# Stress Testing of Task Deadlines: A Constraint Programming Approach


**Stefano Di Alesio [1,2]**

**Shiva Nejati [2]**

**Lionel Briand [2]**

**Arnaud Gotlieb [1]**


**ISSRE 2013**

**05/11/2013**

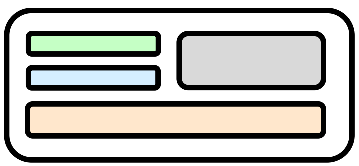

emcelettronica.com


[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 a technique to use Constraint Programming to test deadline misses for RTES

**Performance Requirements vs. Real Time Embedded Systems (RTES)**



**Generating Test Cases that uncover task deadlines using CP**

**How does CP perform w.r.t. the state-of-the-art?**

# RTES are typically safety-critical, and thus bound to meet strict Performance Requirements

ch1group.com

# Performance Requirements are the most difficult requirements to verify

**They depend on the environment the software interacts with (hw devices)**

libelium.com

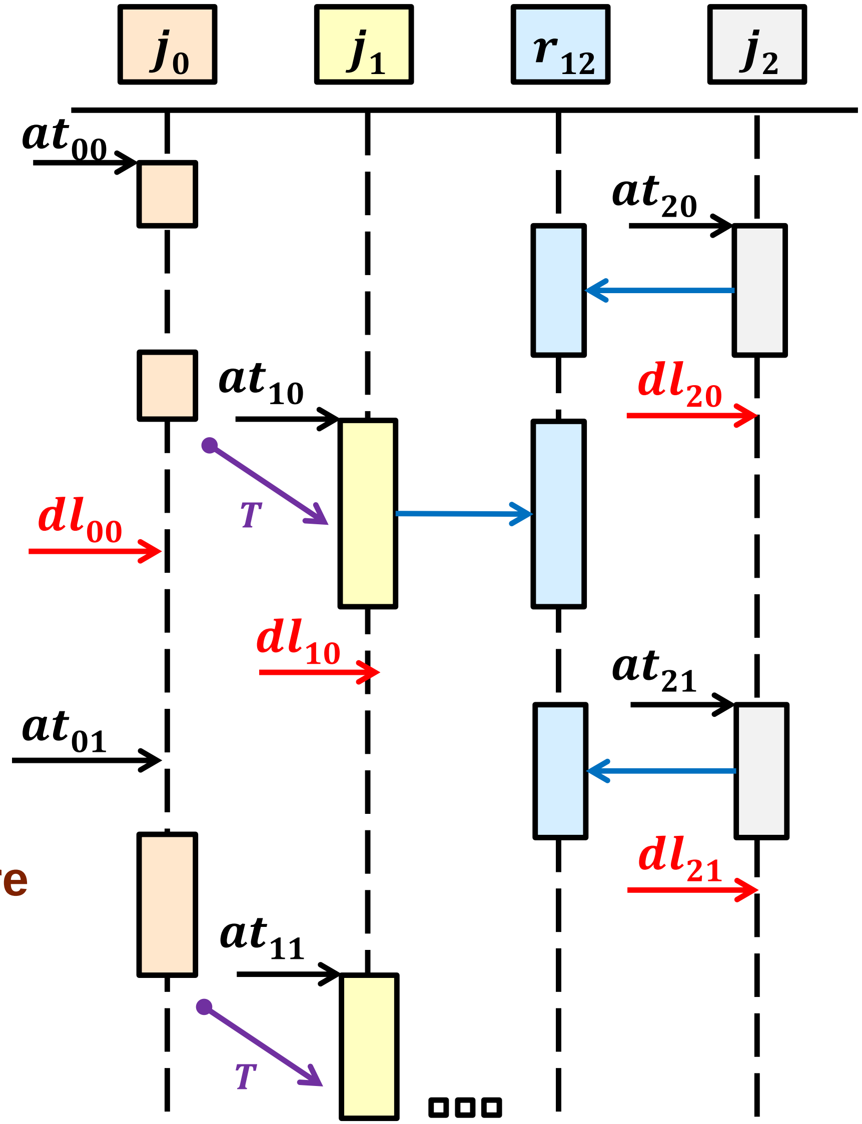
**They depend on the computing platform on which the software runs**


pclaunches.com

deviantart.com

**They constraint the entire system's behavior and thus can't be checked locally**

# RTES have concurrent interdependent tasks which have to finish before their deadlines



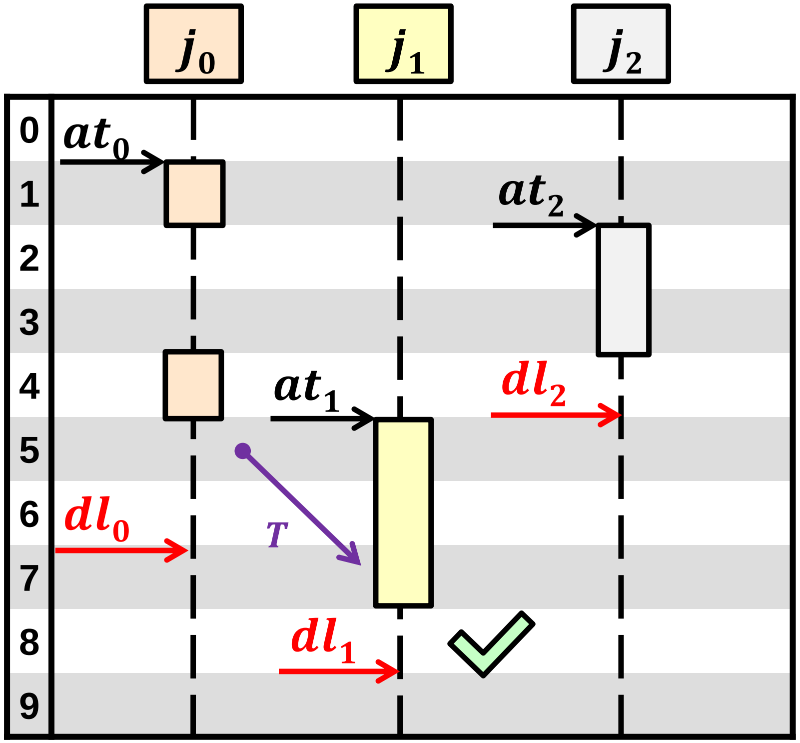

**Each task has a deadline (i.e., latest finishing time) w.r.t. its arrival time**

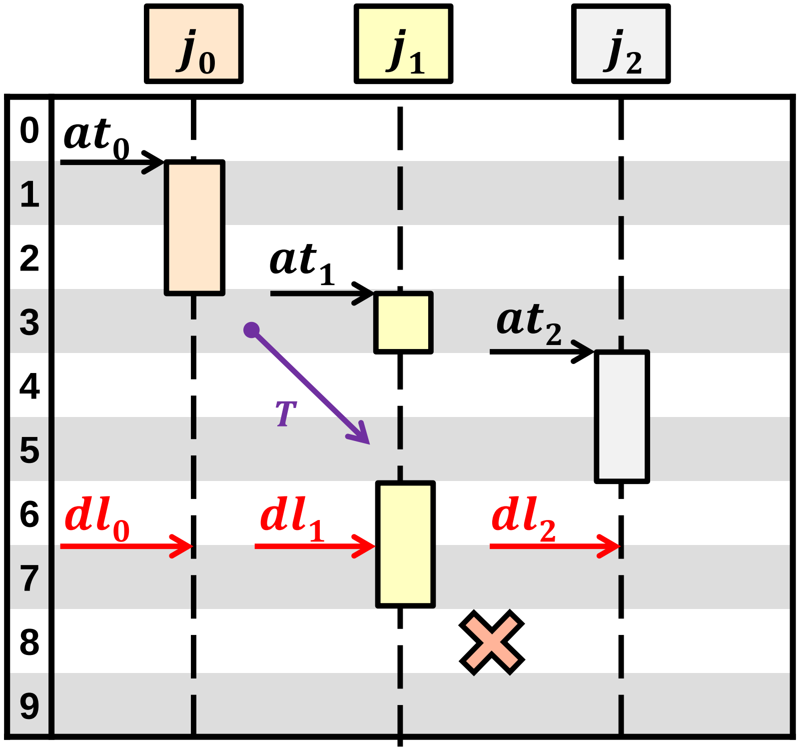
**Some task properties depend on the environment, some are design choices**

**Tasks can trigger other tasks, and can share computational resources with other tasks**

# Particular sequences of arrival times of tasks can determine deadline miss scenarios

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



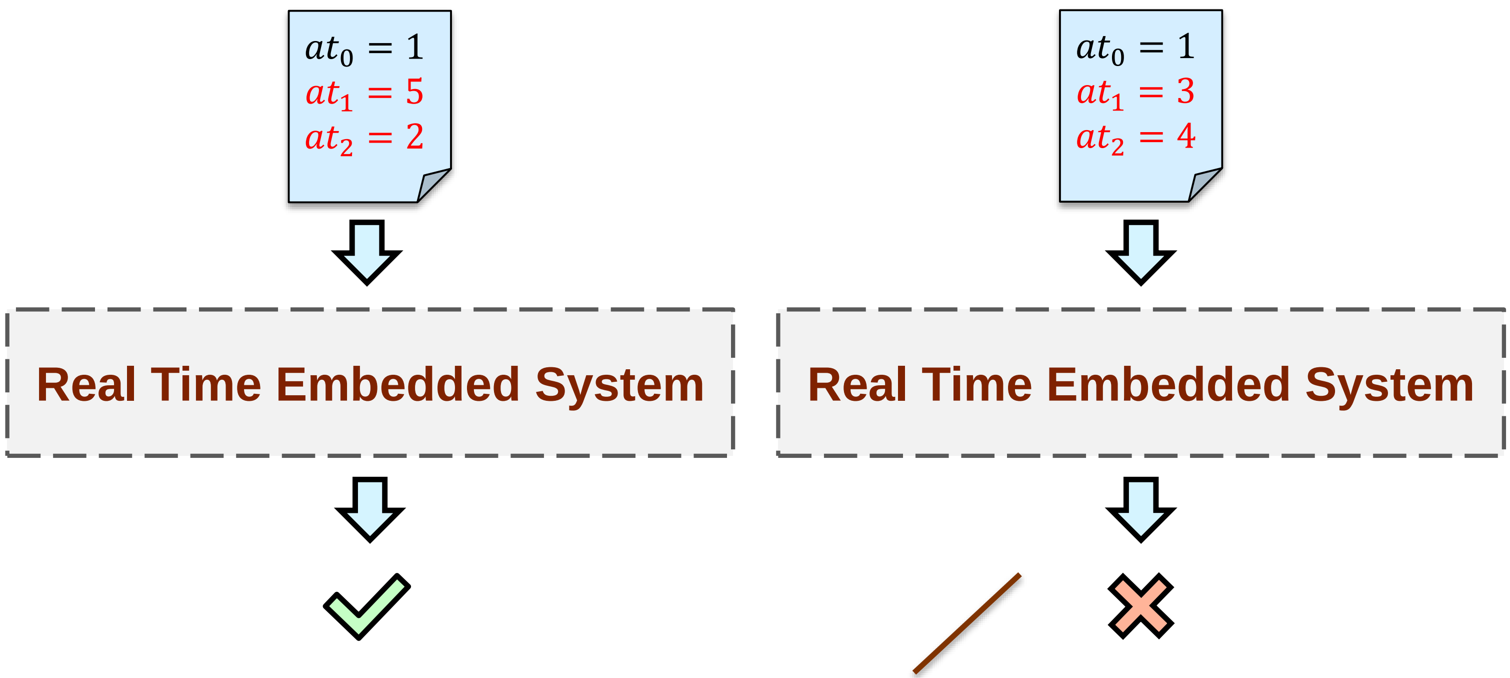




$j_1$ can miss its deadline $dl_1$ depending on when $at_2$ occurs!

# We search for sequences of arrival times maximizing the likelihood of deadline misses

**Arrival times for tasks in a RTES depend on the environment**

**Arrival times can be tuned during software testing**



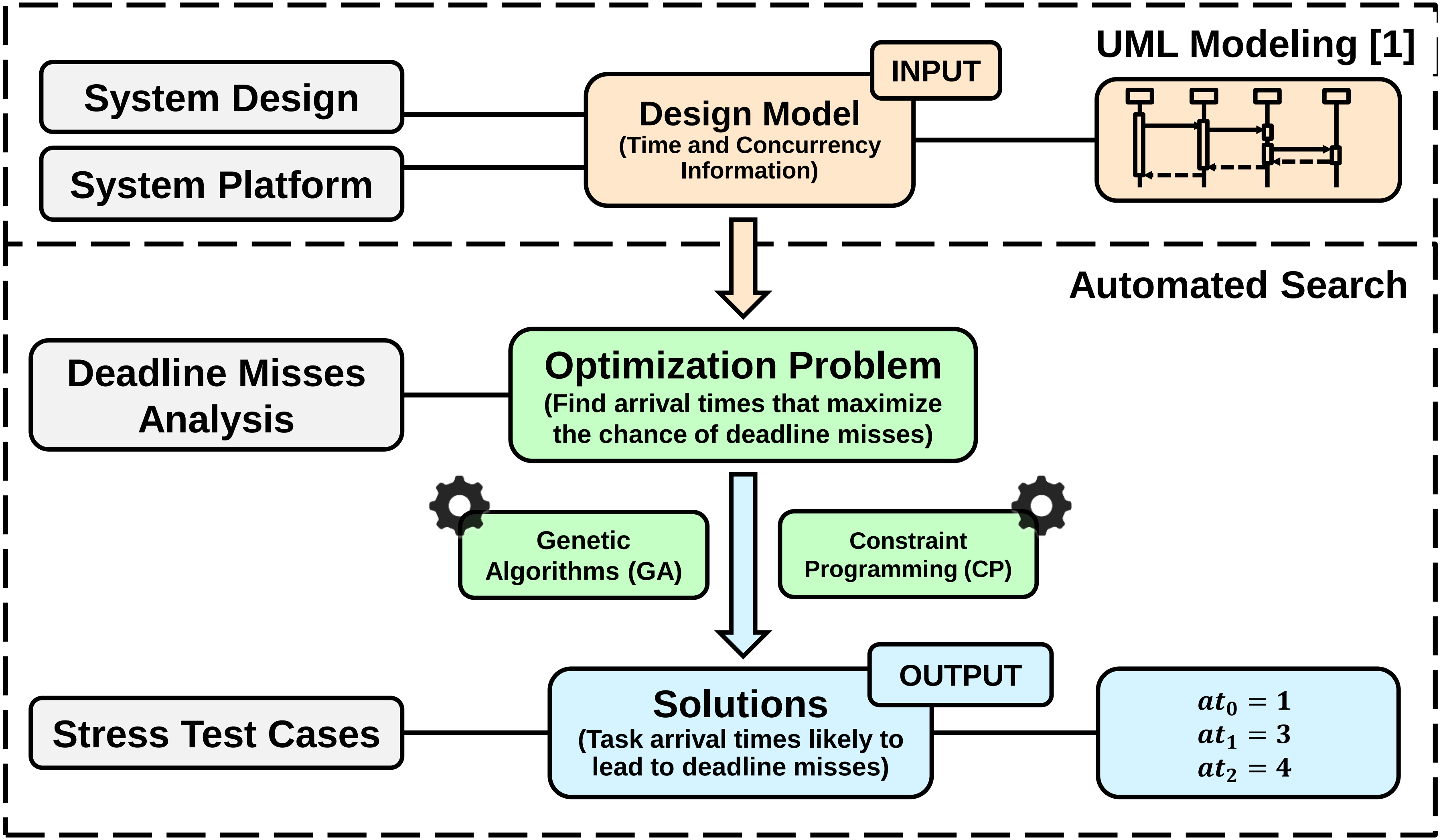

**A sequence of arrival times identified by our approach as likely to lead to a deadline miss characterizes a Stress Test Case**

# This problem is well-known, but each existing approach has its weaknesses

| | Verification | | Testing | |
|---|---|---|---|---|
| | **Schedulability Theory** | **Model Checking** | **Performance Engineering** | **Genetic Algorithms** |
| **Basis** | Mathematical Theory | System Modeling | Practice and Tools | System Modeling |
| **Background** | Queuing Theory | Fixed-point Computation | Profiling, Benchmarking | Meta-Heuristic Search |
| **Key Features** | Theorems [1] | Graph-based, Symbolic [2] | Dynamic Analysis [3] | Non-Complete Search [4] |
| **Weaknesses** | Assumptions, Multi-Core | Complex Modeling | Non Systematic | Low Effectiveness |

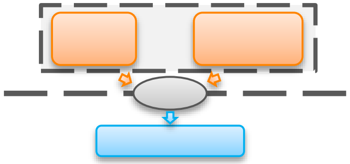

**[1] J. W. S. Liu, "Real-Time Systems". Prentice Hall, 2000**
**[2] M. Mikucionis, K. Larsen, B. Nielsen, J. Illum, A. Skou, S.Palm, J.Pedersen, and P. Hougaaard, "Schedulability analysis using UPPAAL: Herschel-Planck case study", in ISoLA, 2010**
**[3] R. Jain, The art of computer systems performance analysis. John Wiley & Sons, 2008.**
**[4] 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**

# We model System Design, Platform, and Performance Requirement through an Optimization Problem




[1] S. Nejati, S. Di Alesio, M. Sabetzadeh, and L. Briand, "Modeling and analysis of cpu usage in safety-critical embedded systems to support stress testing," in Model Driven Engineering Languages and Systems. Springer, 2012, pp. 759–775.

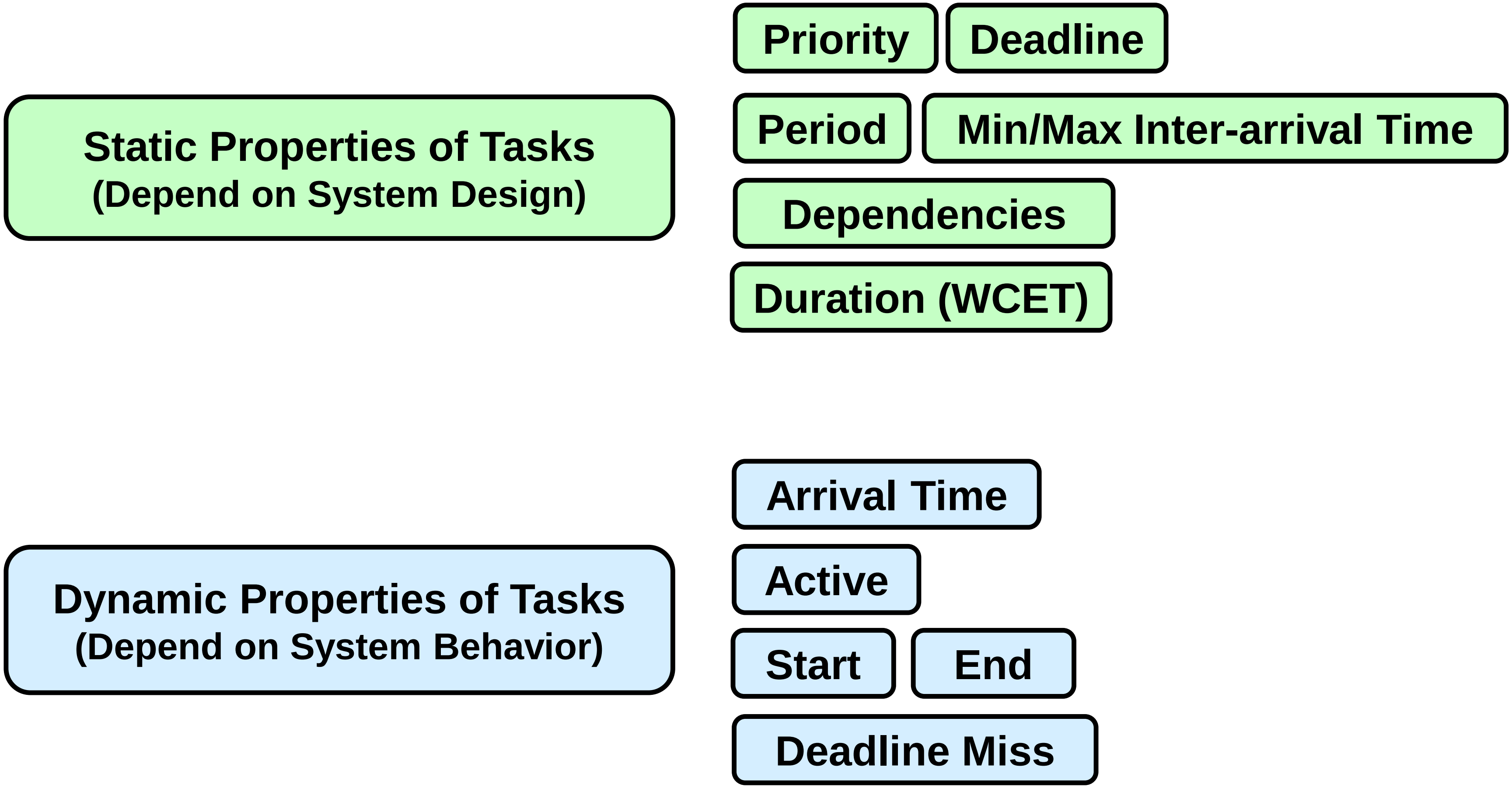
# The goal of our approach is to mitigate the weaknesses found in related work



| | Verification | | Testing | |
|---|---|---|---|---|
| | **Schedulability Theory** | **Model Checking** | **Performance Engineering** | **Genetic Algorithms** |
| **Basis** | Mathematical Theory | System Modeling | Practice and Tools | System Modeling |
| **Background** | Queuing Theory | Fixed-point Computation | Profiling, Benchmarking | Meta-Heuristic Search |
| **Key Features** | Theorems [1] | Graph-based, Symbolic [2] | Dynamic Analysis [3] | Non-Complete Search [4] |
| **Weaknesses** | Assumptions, Multi-Core | Complex Modeling | Non Systematic | Low Effectiveness |

**[1] J. W. S. Liu, "Real-Time Systems". Prentice Hall, 2000**
**[2] M. Mikucionis, K. Larsen, B. Nielsen, J. Illum, A. Skou, S.Palm, J.Pedersen, and P. Hougaaard, "Schedulability analysis using UPPAAL: Herschel-Planck case study", in ISoLA, 2010**
**[3] R. Jain, The art of computer systems performance analysis. John Wiley & Sons, 2008.**
**[4] 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**

# To enable our deadline misses analysis, we first define some timing and concurrency abstractions



# We model the OS scheduler through relationships among the Static and Dynamic Properties of Tasks

OS Scheduler Behaviour

Start + Duration (WCET) ≤ End

Arrival Time ≤ Start

Min. Inter-arrival Time ≤ Arrival Time

Arrival Time ≤ Max. Inter-arrival Time

...

**We consider a pre-emptive priority driven scheduling policy (fixed priority)**

# We defined a function that quantifies how likely arrival times are to trigger deadline misses

**Three "golden rules" [1]:**

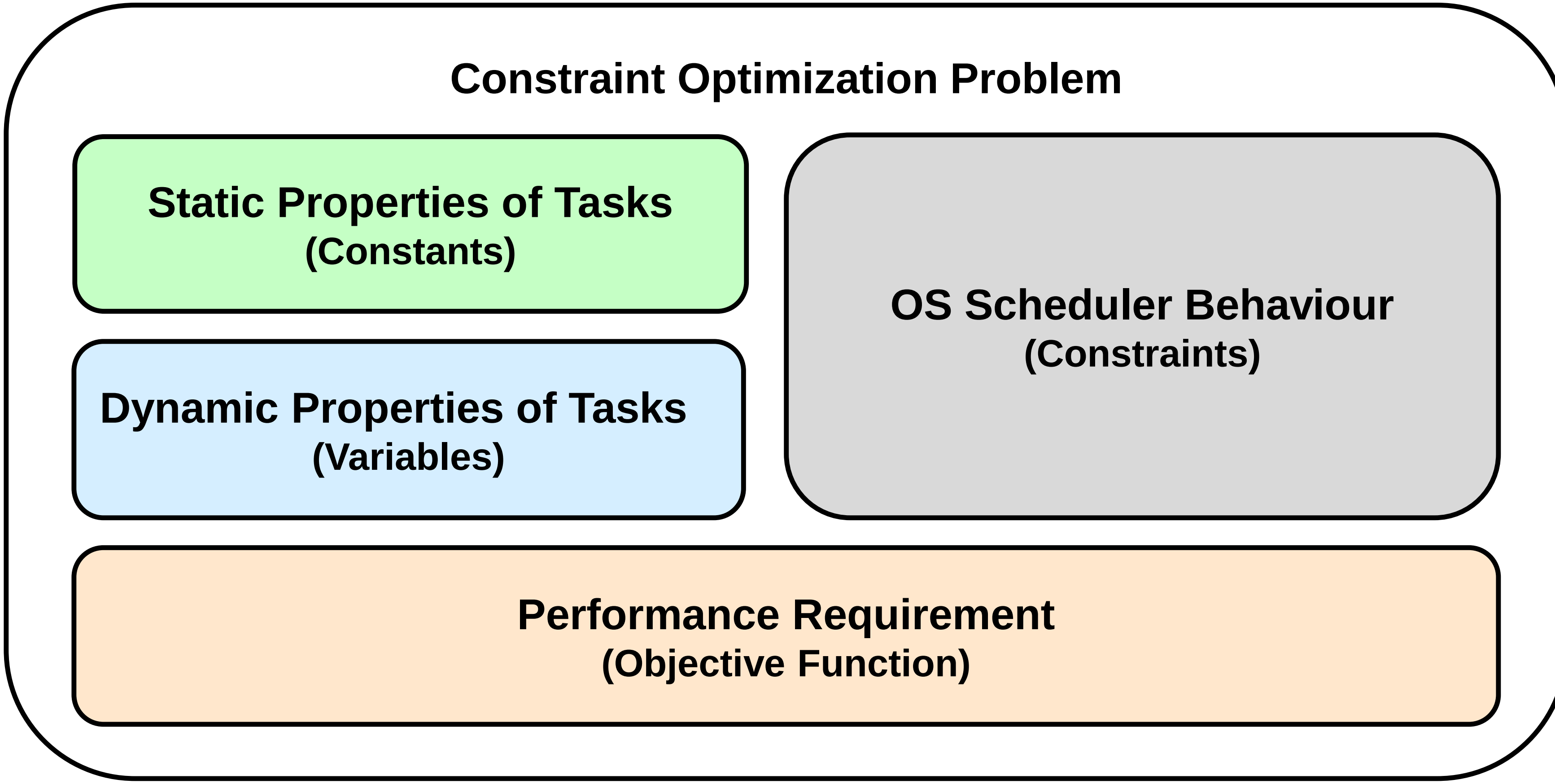
1. **No deadline miss is overshadowed**
2. **The more deadline misses, the higher the value**
3. **The larger the deadline misses, the higher the value**

**Performance Requirement**

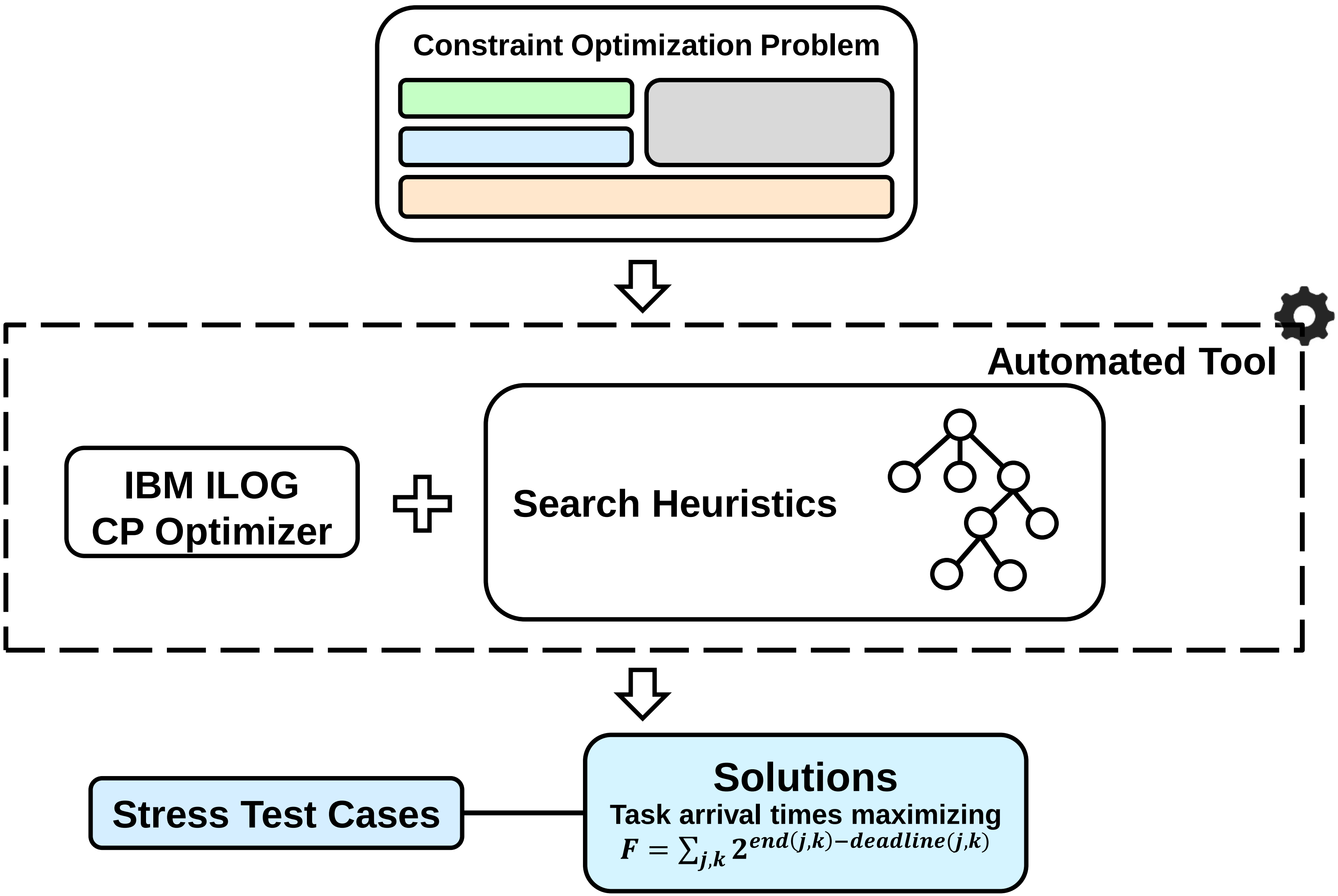
$$F = \sum_{j,k} 2^{end(j,k) - deadline(j,k)}$$

[1] 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

# The key idea is to cast the deadline misses analysis as a Constraint Optimization Problem



# We solve the Constraint Problem in a tool that implements Search Heuristics in CP Optimizer



# We investigated the performance of CP and GA in five case studies from safety-critical domains

| | Domain | Tasks | |
|---|---|---|---|
| | | Periodic | Aperiodic |
| **Ignition Control System [1]** | Automotive | 3 | 3 |
| **Cruise Control System [2]** | Automotive | 8 | 3 |
| **Unmanned Air Vehicle [3]** | Avionics | 12 | 4 |
| **Generic Avionics Platform [4]** | Avionics | 15 | 8 |
| **Herschel-Planck Satellite System [5]** | Aerospace | 23 | 9 |


**[1] M.-A. Peraldi-Frati, Y. Sorel, "From high-level modelling of time in MARTE to real-time scheduling analysis," ACESMB, p. 129, 2008.**
**[2] S. Anssi, S. Tucci-Piergiovanni, S. Kuntz, S. Gérard, and F. Terrier, "Enabling scheduling analysis for AUTOSAR systems," in Object/Component/Service-Oriented Real-Time Distributed Computing, 14th IEEE International Symposium on., 2011, pp. 152–159.**
**[3] K. Traore, E. Grolleau, and F. Cottet, "Simpler analysis of serial transactions using reverse transactions," in Autonomic and Autonomous Systems, International Conference on. IEEE, 2006, pp. 11–11.**
**[4] C. D. Locke, D. R. Vogel, L. Lucas, and J. B. Goodenough, "Generic avionics software specification," DTIC Tech. Rep., 1990.**
**[5] M. Mikučionis, K. G. Larsen, J. I. Rasmussen, B. Nielsen, A. Skou, S. U. Palm, J. S. Pedersen, and P. Hougaard, "Schedulability analysis using UPPAAL: Herschel-Planck case study," in Leveraging Applications of Formal Methods, Verification, and Validation. Springer, 2010, pp. 175–190.**

# To compare CP with the GA for uncovering task deadlines, we answer three Research Questions

**RQ1 – Efficiency: Is CP faster than GA at finding solutions?**

**RQ2 – Effectiveness: Does CP find better solutions than GA?**

**RQ3 – Scalability: How does the size of the system affect the efficiency and effectiveness of CP and GA?**

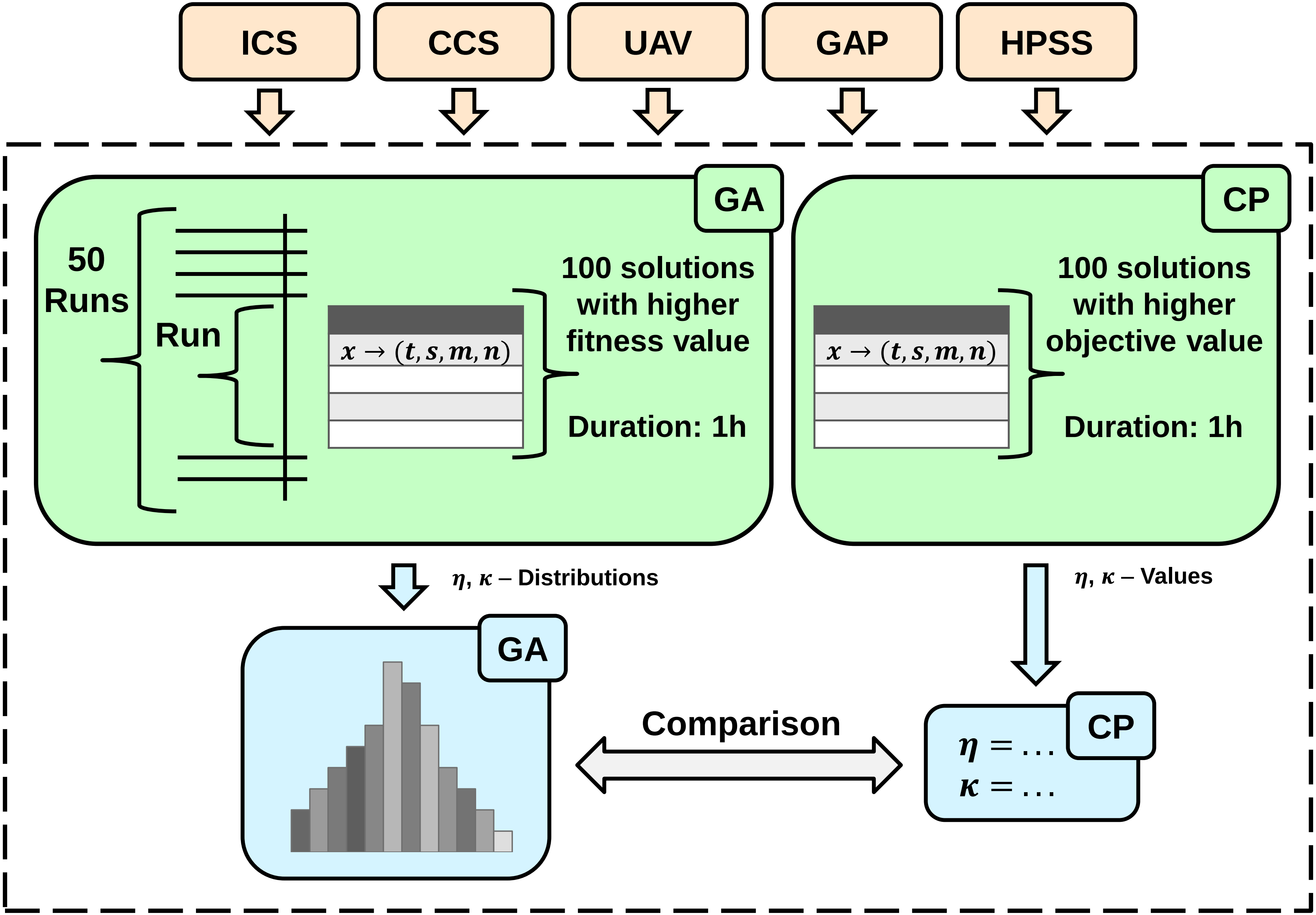
# To answer the Research Questions, one must look into several aspects of practical interest

**Metrics**

- Computation time $t(x)$ of a solution $x$
- Sum $s$ of time quanta in deadline misses – $s^*$, $x_s^*$
- Number $n$ of tasks that miss a deadline – $n^*$, $x_n^*$
- Number $m$ of task executions that miss a deadline – $m^*$, $x_m^*$

**Attributes**

- Efficiency $\eta$: computation time of the best solution
  - $\eta_s = t(x_s^*)$
  - $\eta_n = t(x_n^*)$
  - $\eta_m = t(x_m^*)$
- Effectiveness $\kappa$: metric value of the best solution
  - $\kappa_s = s^*$
  - $\kappa_n = n^*$
  - $\kappa_m = m^*$



ICS
CCS
UAV
GAP
HPSS
GA
50 Runs
Run
$x \rightarrow (t, s, m, n)$
100 solutions with higher fitness value
Duration: 1h
CP
$x \rightarrow (t, s, m, n)$
100 solutions with higher objective value
Duration: 1h
$\eta, \kappa$ – Distributions
$\eta, \kappa$ – Values
GA
Comparison
CP
$\eta = \ldots$
$\kappa = \ldots$

# While GA is more efficient on the smaller case studies, CP is more efficient on the larger ones

| | | $\eta_s$ | | | $\eta_n$ | | | $\eta_m$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | GA | | CP | GA | | CP | GA | | CP |
| ICS | $\bar{x}$ | 15:23 | 40:23 | $\bar{x}$ | 11:05 | 40:23 | $\bar{x}$ | 11:05 | 40:23 |
| | $Q_1$ | 09:33 | | $Q_1$ | 04:33 | | $Q_1$ | 04:33 | |
| | $Q_2$ | 14:07 | | $Q_2$ | 07:49 | | $Q_2$ | 07:49 | |
| | $Q_3$ | 18:05 | | $Q_3$ | 13:32 | | $Q_3$ | 13:32 | |
| | $P$ | 0.98 | | $P$ | 1 | | $P$ | 1 | |
| CCS | $\bar{x}$ | 24:42 | 18:04 | $\bar{x}$ | 07:20 | 18:04 | $\bar{x}$ | 07:20 | 18:04 |
| | $Q_1$ | 15:09 | | $Q_1$ | 05:19 | | $Q_1$ | 05:19 | |
| | $Q_2$ | 22:33 | | $Q_2$ | 06:48 | | $Q_2$ | 06:48 | |
| | $Q_3$ | 30:52 | | $Q_3$ | 08:16 | | $Q_3$ | 08:16 | |
| | $P$ | 0.36 | | $P$ | 1 | | $P$ | 1 | |

**ICS, CCS: GA is more efficient than CP**

**UAV, GAP, HPSS: CP is more efficient than GA**

# CP is more effective than GA, but the difference is more significant on the larger case studies

| | $\kappa_s$ | | | $\kappa_n$ | | | $\kappa_m$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | GA | | CP | GA | | CP | GA | | CP |
| ICS | $\bar{x}$ | 13.22 | 19 | $\bar{x}$ | 1.3 | 2 | $\bar{x}$ | 1.3 | 2 |
| | $Q_1$ | 14 | | $Q_1$ | 1 | | $Q_1$ | 1 | |
| | $Q_2$ | 14 | | $Q_2$ | 1 | | $Q_2$ | 1 | |
| | $Q_3$ | 19 | | $Q_3$ | 2 | | $Q_3$ | 2 | |
| | $P$ | 0.26 | | $P$ | 0.32 | | $P$ | 0.32 | |
| CCS | $\bar{x}$ | 12.14 | 13 | $\bar{x}$ | 2 | 2 | $\bar{x}$ | 2 | 2 |
| | $Q_1$ | 11 | | $Q_1$ | 2 | | $Q_1$ | 2 | |
| | $Q_2$ | 13 | | $Q_2$ | 2 | | $Q_2$ | 2 | |
| | $Q_3$ | 13 | | $Q_3$ | 2 | | $Q_3$ | 2 | |
| | $P$ | 0.52 | | $P$ | 1 | | $P$ | 1 | |

**ICS, CCS: CP is slightly more effective than GA**

**UAV, GAP, HPSS: CP is far more effective than GA**

# Overall, CP has shown to achieve higher efficiency and effectiveness than GA

**RQ1 – Efficiency: Is CP faster than GA at finding solutions?**

**A1: Yes, on the larger case studies**

**RQ2 – Effectiveness: Does CP find better solutions than GA?**

**A2: Yes, especially on the larger case studies**

**RQ3 – Scalability: How does the size of the system affect the efficiency and effectiveness of CP and GA?**

**A3: Within the range covered by our case studies, the larger the case study, the better CP when compared to GA**

**These results are influenced by the runs length**

**Even though…**

**For larger problems, CP may incur in memory problems**

# In summary, Constraint Optimization is a promising approach to derive Stress Test Cases for RTES

**System Platform, Tasks and PRs are modeled in a Constraint Program**

**Solving the CP finds arrival times more likely to stress test the system**

**Significant advantages over other approaches encourage future work**

istockphoto.com

**Questions?**