Combining Genetic Algorithms and Constraint Programming to Support Stress Testing of Task Deadlines

Stefano Di Alesio [1,2]

Lionel Briand [2]

Shiva Nejati [2]

Arnaud Gotlieb [1]

ESEC/FSE 2015

Bergamo, 02/09/2015

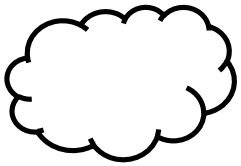
[1] Certus Centre for Software Verification & Validation

[2] securityandtrust.lu

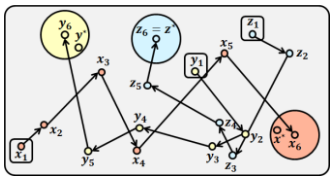
[1] **Certus Centre for Software V&V**
Simula Research Laboratory
Norway

[2] **Interdisciplinary Centre for Reliability, Security and Trust (SnT)**
University of Luxembourg
Luxembourg

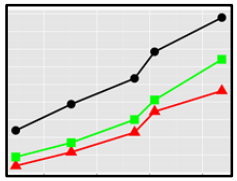
We present GA+CP: a search strategy to identify scenarios likely to violate task deadlines in RTES



Problem Statement: Stress Testing of Task Deadlines in Real Time Embedded Systems (RTES)

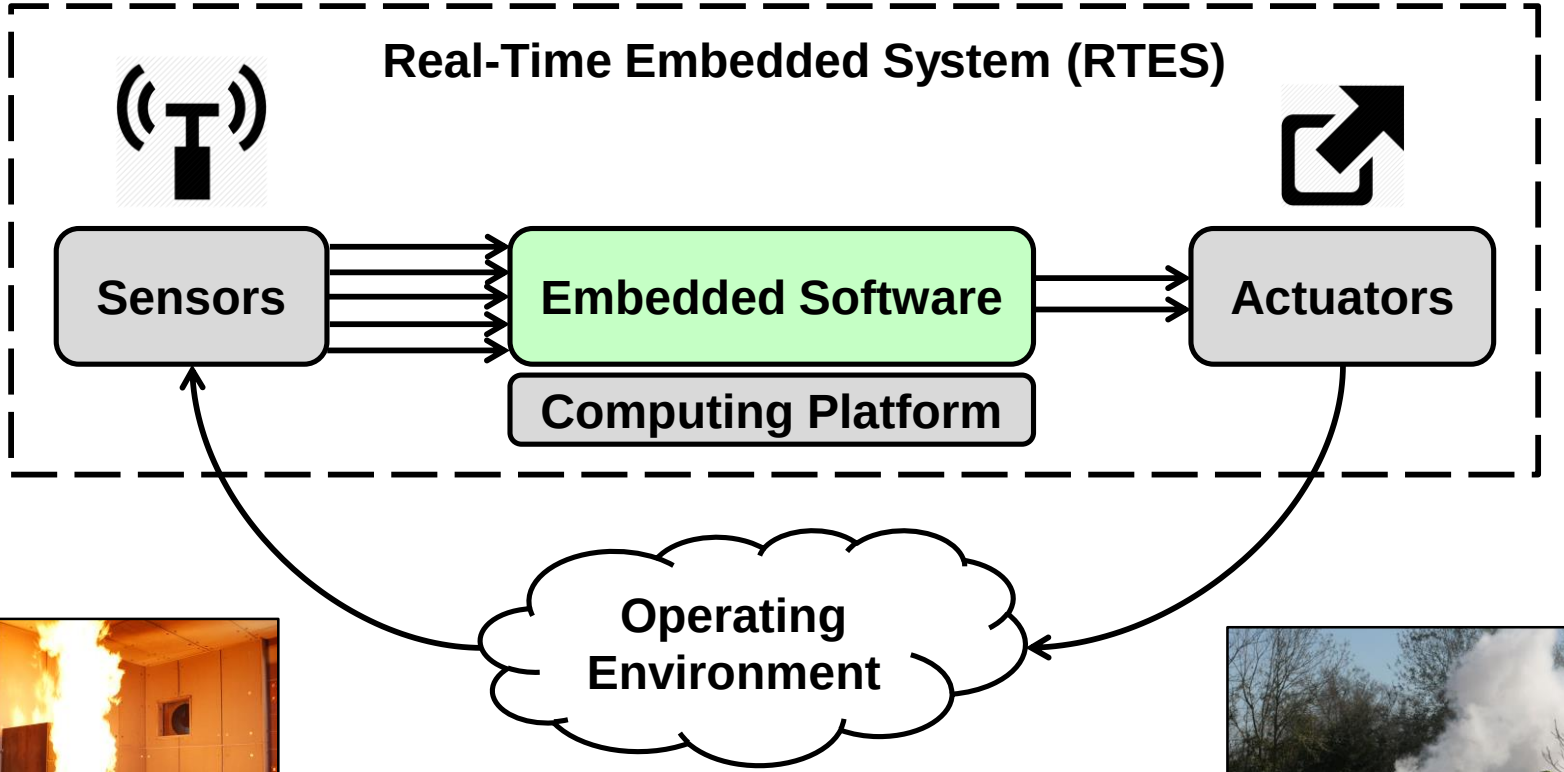


Proposed Solution: Search for worst-case schedules with a combined GA+CP strategy



Key Results: Efficiency, Effectiveness, Diversity and Scalability

Safety-critical RTES have to meet strict Performance Requirements to be deemed safe for operation



Systematic *stress* and performance testing is highly recommended when certifying safety-critical RTES

Stress Testing: ***"Testing in which a system is subjected to […] harsh inputs […] with the intention of breaking it"***

— Boris Beizer

Arrival times for aperiodic tasks

Worst-case scenarios

Table B.6 – Performance testing
(referenced by tables A.5 and A.6)

	Technique/Measure*	Ref	SIL1	SIL2	SIL3	SIL4
1	Avalanche/stress testing	C.5.21	R	R	HR	HR
2	Response timings and memory constraints	C.5.22	HR	HR	HR	HR
3	Performance requirements	C.5.19	HR	HR	HR	HR
*	Appropriate techniques/measures shall be selected according to the safety integrity level.					

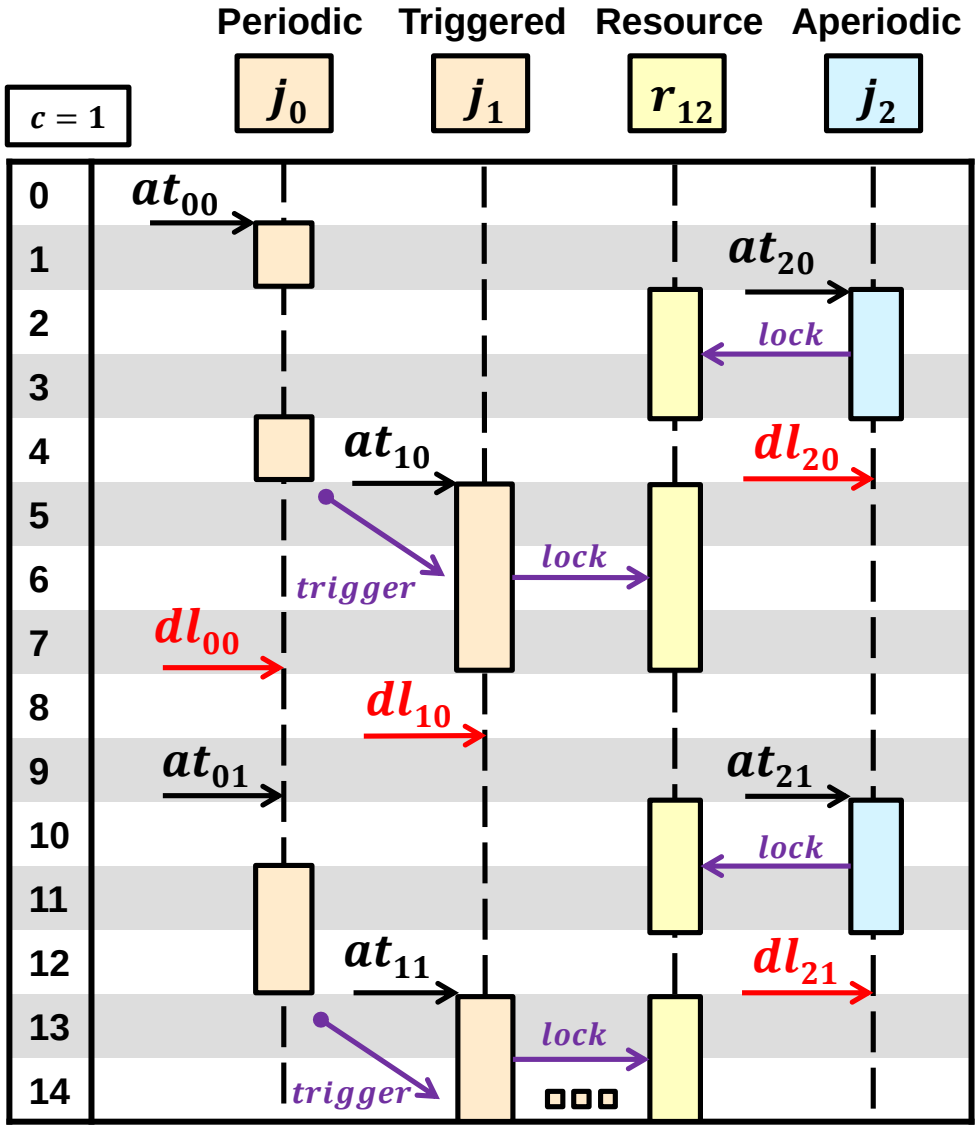
IEC 61508 deems stress testing as *highly recommended* for SIL 3-4

RTES usually have concurrent interdependent tasks executed by a priority-driven preemptive scheduler

Each task has a deadline (i.e., latest finishing time) w.r.t. its arrival time

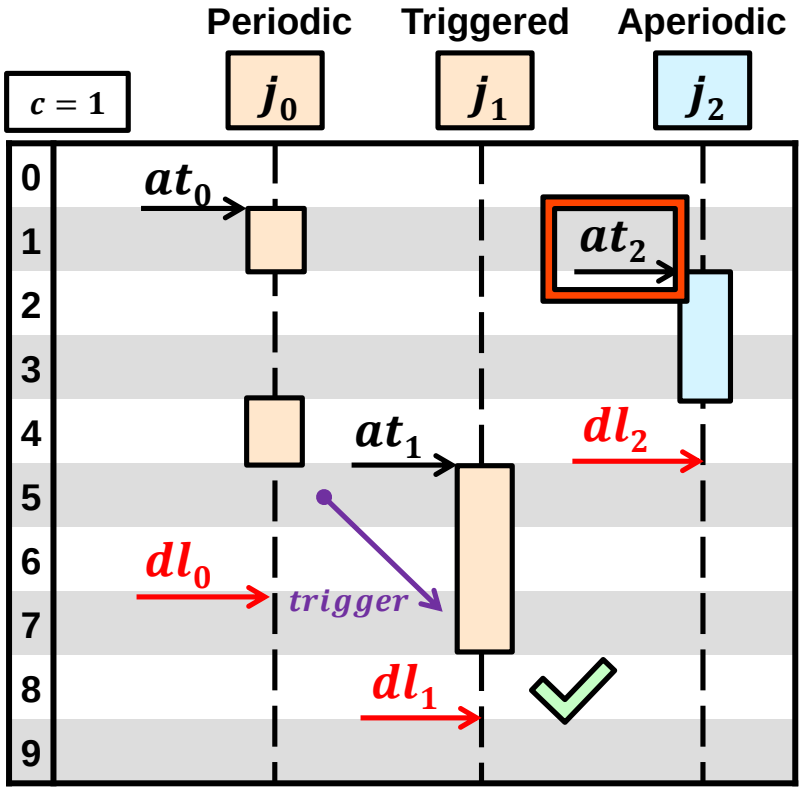
Some task properties depend on the environment, others are design choices

Tasks can trigger other tasks, or share computational resources with them

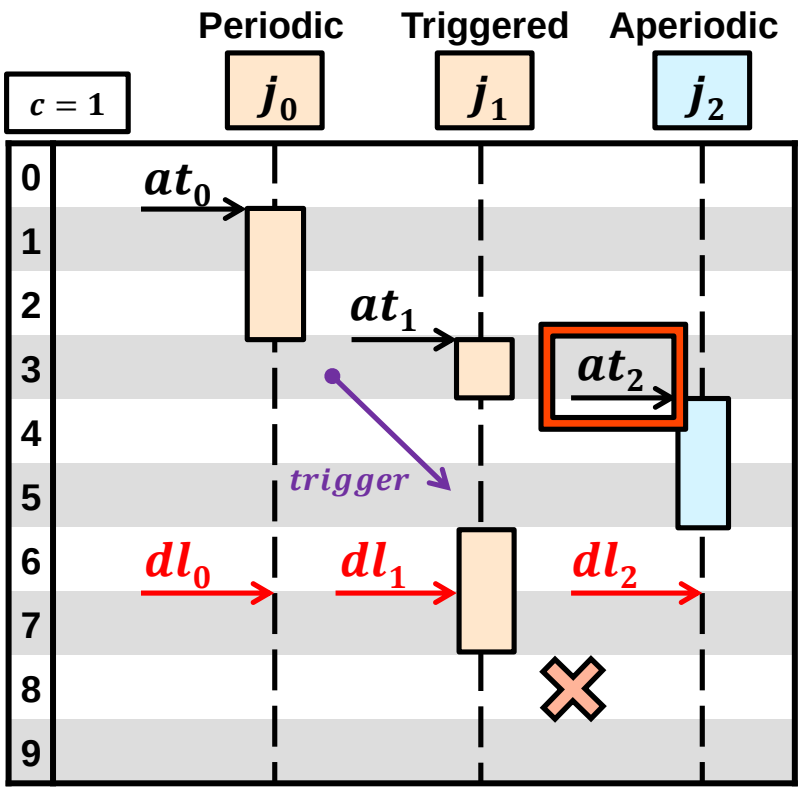


Particular sequences of arrival times may determine scenarios violating task deadlines

j_0, j_1, j_2 arrive at at_0, at_1, at_2 and must finish before dl_0, dl_1, dl_2



j_1 can miss its deadline dl_1 depending on when at_2 occurs!



A sequence of arrival times which is likely to violate a task deadline characterizes a *stress test case*

Several techniques have been used for solving this problem, but each has its own drawbacks

	Formal Verification		Testing		
	Scheduling Theory	**Model Checking**	**Performance Engineering**	**Genetic Algorithms**	**Constraint Programming**
Background	Queuing Theory	Fixed-point Computation	Practice and Tools	Metaheuristics	Artificial Intelligence
Key Features	Theorems	Symbolic Execution	Dynamic Analysis	Randomized Search	Complete Search
Drawbacks	Assumptions	Complex Modeling	Non Systematic	Ineffective [1]	Inefficient [1], Low diversity

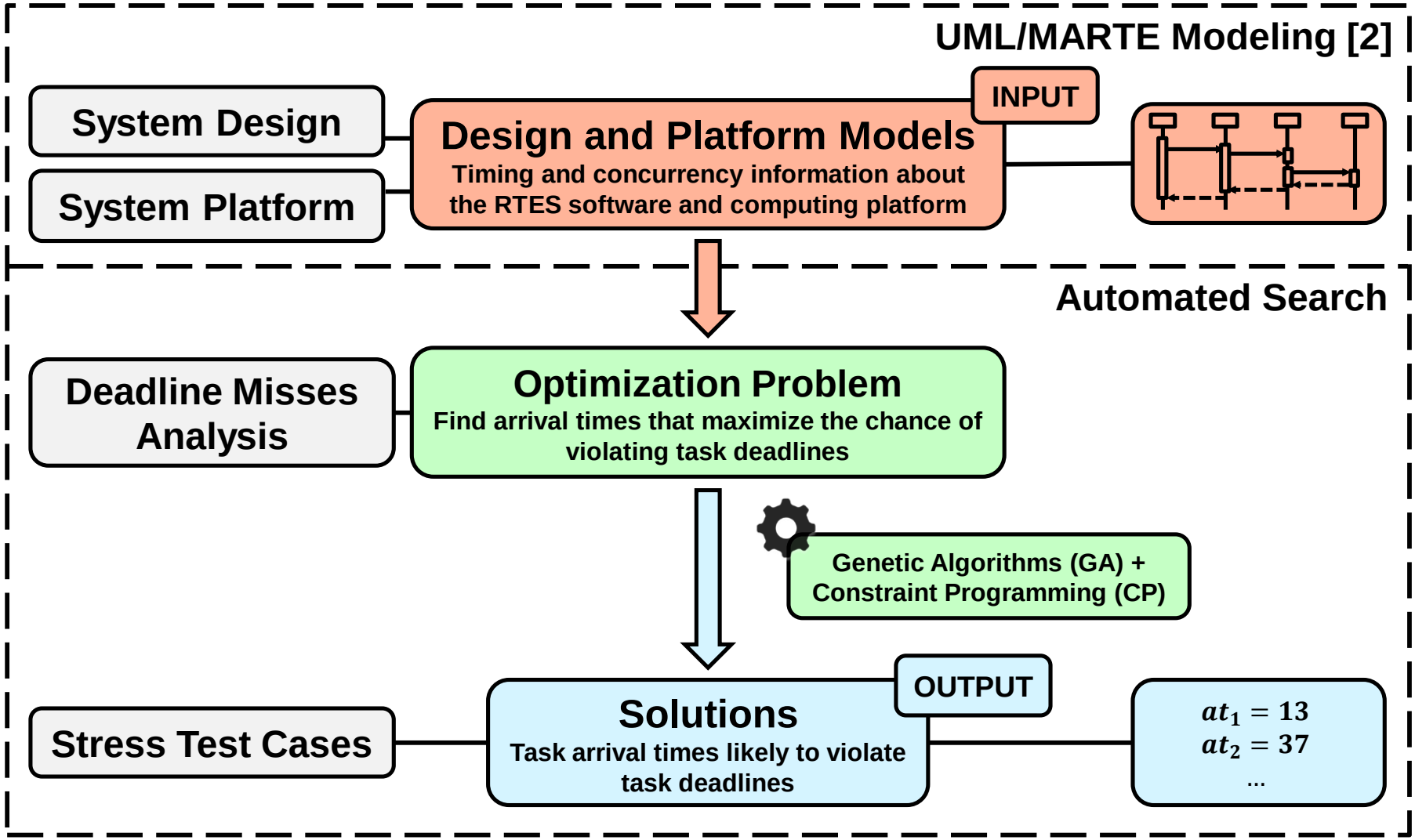
GA is efficient and diverse: *quickly* generates test cases involving *different interactions* between tasks

CP is effective: generates test cases that are *more likely* to violate task deadlines

→ **GA + CP**

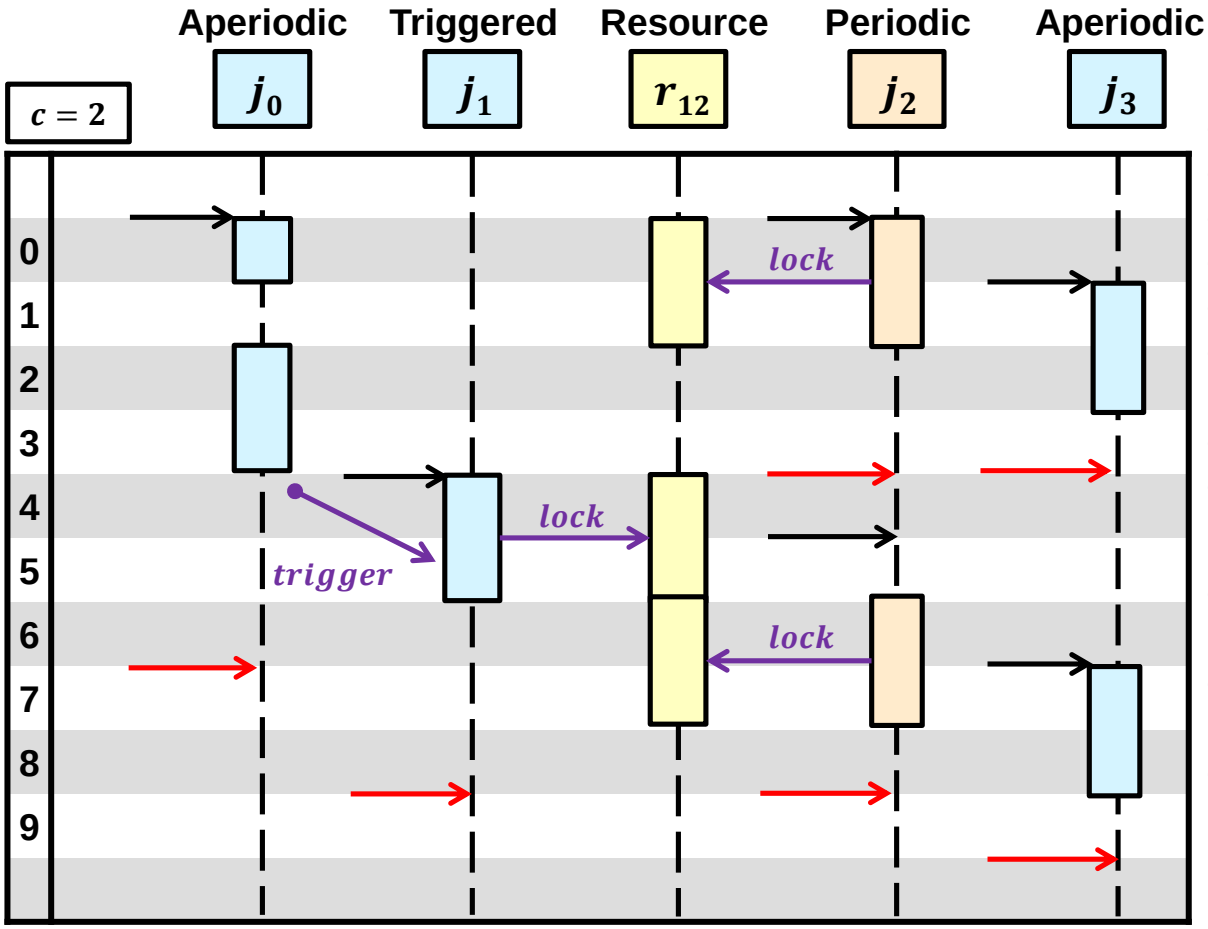
[1] Di Alesio, S., Nejati, S., Briand, L., and Gotlieb, A. (2013). Stress Testing of Task Deadlines: A Constraint Programming Approach. In Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on, pages 158–167. IEEE.

We cast the generation of stress test cases as an Optimization Problem over the task arrival times



[2] Nejati, S., Di Alesio, S., Sabetzadeh, M., Briand, L.: Modeling and Analysis of CPU Usage in Safety-critical Embedded Systems to Support Stress Testing. In: Model Driven Engineering Languages and Systems (MODELS), pp. 759–775. Springer (2012)

Static Properties depend on the RTES design (are known), and express constraints on task execution

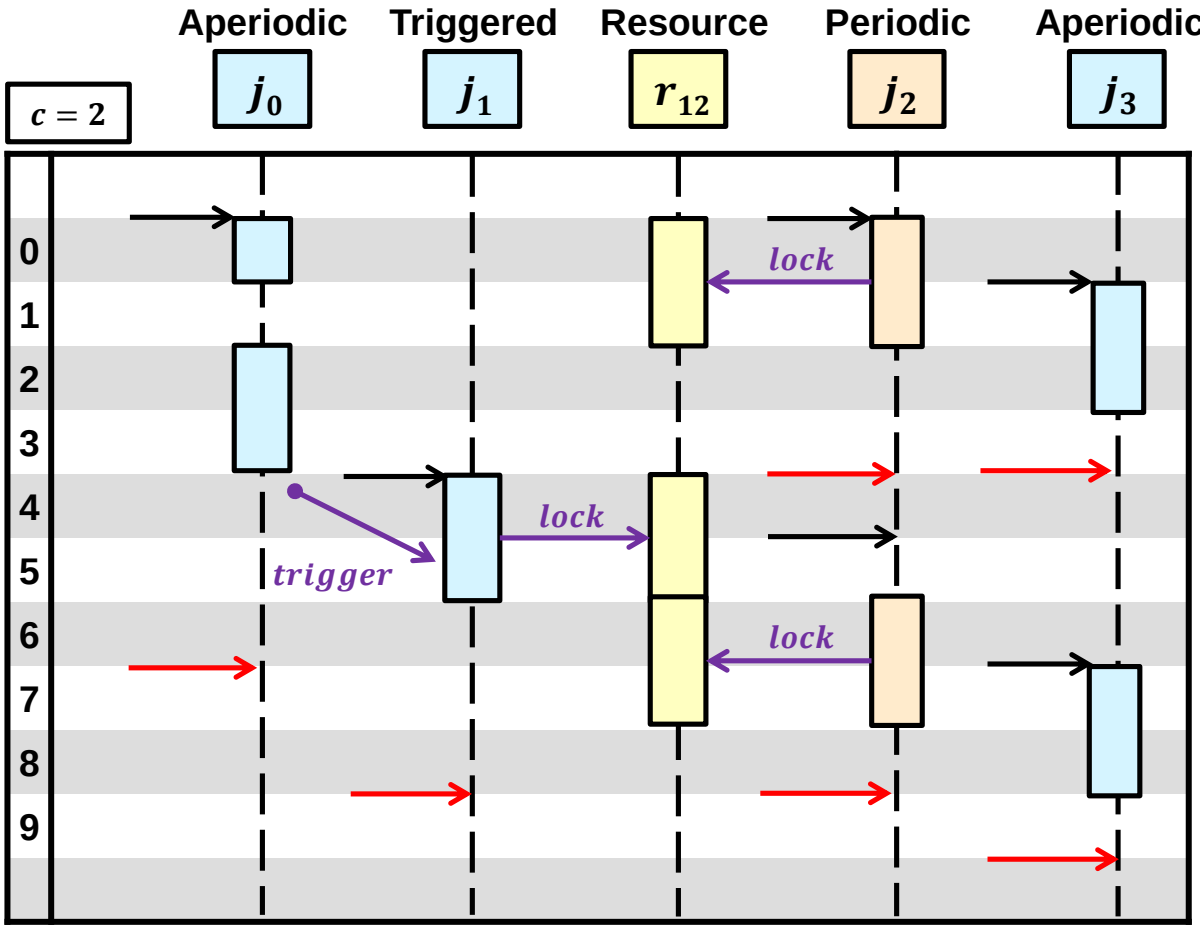


- Observation Interval: $T = [0, 9]$
- Number of cores: $c = 2$
- Set of Tasks: $J = \{j_0, j_1, j_2, j_3\}$
- Priority of Tasks: $priority(j_i) = i$
- Period of Tasks: $period(j_2) = 5$
- Min/Max Inter-arrival time of Tasks: $min\_ia(j_0) = 5, max\_ia(j_0) = 10$
- Duration of Tasks: $duration(j_0) = 3$
- Deadline of Tasks: $deadline(j_0) = 7$

- Triggering Relationship: $triggers(j_0, j_1)$
- Dependency Relationship: $dependent(j_1, j_2)$, $dependent(j_2, j_1)$
- Impacting Relationship: $impacts(j_3, j_2)$, $impacts(j_1, j_2)$, $I(j_2) = \{j_1, j_3\}$

Assumption 1: Time is discretized in *time quanta*
Assumption 2: The time for switching context between tasks is negligible w.r.t. a time quantum

Dynamic Properties depend on the RTES runtime behavior, and are not known prior to the analysis



Independent Properties

- Number of Task Executions:
  $task\_executions(j_0) = 1$,
  $task\_executions(j_3) = 2$
- Arrival time of Aperiodic Task Exec.:
  $arrival\_time(j_0, 0) = 0$,
  $arrival\_time(j_3, 1) = 7$

Dependent Properties

- Active time of Task Executions:
  $active(j_0, 0, 0) = 0$, $active(j_0, 0, 1) = 2$
- Start/End time of Task Executions:
  $start(j_0, 0) = 0$, $end(j_0, 0) = 3$
- Preempted Time Quanta in :
  $preempted(j_0, 0, 1) = 2 - 0 - 1 = 1$
- System Load: $load(0) = 2$
- Deadline Miss of Task Executions:
  $deadline\_miss(j_0, 0) = 3 - 6 = -3$

Independent Properties characterize stress test cases

Dependent Properties characterize the expected reaction of the system to the events modeled by the Independent properties

Both GA and CP cast the search for arrival times that violate task deadlines as an optimization problem

$$F_{DM} = \sum_{j,k} 2^{deadline\_miss(j,k)}$$

Properly rewards scenarios with deadline misses [3]

	Genetic Algorithms [3]	Constraint Programming [4]
Static Properties of Tasks	Chromosomes Properties	Constants
Dynamic Properties of Tasks	Chromosomes Genes	Variables
OS Scheduler Behavior	Chromosomes Evaluation	Constraints
Deadline Misses Requirement	Fitness Function	Objective Function

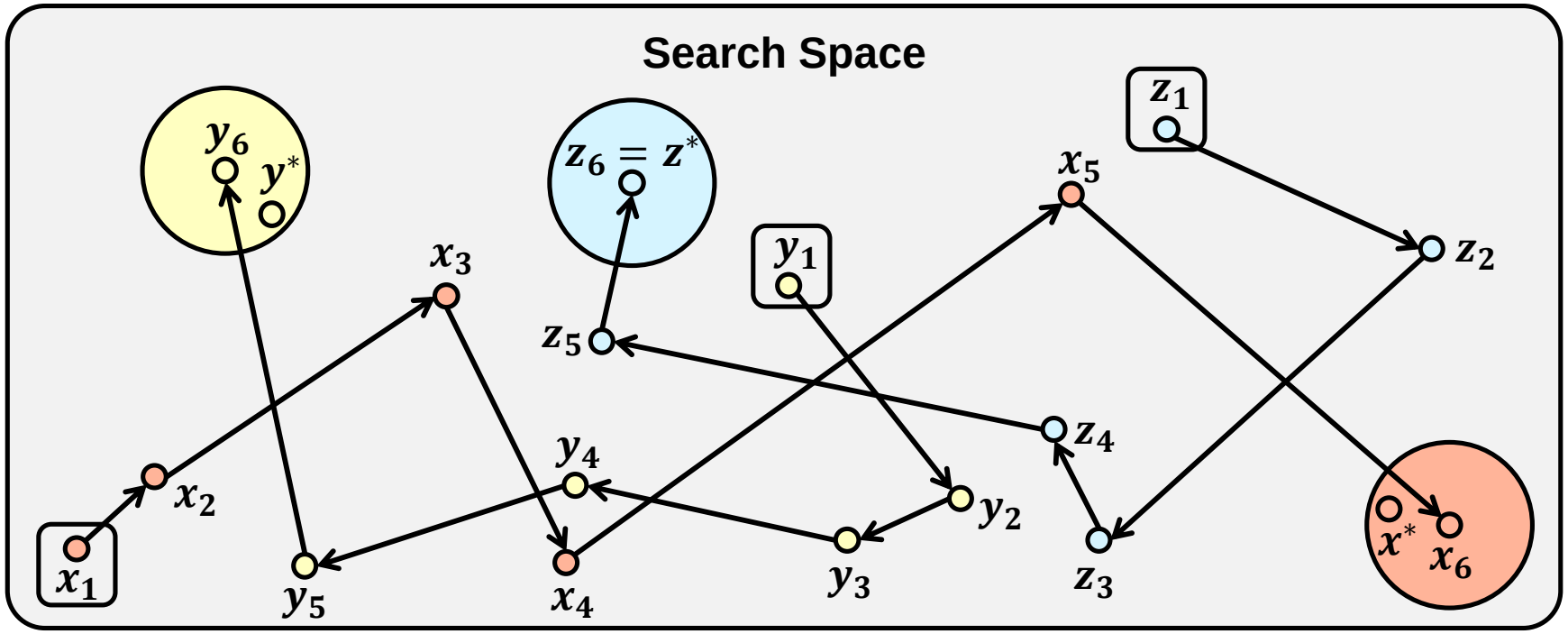
Efficient and diverse, but ineffective

Effective, but inefficient and non-diverse

[3] L. Briand, Y. Labiche, and M. Shousha, "Using Genetic Algorithms for Early Schedulability Analysis and Stress Testing in Real-Time Systems", Genetic Programming and Evolvable Machines, vol. 7 no. 2, pp. 145-170, 2006
[4] Di Alesio, S., Nejati, S., Briand, L., and Gotlieb, A. (2014). Worst-Case Scheduling of Software Tasks – A Constraint Optimization Model to Support Performance Testing. In Principles and Practice of Constraint Programming (CP 2014)

Therefore, we looked into a way to retain the practical advantages of GA and CP in isolation

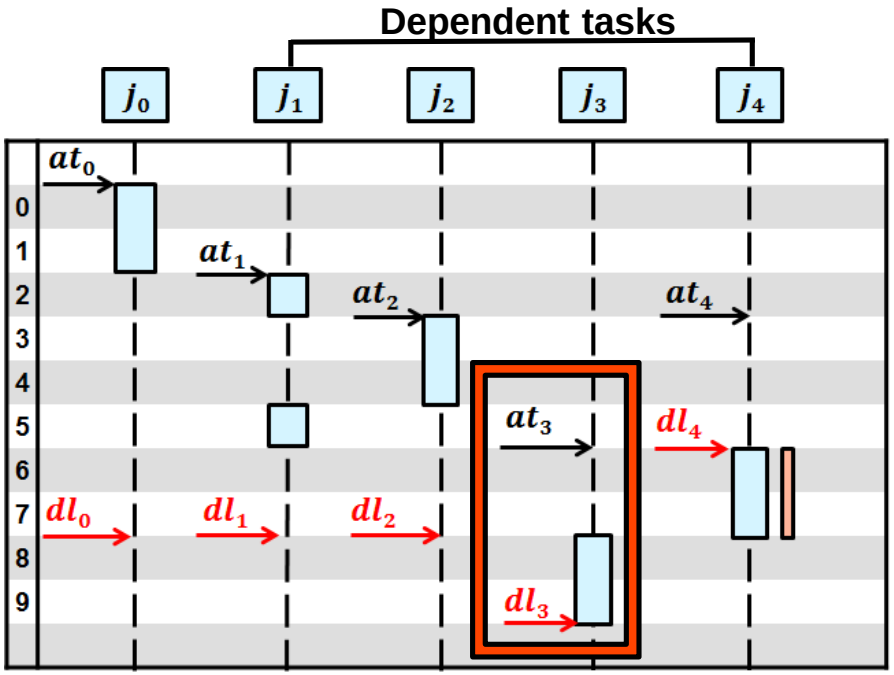
The key idea behind GA+CP is to run complete searches with CP in the neighborhood of solutions found by GA



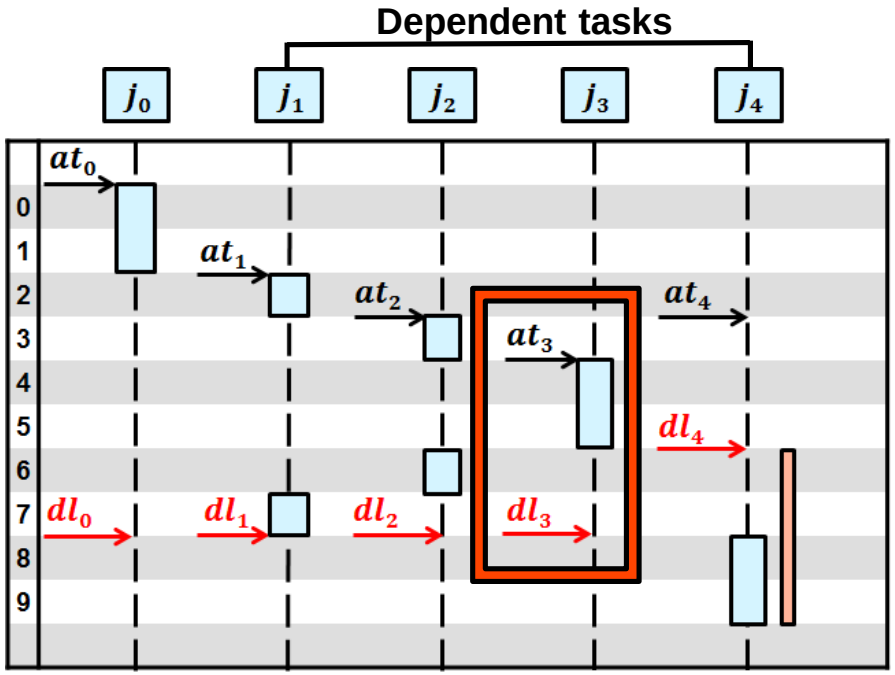
2-steps strategy

1. **GA-step: x_1, y_1, z_1 evolve into x_6, y_6, z_6**
2. **CP-step: x_6, y_6, z_6 are optimized into x^*, y^*, z^***

GA+CP only looks in the neighborhood of tasks that can have an impact on task deadlines in a solution



$$x_{GA} = [[0], [2], [3], [6], [3]]$$



$$x_{GA+CP} = [[0], [2], [3], [4], [3]]$$

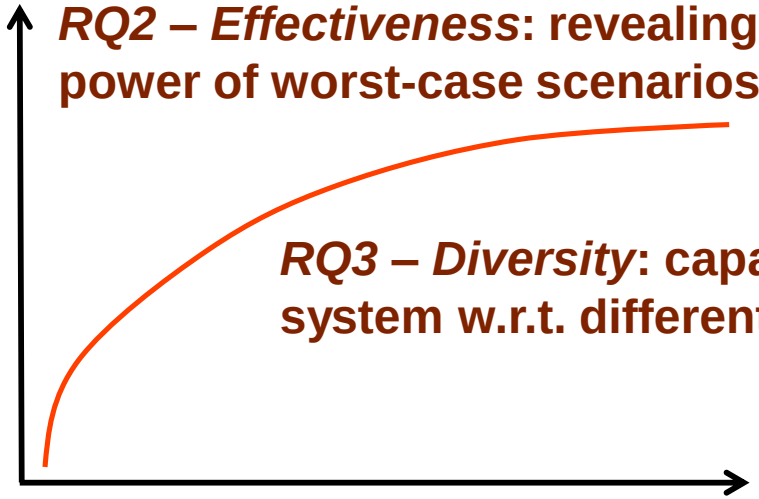
j_1 has a *direct* impact on j_4 because it depends on j_4

j_2 and j_3 have an *indirect* impact on j_4 because they have higher priority than j_1

$$J^*(x) = \{j_4\}$$
$$I_{j_4}(x) = \{j_1, j_2, j_3, j_4\}$$
$$arrival\_time(j_0, 0) = 0$$
$$2 - D \leq arrival\_time(j_1, 0) \leq 2 + D$$
$$3 - D \leq arrival\_time(j_2, 0) \leq 3 + D$$
$$6 - D \leq arrival\_time(j_3, 0) \leq 6 + D$$
$$3 - D \leq arrival\_time(j_4, 0) \leq 3 + D$$

We compared GA+CP with GA and CP in isolation on 5 systems from safety-critical domains

	Domain	Tasks		Logsize
		Periodic	Aperiodic	
ICS: Ignition Control System	Automotive	3	3	446.7
CCS: Cruise Control System	Automotive	8	3	551.6
UAV: Unmanned Air Vehicle	Avionics	12	4	671.5
GAP: Generic Avionics Platform	Avionics	15	8	709.4
HPSS: Herschel-Planck Satellite System	Aerospace	23	9	836.6



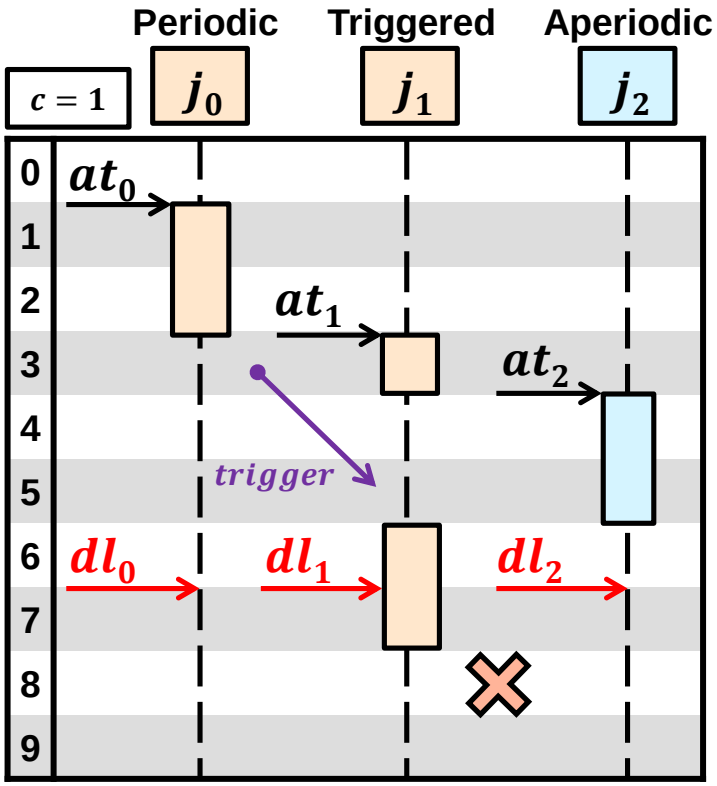
***RQ2 – Effectiveness*: revealing power of worst-case scenarios**

***RQ4 – Scalability*: extent to which the system size affects the efficiency**

***RQ3 – Diversity*: capability to exercise the system w.r.t. different patterns (i.e., *coverage*)**

***RQ1 – Efficiency*: time needed to generate test cases**

We formalize aspects of practical interest related to the Research Questions as *metrics* and *attributes*



$X = \{x\}$
$x = [[1], [3], [4]]$
$s = 1$
$n = 1$
$m = 1$

Metrics

- Computation time $t(x)$ of a solution $x \in X$
- Sum $s(x)$ of time quanta in deadline misses

  s^* x_s^* X_s^*
- Number $n(x)$ of tasks that miss a deadline

  n^* x_n^* X_n^*
- Number $m(x)$ of task execs. that miss a deadline

  m^* x_m^* X_s^*

Attributes

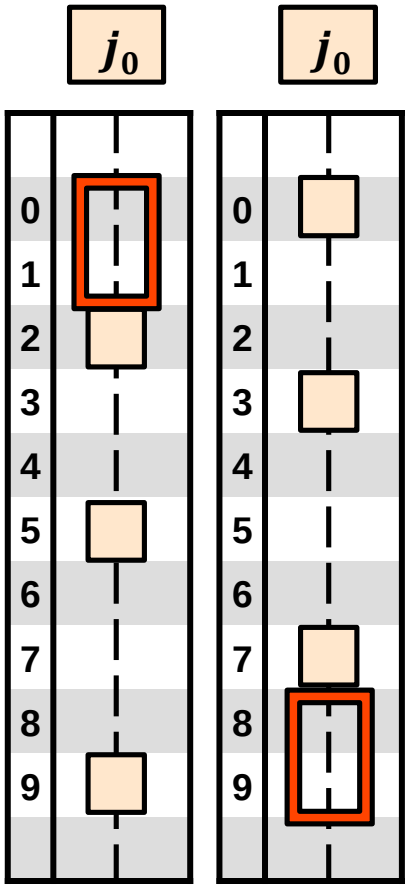
- RQ1 – Efficiency η: computation time of the best solution

  $\eta_s = t(x_s^*)$ $\eta_n = t(x_n^*)$ $\eta_m = t(x_m^*)$
- RQ2 – Effectiveness κ: metric value of the best solution

  $\kappa_s = s^*$ $\kappa_n = n^*$ $\kappa_m = m^*$
- RQ3 – Number N of best solutions

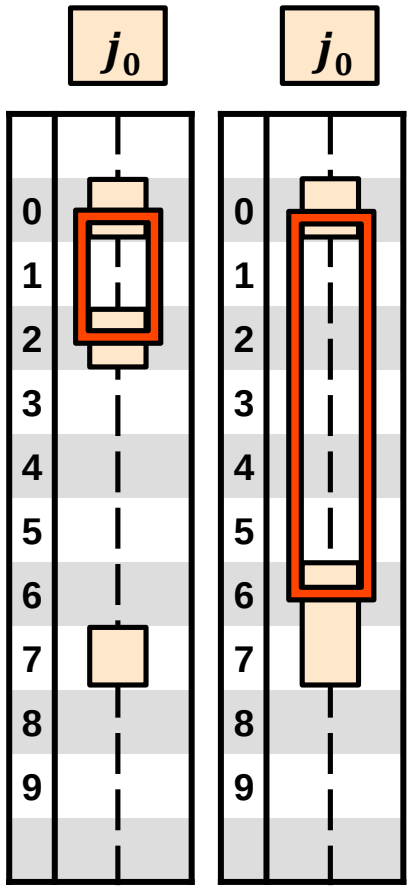
  $N_s = |X_s^*|$ $N_n = |X_n^*|$ $N_m = |X_m^*|$
- RQ3 – Diversity δ: extent to which solutions exercise the system in different ways

High diversity entails that test cases thoroughly exercise interactions between task executions



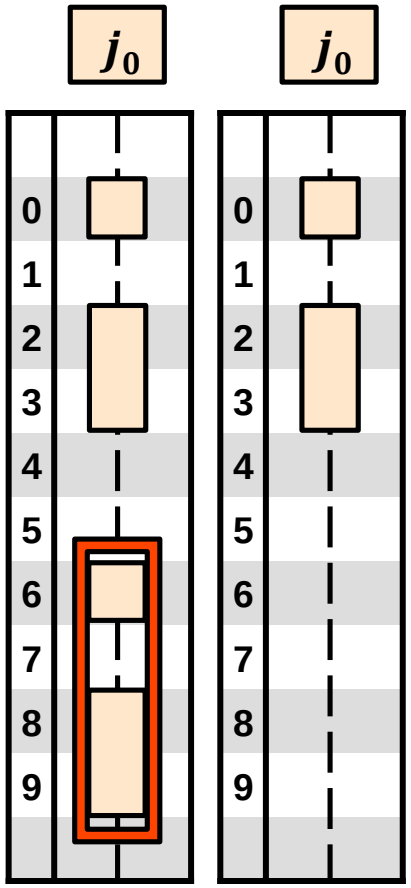
Diversity δ_h w.r.t. execution shift

$\delta_h = |2-0| + |9-7| = 4$



Diversity δ_r w.r.t. execution pattern

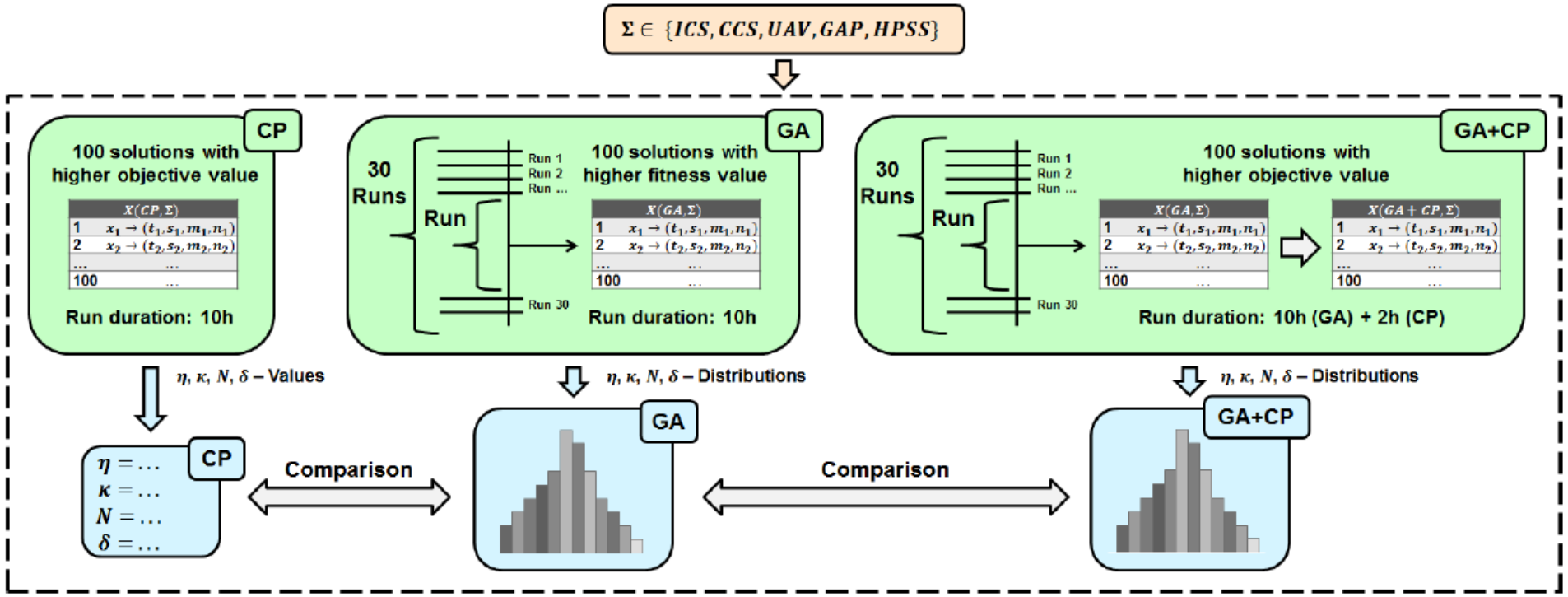
$\delta_r = |1-5| + |4-0| = 8$



Diversity δ_e w.r.t. number of executions

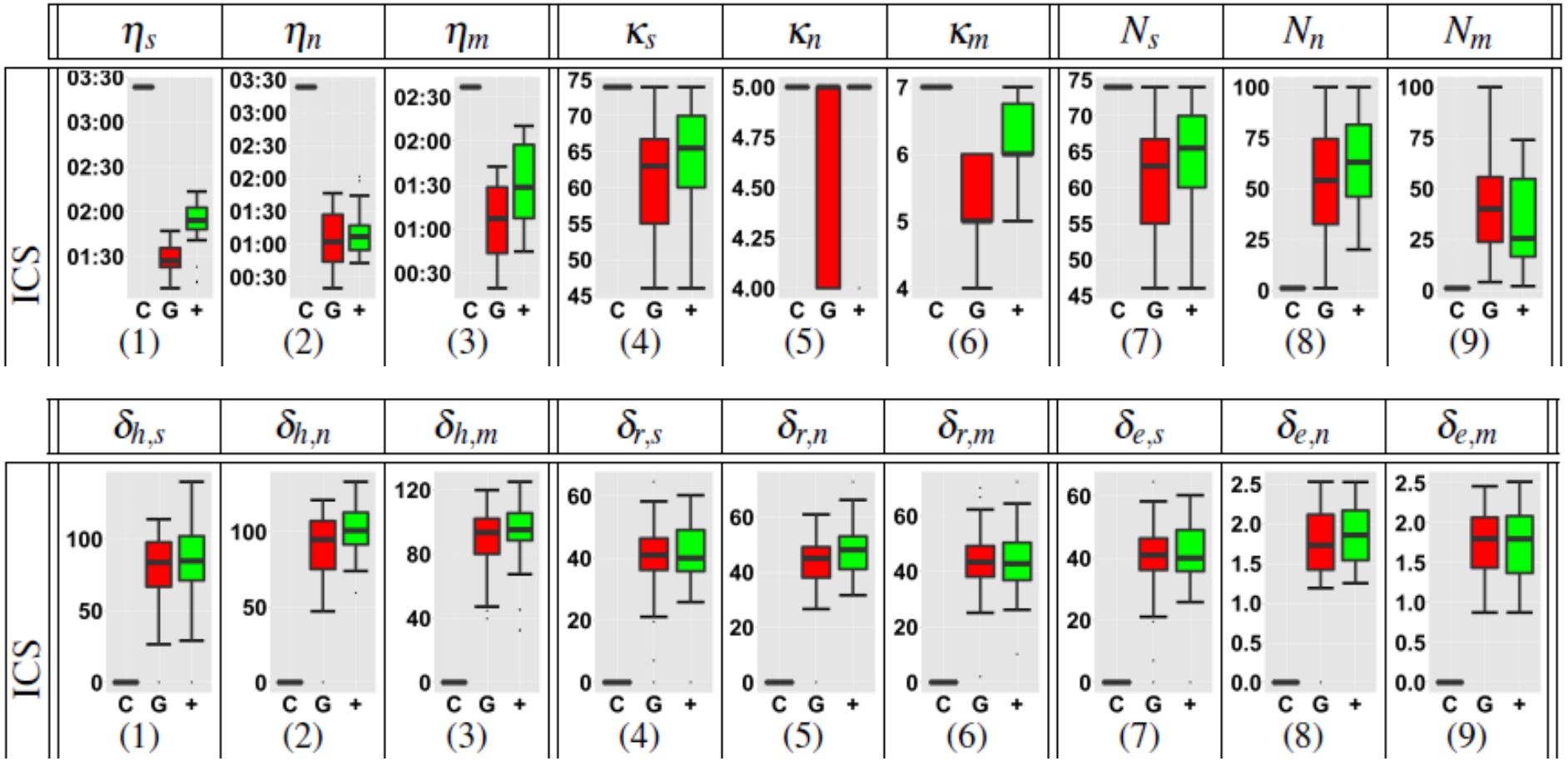
$\delta_e = |2-1| = 1$

In GA+CP, we instructed CP to terminate the local search after two hours



The time taken by CP to terminate the local searches was not significant with respect to the time taken by GA to generate its solutions

GA+CP achieves trade-off between the efficiency and diversity of GA, and the effectiveness of CP

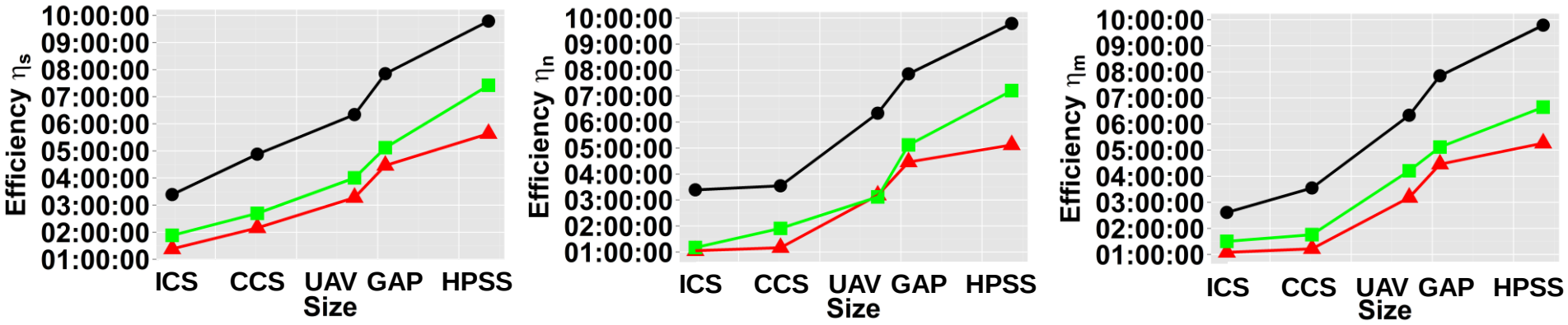


Wilcoxon rank-sum test (GA+CP vs GA)
Wilcoxon signed-rank test (GA+CP vs CP)

$$\eta(GA+CP) \approx \eta(GA) > \eta(CP)$$
$$\kappa(GA+CP) \approx \kappa(CP) > \kappa(GA)$$
$$\delta(GA+CP) \approx \delta(GA) > \delta(CP)$$

The effect the system size has over the efficiency of GA+CP (RQ4 – *Scalability)* is similar to that of GA

CP (•) GA (■) GA+CP (▲)

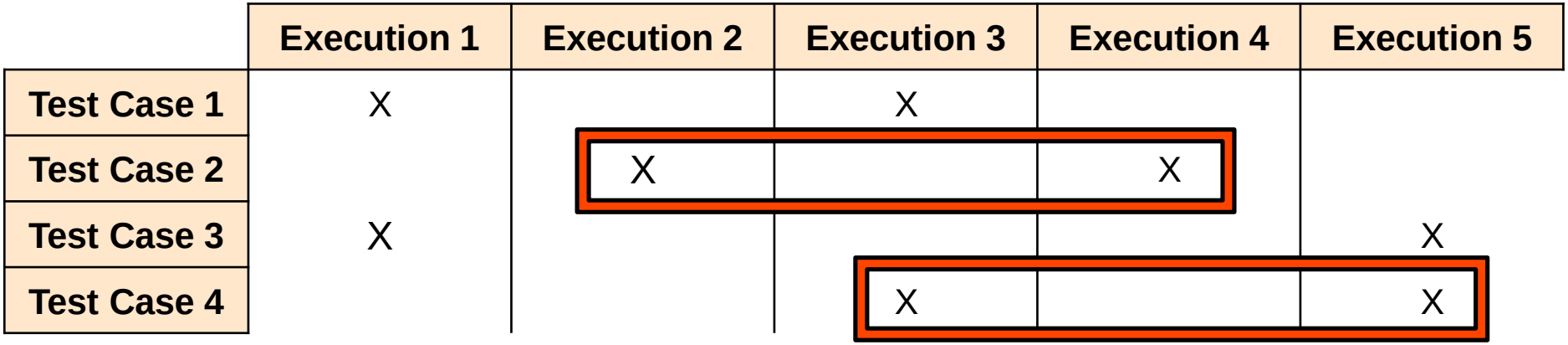


Our experiment was performed on 5 case studies
→ No quantitative scalability study

In our experiments, $D \approx \frac{T}{100}$ proved to yield satisfactory results
→ But in larger case studies D might also have to be larger

Future directions include investigating test suite reduction strategies and multiobjective optimization

Find the minimal set of test cases that retain some relevant property (e.g.: cover all the task executions predicted to miss a deadline)



Multiobjective optimization allows to investigate scenarios where multiple requirements are close to be violated

Deadline Misses ⟺ **CPU Usage** ⟺ **Response Time**

In summary, GA+CP casts stress testing of task deadlines misses as an optimization problem

GA explores the search space, and CP exploits the solutions found by GA

The CP search is complete, but only along "promising" directions (*impacting tasks*)

This design yields trade-off in efficiency, diversity (GA) and effectiveness (CP)

Questions?