Abstract:- *We present a procedure for representing the schedulability condition for preemptive uniprocessor fixed-priority scheduling of constrained-deadline sporadic task systems as an Integer Linear Program.*

We consider here the fixed-priority scheduling of a constrained-deadline sporadic task system upon a single preemptive processor. Each task τ_i is characterized by three parameters: $\tau_i \stackrel{\text{def}}{=} (C_i, D_i, T_i)$ with the interpretation that

- C_i denotes the *worst-case execution time* of each job of the task;
- T_i is the task's *period*; and
- D_i is the task's *relative deadline* parameter. (We require that $D_i \leq T_i$: hence the qualifier *constrained-deadline*.)

A sporadic task system Γ comprises multiple such tasks: $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$; without loss of generality, we assume that tasks are indexed in decreasing order of priority. The problem of determining whether such a system is schedulable has been shown to be NP-hard [1, 2]. It is also known [3] that a necessary and sufficient condition for Γ to be correctly scheduled is that for each i, $1 \leq i \leq n$, there should be some value of t satisfying the recurrence below that is $\leq D_i$:

$$t \geq C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil \times C_j \tag{1}$$

It is fairly straightforward to show that determining whether such a recurrence has a solution for t that is $\leq D_i$ can be done in pseudo-polynomial time; indeed such a pseudo-polynomial time algorithm forms the basis of the widely-used Rate-Monotonic Analysis methodology [4] for safety-critical real-time system design.

Since (as stated above) the underlying problem is NP-hard, we should not expect to obtain polynomial-time algorithms: from an asymptotic perspective one is unlikely to find a more efficient algorithm than the one in [3]. We do not attempt to do so; rather we describe how preemptive uniprocessor FP schedulability for constrained-deadline sporadic task systems may be represented as an Integer Linear Program (ILP) which can then be solved using off-the-shelf ILP solvers.

Here are the steps for writing down the ILP for a given task system.

1. For each $i = 1 \ldots n$, define a non-negative real-valued variable R_i, with the intended interpretation that R_i denotes some value of t satisfying Recurrence 1 above.
2. For each $i = 1 \ldots n$, specify a constraint
$$R_i \leq D_i \tag{2}$$
to represent the correctness requirement that Recurrence 1 have a solution not exceeding D_i.
3. For each $i = 1, \ldots, n$, and for each $j = 1, \ldots, (i-1)$, define a non-negative integer variable Z_{ij} with the intended interpretation that Z_{ij} represent the term $\lceil R_i/T_j \rceil$.
4. To enforce this intended interpretation on the Z_{ij} variables, add the constraint
$$Z_{ij} \geq \left(\frac{R_i}{T_j} \right) \tag{3}$$
for each $i = 1, \ldots, n$, $j = 1, \ldots, (i-1)$. Since Z_{ij} is specified to be an *integer* variable, it will take on a value $\geq \lceil R_i/T_j \rceil$ (i.e., it respects the $\lceil\ \rceil$ operator that appears in Recurrence 1).
5. And finally add a constraint of the following form for each $i = 1, 2, \ldots, n$, to represent Recurrence 1:
$$C_i + \sum_{j=1}^{i-1} Z_{ij} \times C_j \leq R_i \tag{4}$$

This is simply a restatement of Recurrence 1, except that Constraint 3 sets Z_{ij} to be $\geq \lceil R_i/T_j \rceil$ (rather than exactly equal to $\lceil R_i/T_j \rceil$). We now explain why this is OK. Suppose that there is a *feasible solution* to our ILP — an assignment of values to all the variables that results in all the constraints being satisfied. Note that the Z_{ij} variables appear on the LHS of the $\leq$ inequality of Constraints 4 above. Hence if the value of Z_{ij} satisfying the Constraint 3 in our feasible solution is strictly greater than $\lceil R_i/T_j \rceil$, then Constraints 4 will continue to be satisfied if the value of Z_{ij} is reduced to be exactly equal to $\lceil R_i/T_j \rceil$.

6. Although unneeded for the purposes of determining schedulability, adding the **objective function**

$$\textbf{minimize} \sum_{i=1}^{n} R_i \tag{5}$$

to the ILP would ensure that in the feasible solution each R_i takes on the smallest value of t satisfying Constraint 1, and hence gives the value of the *worst-case response times* of the tasks in Γ.

References

[1] Pontus Ekberg and Wang Yi. Fixed-priority schedulability of sporadic tasks on uniprocessors is NP-hard. In *2017 IEEE Real-Time Systems Symposium, RTSS 2017, Paris, France, December 5-8, 2017*, pages 139–146. IEEE Computer Society, 2017.

[2] Pontus Ekberg. Rate-monotonic schedulability of implicit-deadline tasks is NP-hard beyond Liu and Layland's bound. In *41st IEEE Real-Time Systems Symposium, RTSS 2020, Houston, TX, USA, December 1-4, 2020*, pages 308–318. IEEE, 2020.

[3] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, October 1986.

[4] Mark H. Klein, Thomas Ralya, Bill Pollak, Ray Obenza, and Michael González Harbour. *A Practitioner's Handbook for Real-time Analysis.* Kluwer Academic Publishers, Norwell, MA, USA, 1993.