OPTIMAL PRIORITY ASSIGNMENT ALGORITHMS FOR PROBABILISTIC REAL-TIME SYSTEMS

Dorin Maxim*, Olivier Buffet*,
Luca Santinelli*, Liliana Cucu-Grosjean*

and Robert I. Davis#

*INRIA, Nancy Grand-Est, France, firstname.lastname@inria.fr,
#University of York, Real-Time Systems Research Group, United Kingdom, rob.davis@cs.york.ac.uk
PROBABILISTIC REAL-TIME SYSTEMS?

- Deterministic analysis can lead to significant overprovision in the system architecture.
PROBABILISTIC REAL-TIME SYSTEMS?

- Deterministic analysis can lead to significant overprovision in the system architecture.

- An alternative approach is to use probabilistic analysis. System reliability is typically expressed in terms of probabilities for hardware failures, memory failures, software faults, etc.
Deterministic analysis can lead to significant overprovision in the system architecture.

An alternative approach is to use probabilistic analysis. System reliability is typically expressed in terms of probabilities for hardware failures, memory failures, software faults, etc.

For example, the reliability requirements placed on the timing behaviour of a system might indicate that the timing failure rate must be less than $10^{-9}$ per hour of operation.
THE PROBABILISTIC REAL-TIME SYSTEM

- Probabilistic execution times
- Pre-emptive
- Single processor
- Fixed priorities
- Synchronous
- Constrained deadline
- Periodic

The goal: Finding an optimal* priority assignment

*Optimal in the sense that it optimizes some metric related to the probability of deadline failures
A set of $n$ independent periodic tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$

Each task $\tau_i$ generates an infinite number of jobs

Jobs are independent of other jobs of the same task and those of other tasks

$\tau_i$ is characterized by:

$$\tau_i = (C_i, T_i, D_i)$$

$T_i$ being its period;

$D_i$ being its relative deadline;

$C_i$ being its execution time described by a random variable:

$$C_i = \left( \frac{c_{i,k}}{P(C_i = c_{i,k})} \right)$$
THE PROBABILISTIC EXECUTION TIME

The execution time of task $\tau_i$ is assumed to have a known probability function

$$f_{c_i}(\bullet) = P(C_i = c)$$

giving the probability that $\tau_i$ has a computation time equal to $c$

Example: $C_i = \begin{pmatrix} 2 & 3 & 4 \\ 0.5 & 0.45 & 0.05 \end{pmatrix}$
THE PROBABILISTIC EXECUTION TIME

The execution time of task \( \tau_i \) is assumed to have a known probability function

\[
f_{C_i}(\bullet) = P(C_i = c)
\]

giving the probability that \( \tau_i \) has a computation time equal to \( c \)

Example: \( C_i = \begin{pmatrix} 2 & 3 & 4 \\ 0.5 & 0.45 & 0.05 \end{pmatrix} \)
THE PROBABILISTIC EXECUTION TIME

The execution time of task $\tau_i$ is assumed to have a known probability function

$$f_{C_i}(\bullet) = P(C_i = c)$$

giving the probability that $\tau_i$ has a computation time equal to $c$

Example: $C_i = \begin{pmatrix} 2 & 3 & 4 \\ 0.5 & 0.45 & 0.05 \end{pmatrix}$
\[ \tau_1 = \left( \begin{pmatrix} 2 \\ 0.9 \\ 0.1 \end{pmatrix}, 5, 5 \right) \]

\[ \tau_2 = \left( \begin{pmatrix} 2 \\ 0.9 \\ 0.1 \end{pmatrix}, 5, 5 \right) \]
\[ \tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), \quad \tau_2 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right) \]

Response time = 4
Scenario probability = 81\%
\[ \tau_1 = \left( \begin{pmatrix} 2 & 3 \\ 0.9 & 0.1 \end{pmatrix}, 5, 5 \right) \]

\[ \tau_2 = \left( \begin{pmatrix} 2 & 3 \\ 0.9 & 0.1 \end{pmatrix}, 5, 5 \right) \]

- \( \tau_{1,1} \) with Response time = 4 and Scenario probability = 81%
- \( \tau_{2,1} \) with Response time = 5 and Scenario probability = 9%
\[ \tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right) \begin{pmatrix} 5 \\ 5 \end{pmatrix} \]

\[ \tau_2 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right) \begin{pmatrix} 5 \\ 5 \end{pmatrix} \]

Response time = 4
Scenario probability = 81%

Response time = 5
Scenario probability = 9%

Response time = 5
Scenario probability = 9%
\[ \tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), 5, 5 \]

\[ \tau_2 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), 5, 5 \]

Response time = 4
Scenario probability = 81%

Response time = 5
Scenario probability = 9%

Job misses its deadline 1% of the time
Combining the four scenarios we have:

\[
\mathbf{R}_{2,1} = \begin{pmatrix} 4 & 5 & 5 \\ 0,81 & 0,09 & 0,09 \\ 6 & 0,1 \end{pmatrix} = \begin{pmatrix} 2 & 3 \\ 0,9 & 0,1 \end{pmatrix} \otimes \begin{pmatrix} 2 & 3 \\ 0,9 & 0,1 \end{pmatrix} =
\]

\[
= \begin{pmatrix} 4 & 5 \\ 0,81 & 0,18 \end{pmatrix}
\]

**Definition:** The *Response Time* of a job is the elapsed time between its *release* and its *completion*.

**Note:** The response time of a job/task is, as the execution time, described by a random variable.
RESPONSE TIME COMPUTATION

\[ R_{i,j} = B_i(\lambda_{i,j}) \otimes I_i(\lambda_{i,j}) \otimes C_i \]
RESPONSE TIME COMPUTATION

\[ R_{i,j} = B_i(\lambda_{i,j}) \otimes I_i(\lambda_{i,j}) \otimes C_i \]
RESPONSE TIME COMPUTATION

\[ R_{i,j} = B_i(\lambda_{i,j}) \otimes I_i(\lambda_{i,j}) \otimes C_i \]
RESPONSE TIME COMPUTATION

\[ R_{i,j} = B_i(\lambda_{i,j}) \otimes I_i(\lambda_{i,j}) \otimes C_i \]
**DEFINITIONS**

**Definition (Job Deadline Miss Probability):**

\[ DMP_{i,j} = P(R_{i,j} > D_i) \]

is the probability that \( \tau_{i,j} \) misses its deadline.
DEFINITIONS

Definition (Job Deadline Miss Probability):

\[ DMP_{i,j} = P(R_{i,j} > D_i) \]

is the probability that \( \tau_{i,j} \) misses its deadline.

Example: \( \mathcal{R}_{2,1} = \begin{pmatrix} 4 & 5 & 6 \\ 0.81 & 0.18 & 0.01 \end{pmatrix} \)

\[ DMP_{2,1} = 0.01 \]
\[ D_2 = 5 \]
DEFINITIONS

Definition (Task Deadline Miss Ratio):

\[
DMR_i(a, b) = \frac{P(R_i[a, b] > D_i)}{n[a,b]} = \frac{1}{n[a,b]} \sum_{j=1}^{n[a,b]} DMP_{i,j}
\]

is the deadline miss ratio of task \( \tau_i \) in the interval \([a,b]\) and

\[n[a,b] = \left\lfloor \frac{b-a}{T_i} \right\rfloor\]

is the number of jobs of \( \tau_i \) released in \([a,b]\)
DEFINITIONS

**Definition** (Task Deadline Miss Ratio): Is the deadline miss ratio of task $\tau_i$ in an interval.

$$\tau_1 = \left( \begin{pmatrix} 2 \\ 0,9 \\ 0,1 \end{pmatrix}, 10, 10 \right) \quad \tau_2 = \left( \begin{pmatrix} 2 \\ 0,9 \\ 0,1 \end{pmatrix}, 5, 5 \right)$$
**DEFINITIONS**

**Definition** (Task Deadline Miss Ratio): Is the deadline miss ratio of task $\tau_i$ in an interval.

$$\tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), \ 10, \ 10$$

$$\tau_2 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), \ 5, \ 5$$
DEFINITIONS

**Definition** (Task Deadline Miss Ratio): Is the deadline miss ratio of task $\tau_i$ in an interval.

$$\tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), 10, 10) \quad \tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.9 & 0.1 \end{array} \right), 5, 5$$

$$R_{2,1} = \left( \begin{array}{ccc} 4 & 5 & 6 \\ 0.81 & 0.18 & 0.01 \end{array} \right)$$

$$R_{2,2} = \left( \begin{array}{ccc} 2 & 3 & 4 \\ 0.991 & 0.008 & 0.001 \end{array} \right)$$

$$DMR_2 = \frac{DMR_{2,1} + DMR_{2,2}}{2} = \frac{0.01 + 0}{2} = 0.005$$
PROBLEMS TO SOLVE
PROBLEMS TO SOLVE

1. Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$DMR_i(\Phi) \leq p_i$$
PROBLEMS TO SOLVE

1. Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$\text{DMR}_i(\Phi) \leq p_i$$

2. Minimization of the Maximum Priority Assignment Problem (MPAP): involves finding a priority assignment that minimizes the maximum deadline miss ratio of any task.

$$\max_i \{\text{DMR}_i(a, b, \Phi^*)\} = \min_{\Phi} \{\max_i \text{DMR}_i(a, b, \Phi)\}$$
PROBLEMS TO SOLVE

1. Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$\text{DMR}_i(\Phi) \leq p_i$$

2. Minimization of the Maximum Priority Assignment Problem (MPAP): involves finding a priority assignment that minimizes the maximum deadline miss ratio of any task.

$$\max_i \{\text{DMR}_i(a, b, \Phi^*)\} = \min_\Phi \{\max_i \text{DMR}_i(a, b, \Phi)\}$$

3. Average Priority Assignment Problem (APAP): involves finding a priority assignment that minimizes the sum of the deadline miss ratios for all tasks.

$$\Sigma_i \text{DMR}_i(a, b, \Phi^*) = \min_\Phi \{\Sigma_i \text{DMR}_i(a, b, \Phi)\}.$$
**Theorem 1** (Order of higher priority tasks). Considering a task $\tau_i$, if membership of the sets $\text{HP}(i)$ and $\text{LP}(i)$ are unchanged, then the response time $R_{i,j}$ of any job of $\tau_{i,j}$ is unchanged and the response time $R_i^{[a,b]}$ of task $\tau_i$ is unchanged whatever the priority order of tasks within $\text{HP}(i)$ and within $\text{LP}(i)$.
**Theorem 2** (Monotonicity of the response time). Let $\Phi_1$ and $\Phi_2$ be two priority assignments with the same partial order for all tasks except for $\tau_i$ and $\tau_i$ is of lower in $\Phi_1$ than in $\Phi_2$, then the response time of any of its jobs is such that $R_{i,j}(\Phi_1) \geq R_{i,j}(\Phi_2)$. Consequently, the task response time $R_{i,[a,b]}(\Phi_1) \geq R_{i,[a,b]}(\Phi_2)$. 
Theorem 2 (Monotonicity of the response time). Let $\Phi_1$ and $\Phi_2$ be two priority assignments with the same partial order for all tasks except for $\tau_i$ and $\tau_i$ is of lower in $\Phi_1$ than in $\Phi_2$, then the response time of any of its jobs is such that $R_{ij}(\Phi_1) \geq R_{ij}(\Phi_2)$. Consequently, the task response time $R_i^{[a,b]}(\Phi_1) \geq R_i^{[a,b]}(\Phi_2)$.
**Theorem 2** (Monotonicity of the response time). Let $\Phi_1$ and $\Phi_2$ be two priority assignments with the same partial order for all tasks except for $\tau_i$ and $\tau_i$ is of lower in $\Phi_1$ than in $\Phi_2$, then the response time of any of its jobs is such that $R_{ij}(\Phi_1) \geq R_{ij}(\Phi_2)$. Consequently, the task response time $R_{i}^{[a,b]}(\Phi_1) \geq R_{i}^{[a,b]}(\Phi_2)$.

**Corollary**

(Monotonicity of DMP and DMR).

In the same conditions as above:

$DMP_{i,j}(\Phi_1) \geq DMP_{i,j}(\Phi_2)$ and

$DMR_{i}^{[a,b]}(\Phi_1) \geq DMR_{i}^{[a,b]}(\Phi_2)$. 

---

REAL TIME AND NETWORK SYSTEMS, NANTES 30/09/2011 1219
BPAP

Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$\text{DMR}_i(\Phi) \leq p_i$$
BPAP

Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$DMR_i(\Phi) \leq p_i$$

**The Rate Monotonic priority assignment policy is not optimal for BPAP.**
BPAP

Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$DMR_i(\Phi) \leq p_i$$

$$\tau_1 = \left( \begin{array}{cc} 2 & 3 \\ 0.5 & 0.5 \end{array} \right), \quad 4, 4, 0.5$$

$$\tau_2 = \left( \begin{array}{cc} 2 & 3 \\ 0.5 & 0.5 \end{array} \right), \quad 8, 8, 0.1$$

If $\tau_1$ has the higher priority and $\tau_2$ the lower one, as RM dictates:

$$\mathcal{R}_2 = \left( \begin{array}{cccc} 4 & 7 & 8 & D_2^+ \\ 0.25 & 0.25 & 0.375 & 0.125 \end{array} \right)$$

If $\tau_2$ has the higher priority and $\tau_1$ the lower one:

$$\mathcal{R}_1 = \left( \begin{array}{cccc} 2 & 3 & 4 & D_1^+ \\ 0.0625 & 0.1875 & 0.3125 & 0.4375 \end{array} \right)$$
BPAP

Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$DMR_i(\Phi) \leq p_i$$

*Arranging the tasks in increasing order of their maximum permitted deadline miss ratio is not a solution for BPAP.*

Example in the paper.
Basic Priority Assignment Problem (BPAP): Considering that each task has a maximum permitted deadline miss ratio $p_i$, we search for a priority assignment $\Phi$ such that

$$DMR_i(\Phi) \leq p_i$$

Main idea: assigning priorities to tasks starting at the lowest priority level and continuing up to the highest priority level.
MPAP

Minimization of the Maximum Priority Assignment Problem (MPAP): involves finding a priority assignment that minimizes the maximum deadline miss ratio of any task.

The Lazy and Greedy algorithm incrementally builds a solution as a sequence of tasks, starting with the lowest priority first, and adding to $\Phi$ at each iteration an unassigned task.
MPAP

Minimization of the Maximum Priority Assignment Problem (MPAP): involves finding a priority assignment that minimizes the maximum deadline miss ratio of any task.

Greedy: At each iteration, it performs a for loop over the unassigned tasks to search for the one that has the best DMR at the current priority level.

Lazy: whenever a task is found with a deadline miss ratio better than or equal to the current worst DMR, the search is cancelled and this task is assigned.

Algorithm 2: Lazy and Greedy Algorithm

Input: $\Gamma = \{\tau_i, i \in 1..n\}$ /* source set of tasks */
Output: $\Phi$ /* sequence of tasks */, $\text{DMR}_{\text{worst}}$ /* worst DMR */

$\Phi \leftarrow ()$
$\text{DMR}_{\text{worst}} \leftarrow 0$
/* Loop over the priority levels (from lowest to highest) */
for $l \in n..1$ do
  /* Search among unassigned tasks */
  $(\tau_{\text{best}}, \text{DMR}_{\text{best}}) \leftarrow (0, +\infty)$
  for $\tau_i \in \Gamma$ do
    /* Compute DMR of current task $\tau_i$ */
    $\delta \leftarrow \text{DMR}_i(\Phi)$
    /* If this DMR is better than (or equal to) the current worst DMR
    in $\Phi$, be lazy: pick this task and stop the search. */
    if $\delta \leq \text{DMR}_{\text{worst}}$ then
      $(\tau_{\text{best}}, \text{DMR}_{\text{best}}) \leftarrow (\tau_i, \delta)$
      break
    /* If this DMR improves on other unassigned tasks, remember
    this task. */
    if $\delta < \text{DMR}_{\text{best}}$ then
      $(\tau_{\text{best}}, \text{DMR}_{\text{best}}) \leftarrow (\tau_i, \delta)$
  /* The search is done. The task in $\tau_{\text{best}}$ can be assigned at the current
  priority level. */
  $\Gamma \leftarrow \Gamma \setminus \{\tau_{\text{best}}\}$
  $\Phi \leftarrow \Phi \cup \{\tau_{\text{best}}\}$
  /* Update the value of the worst DMR in $\Phi$. */
  if $\text{DMR}_{\text{worst}} < \text{DMR}_{\text{best}}$ then
    $\text{DMR}_{\text{worst}} \leftarrow \text{DMR}_{\text{best}}$
return $(\Phi, \text{DMR}_{\text{worst}})$
APAP

Average Priority Assignment Problem (APAP): involves finding a priority assignment that minimizes the sum of the deadline miss ratios for all tasks.

The Lazy and Greedy Algorithm is not optimal for this problem.

Counter example in the paper.
APAP

Average Priority Assignment Problem (APAP): involves finding a priority assignment that minimizes the sum of the deadline miss ratios for all tasks.

To optimize $g(\Phi) = \sum_i DMR_i(\Phi)$, a simple approach is to use a tree search algorithm enumerating all solutions.

Among various possible tree search algorithms, we choose here Depth-First Search (DFS), which explores each branch as far as possible before backtracking.

Algorithm 3: Depth-First Search

```
\textbf{f}_{\text{best}} = +\infty /* best value so far (glob. var.)*/
\Gamma = \{\tau_i, i \in 1..n\} /* source set of tasks */
(\Phi, g) \leftarrow \text{RECUR}(\Gamma, (.), n, 0)
\text{return} (\Phi, g)
```

/* Function recursively completing the current solution $\Phi$. */
RECUR($\Gamma, \Phi, l, g$) /* Note: $g = g(\Phi)$ */
/* If priority level 0 is attained, we have a complete solution. */
if $l = 0$ then
  /* Is this solution the new best solution? */
  if $g < g^{\text{best}}$ then
    $g^{\text{best}} \leftarrow g$
  return $(\Phi, g)$
/* Otherwise, if the current partial solution is worse than the best solution so far, then backtrack. */
if $g \geq g^{\text{best}}$ then
  return $(\Phi, g)$
/* Try each unassigned task $\tau_i$ at the current priority level. */
($\Phi^{\text{min}}, g^{\text{min}}$) \leftarrow ((.), +\infty)
for $\tau_i \in \Gamma$ do
  $\delta \leftarrow \text{DMR}_i(\Phi)$
  /* Get the best solution completing $\Phi, \tau_i$. */
  ($\Phi', g'$) \leftarrow \text{RECUR}(\Gamma\setminus\{\tau_i\}, \Phi, \tau_i, l - 1, g + \delta)
  /* Memorize the best completed solution. */
  if $g' < g^{\min}$ then
    ($\Phi^{\min}, g^{\min}$) \leftarrow ($\Phi', g'$)
  /* If task $\tau_i$ has a null DMR, then backtrack. */
  if $\delta = 0$ then
    break
/* Return the best completed solution. */
return ($\Phi^{\min}, g^{\min}$)
APAP

Average Priority Assignment Problem (APAP): involves finding a priority assignment that minimizes the sum of the deadline miss ratios for all tasks.

As in previous algorithms, we start with the lowest priority, extending the partial priority ordering $\Phi$ progressively as we go down the tree.

Because of the different criteria optimized in APAP, one cannot be as lazy as in MPAP. Nevertheless, if a task is encountered with a DMR equal to zero, then the search loop can also be interrupted early.
CONCLUSIONS

Probabilistic analysis as a way to reduce the overprovisioning of real-time systems

Three analysis frameworks (problems):
- **BPAP** – Greedy (Audsley) algorithm
- **MPAP** – Greedy and Lazy algorithm
- **APAP** – Tree Search Algorithm
CONCLUSIONS

Probabilistic analysis as a way to reduce the overprovisioning of real-time systems

Three analysis frameworks (problems):
- **BPAP** – Greedy (Audsley) algorithm
- **MPAP** – Greedy and Lazy algorithm
- **APAP** – Tree Search Algorithm

FUTURE WORK

Probabilistic arrivals (periodic model)

More than one probabilistic parameter (for example probabilistic execution and probabilistic arrival)
THANK YOU FOR YOUR ATTENTION