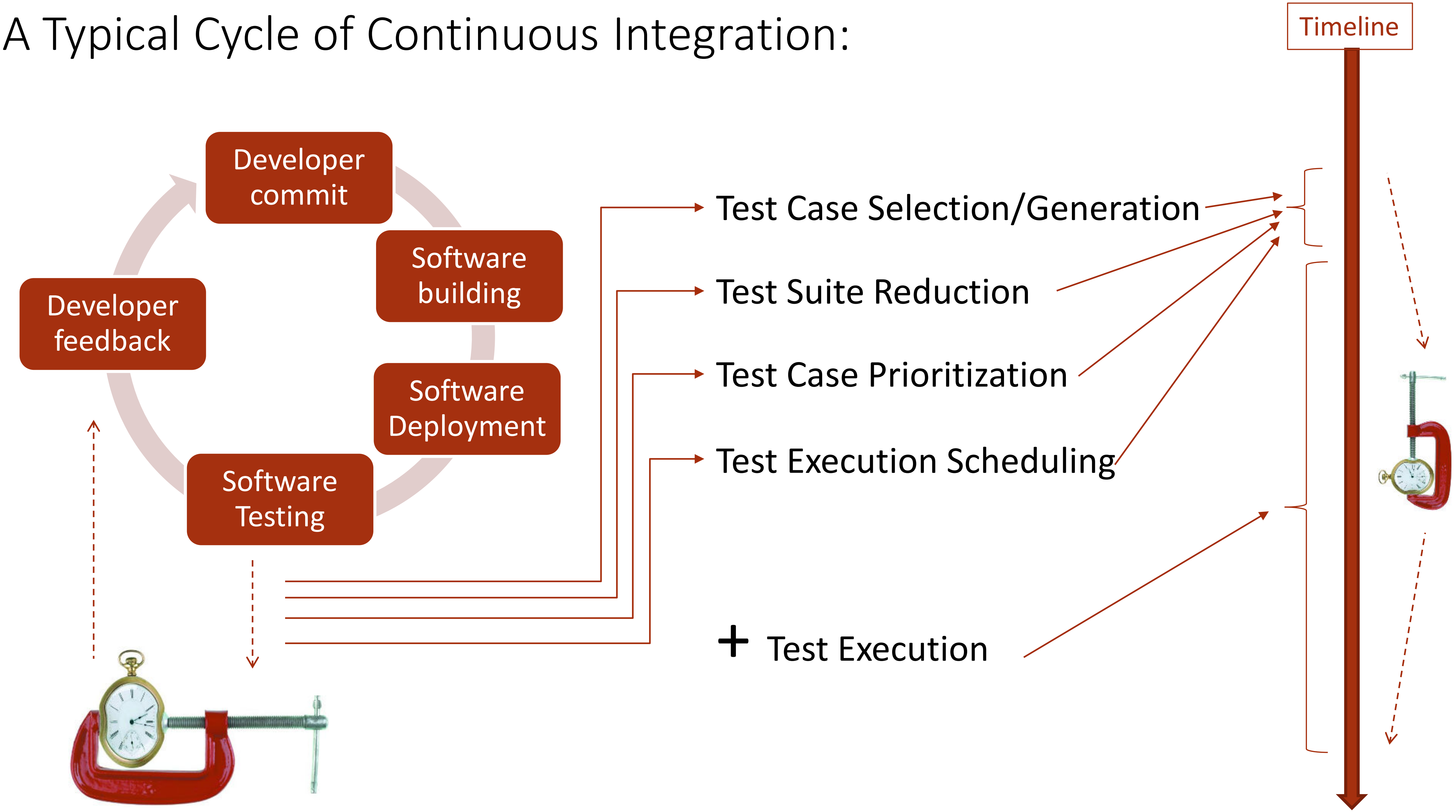
- The validation of industrial robots still involve too much human labour
- "*Hurry-up, the robots are uncaged!*": Failures are not anymore handled using fences
- Robot behaviours evolve with changing working conditions
- Today, industrial robots can be taught by-imitation.
  Tomorrow, they will learn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

More automation in testing

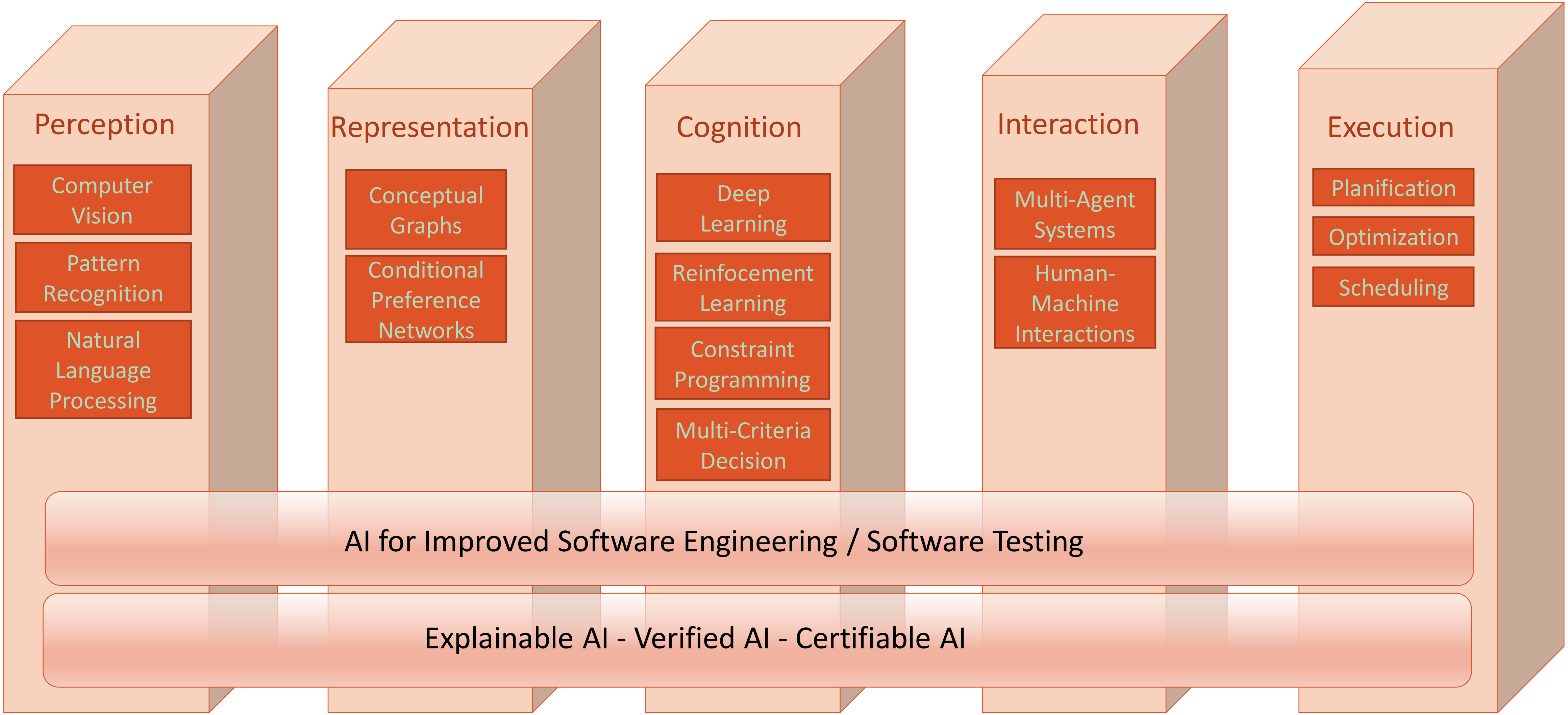
More diversity in testing


More efficiency in testing

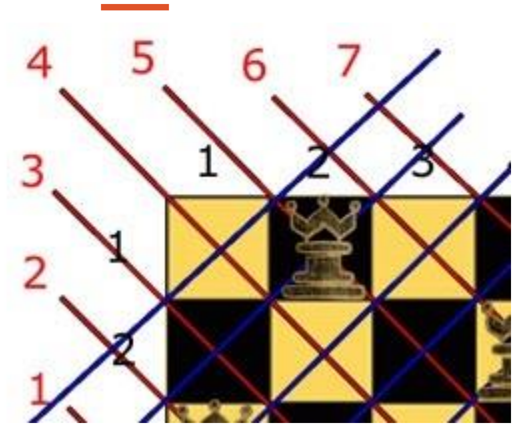
# A Typical Cycle of Continuous Integration:



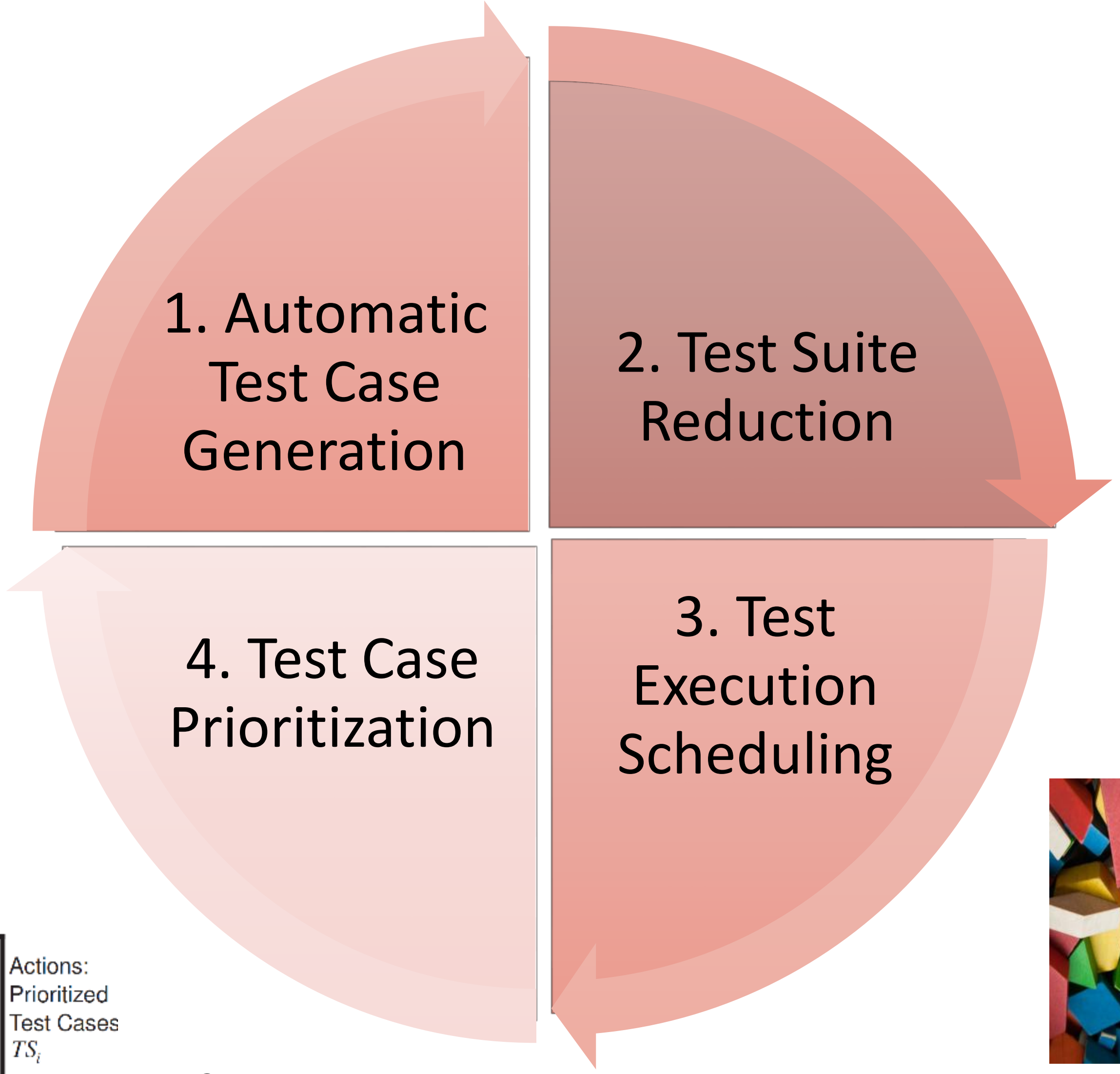
# Artificial Intelligence in a Nutshell

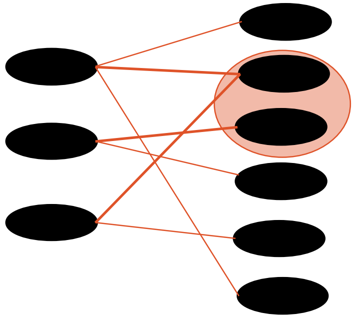


# Our Focus : Artificial Intelligence for Improving Software Testing

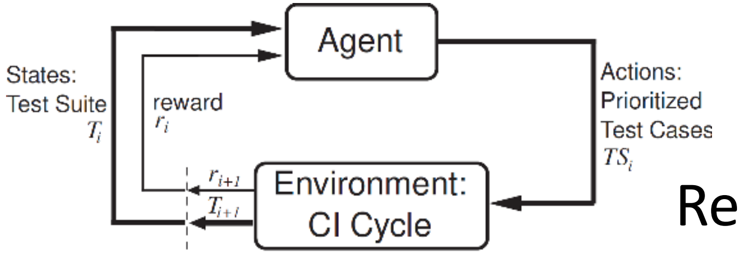


Constraint Modelling

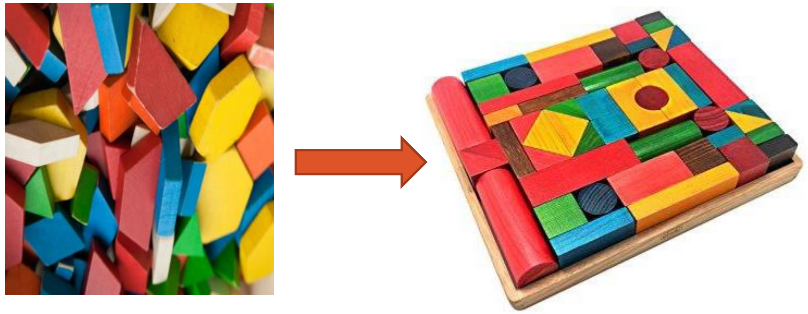


Global Constraints

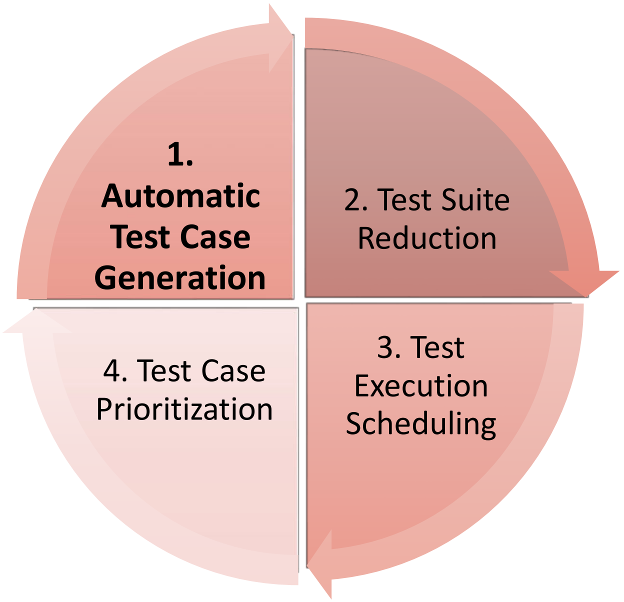


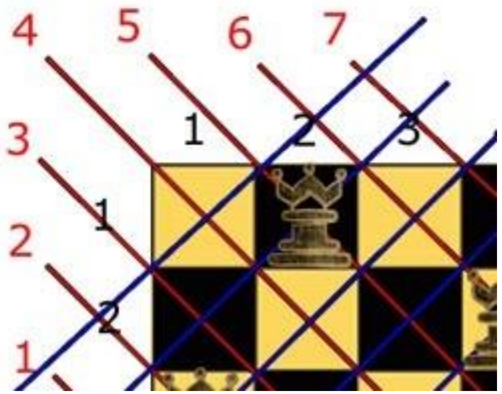

Reinforcement Learning



Constraint-based Scheduling

simula



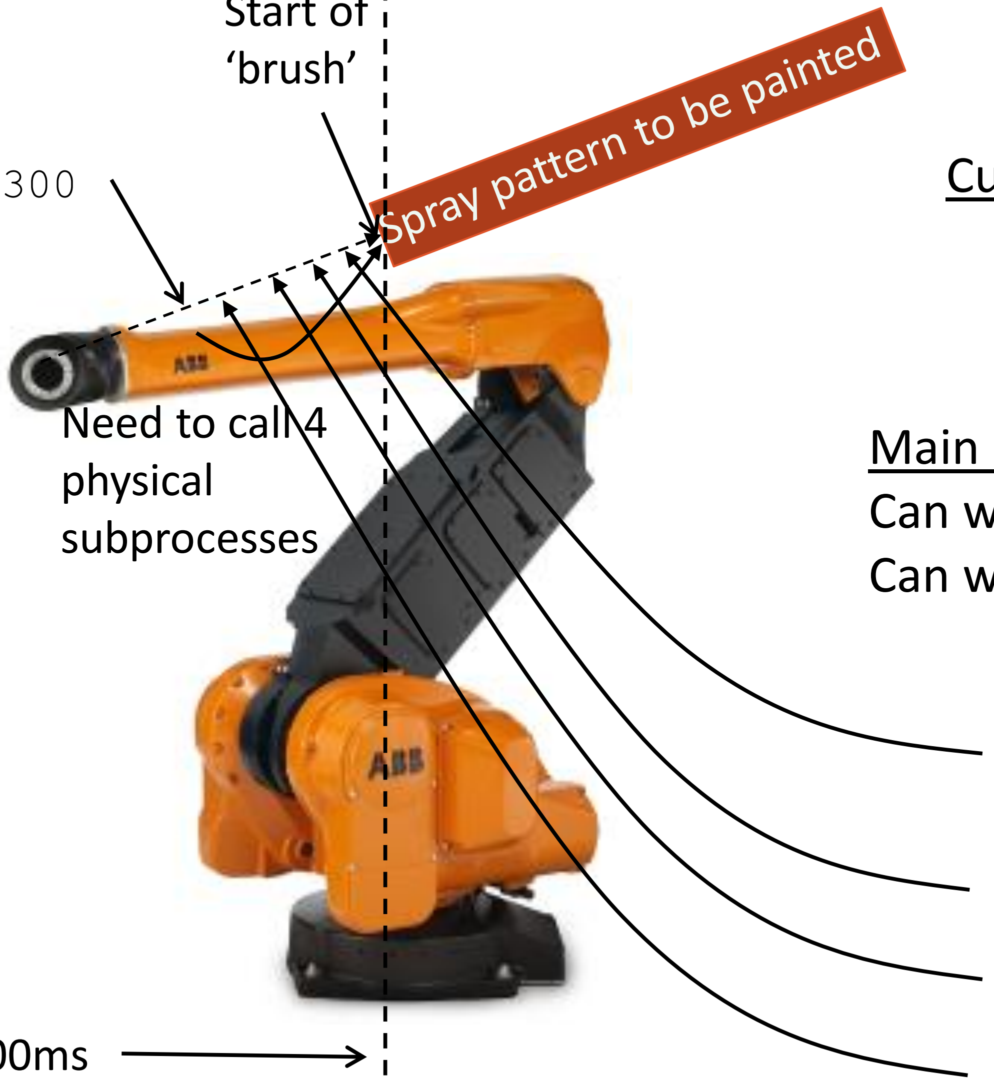




Constraint Modelling

# 1. Automatic Test Case Generation

simula

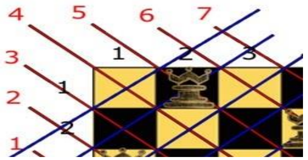
# A Typical Robot Painting Scenario



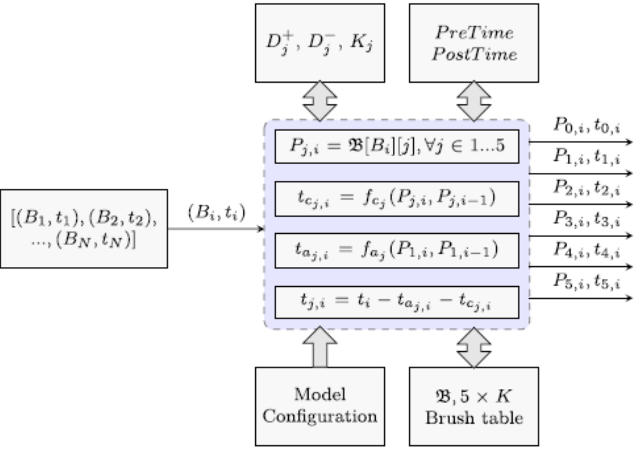

<u>Crucial test objective:</u>
to validate that physical outputs are triggered on expected time

<u>Current practice:</u>

<u>Main issue:</u>
Can we automate this testing process?
Can we integrate an AI model into Continous Testing?




# AI-Powered Model of IPS




Test sequence

| $t_i$ | $B_i$ |
|---|---|
| 300 | 1 |
| 600 | 2 |
| 900 | 1 |
| 1200 | 0 |

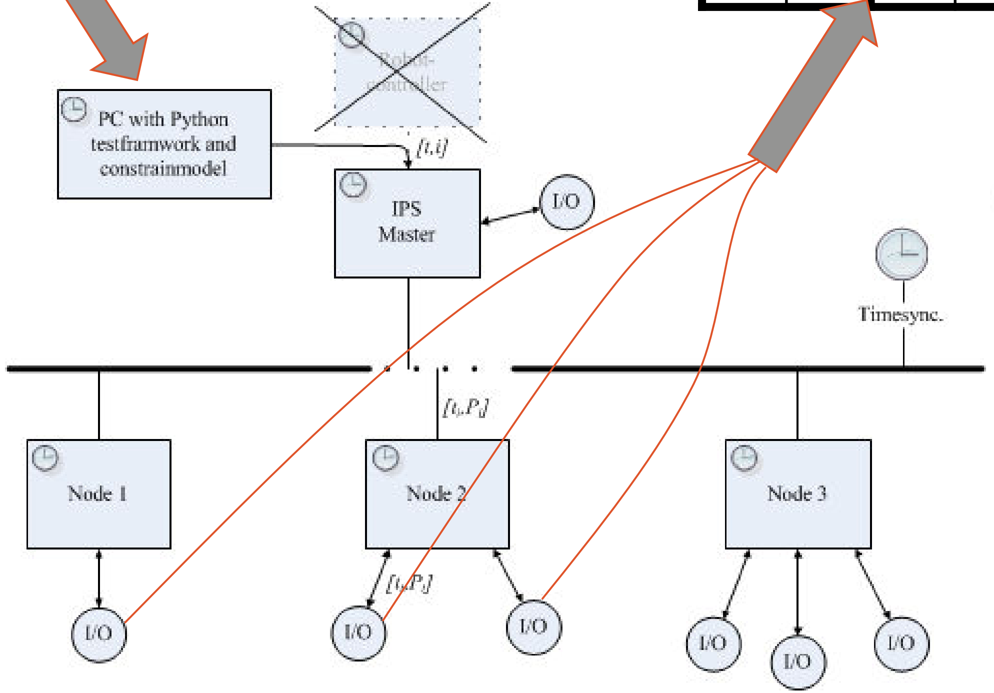
Test oracle

| $t_t$ | I/O-1 | $t_t$ | I/0-2 | $t_t$ | I/O-3 |
|---|---|---|---|---|---|
| 295 | 75 | 120 | 150 | 205 | 75 |
| 579 | 500 | 500 | 175 | 585 | 150 |
| 879 | 75 | 780 | 150 | 881 | 75 |
| 1195 | 0 | 1130 | 0 | 1231 | 0 |

Compare


Test results

| $t_t$ | I/O-1 | $t_t$ | I/0-2 | $t_t$ | I/O-3 |
|---|---|---|---|---|---|
| 294 | 75 | 121 | 150 | 205 | 75 |
| 579 | 500 | 501 | 175 | 585 | 150 |
| 880 | 75 | 792 | 150 | 880 | 75 |
| 1197 | 0 | 1131 | 0 | 1232 | 0 |





## Issues for deployment:

1. Can we control the solving time wrt the test execution time?

2. Is this Constraint-based Testing approach interesting to find bugs?

3. Can we ensure enough diversity in the generated test scenarii?

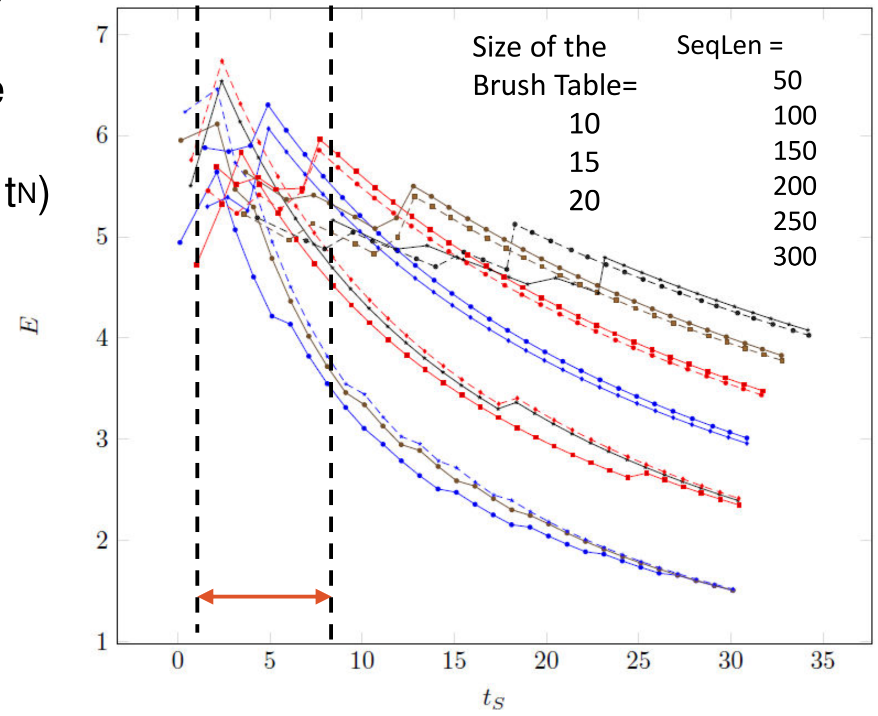
simula

# Industrial Deployment

[Mossige et al. CP’14, IST’15]

E: Efficiency factor
$t_S$ : Solving time
$t_N$ : Test exec. time

$E = SeqLen / (t_S + t_N)$



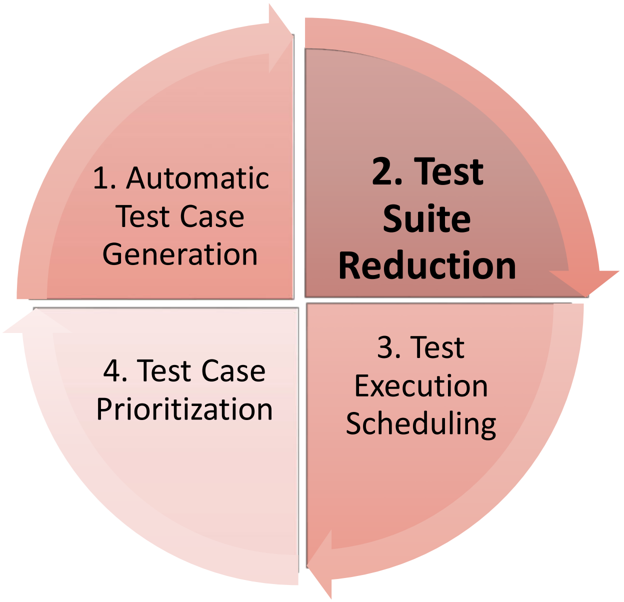

- Integrated throug ABB’s Continuous Integration process
- Constraint model is solved ~15 times per day

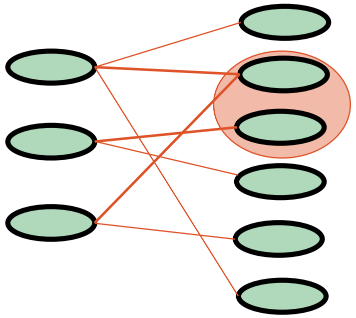
During initial deployment, it found 5 critical bugs
+ dozens of (non-critical) new bugs

But, since then, bug discovery has decreased!
still working on

1. Maximizing the diversity among test scenarii
2. Generating test scenarios for multi-robots

simula



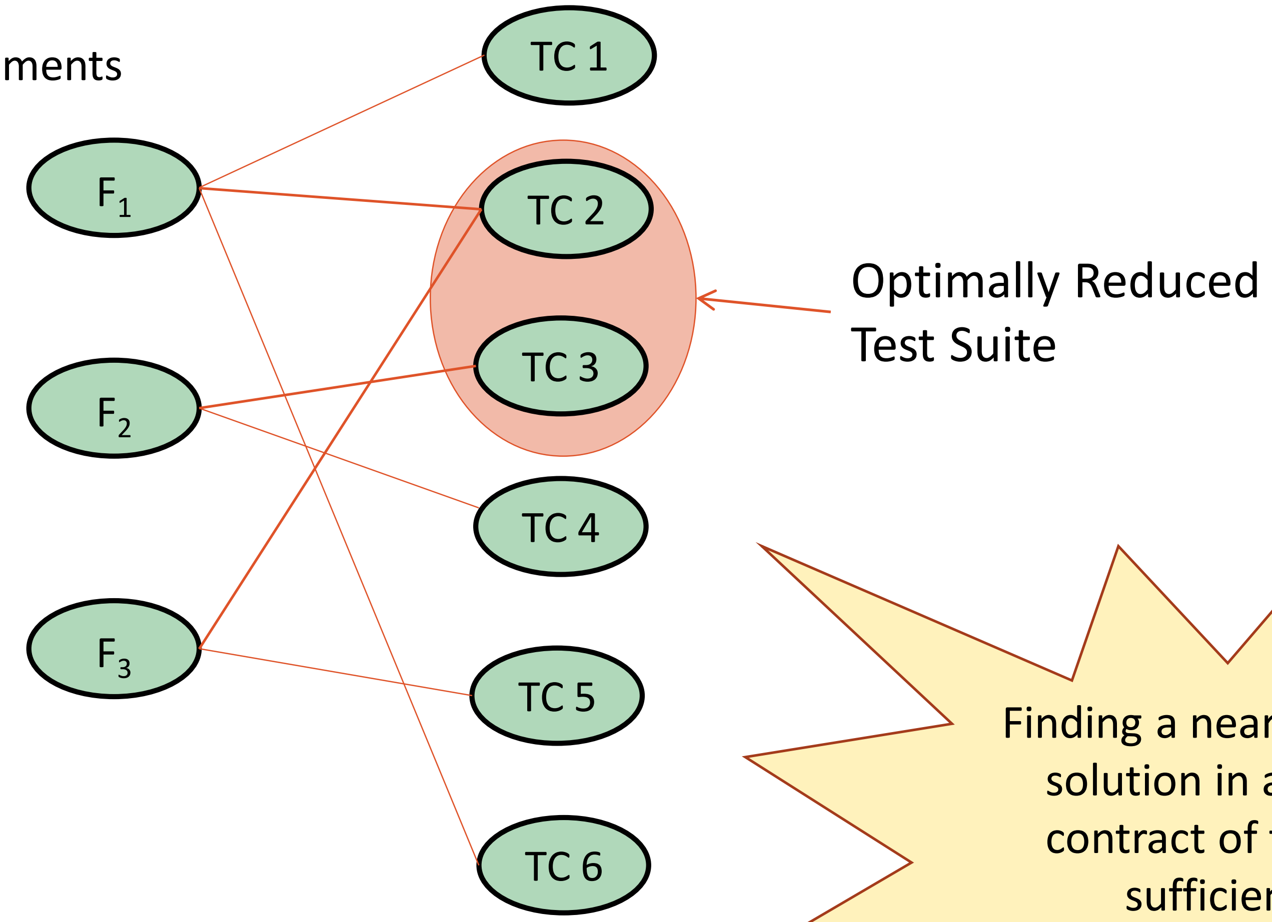
Global Constraints

# 2. Test Suite Reduction

simula

# Test Suite Reduction: the core problem

$F_i$: Features / User Requirements
TC: Test Cases

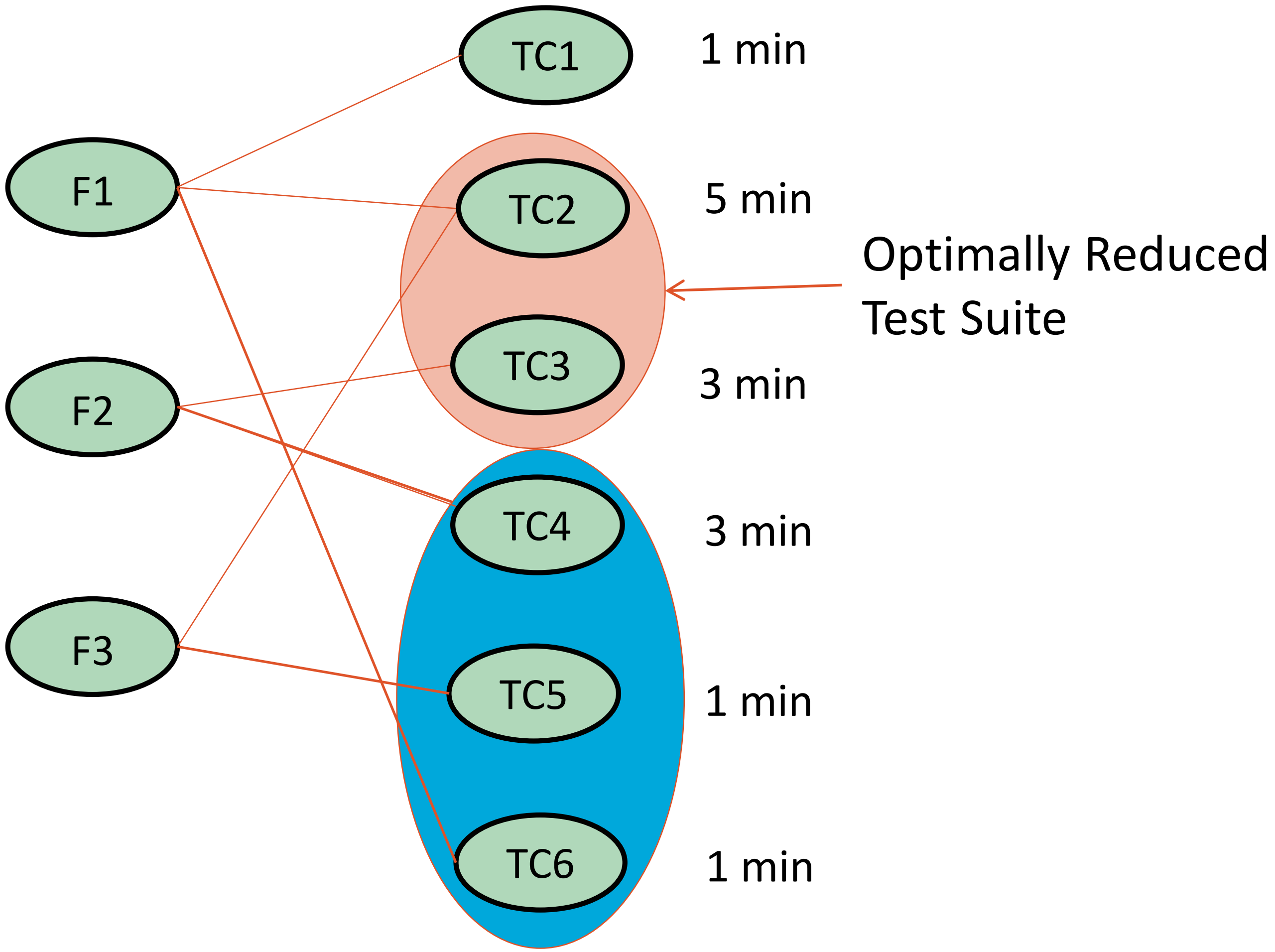



NP-hard problem!

Finding a near-optimal solution in a given contract of time is sufficient!

# Other criteria to minimize

**Feature coverage**
is always a prerequiste

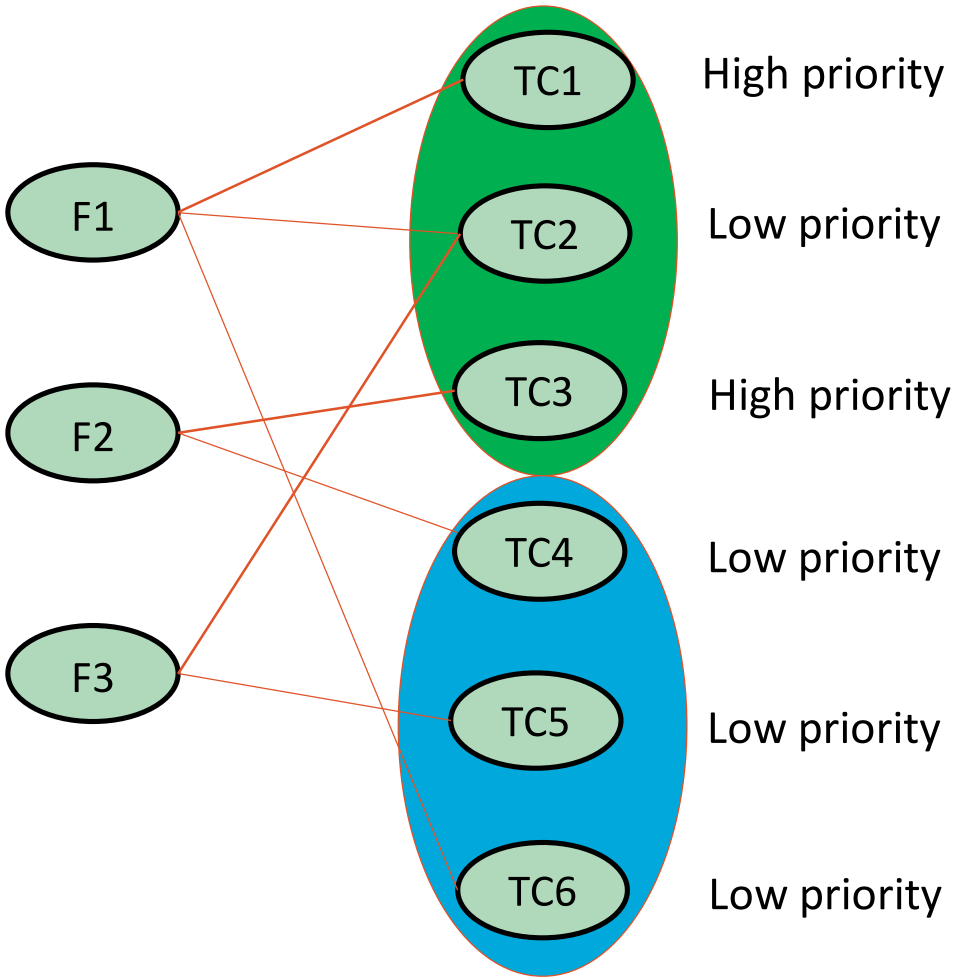



Execution time!

simula

# Other criteria to minimize

**Feature coverage**
is always a prerequiste




Fault revealing capabilities!

simula

# Test Suite Reduction: Existing Approaches

- Exact methods: Integer Linear Programming *[Hsu Orso ICSE 2009, Campos Abreu QSIC 2013,…]*

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

subject to $\begin{cases} x_1 + x_2 + x_6 \geq 1 \\ x_3 + x_4 \geq 1 \\ x_2 + x_5 \geq 1 \end{cases}$ (cover every feature. at least once)

- Approximation algorithms (greedy, search-based methods) *[Harrold et al. TOSEM 1993, …]*

```
F = Set of reqs, Current = Ø
    while( Current ǂ F)
        Select a test case that covers the most uncovered features ;
        Add covered features to Current ;
    return Current
```

- AI-powered method:
  **Constraint Programming with Global Constraints** *[Gotlieb et al. ISSTA 2014, AI Magazine 2016, …]*
  **Multi-Criteria Test Minimization** *[Wang et al. JSS 2015, ESE 2015, …]*


# **gcc**: global cardinality constraint
# Powerfull AI combinatorial tool

$F_1,..,F_n$: Features

$t_1,..,t_m$: Test cases

$c_1,..,c_m$: Unit cost for each test case

This cost value aggregates different criteria (e.g., execution time, …)

Minimize TotalCost

s.t

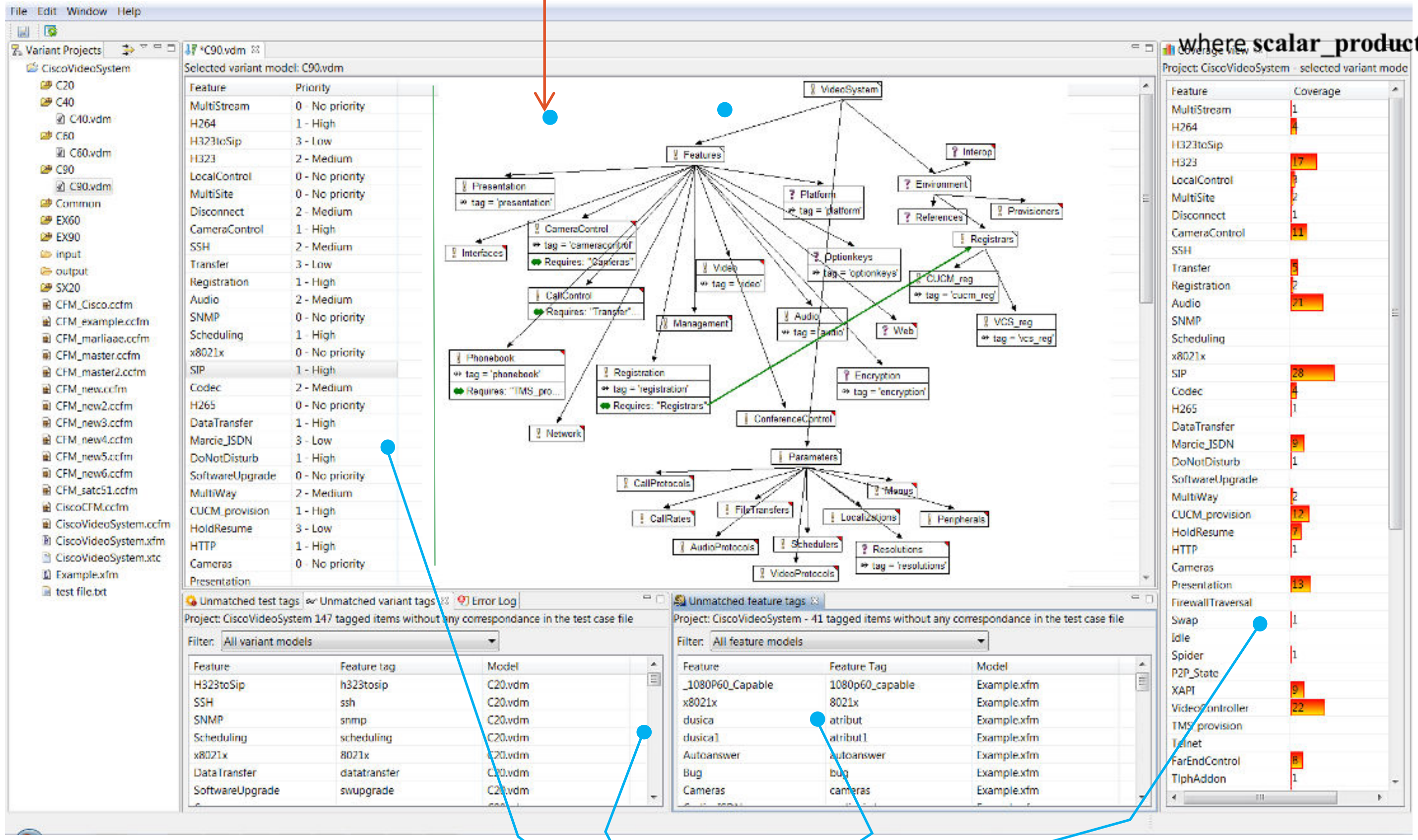
**gcc**$([F_1, …, F_n], [t_1, …, t_m], [O_1, …, O_m])$

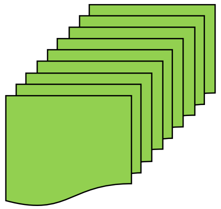
for i=1 to m do $B_i = (O_i > 0)$

**scalar_product**$([B_1, …, B_m], [c_1, …, c_m], \text{TotalCost})$

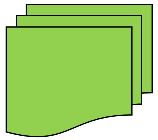
where **scalar_product** encodes $B_1*c_1 + .. + B_m*c_m = \text{TotalCost}$

**Variability model to describe a product line**





**Unoptimized test suite**

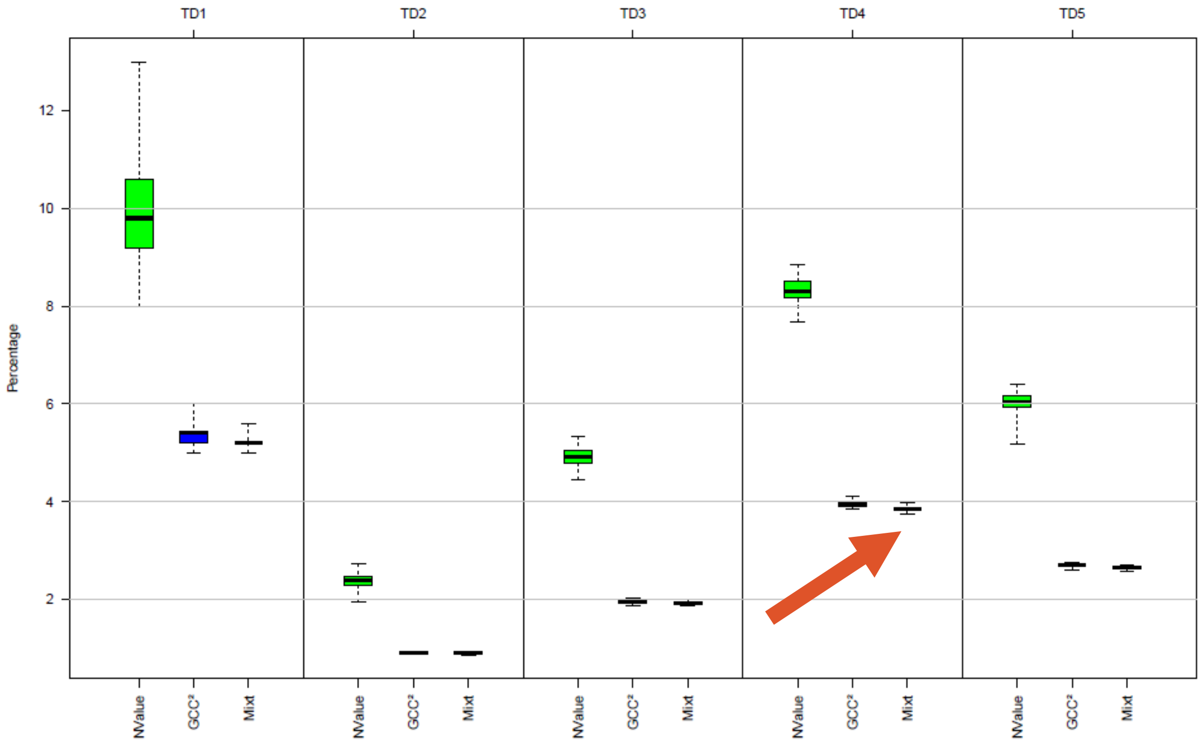


**Optimized (reduced) test suite**

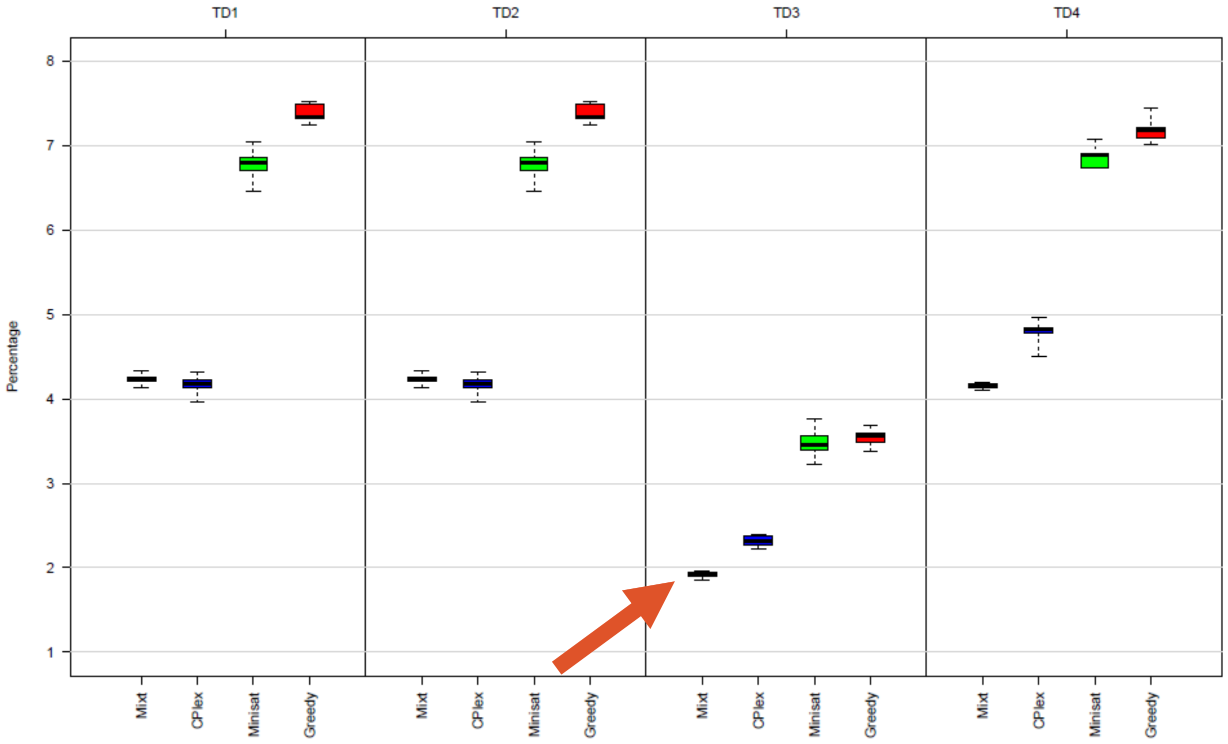
**Diagnostic views, feature coverage**

# Comparison with CPLEX, MiniSAT, Greedy (uniform costs)

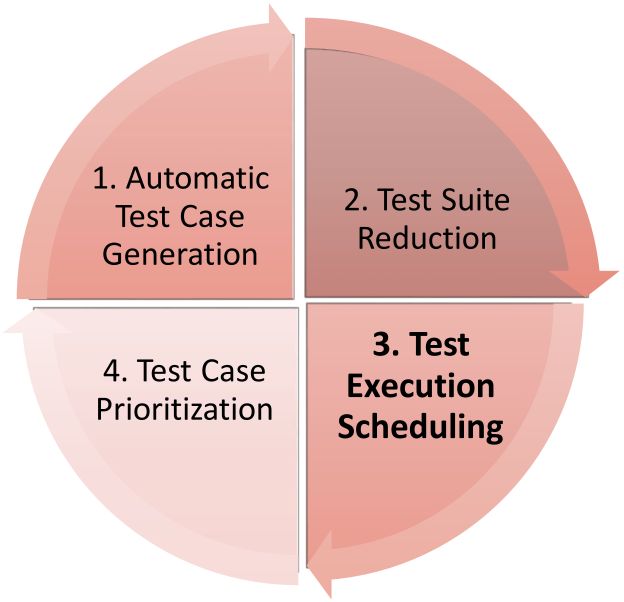
(Reduced Test Suite percentage in 60 sec)

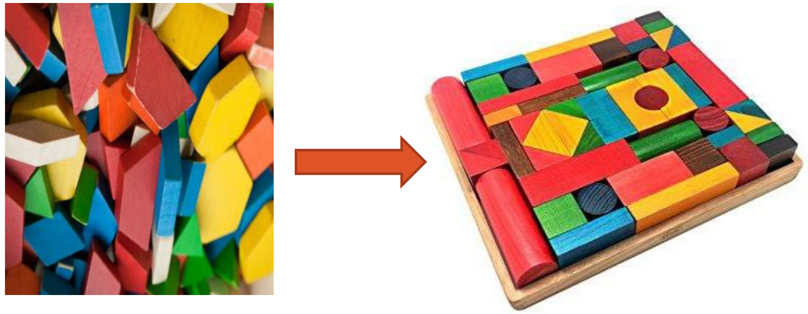



| | TD1 | TD2 | TD3 | TD4 | TD5 |
|---|---|---|---|---|---|
| Requirements | 250 | 500 | 1000 | 1000 | 1000 |
| Test cases | 500 | 5000 | 5000 | 5000 | 7000 |
| Density | 20 | 20 | 20 | 8 | 8 |




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



Constraint-based Scheduling

# 3. Test Execution Scheduling

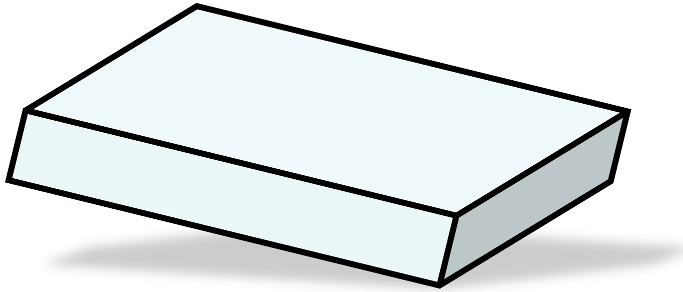
# Test Execution Scheduling

Test Cases
with distinct
characteristics

Schedule

Assignment of Test Cases to Agents such that:

1. Capacity constraints are not exceeded
2. Test Agents are well occupied
3. Test Execution Time is minimized



Test Agents
(Robots)
with limited
(time or resources)
capacity

Additionally, there can be some shared global resources among test cases (e.g., flow meter, oscilloscope, camera, …)

simula

# Constraint Models for Test Scheduling

ABB

Test Cases Repository:
~10,000 Test Cases (TC)
~25 distinct Test Robots
Diverse tested features

10..30 code changes per Day

python™ SICStus 4

Test Cases:
- duration
[- priority]
[- history]

**Constraint-based scheduling Models**

1. Greedy approach
2. Constraint-based scheduling
3. Advanced Constraint-based scheduling using bin-packing

| | |
|---|---|
| 1 | Deployed at ABB in CI / «Good Enough» |
| 2 | Evaluated / Needs Improvements |
| 3 | Deployment in progress |

T2, T5, T34

T3, T6, T45, T78

T7, T23

T4, T56, T67

T45, T55




simula

# Experimental results (Comparing model 3 vs model 1)

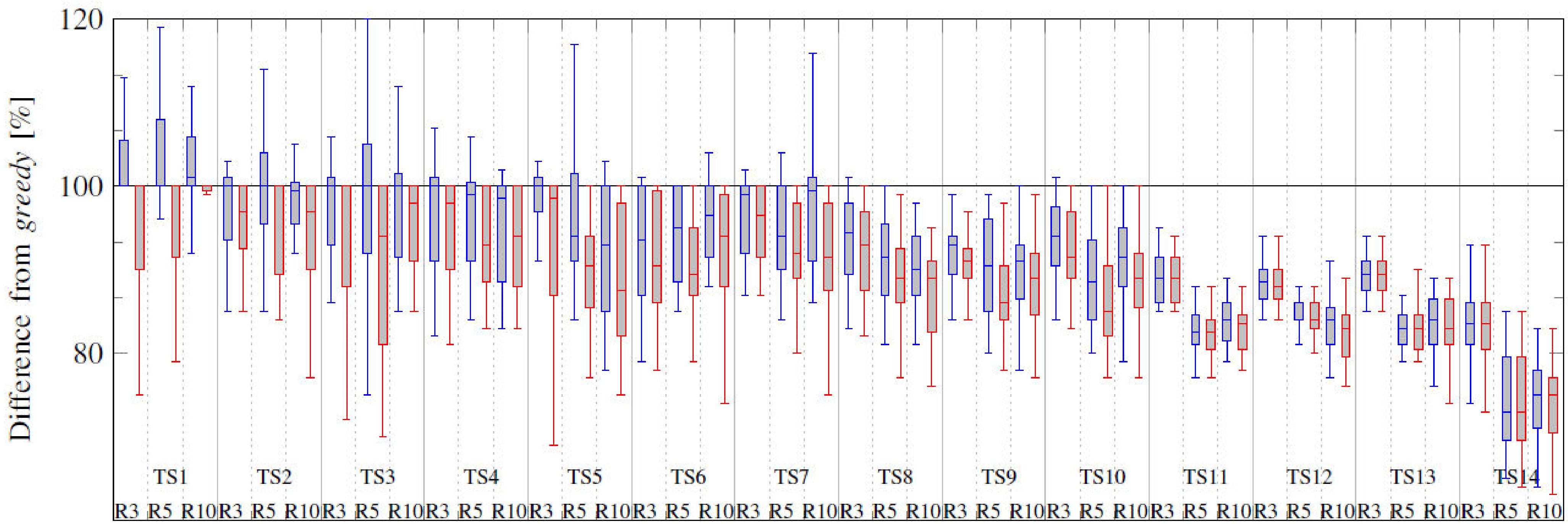
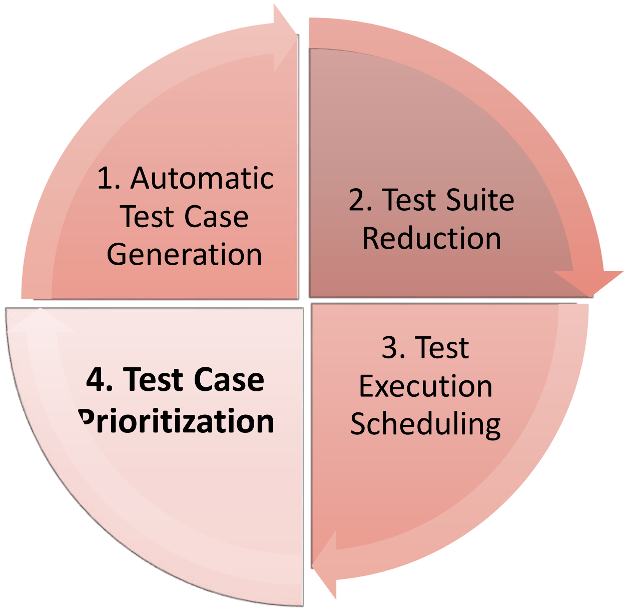


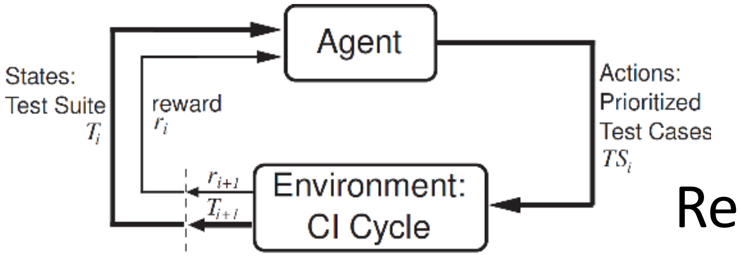

Fig. 5. The differences in schedule execution times produced by the different methods for test suites TS1–TS14, with *greedy* as the baseline of 100%. The blue is the difference between $C_f^*$ and *greedy* and the red shows the difference between $C_l^*$ and *greedy*.

| # machines | # of tests | 20 | 30 | 40 | 50 | 100 | 500 |
|---|---|---|---|---|---|---|---|
| | 100 | - | - | - | - | - | TS11 |
| | 50 | - | - | - | - | TS8 | TS12 |
| | 20 | - | TS2 | TS4 | TS6 | TS9 | TS13 |
| | 10 | TS1 | TS3 | TS5 | TS7 | TS10 | TS14 |

But, handling test case diversity is challenging!




# 4. Test Case Prioritization




Reinforcement Learning

simula

# Motivation: Learning from previous test runs of the robot control systems

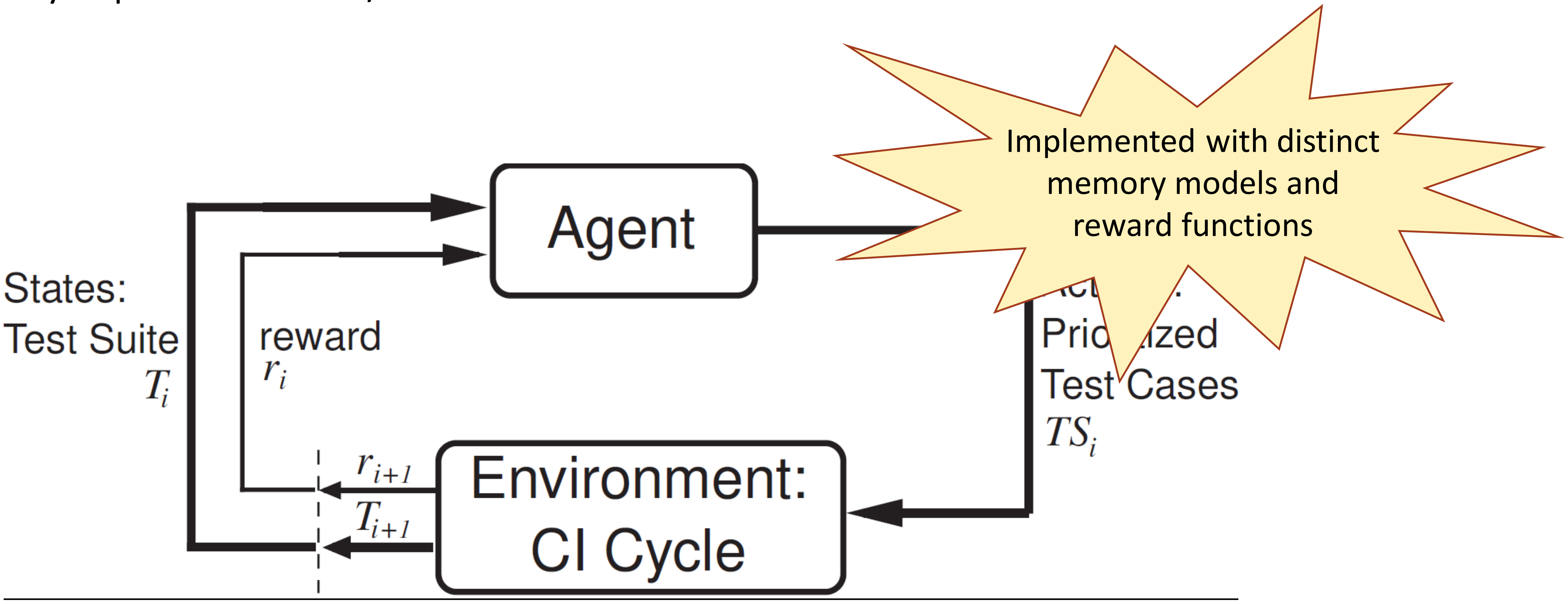
- Adapt testing to focus on the more error-prone parts of the tested system
- Adapt testing to the execution environment (available robots and devices, limited testing time and resources, experiences from previous cycles in continuous integration)


Robot learning different testing techniques

simula

# Using Reinforcement Learning to prioritize test case execution

- Considering test case meta-data only (test verdicts, tested robots, execution time, ...) → lightweight method
- Reward function based on test verdicts from the previous CI-cycles → online ML
- Limited memory of past executions / test results



simula

# Does it learn?

3 Industrial data sets (1 year of CI cycles)
NAPFD: Normalized Average Percentage of Faults Detected

*Reward Function 1. Failure Count Reward*

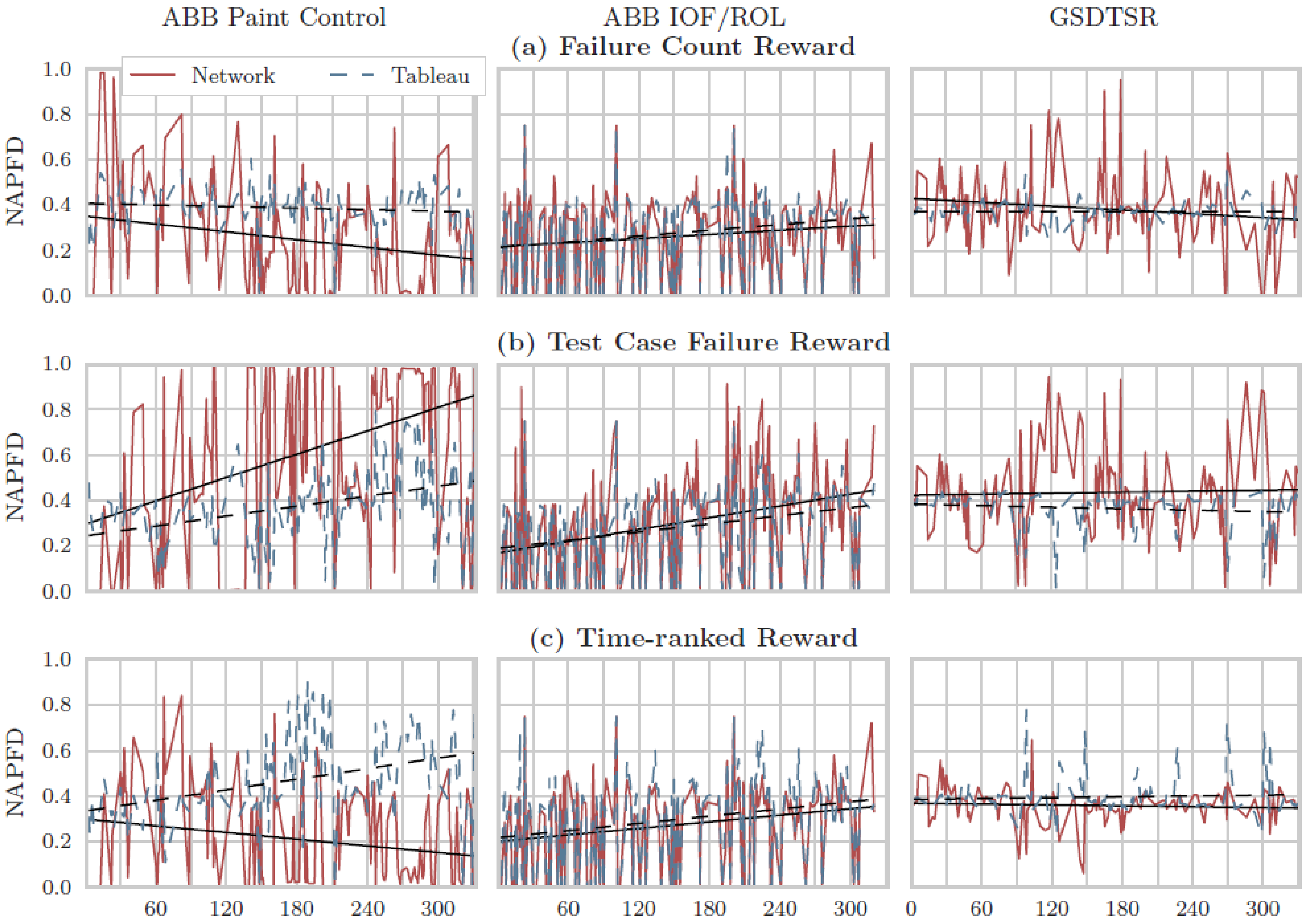
$$reward_i^{fail}(t) = |\mathcal{TS}_i^{fail}| \qquad (\forall\, t \in \mathcal{T}_i)$$

*Reward Function 2. Test Case Failure Reward*

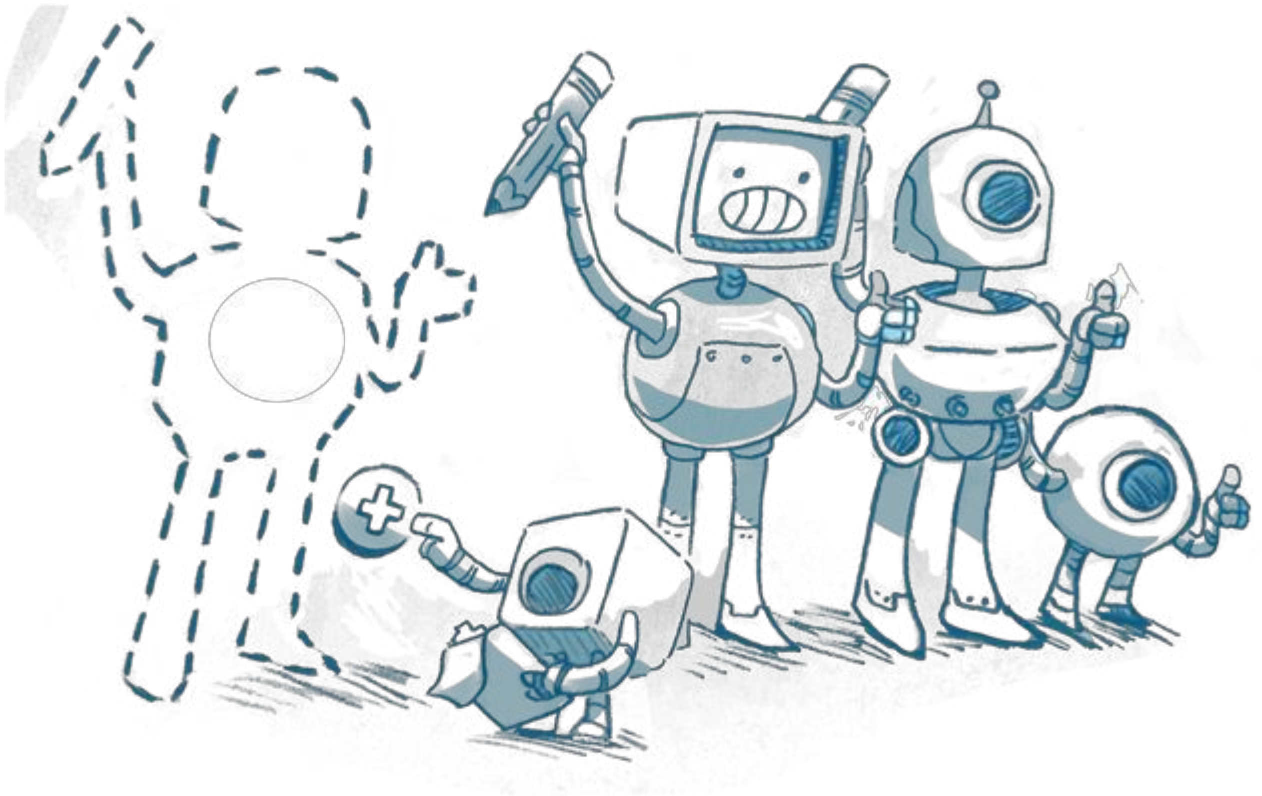
$$reward_i^{tcfail}(t) = \begin{cases} 1 - t.verdict_i & \text{if } t \in \mathcal{TS}_i \\ 0 & \text{otherwise} \end{cases}$$

*Reward Function 3. Time-ranked Reward*

$$reward_i^{time}(t) = |\mathcal{TS}_i^{fail}| - t.verdict_i \times \sum_{\substack{t_k \in \mathcal{TS}_i^{fail} \wedge \\ rank(t) < rank(t_k)}} 1$$



# Lessons Learned and Emerging Topics



simula

# Lessons learned

- Industrial Robotics is an interesting application field for AI-powered software testing approaches

- More automation is highly desired in industrial robotics
  AI is a key-enabler for *Release better, release faster, release cheaper!*

- **Adoption of (robust) AI techniques** beneficial in test automation and optimization:

  Constraint Programming, Scheduling, Reinforcement Learning, …

Many Emerging Challenges!

simula

# Emerging Topics

- Testing Learning Robots (**RCN T-LARGO Project**)
- Machine Learning in Continuous Testing Processes (**Collaboration Smartesting**)
- AI-on-demand platform for performance testing of industrial robots **(AI4EU H2020 Proposal)**
- Testing Human Perception of Robot Safety


Thanks to:

**Dusica Marijan** (SIMULA, Norway)
**Morten Mossige** (ABB Robotics, Norway)
**Helge Spieker** (SIMULA, Norway)
**Shuai Wang** (SIMULA, Norway)
**Marius Liaeen** (CISCO Systems, Norway)
**Mats Carlsson** (SICS, Sweden)
**Carlo Ieva** (SIMULA, Norway)
....

# References


1. [Spieker et al. 2017] H. Spieker, A. Gotlieb, D. Marijan and M. Mossige
   **Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration**
   In Proc. of 26th Int. Symp. on Soft. Testing and Analysis (ISSTA-17), Santa Barbara, USA, July 2017.
2. [Gotlieb Marijan 2017] A. Gotlieb and D. Marijan
   **Using Global Constraints to Automate Regression Testing** AI Magazine 38, Spring, 2017.
3. [Marijan et al. 2017] D. Marijan, A. Gotlieb, M. Liaaen, S. Sen and C. Ieva
   **TITAN: Test Suite Optimization for Highly Configurable Software**
   In Int. Conf. on Soft. Testing, Verification and Validation (ICST-17), Tools Track, Tokyo, Japan, 2017.
4. [Mossige et al. 2017] M.Mossige, A. Gotlieb, H. Spieker, H. Meling, M. Carlsson
   **Time-aware Test Case Execution Scheduling for Cyber-Physical Systems**
   In Principles and Practice of Constraint Programming (CP-17) – Application Track, Melbourne, Australia, Aug. 2017
5. [Gotlieb et al., 2016] A. Gotlieb, M. Carlsson, D. Marijan and A. Petillon
   **A New Approach to Feature-based Test Suite Reduction in Software Product Line Testing**
   In 11th Int. Conf. on Software Engineering and Applications (ICSOFT-16), Lisbon, July 2016, **Awarded Best Paper**
6. [Mossige et al., 2015] M. Mossige, A. Gotlieb, and H. Meling.
   **Testing robot controllers using constraint programming and continuous integration.**
   Information and Software Technology, 57:169-185, Jan. 2015.
7. [Wang et al., 2015] S. Wang, S. Ali, and A. Gotlieb.
   **Cost-effective test suite minimization in product lines using search techniques.**
   Journal of Systems and Software 103: 370-391, 2015.
8. [Gotlieb et al., 2014] 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.
9. [Mossige et al., 2014] M. Mossige, A. Gotlieb, and H. Meling.
   **Using CP in automatic test generation for ABB robotics' paint control system.**
   In Principles and Practice of Constraint Programming (CP-14) – **Awarded Best Application Paper**, Lyon, Fr., Sep. 2014.