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

Stefano Di Alesio¹,²  
Lionel Briand²  
Shiva Nejati²  
Arnaud Gotlieb¹

ESEC/FSE 2015  
Bergamo, 02/09/2015

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)

<table>
<thead>
<tr>
<th>Technique/Measure*</th>
<th>Ref</th>
<th>SIL1</th>
<th>SIL2</th>
<th>SIL3</th>
<th>SIL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Avalanche/stress testing</td>
<td>C.5.21</td>
<td>R</td>
<td>R</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>2 Response timings and memory constraints</td>
<td>C.5.22</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
</tr>
<tr>
<td>3 Performance requirements</td>
<td>C.5.19</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
<td>HR</td>
</tr>
</tbody>
</table>

* Appropriate techniques/measures shall be selected according to the safety integrity level.

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

Stefano Di Alesio – 4/21
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.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|
| $a_t_{00}$ | | | | $a_t_{10}$ | | | | | $dl_{00}$ | | | | |
| $at_{20}$ | | | | | | | | | | | | | |
| $a_t_{11}$ | | | | | | | | $dl_{10}$ | | | | | |
| | | | | | | $dl_{20}$ | | | | | | | |
| | | | | | $a_t_{01}$ | | | | | | | |
| | | | | | | | | | | | | |

**Periodic**  **Triggered**  **Resource**  **Aperiodic**

$c = 1$

\[ j_0 \quad j_1 \quad r_{12} \quad j_2 \]

Stefano Di Alesio – 5/21
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.

<table>
<thead>
<tr>
<th>Formal Verification</th>
<th>Testing</th>
<th>Constraint Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduling Theory</td>
<td>Model Checking</td>
<td>Performance Engineering</td>
</tr>
<tr>
<td>Queuing Theory</td>
<td>Fixed-point Computation</td>
<td>Practice and Tools</td>
</tr>
<tr>
<td>Theorems</td>
<td>Symbolic Execution</td>
<td>Dynamic Analysis</td>
</tr>
<tr>
<td>Assumptions</td>
<td>Complex Modeling</td>
<td>Non Systematic</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Background</th>
<th>Key Features</th>
<th>Drawbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queuing Theory</td>
<td>Theorems</td>
<td>Assumptions</td>
</tr>
<tr>
<td>Fixed-point Computation</td>
<td>Symbolic Execution</td>
<td>Complex Modeling</td>
</tr>
<tr>
<td>Practice and Tools</td>
<td>Dynamic Analysis</td>
<td>Non Systematic</td>
</tr>
<tr>
<td>Metaheuristics</td>
<td>Randomized Search</td>
<td>Ineffective [1]</td>
</tr>
<tr>
<td>Artificial Intelligence</td>
<td>Complete Search</td>
<td>Inefficient [1], Low diversity</td>
</tr>
</tbody>
</table>

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.

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

1. **System Design**
2. **System Platform**
3. **Design and Platform Models**
   - Timing and concurrency information about the RTES software and computing platform
4. **Deadline Misses Analysis**
5. **Optimization Problem**
   - Find arrival times that maximize the chance of violating task deadlines
6. **Solutions**
   - Task arrival times likely to violate task deadlines
7. **Stress Test Cases**
8. **UML/MARTE Modeling [2]**

---

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: $\text{priority}(j_i) = i$
- Period of Tasks: $\text{period}(j_2) = 5$
- Min/Max Inter-arrival time of Tasks: 
  $\text{min}_{ia}(j_0) = 5, \text{max}_{ia}(j_0) = 10$
- Duration of Tasks: $\text{duration}(j_0) = 3$
- Deadline of Tasks: $\text{deadline}(j_0) = 7$

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

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:
  \( \text{task_executions}(j_0) = 1 \), \( \text{task_executions}(j_3) = 2 \)

• Arrival time of Aperiodic Task Exec.:  
  \( \text{arrival_time}(j_0, 0) = 0 \), \( \text{arrival_time}(j_3, 1) = 7 \)

Dependent Properties

• Active time of Task Executions:  
  \( \text{active}(j_0, 0, 0) = 0 \), \( \text{active}(j_0, 0, 1) = 2 \)

• Start/End time of Task Executions:  
  \( \text{start}(j_0, 0) = 0 \), \( \text{end}(j_0, 0) = 3 \)

• Preempted Time Quanta in:  
  \( \text{preempted}(j_0, 0, 1) = 2 - 0 - 1 = 1 \)

• System Load: \( \text{load}(0) = 2 \)

• Deadline Miss of Task Executions:  
  \( \text{deadline_miss}(j_0, 0) = 3 - 6 = -3 \)

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.

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

Properly rewards scenarios with deadline misses [3]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chromosomes Properties</td>
<td>Constants</td>
<td></td>
</tr>
<tr>
<td>Dynamic Properties of Tasks</td>
<td>Chromosomes Genes</td>
<td>Variables</td>
</tr>
<tr>
<td>OS Scheduler Behavior</td>
<td>Chromosomes Evaluation</td>
<td>Constraints</td>
</tr>
<tr>
<td>Deadline Misses Requirement</td>
<td>Fitness Function</td>
<td>Objective Function</td>
</tr>
</tbody>
</table>

Efficient and diverse, but ineffective

Effective, but inefficient and non-diverse

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

$j_1$ 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 $j_1$
We compared GA+CP with GA and CP in isolation on 5 systems from safety-critical domains.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Tasks</th>
<th>Logsize</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Periodic</td>
<td>Aperiodic</td>
</tr>
<tr>
<td>ICS: Ignition Control System</td>
<td>Automotive</td>
<td>3</td>
</tr>
<tr>
<td>CCS: Cruise Control System</td>
<td>Automotive</td>
<td>8</td>
</tr>
<tr>
<td>UAV: Unmanned Air Vehicle</td>
<td>Avionics</td>
<td>12</td>
</tr>
<tr>
<td>GAP: Generic Avionics Platform</td>
<td>Avionics</td>
<td>15</td>
</tr>
<tr>
<td>HPSS: Herschel-Planck Satellite System</td>
<td>Aerospace</td>
<td>23</td>
</tr>
</tbody>
</table>

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

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

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

**RQ4 – Scalability**: extent to which the system size affects the efficiency
We formalize aspects of practical interest related to the Research Questions as *metrics* and *attributes*.

**Metrics**
- Computation time $t(x)$ of a solution $x \in X$
- Sum $s(x)$ of time quanta in deadline misses
  
  
  $s^* \quad x_s^* \quad X_s^*$

- Number $n(x)$ of tasks that miss a deadline
  
  
  $n^* \quad x_n^* \quad X_n^*$

- Number $m(x)$ of task execs. that miss a deadline
  
  
  $m^* \quad x_m^* \quad X_s^*$

**Attributes**
- RQ1 – Efficiency $\eta$: computation time of the best solution
  
  
  $\eta_s = t(x_s^*) \quad \eta_n = t(x_n^*) \quad \eta_m = t(x_m^*)$

- RQ2 – Effectiveness $\kappa$: metric value of the best solution
  
  
  $\kappa_s = s^* \quad \kappa_n = n^* \quad \kappa_m = m^*$

- RQ3 – Number $N$ of best solutions
  
  
  $N_s = |X_s^*| \quad N_n = |X_n^*| \quad N_m = |X_m^*|$

- RQ3 – Diversity $\delta$: extent to which solutions exercise the system in different ways

**Diagram**

- **Periodic**
  - $c = 1$
  - $j_0$
  - $j_1$
  - $j_2$

- **Triggered**
  - $at_0$
  - $at_1$
  - $at_2$
  - *trigger*

- **Aperiodic**
  - $dl_0$
  - $dl_1$
  - $dl_2$

**Variables**

- $X = \{x\}$
- $x = \{[1], [3], [4]\}$
- $s = 1$
- $n = 1$
- $m = 1$
High diversity entails that test cases thoroughly exercise interactions between task executions.

Diversity $\delta_h$ w.r.t. execution shift

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

Diversity $\delta_r$ w.r.t. execution pattern

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

Diversity $\delta_e$ 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.

<table>
<thead>
<tr>
<th>$\eta_s$</th>
<th>$\eta_n$</th>
<th>$\eta_m$</th>
<th>$\kappa_s$</th>
<th>$\kappa_n$</th>
<th>$\kappa_m$</th>
<th>$N_s$</th>
<th>$N_n$</th>
<th>$N_m$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="chart1.png" alt="" /></td>
<td><img src="chart2.png" alt="" /></td>
<td><img src="chart3.png" alt="" /></td>
<td><img src="chart4.png" alt="" /></td>
<td><img src="chart5.png" alt="" /></td>
<td><img src="chart6.png" alt="" /></td>
<td><img src="chart7.png" alt="" /></td>
<td><img src="chart8.png" alt="" /></td>
<td><img src="chart9.png" alt="" /></td>
</tr>
</tbody>
</table>

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

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

Stefano Di Alesio – 18/21
The effect the system size has over the efficiency of GA+CP (RQ4 – Scalability) is similar to that of GA

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)

<table>
<thead>
<tr>
<th>Execution 1</th>
<th>Execution 2</th>
<th>Execution 3</th>
<th>Execution 4</th>
<th>Execution 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Case 1</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Test Case 2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Test Case 3</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Test Case 4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

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?