Advanced Real-Time Systems

Lecture 3/6

Enrico Bini

October 31, 2012

Outline

Motivation

- Schedulability tests says yes/no
- Sensitivity analysis returns the margins of variation

- What if some task parameters are left unspecified and need to be set by the designer?

Optimal design of RT systems

Problem formulation

Given:

- a task set $\mathcal{N}$ with a set $\mathcal{X}$ of parameters that are not specified
- a performance function of $J : \mathcal{X} \to \mathbb{R}$, which returns the performance $J(x)$ of assigning the unspecified parameters equal to x

Solve the following problem

$$\max_{x \in \mathcal{X}} J(x)$$
$$s.t.\ \mathcal{N}(x) \text{ is schedulable by FP}$$

It requires to understand the geometry of the feasible region $\mathcal{X}$.

Sustainable sched test

Definition (Def. 1 in [?])

A schedulability test for a scheduling policy is *sustainable* if any system deemed schedulable by the schedulability test remains schedulable when the parameters of one or more individual job[s] are changed in any, some, or all of the following ways:

1. decreased execution requirements;
2. later arrival times;
3. smaller jitter; and
4. larger relative deadlines.

Theorem (in [?])

Exact tests of FP and EDF are sustainable.

Graphical interpretation

Outline

Execution time are unknowns

- $\mathcal{X} = \{C_1, \ldots, C_n\}$.
- Periods T_i and deadlines D_i are given
- Execution times have to be found so that the performance $J(C_1, \ldots, C_n)$ (function of the computation times) is maximized

Graphical interpretation of the exact solution

Example of cost function

An example of cost function can be

$$\max_{C_1,\ldots,C_n} \min_i J_i(C_i)$$
$$\text{s.t } \mathcal{N} \text{ is FP-schedulable}$$

with $J_i(C_i)$ task dependent performance.

- Typically $J_i(C_i)$ is non-decreasing.

Solution is such that

- $\forall i = 1, \ldots, n-1 \quad J_i(C_i) = J_{i+1}(C_{i+1})$
- the task set is *barely FP-schedulable*: by increasing some computation time by any small amount, we make it non-schedulable.

Graphical interpretation.

Using sufficient test

- If exact solution is too complicated, one can always use sufficient (simpler) tests

An example with LL test

$$\max_{C_1,\ldots,C_n} \min_i J_i(C_i)$$

$$\text{s.t } \sum_i \frac{C_i}{T_i} \leq U_{\mathsf{LL}}$$

with $J_i(C_i)$ task dependent performance Solution is such that

- $\forall i = 1, \ldots, n-1 \quad J_i(C_i) = J_{i+1}(C_{i+1})$
- $\sum_i \frac{C_i}{T_i} = U_{\mathsf{LL}}$

Outline

Periods are unknowns

- $\mathcal{X} = \{T_1, \ldots, T_n\}$
- Typical problem in control systems: choosing the sampling periods of all controlers
- sometime it is convenient to view them as activation frequencies $f_i = \frac{1}{T_i}$, so that $\mathcal{X} = \{f_1, \ldots, f_n\}$
- Execution times C_i are specified
- Deadlines are specified either as absolute value D_i or normalized to the periods $\frac{D_i}{T_i}$.

The problem then is

$$\max_{T_1,\ldots,T_n} \; J(T_1, \ldots, T_n)$$

s.t $\mathcal{N}$ is FP-schedulable

What is the geometry of the FP-schedulable periods?

hw1, problem 3: lesson learned

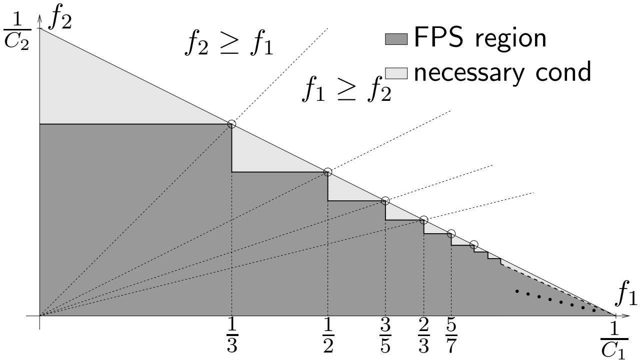
Let us assume $D_i = T_i$

- Starting from a schedulable configuration, we can reduce any period, then another, ...until we reach a point with $\sum_i U_i = 1$. Hence at this final point no other period can be further reduced;
- the point that we reach changes depending on the order that we follow for reducing the periods.
- FP is sustainable.

Graphical interpretation
View in the space of activation frequencies $f_i = \frac{1}{T_i}$. In the f_i space the constraint $\sum_i U_i \leq 1$ is linear.

Space of feasible frequecies

if $n = 2$ and $C_1 = 1$, $C_2 = 2$ we have



If $\frac{\partial J}{\partial f_i} \geq 0$ all "vertices" are local optima.
Why the HB region is not contained?

Heuristic search algorithm

1. find a good starting point over a simple constraint (for example, by using $\sum_i U_i \leq 1$)
2. using the "Min schedulable speed" find the intercept with the FP boundary
   - scaling the computation times by α is equivalent to scaling the task periods by $\frac{1}{\alpha}$
3. since $\frac{\partial J}{\partial f_i} \geq 0$, then by increasing task frequency we certainly increase the performance J

No guarantee of optimality

Optimal search algorithm

1. Start from an initial FP-schedulable solution $\mathbf{f}^{\text{first}}$ (such as the one found previously) and set it as current solution $\mathbf{f}^{\text{cur}}$
2. Use a branch and bound algorithm to enumerate, and possibly prune, all vertices $\mathbf{f}$ with better performance $J(\mathbf{f}) > J(\mathbf{f}^{\text{first}})$
3. if a better solution is found, then update $\mathbf{f}^{\text{cur}}$
4. when all vertices have been check $\mathbf{f}^{\text{cur}}$ will be the optimum

Optimal search algorithm

1. Start from an initial FP-schedulable solution $\mathbf{f}^{\text{first}}$ (such as the one found previously) and set it as current solution $\mathbf{f}^{\text{cur}}$
2. Use a branch and bound algorithm to enumerate, and possibly prune, all vertices $\mathbf{f}$ with better performance $J(\mathbf{f}) > J(\mathbf{f}^{\text{first}})$
3. if a better solution is found, then update $\mathbf{f}^{\text{cur}}$
4. when all vertices have been check $\mathbf{f}^{\text{cur}}$ will be the optimum

How do we enumerate the vertices?

Yet another exact FP test

Theorem (Proposition 2.1 in [?])

The task set $\mathcal{N}$ is schedulable by FP **if and only if***:*

$$\forall i = 1, \ldots, n \quad \exists\, \mathbf{k}^{(i)} \in \mathbb{N}^{i-1}$$

such that

$$\begin{cases} C_i + \sum_{j=1}^{i-1} k_j^{(i)}\, C_j \leq T_i \\ (k_\ell^{(i)} - 1) T_\ell < C_i + \sum_{j=1}^{i-1} k_j^{(i)} C_j \leq k_\ell^{(i)} T_\ell \quad \ell = 1, \ldots, i-1 \end{cases}$$

Proof sketch: it is equivalent to $\forall i, \exists t \in [0, T_i] \ldots$, by setting $t = C_i + \sum_{j=1}^{i-1} k_j^{(i)} C_j$ and by setting $k_j^{(i)} = \left\lceil \frac{t}{T_j} \right\rceil$

Implications over the $\mathbf{f}$-space

The task set $\mathcal{N}$ is schedulable by FPS **if and only if**:

$$\forall i = 1, \ldots, n \quad \exists\, \mathbf{k}^{(i)} \in \mathbb{N}^{i-1}$$

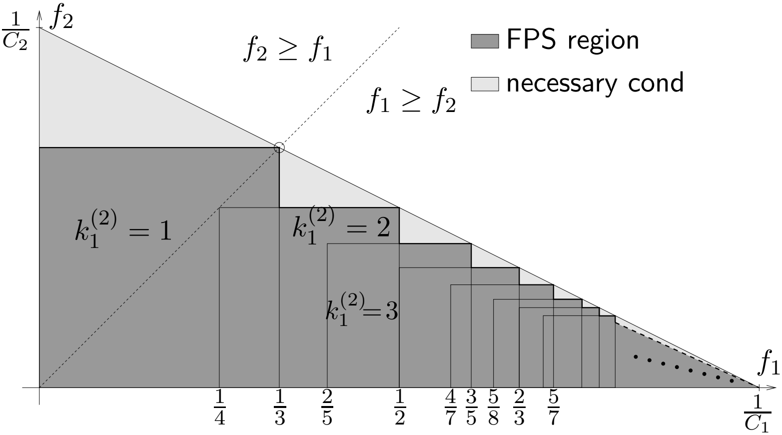
such that:

$$\begin{cases} 0 \leq f_i \leq \boxed{\dfrac{1}{C_i + \sum_{j=1}^{i-1} k_j^{(i)}\, C_j}} \\ \dfrac{k_\ell^{(i)} - 1}{C_i + \sum_{j=1}^{i-1} n_j^{(i)} C_j} < f_\ell \leq \boxed{\dfrac{k_\ell^{(i)}}{C_i + \sum_{j=1}^{i-1} k_j^{(i)} C_j}} & \ell = 1, \ldots, i-1 \end{cases}$$

which are the coordinates of the vertices.

Viewing the f-space

with $C_1 = 1$ and $C_2 = 2$



Branch and Bound

- A branch and bound algorithm can be used to search for the optimal task frequencies.
- More details can be found in [**?**].

Using sufficient tests

Seto et al. [**?**] proposed to simply use a utilization based test

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq U^{\mathsf{ub}}$$

to assign the periods.

- sub-optimal
- explicit solution can be found

Outline

Performance is not just the task periods

Is the performance J function of the only task periods/frequencies?

- In control systems "periodic" means that both sampling and actuation occur at the same instant;
- In reality when a "periodic" task is scheduled over a CPU, it has a *delay* and a *jitter* in its execution, which depends on the parameters of the other tasks, too.

Add a drawing

Task schedule and performance

Ideally we should:

- identify how the task periods and priorities affect the schedule $\mathcal{S}$;
- identify how the schedule affect the (control) performance;
- just perform the optimization.

Task schedule and performance

Ideally we should:

- identify how the task periods and priorities affect the schedule $\mathcal{S}$;
- identify how the schedule affect the (control) performance;
- just perform the optimization.

- relatively recent research area
- some results do exist (will be included in the course notes)
- still space for new contributions

Selection of open problems

- How do task parameters affect the entire task schedule?
  - pattern of job start times
  - pattern of job response time
  - distribution of job response time
  - joint distribution of job response times of k consecutive jobs
- How does the task schedule affect performance?
  - can controllers be improved by the exact knowledge of sampling and actuation times?
  - how robust are controllers against variations of the task schedule?