Leveraging AI Methods for Testing Non-testable Autonomous Systems

Arnaud Gotlieb
Simula Research Laboratory
Norway

EDCC 2021 - Munich, Germany
Sep. 15, 2021

Autonomous Software-Systems

- **Systems which have a certain degree of self-decision capabilities**,
  e.g., self-driving cars, industrial robots, smart transportation systems, smart communication systems, …

- Systems with increased capabilities of **planning** (what, how), **scheduling** (when, who) and **executing complex functions**, with limited human intervention, **managing unexpected events**, such as faults or hazards

- **Not equal to *automated systems***, which have limited capacity to learn and adapt to unexpected events

- In robotics and automated driving, the main focus for autonomy is to complement human's capacity to take **decisions based on vast amounts of uncertain raw data**

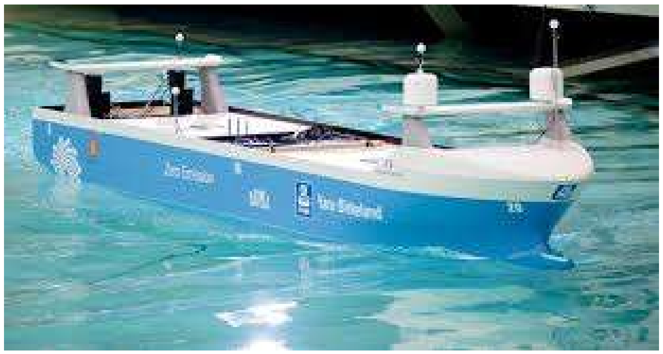
IEEE Spectrum – Self-driving car

Universal Robot – UR3 Cobot

ABB Robotics – YUMI Cobot

Kongsberg Maritime – Yara Birkeland

Norwegian Yara Birkeland

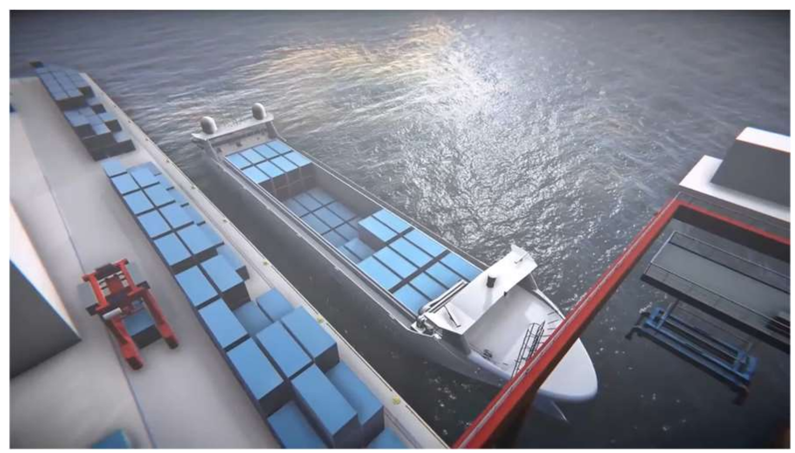


This electrical autonomous cargo vessel will transport fertiliser from Yara's Porsgrunn plant via inland waterways to the deep-sea ports of Larvik and Brevik (31 nautical miles). Removing up to 40,000 truck journeys annually.

Norwegian shore

Automated Mooring System

The system is based on a seven-axis robotic arm that takes the mooring ropes with loops and wraps them around bollards on the dock.
The mooring system has redundant kinematics, with built-in movement compensation and track planning.
The vessel’s position against the quay will inform the robotic arm where each bollard is located, and the track planning is automatically generated by the control system.

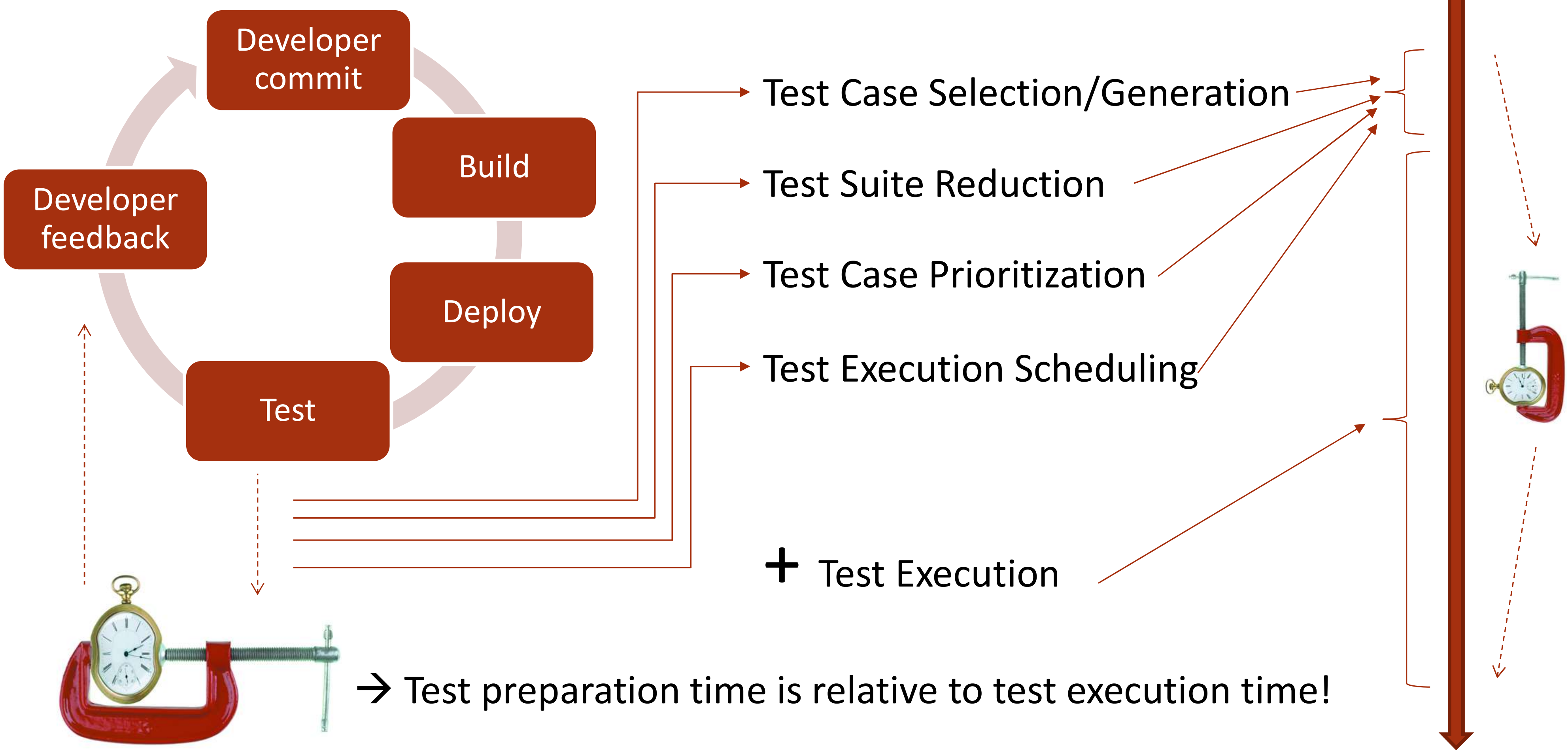


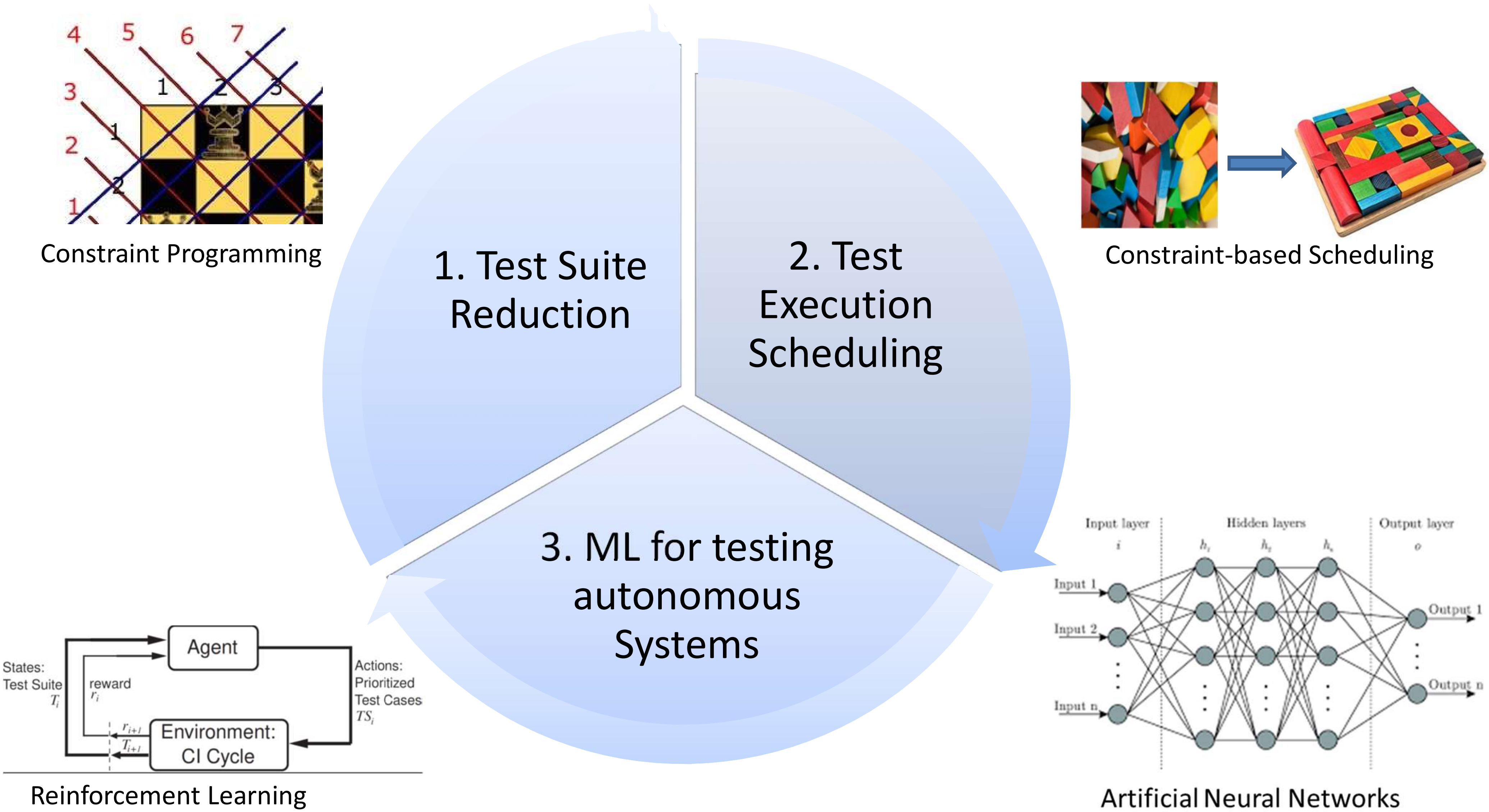
Source: MacGregor Inc.

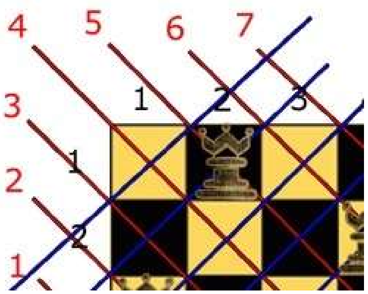
Testing Non-testable Autonomous Systems

- Testing perception systems needs to generate tests with (environment) hazards
- Test coverage over high-dimensional inputs is limited
- Non-linear motion planning involves solving complex constraint models
- Validation of learning systems needs test oracles which can hardly be defined
- Continuous testing is key but needs high control and more diversity

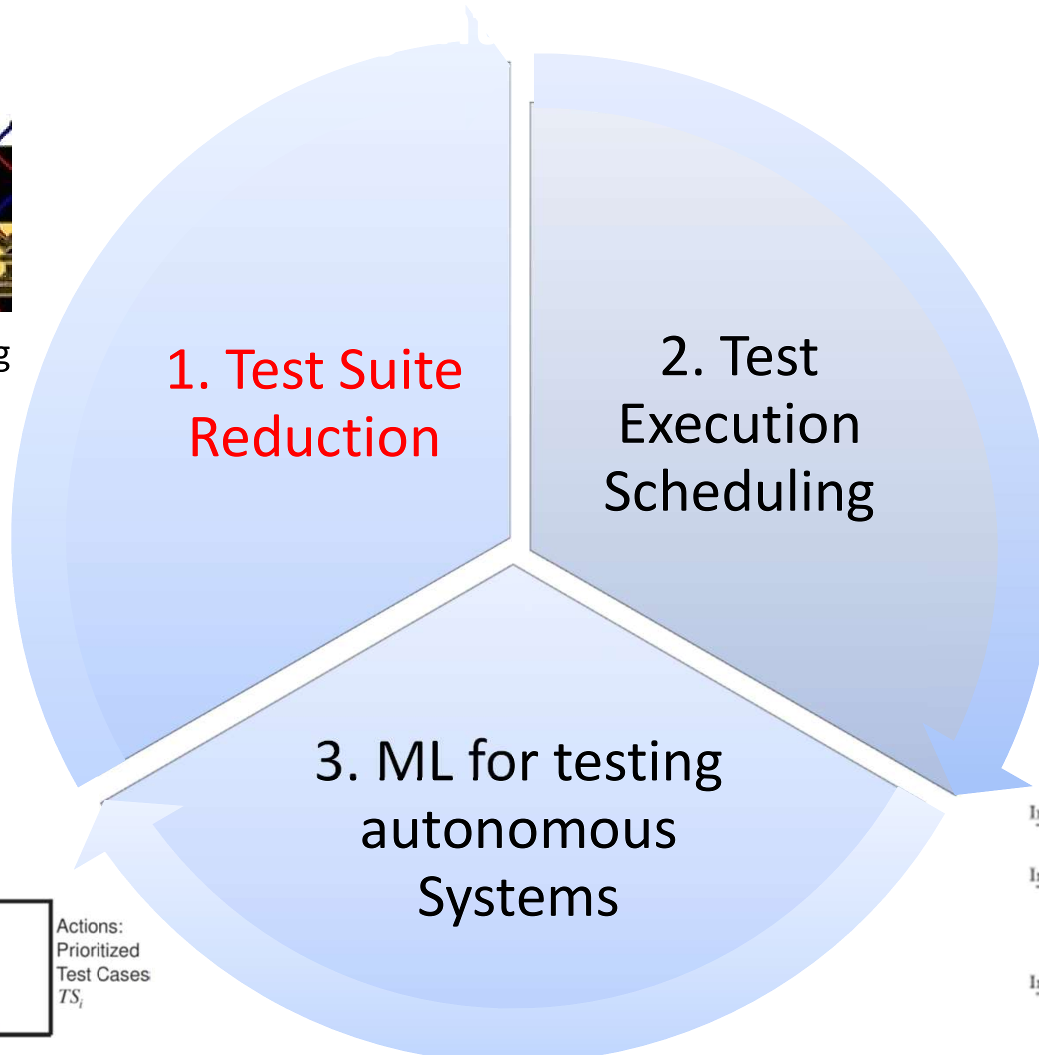
An Ideal Cycle of Continuous Integration and its Timing Challenges

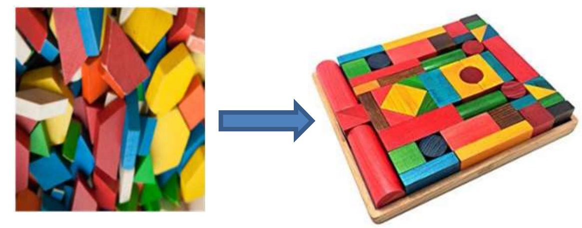



1. Test Suite Reduction
2. Test Execution Scheduling
3. ML for testing autonomous Systems
Constraint Programming
Constraint-based Scheduling
Agent
States:
Test Suite
T_i
reward
r_i
Actions:
Prioritized
Test Cases
TS_i
r_{i+1}
T_{i+1}
Environment:
CI Cycle
Reinforcement Learning
Input layer
i
Hidden layers
h_1
h_2
h_n
Output layer
o
Input 1
Input 2
Input n
Output 1
Output n
Artificial Neural Networks

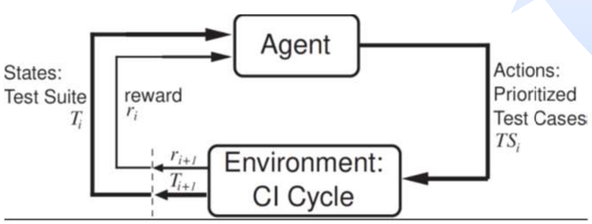


Constraint Programming

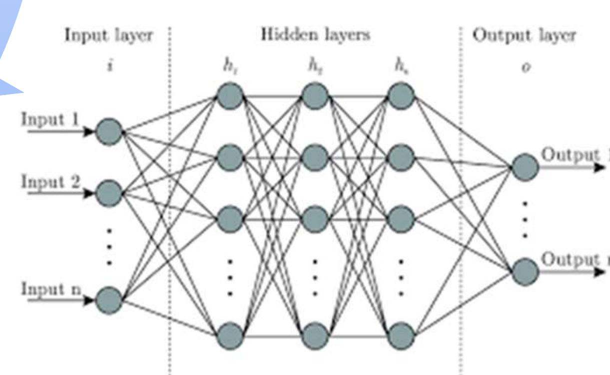




Constraint-based Scheduling



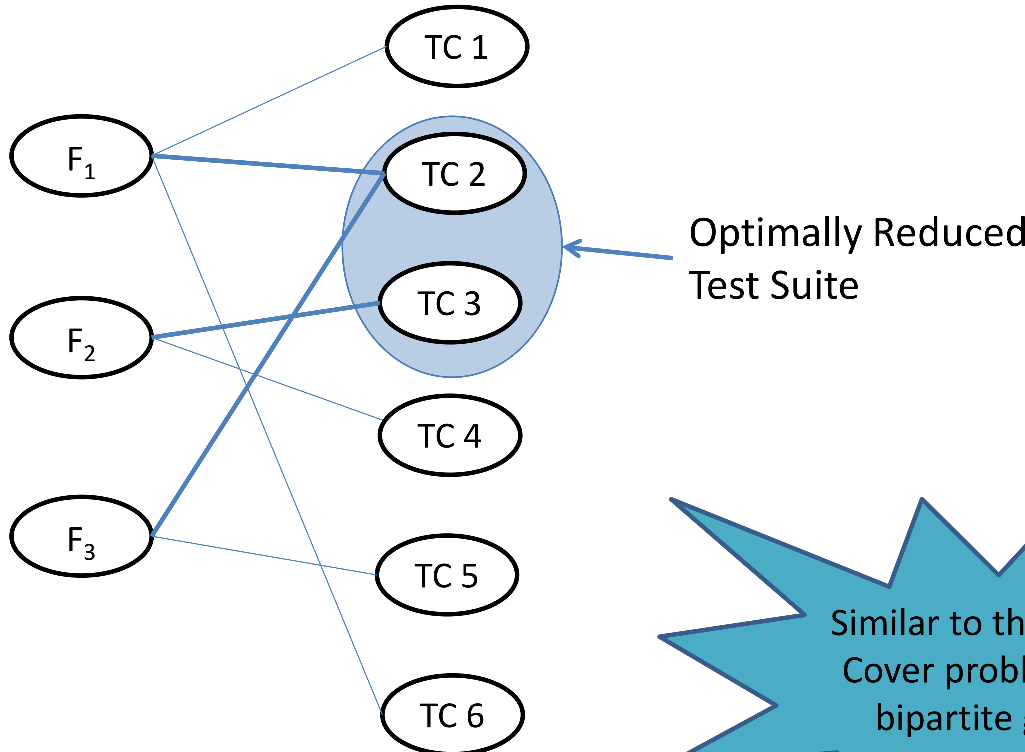
Reinforcement Learning



Artificial Neural Networks

Optimal Test Suite Reduction

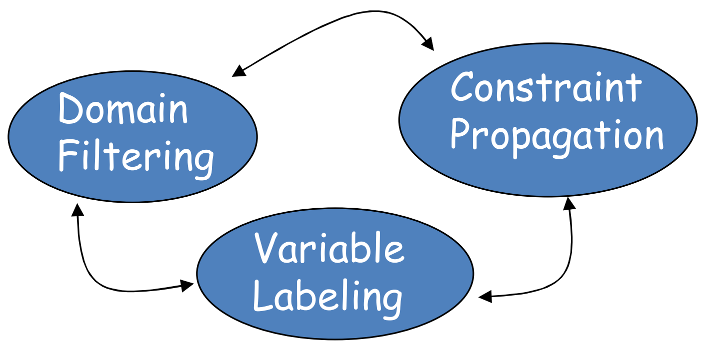
F_i: Requirements
TC: Test Cases



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 $N = card(\{V_i\}_{i \; in \; 1..k})$

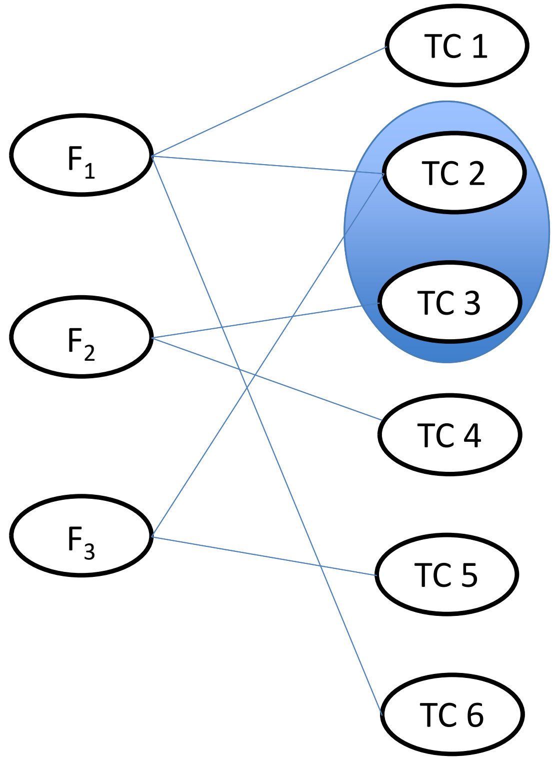
nvalue(N, [3, 1, 3]) entails N = 2

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

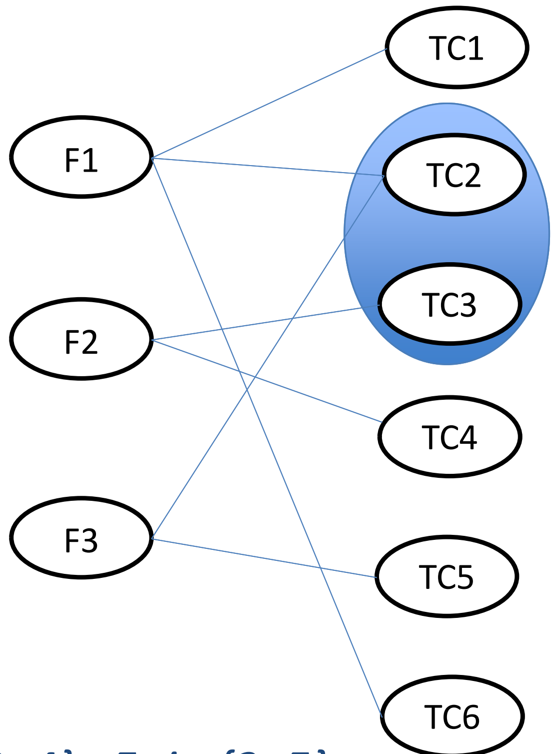
d = $[d_1, \ldots., d_k]$ is a vector of k values

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

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

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

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)

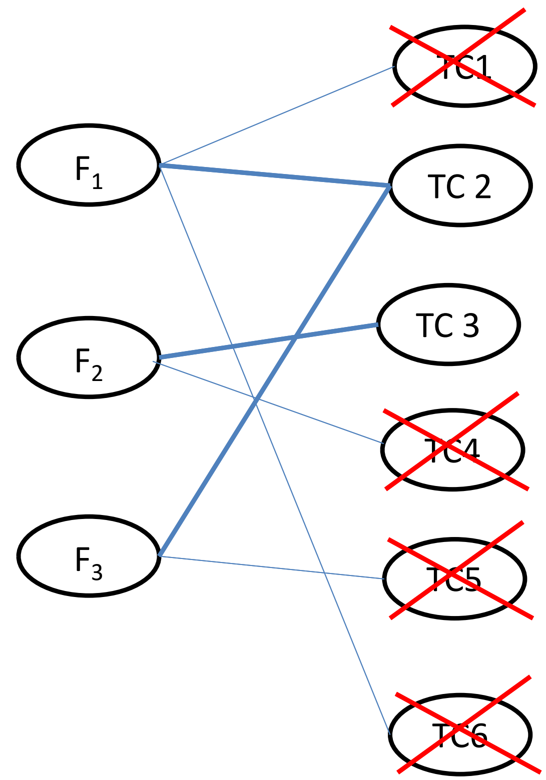
Model pre-processing

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

F_3 is covered → withdraw TC_5

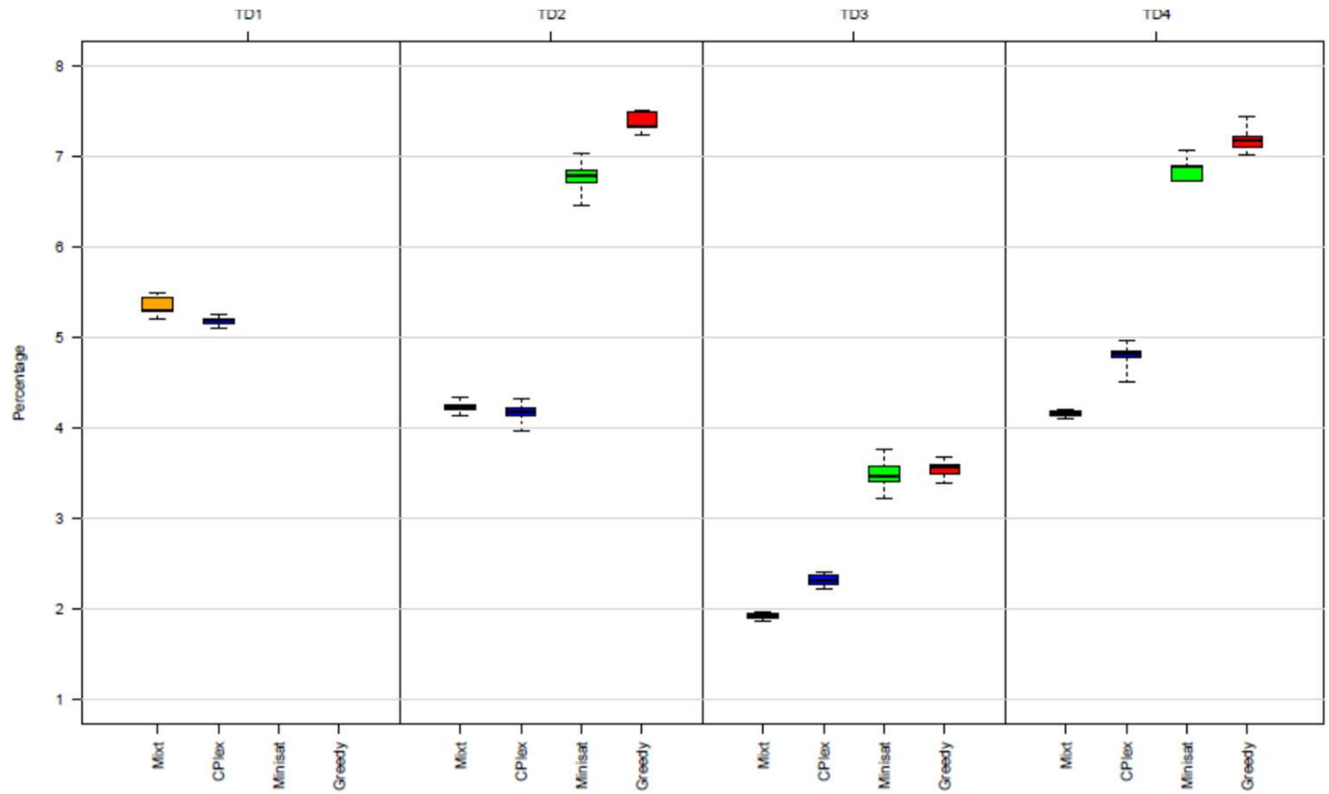
F_2 in {3,4} → e.g., $F_2 = 3$, withdraw TC_4

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



Comparison with CPLEX, MiniSAT, Greedy (uniform costs)

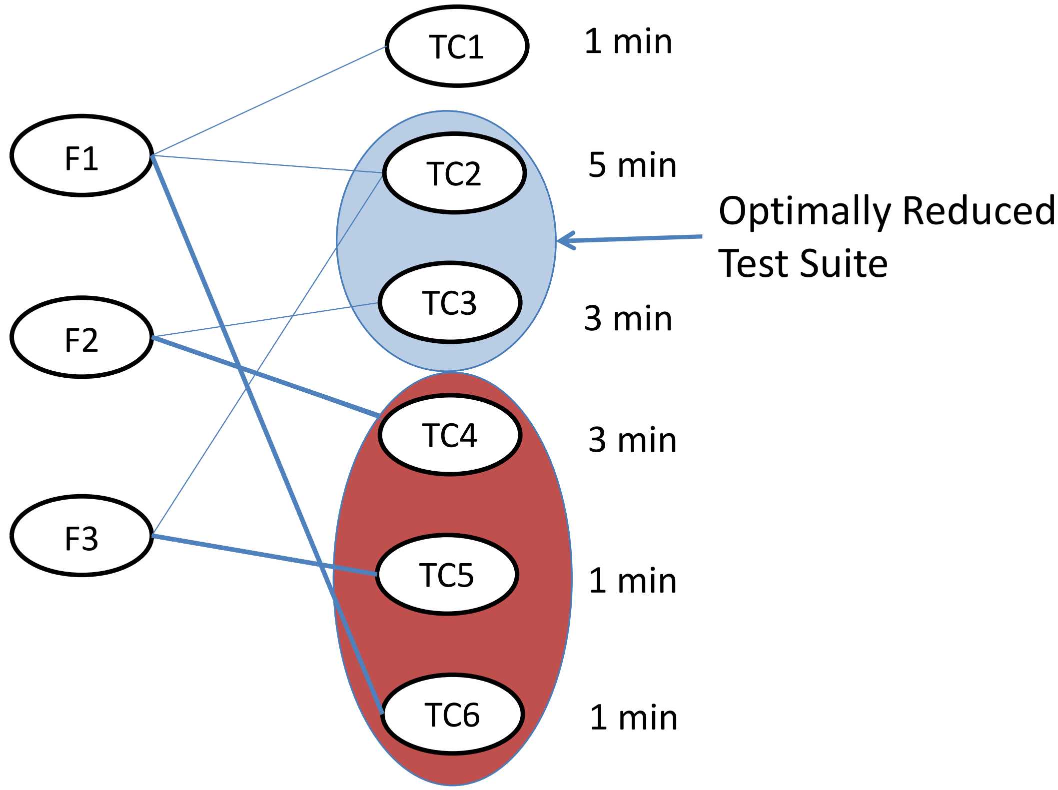
(Reduced Test Suite percentage in 60 sec)



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

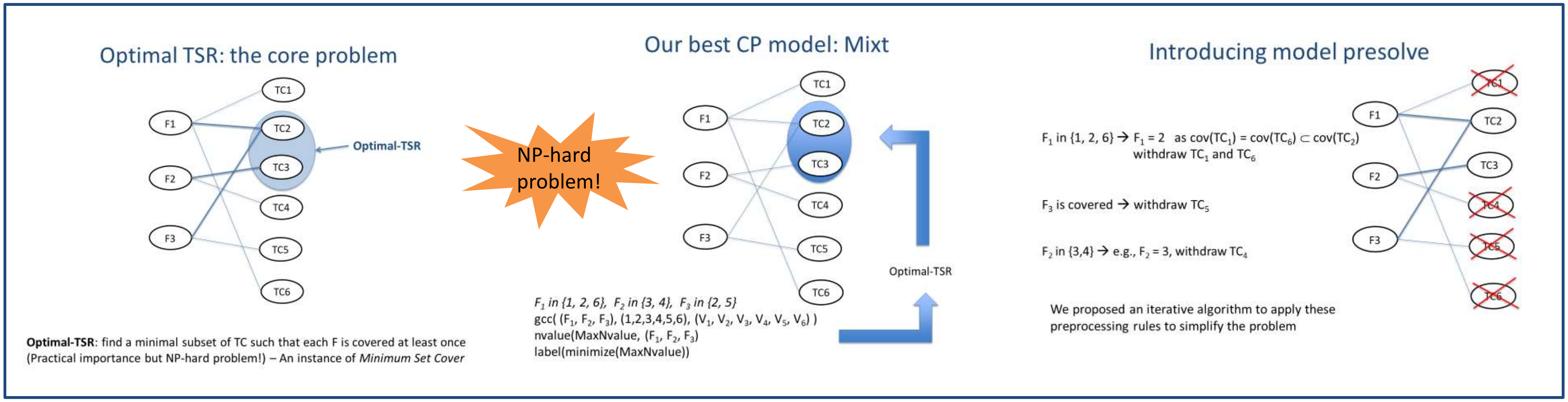
Other criteria to minimize

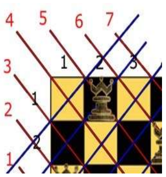
Requirement coverage is always a prerequiste



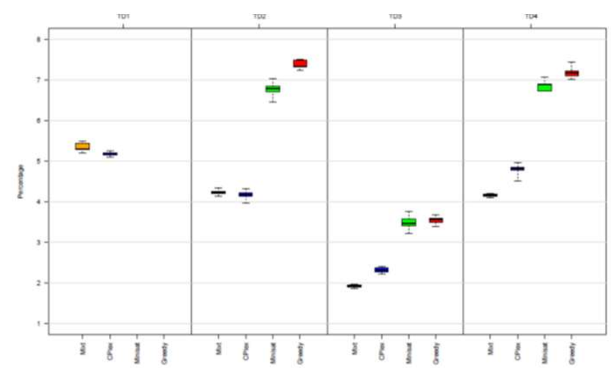
Execution time!

Time-bounded Test Suite Reduction with Constraint Programming (CP)



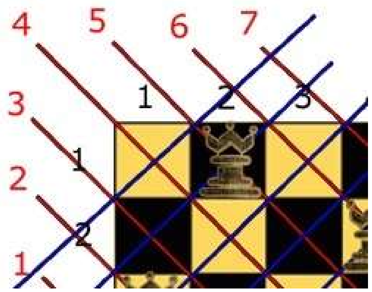


CP with **global constraints** (nvalue, gcc) and search heuristic and **presolve**
Time-contract solving of the **multi-criteria optimisation** problem

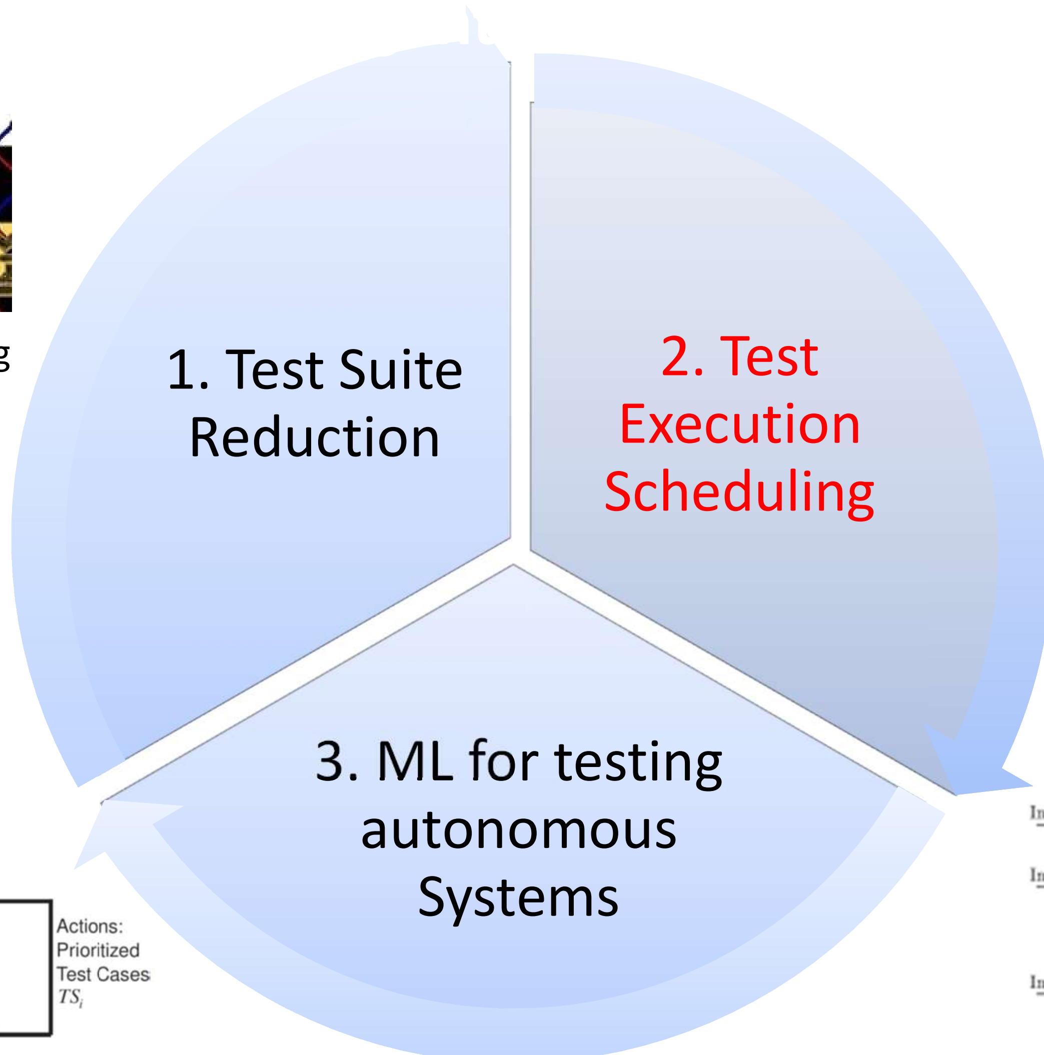


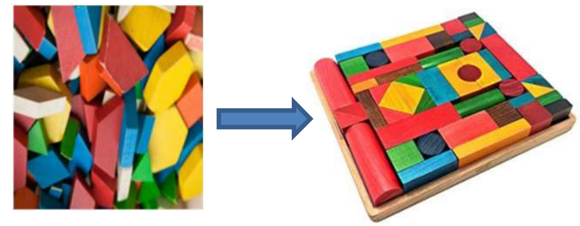
	TD1	TD2	TD3	TD4
Requirements	1000	1000	1000	2000
Test cases	2000	5000	5000	5000
Density	20	7	20	20

*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).

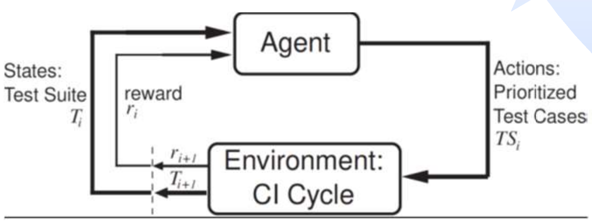


Constraint Programming

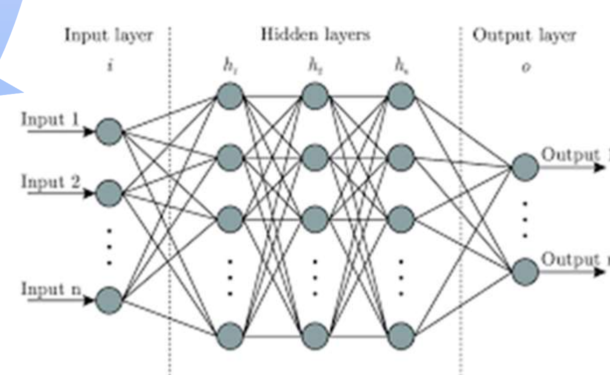




Constraint-based Scheduling

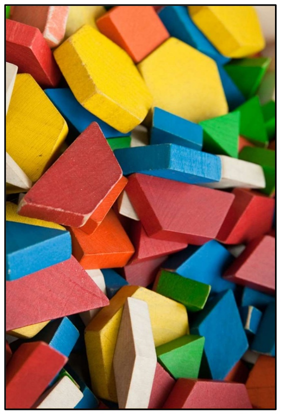


Reinforcement Learning



Artificial Neural Networks

Constraint-Based Scheduling

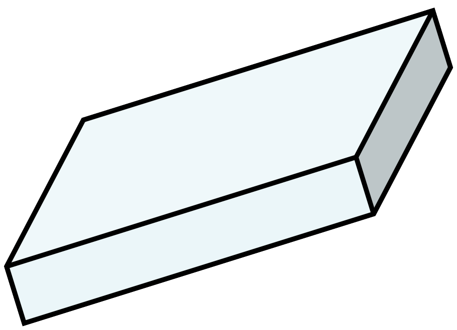


Tasks
with distinct characteristics

Schedule

<ins>Assignment of **Tasks to Agents** such that:</ins>

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



Agents
with limited time or resources capacity

<ins>Goal:</ins>
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: $T \rightarrow N$ *estimated duration*
g: $T \rightarrow 2^G$ *usage of global resources*
f: $T \rightarrow 2^M$ *possible machines*

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

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

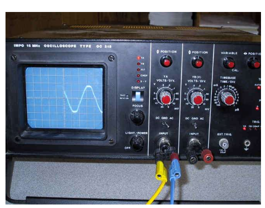
In practice, global optimality is desired but not mandatory, it's more important to control T_S w.r.t T_E
→ Time-contract global optimization

A simple example

Test	Duration	Executable on	Use of global resource
	d	f	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 $$\sum_{\substack{i=1 \\ t_i \leq t \leq t_i + d_i}}^{N} r_i \leq m$$

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$(MaxSpan, (E_1, ..., E_{10})),$
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:

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

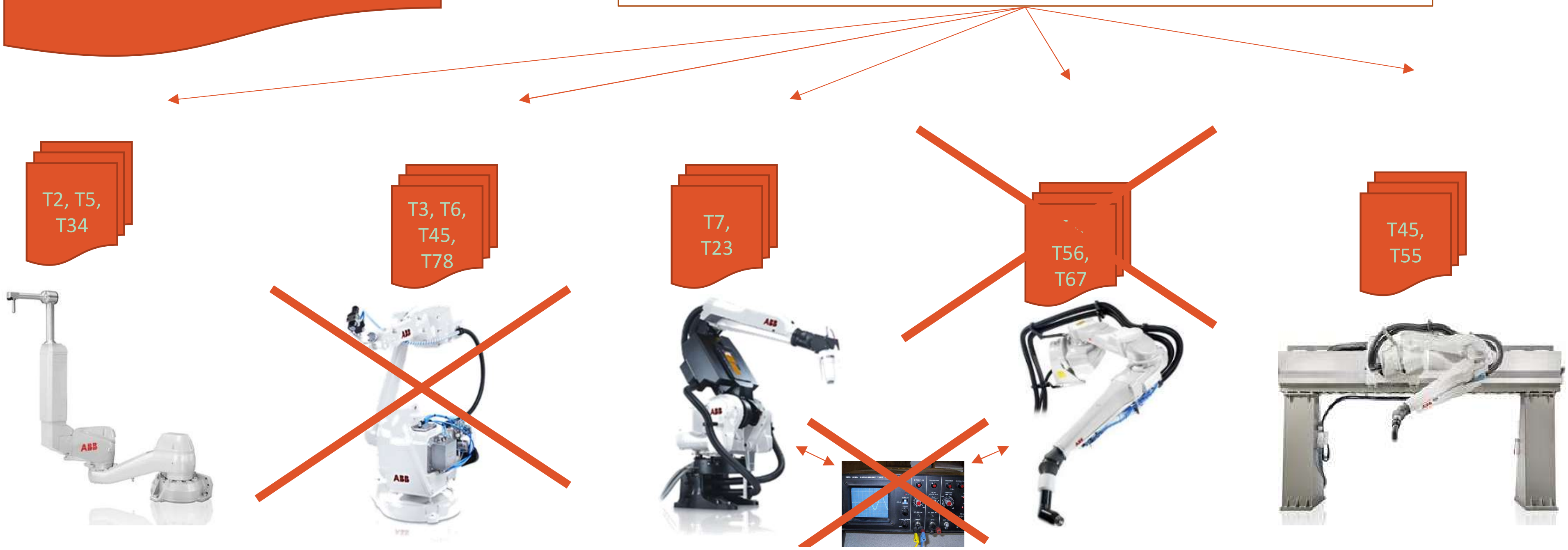
Limitations of this model

- 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

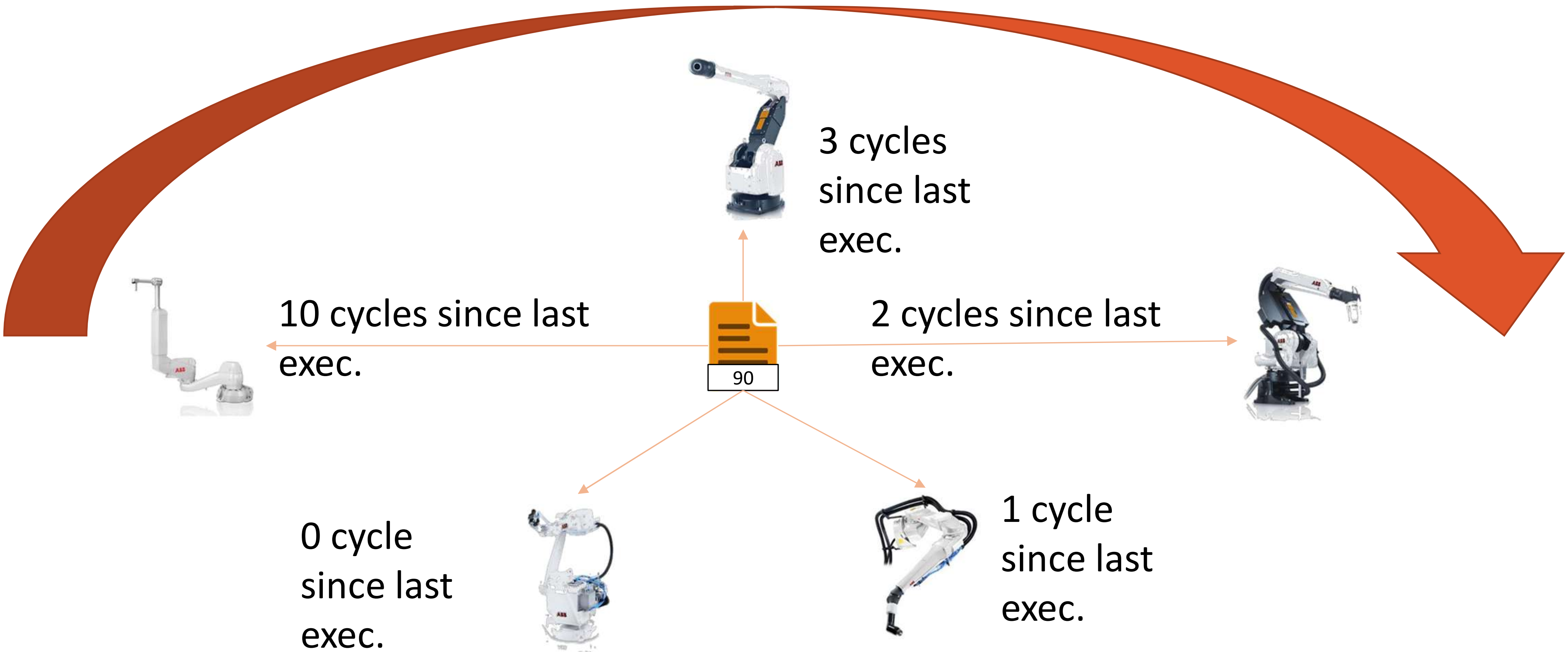
A New Approach Based on Priority and Affinity

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



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

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

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

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

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) $v_{ij} \triangleq p_{ij}^{\alpha} \cdot a_{ij}^{\beta}$

with

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

k : Index of the current cycle

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

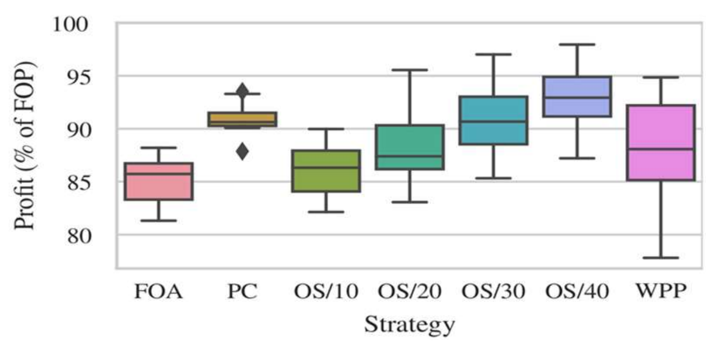
$\mathcal{T}^k$: A set of integers j labeling n tasks

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

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

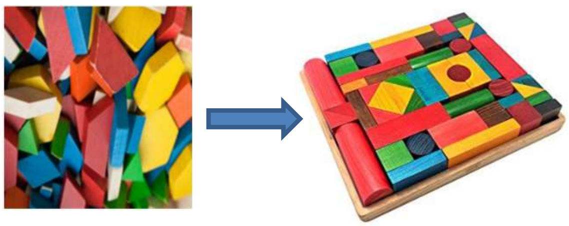


Agents	20	20	20	30	
Tasks	750	1500	3000	3000	Total
FOA	15 (24.4)	6 (15.7)	3 (9.5)	3 (8.5)	27 (14.5)
OS/10	14 (22.2)	6 (15.5)	**3 (9.4)**	**3 (8.4)**	26 (13.9)
OS/20	9 (18.6)	6 (15.3)	3 (9.2)	3 (8.3)	21 (12.9)
OS/30	7 (16.9)	5 (14.3)	3 (9.1)	3 (8.1)	18 (12.1)
OS/40	7 (16.2)	4 (13.1)	3 (8.9)	3 (7.9)	17 (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 Time-aware 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



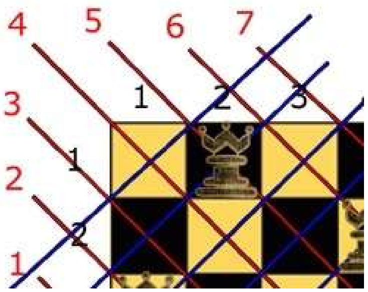
Constraint-based Scheduling

CP with **global constraints (cumulative, binpacking)** and rotational diversity can solve the test execution scheduling problem

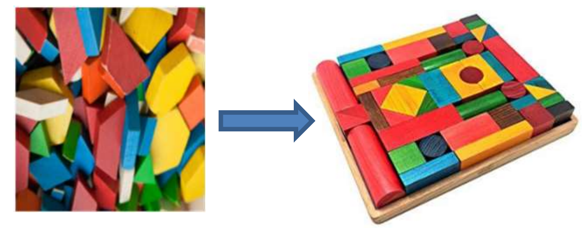
"SWMOD deployed at ABB Robotics and used every day to schedule tests throughout several ABB centers in the world (Norway, Sweden, India, China)"

Morten Mossige, Arnaud Gotlieb, Helge Spieker, Hein Meling, and Mats Carlsson. ***Time-aware test case execution scheduling for cyber-physical systems****. In Proc. of Principles of Constraint Programming (CP'17), Aug. 2017.*

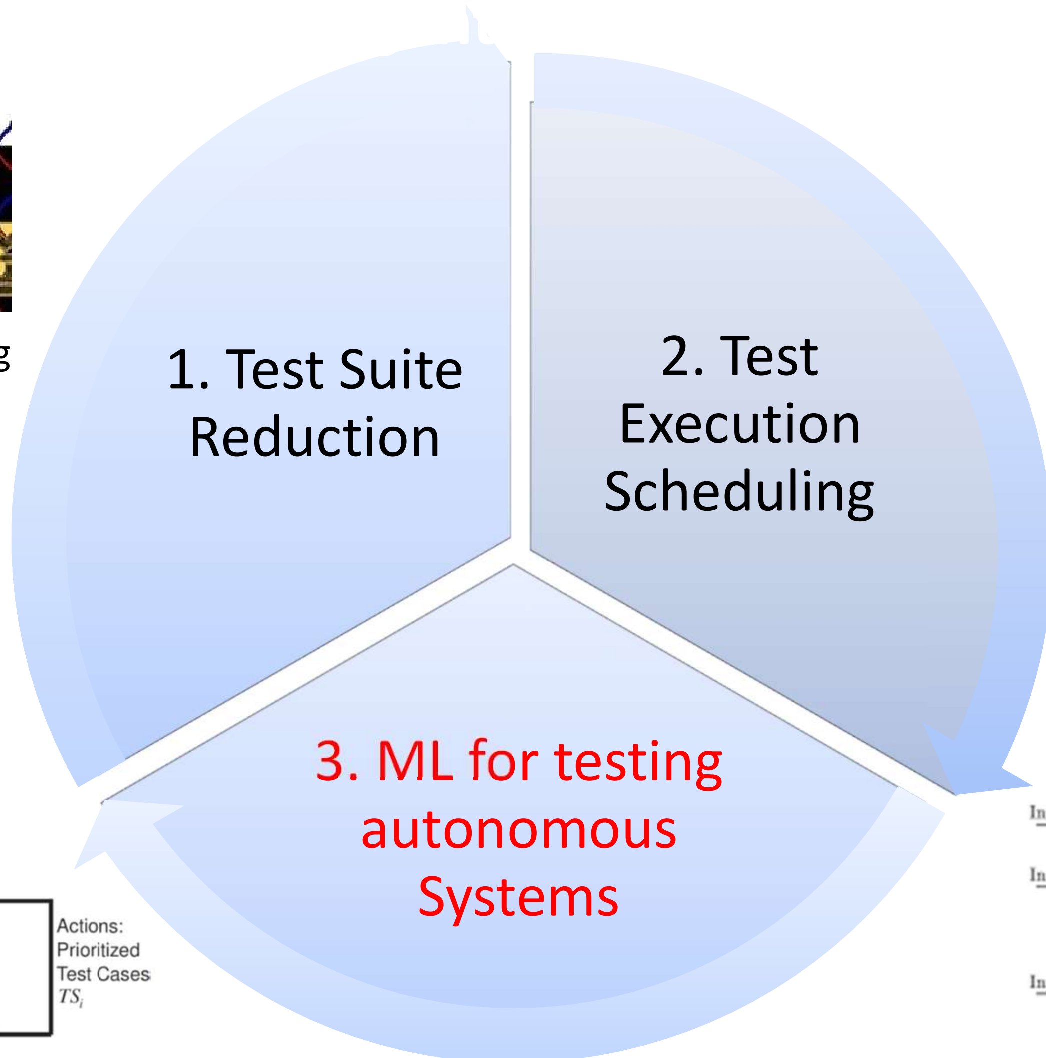
H. Spieker, A. Gotlieb and M. Mossige - ***Rotational Diversity in Multi-Cycle Assignment Problems*** *- In Proc. of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19). Feb. 2019.*

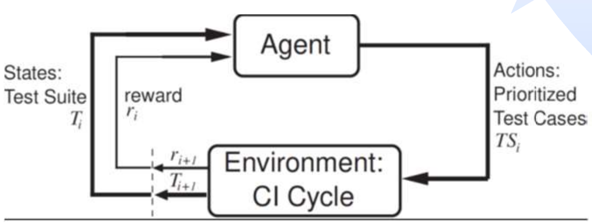


Constraint Programming

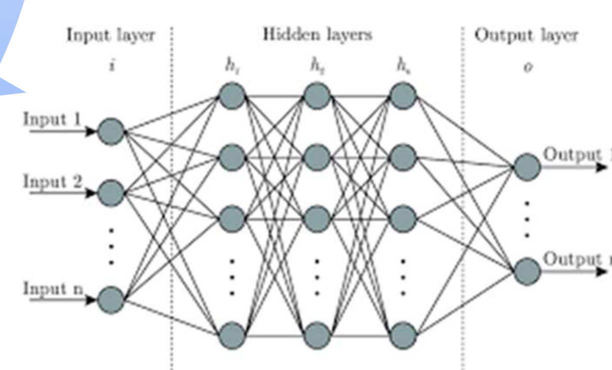


Constraint-based Scheduling





Reinforcement Learning



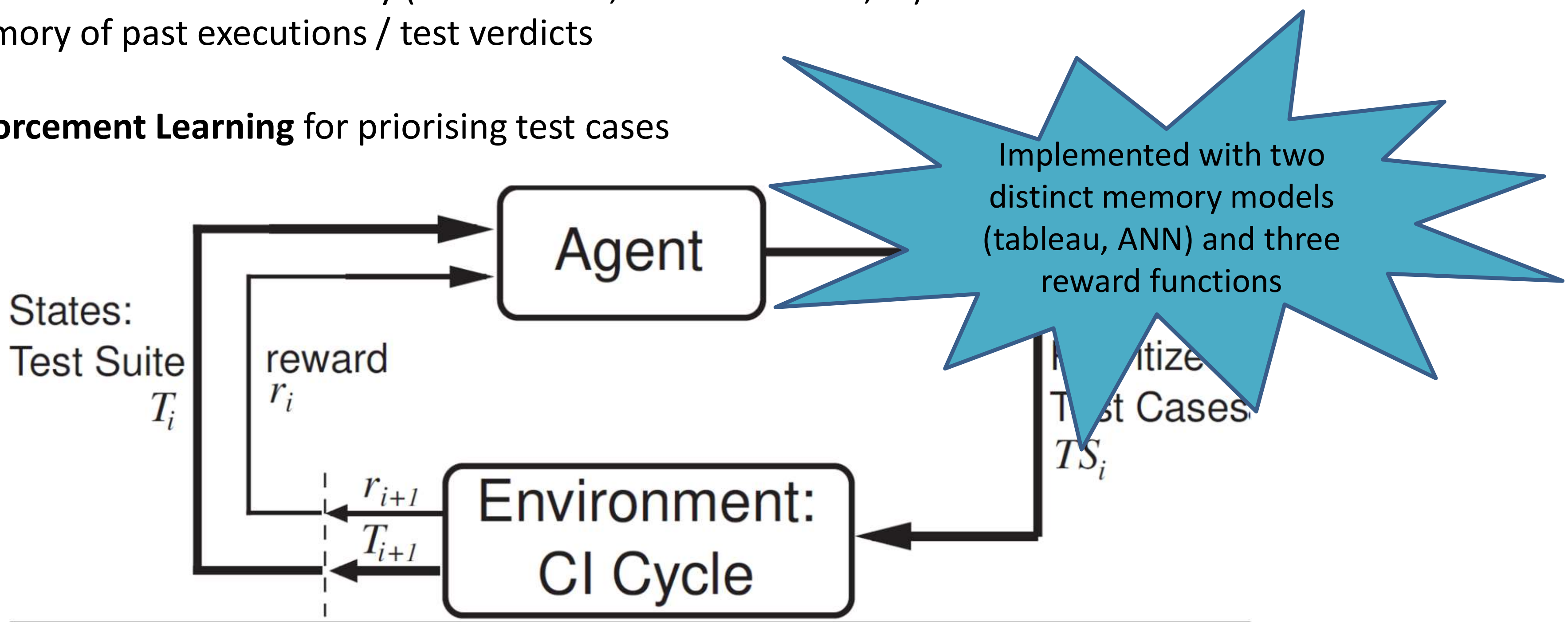
Artificial Neural Networks

Test Prioritization: Learning from previous test runs

Motivation:
Adapting priorities to the most interesting test cases based on past test verdicts (from previous CI cycles)

- Considering test case meta-data only (test verdicts, execution time, ...)
- Limited memory of past executions / test verdicts

- Using **Reinforcement Learning** for priorising test cases



Reward Functions and Experimental Evaluation

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

Reward Function 1. Failure Count Reward

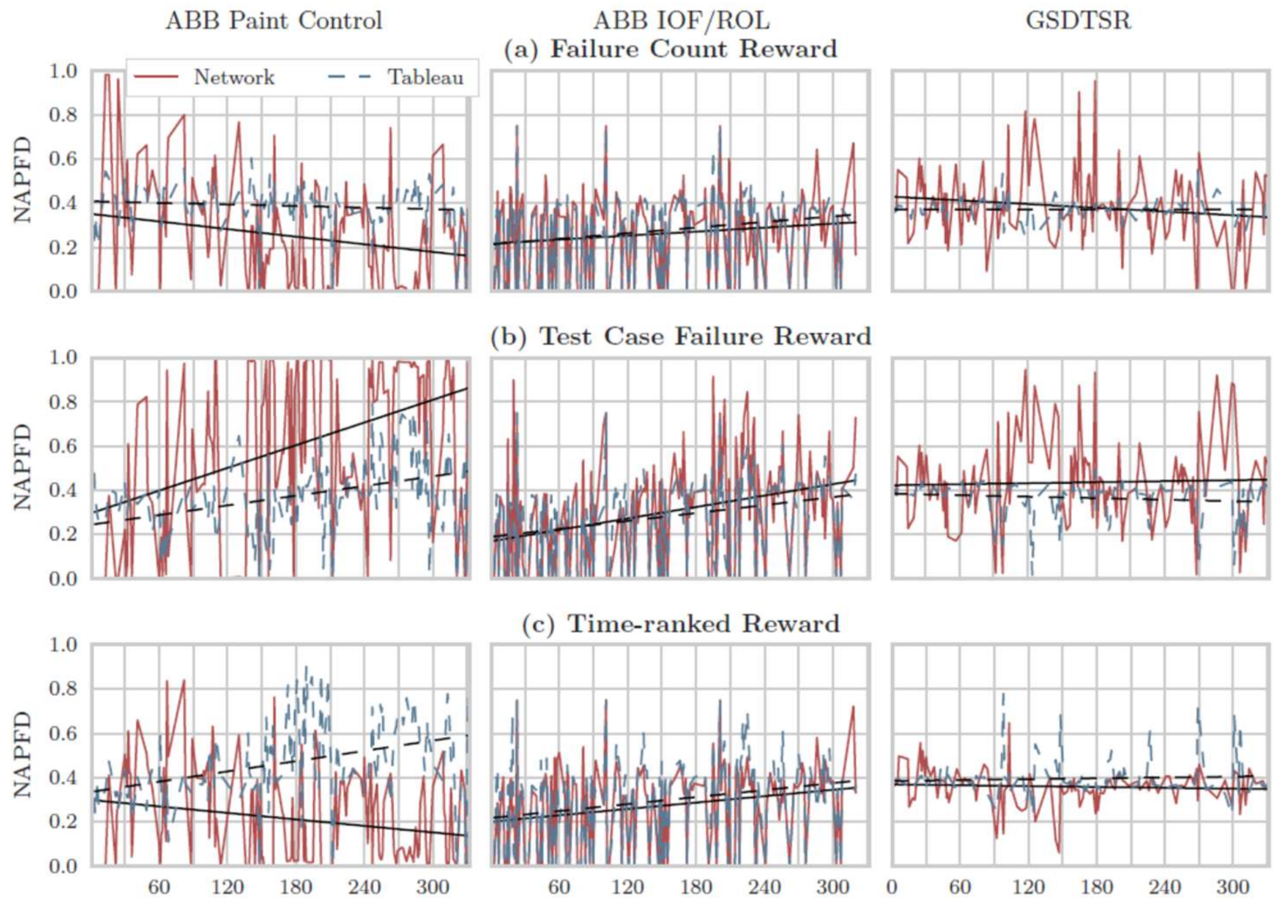
$$reward_i^{fail}(t) = |\mathcal{TS}_i^{fail}| \qquad (\forall t \in \mathcal{T}_i)$$

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

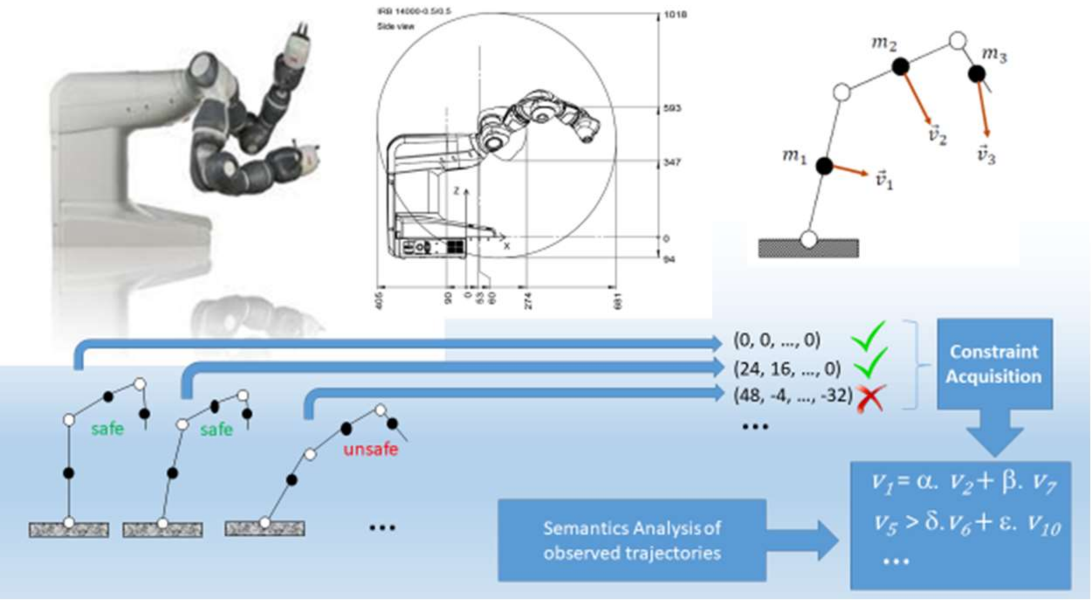
$$reward_i^{time}(t) = |\mathcal{TS}_i^{fail}| - t.verdict_i \times \sum_{\substack{t_k \in \mathcal{TS}_i^{fail} \wedge \\ rank(t) < rank(t_k)}} 1$$



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

Constraint Acquisition for Testing Collaborative Robots

Motivation: Learning how to safely interact with a collaborative robot



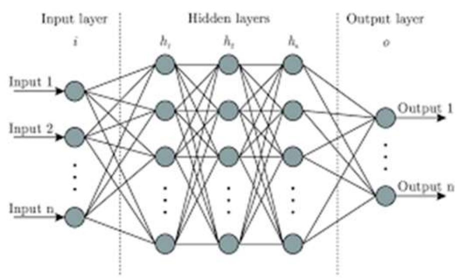
Instance	Q$^+$= $size(L)$	LQCN		GEQCA	
		Q$^-$	eF	Q$^-$	eF
Ins_10_8	45	228	47%	**112**	**26%**
Ins_10_25	145	**177**	**55%**	201	59%
Ins_10_46	268	189	78%	**106**	**63%**
Ins_25_8	300	2,034	60%	**364**	**17%**
Ins_25_28	1,100	**695**	**46%**	748	47%
Ins_25_58	2,253	**703**	**76%**	794	78%
Ins_50_8	1,225	7,308	53%	**748**	**12%**
Ins_50_48	7,680	**1,500**	**58%**	1,763	59%
Ins_50_52	8,316	**1,689**	**63%**	2,051	65%
Ins_100_8	4,950	23,754	44%	**1,634**	**10%**
Ins_100_32	20,395	24,276	69%	**1,653**	**34%**
Ins_100_37	23,942	24,974	76%	**1,629**	**40%**

Using **active constraint acquisition** to learn temporal constraints (task precedence constraints) and deploy it to generate test schedules for collaborative robots

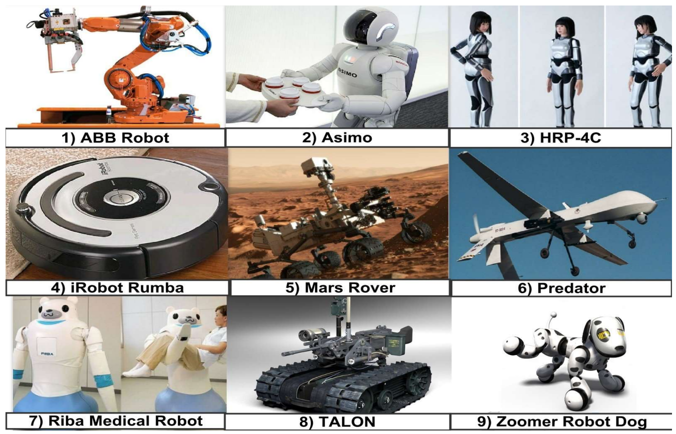
B. Belaid, N. Belmecherie, A. Gotlieb, N. Lazaar, H. Spieker – **GEQCA: Generic Qualitative Constraint Acquisition –** Under Review

Adaptive Metamorphic Testing

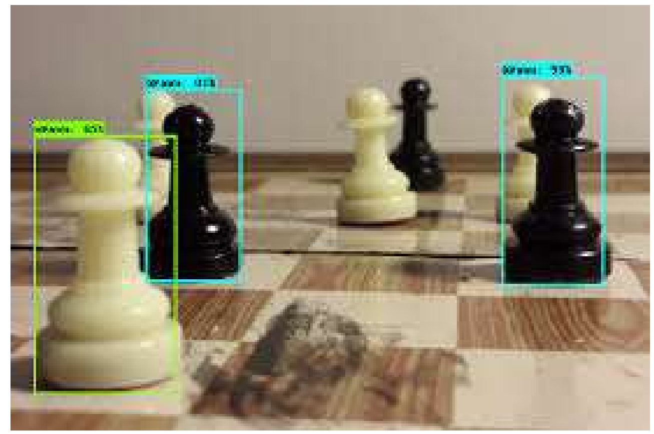
Motivation: Learning which *Metamorphic Relation* works best to test vision-based systems



Artificial Neural Networks

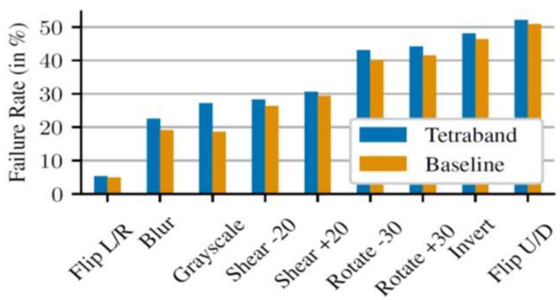


TensorFlow.org - Image classification – dataset of 10,000 images



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

Using Contextual Bandits (Reinforcement Learning) to learn how to select metamorphic relations
→ Adaptive Metamorphic Testing



(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.

*H. Spieker, A. Gotlieb – **Adaptive Metamorphic Testing with Contextual Bandits** – Journal of Systems and Software. 165: 110574 (2020)*

A. Gotlieb, D. Marijan and H. Spieker**: Testing Industrial Robotic Systems: A New Battlefield!**

In Software Engineering for Robotics, 465p. Edited by A. Cavalcanti, B. Dongol, R. Hierons, J. Timmis, J. Woodcock ed. Springer Nature, 2021.

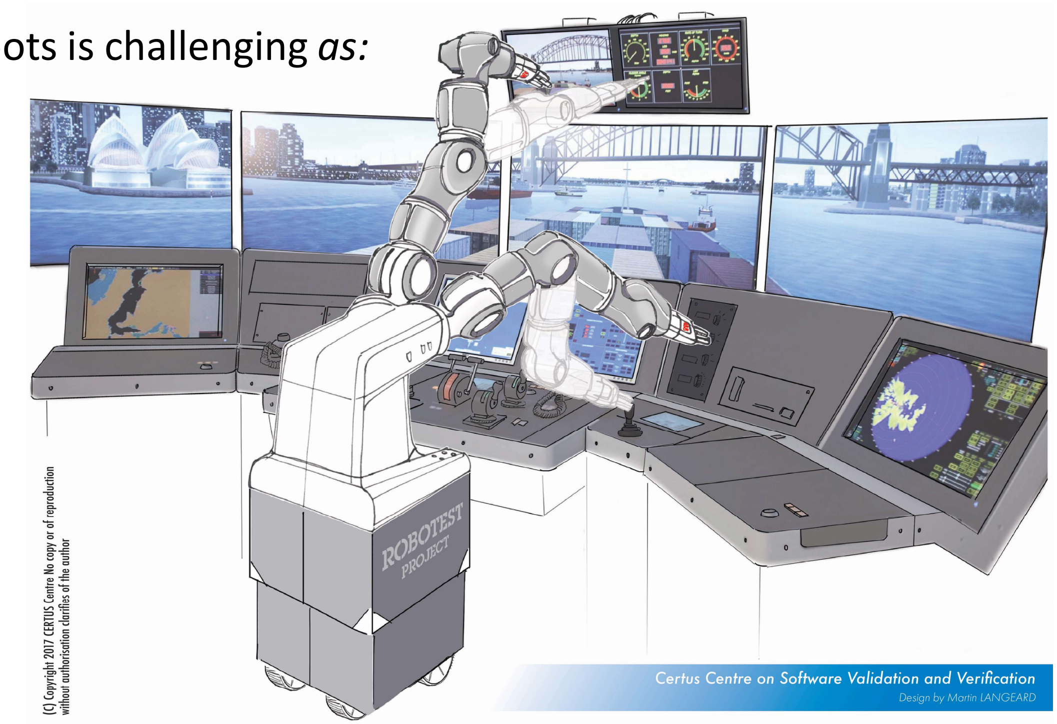
Take Away Message

- *Testing autonomous systems* brings new interesting challenges for software V&V research

- **Some AI techniques such as Constraint Programming (CP) and global constraints** are already very successful in test case generation, test suite reduction and test execution scheduling

- Testing autonomous systems such as collaborative robots is challenging *as:*
  - **Expected behaviours** cannot be specified in advance
  - **Interactions with humans** involve more safety issues

We are currently exploring the usage of **Reinforcement Learning** and **Active Learning** methods for testing collaborative robots



Thanks to Helge Spieker, Dusica Marijan, M. Bachir Belaid, M. Kumar Ahuja, Aizaz Sharif, Pierre Bernabé.

The VIAS Dept. at Simula

3 Permanent Research Scientists, 3 PhD Students, 2 Postdoc

Partners: ABB Robotics, CISCO Systems, Kongsberg Maritime, Cancer Registry, Tax/Toll Dept., etc.

Several National and European Projects (RCN-FRIPRO, RCN-IKTPLUSS, H2020, etc.)

1st Simula-Inria Associate Team (2021)

simula

Inria

The Research Council of Norway

THE FRAMEWORK PROGRAMME FOR RESEARCH AND INNOVATION HORIZON 2020

AI4EU