Advanced Real-Time Systems

Enrico Bini

November 11, 2012

1 Intro to Sched Prob

Basic elements of a scheduling problem:

- a set $\mathcal{N}$ of *tasks* (aka demands, works, jobs) requiring work to be made. Since they are finite, we represent them by $\mathcal{N} = \{1, 2, \ldots, n\}$;
- a set $\mathcal{R}$ of *resources* (aka processors, machines, workers, etc.) capable to perform some work (one/many machine, heterogeneous multicore, different machines in manufacturing, etc.). Since they are finite we represent them by $\mathcal{R} = \{1, 2, \ldots, m\}$;
- a *time set* $\mathcal{T}$, over which the scheduling is performed (typically $\mathbb{N}$ or $[0, \infty)$);
- a *scheduling algorithm* $\mathcal{A}$ which produces a *schedule* S for given $\mathcal{R}$ and $\mathcal{N}$.

Characteristics of tasks:

- amount of work,
- recurrent/non-recurrent, does a task repeat over time? How often?
- on-line/off-line, do we know the parameters in advance?
- precedence constraints,
- deadlines, "the work must be completed by this instant",
- affinity, not all resources are the same,
- sequential (only one resource at time), parallel (more than one resource at time), parallelizable (one or more resources at time).

Characteristics of a resource:

- type, (coffee machine $\neq$ a CPU)
- execution rate r_k (speed), which may be time-varying or demand-varying;

- operating modes (variable speed over time, etc.)

A schedule is a function

$$S : \mathcal{R} \times \mathcal{T} \to \mathcal{N} \cup \{0\}$$

If $m = 1$ then just $S : \mathcal{T} \to \mathcal{N} \cup \{0\}$.

- If $S(k,t) = i$ then the resource k is assigned to the i-task at time t.
- If $S(k,t) = 0$ then the resource k is not assigned at time t (we say that the k-th resource is *idle* at t).
- This definition of schedule implies that at every instant t each resource is assigned to at most one task.
- Conversely, at every instant each task may be assigned any number of resources in $\mathcal{R}$.

The inverse image of i under S, $s_i \subseteq \mathcal{R} \times \mathcal{T}$, that is

$$s_i = S^{-1}(i) = \{(k,t) \in \mathcal{R} \times \mathcal{T} : S(k,t) = i\}$$

represents the resources allocated to the i-th task.

Goal of scheduling algorithm $\mathcal{A}$: find a schedule S such that:

- the constraints are met (in this case constraints have to be specified): all task deadlines are met,
- some target function is minimized/maximized (minimum makespan/delay, best "performance": requires to know how the timing affect the "performance")

Characteristics of scheduling algorithms:

- (non-)work-conserving: no idle resource if pending tasks exist;
- (non-)preemptive, I can interrupt a task while it executes;
- time-complexity: how long does it take to decide the resource assignment?

Examples of scheduling algorithms:

- First In First Out (FIFO), schedule tasks in order of arrivals;
- Round Robin (RR), divide the time in slices and assign slices in round;
- Shortest Job First (SJF) and its preemptive version Shortest Remaining Time First (SRTF);
- Earliest Deadline First (EDF), assigns priority according to the deadlines d;

- Least Laxity First (LLF), aka Least Slack Time (LST), at t assigns priority according to the smallest "laxity" $(d-t)-c'$
- Fixed Priorities (FP), tasks are prioritized.

Definition 1 *A task set $\mathcal{N}$ is* feasible *is it exists a schedule which satisfies the task constraint.*

Definition 2 *A task set $\mathcal{N}$ is* schedulable *by the scheduling algorithm $\mathcal{A}$, if $\mathcal{A}$ can produce a schedule S which does not violate any constraint of $\mathcal{N}$.*

Obviously: schedulability by any algorithm implies feasibility.

1.1 Real-Time Scheduling

Many books exist for surveys of non-real-time scheduling [26, 47]. From now on we focus on RT scheduling.

- Tasks have a an execution requirement C_i;
- Tasks are recurrent and activated *sporadically*: with a minimum interarrival (or *period*) T_i; Each activation of a task is called *job* (job $\neq$ task);
- Tasks have a *relative deadline* D_i, relative (if $D_i = T_i$ then *implicit deadline*, if $D_i \leq T_i$ then *constrained deadline*, if D_i unrelated to T_i then *arbitrary deadline*);

Also the quantity $U_i = C_i/T_i$ is called task utilization and represents the fraction of time needed by task i.

More elaborated task model do exist. Below some references.

- Models for the execution time: Stigge et al. proposed the Di-Graph RT Task Model [58], which is probably the richest task model with a tractable analysis. In the paper, which contains also a comparison with other task models such as the multiframe task model [45], the recurring RT task model [9], and the non-cyclic generalized multiframe [59], the EDF analysis is described. In reward-based scheduling [7] the longer a task executes the higher system utility is achieved. Probabilistic execution times were also considered [32] even in the case of dependence among the distributions [13].
- Models for the activation pattern: the rate-based task model specifies the number of task activations in an interval [37], in the event stream model the number of task activations are lower and upper bounded by functions [51], Velasco et al. [60] analyzed the activation pattern of control tasks activated by events.
- Models for the deadline: a deadline model in which only m deadlines have to be guaranteed among any k consecutive ones [49, 48].

Resources: single processor, multiprocessor (with m processors/cores).
A necessary condition for feasibility is:

$$\sum_{i=}^{n} U_i \leq m \tag{1}$$

From now on we focus on single processor only ($m = 1$).

Below we report a list of the most relevant works on multiprocessor RT scheduling.

- Baruah et al. [11] introduced the notion of fairness and proposed the scheduling algorithm *P-fair* which can always produce a feasible schedule, if (1) is true. Later, Anderson and Srinivasan [2] extended to sporadic tasks. P-fair requires to divide the time in quanta, hence it may have a high overhead is the cost of preemptions is not negligible.

- Other algorithms which can feasibly schedule a task set, provided that (1) is true, with fewer preemption than P-fair, are Edf with task splitting and K processors in a Group (EKG) [5], Local Largest Remaining Execution time First (LLREF) [27], and Reduction to UNiprocessor (RUN) [50].

- A simpler scheduling algorithm is certainly Global EDF (EDF). However, GEDF cannot schedule all tasks satisfying (1). Hence some schedulability tests are needed [35, 8, 14, 23]. Some variations of GEDF can significantly improve the capacity to schedule task sets. Examples are: EDF-US [57] and Earliest Deadline Zero Laxity (EDZL) [29]. If a task set with implicit deadlines with $\sum_i U_i = m$ is scheduled by GEDF, clearly some deadline will be missed. However it is possible to estimate the amount of deadline violation [31, 33], which often may be acceptable.

- Examples of global fixed priority scheduling over multiprocessor are RM-US [4] and SM-US [3].

2 Fixed Priority (FP): basics

- Tasks are sorted in decreasing priority order
  - τ_1 is the highest priority one,
  - τ_n is the lowest priority one.

Theorem 1 (Liu, Layland, 1973 [44]) *If $D_i = T_i$ then Rate Monotonic (RM) is **optimal**: if some priority assignment can schedule the task set, then RM can schedule the task set.*

Theorem 2 (Leung, Whitehead, 1982 [41]) *If $D_i \leq T_i$ then Deadline Monotonic (DM) is optimal.*

Liu and Layland [44] also proved the most popular *utilization upper bound* (checked on 24/10/2012: cited 7914 in Google Scholar).

Theorem 3 *If* $D_i = T_i$

$$\sum_{i=1}^{n} U_i \leq n(\sqrt[n]{2} - 1)$$

then $\mathcal{N}$ *is schedulable by RM.*

- The RHS is called *utilization upper bound*.
- As $n \to \infty$ the bound tends to $\log 2 \approx 0.69315$
- Is the LL bound tight? Is there any non-schedulable task set with $\sum_i U_i > U_{\mathsf{LL}}$? Draw the example.

Theorem 4 (Hyperbolic Bound [19]) *If* $D_i = T_i$ *and*

$$\prod_{i=1}^{n} (1 + U_i) \leq 2$$

then $\mathcal{N}$ *is schedulable by RM.*

- Visualization of the bound and interpretation of HB in the utilization space.
- Still some space for uncertainty. What happens in between?

3 FP exact analysis: response time

Definition 3 *The* response time R_i *of task* τ_i *is the longest time that can elapse from the activation of any job to its completion.*

$$R_i = \max_{j \geq 1} \{R_{i,j}\}$$

with $R_{i,j}$ *response time of the* j*-th job of* τ_i*.*

The idea: to compute the response time and check whether or not $R_i \leq D_i$

- if so, then all jobs of τ_1 will meet their deadline (schedulable by FP).
- if not, then some job will miss its deadline (not schedulable by FP).

Definition 4 *We define the* level-i interference $I_i(t)$ *as the maximum amount of work which can be requested by tasks with priority higher than* i *in an interval of length* t*.*

For our simple task model, it is

$$I_i(t) = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$$

and it corresponds to the scenario with all tasks activated together at 0 at the highest possible rate.

- Example of interference with different activation patterns (two alternating periods)
- The response time of the first job of τ_i is found [38, 6] as the smallest fixed point of the following equation

$$\begin{cases} R_{i,1}^{(0)} = C_i \\ R_{i,1}^{(k+1)} = C_i + I_i(R_{i,1}^{(k)}) \end{cases} \tag{2}$$

- It converges iff $\sum_{j=1}^{i-1} U_j < 1$
- Explain its rationale.
- if $R_{i,1} \leq T_i$ then $R_i = R_{i,1}$ is the longest response time among all jobs belonging to τ_i.
- otherwise ($R_{i,1} > T_i$) it is not guaranteed that the maximum job response time occurs at the first job! In such a case we have to compute the response time $R_{i,k}$ of all subsequent jobs [39].
- We care only if $D_i > T_i$ (arbitrary deadline)
- We can compute the *absolute job response time* $r_{i,j}$ as follows

$$\begin{cases} r_{i,1} = R_{i,1} \\ r_{i,j}^{(0)} = r_{i,j-1} + C_i \\ r_{i,j}^{(k+1)} = j\, C_i + I_i(r_{i,j}^{(k)}) \\ R_{i,j} = r_{i,j} - (j-1)T_i \end{cases} \tag{3}$$

  until $r_{i,j} \leq j\ T_i$.
- The interval $[0, r_{i,j^*}]$, with j^* equal to the index of job where it first is $r_{i,j} \leq j\ T_i$, is called *level-i busy period*.
- If $\sum_{j=1}^{i} U_j < 1$, j^* is finite
- If $\sum_{j=1}^{i} U_j > 1$, $\lim_j R_{i,j} = \infty$ (*level-i overload*)
- If $\sum_{j=1}^{i} U_j = 1$ and $\{T_1, \ldots, T_i\}$ rational, j^* finite because the schedule will repeat after the least common multiple of the periods

- If $\sum_{j=1}^{i} U_j = 1$ and $\{T_1, \ldots, T_i\}$ irrational, j^* infinite, but R_i can still be defined as $R_i = \sup_j R_{i,j}$
- In human cases $R_i = \max_{j \leq j^*} R_{i,j}$ and we can check $R_i \leq D_i$.
- Iterating the response time equation Eq. (2) may be too time consuming, especially if it has to be executed on-line.
- One may want to forget necessity by computing a response time upper bound.
- Suppose we have a linear upper bound of the interference $I_i(t) \leq \overline{I}_i(t) = \alpha_i t + \beta_i$. Then from (2)

$$\begin{aligned} R_i &\leq C_i + \overline{I}_i(R_i) = C_i + \alpha_i R_i + \beta_i \\ R_i - \alpha_i R_i &\leq C_i + \beta_i \\ R_i &\leq \frac{C_i + \beta_i}{1 - \alpha_i} = \overline{R}_i \end{aligned}$$

  and have the following as a just sufficient (faster) test

$$\forall i, \quad \overline{R}_i \leq D_i$$

Finding suitable α_i and β_i. By upper bounding the $\lceil x \rceil$ with $x + 1$ in $I_i(t)$ we quickly find

$$\alpha_i = \sum_{j=1}^{i-1} U_j \qquad \beta_i = \sum_{j=1}^{i-1} C_j$$

however β_i can be made [22] a bit tighter by choosing

$$\beta_i = \sum_{j=1}^{i-1} C_j (1 - U_j)$$

this bound is still valid in the arbitrary deadline case.

4 FP: scheduling points

- The drawback of the response-time test is its "black-box" nature: all task parameters are specified and we get a yes/no answer.
- In reality, it is often more desirable to have a margin on the system parameters that guarantee schedulability: *sensitivity analysis*.

An alternate exact test is the following one.

Theorem 5 (Lehoczky et al. [40]) *A constrained deadline (with $D_i \leq T_i$) task set is schedulable by FP if and only if*

$$\forall i \in \mathcal{N}, \ \exists t \in [0, D_i], \quad C_i + I_i(t) \leq t$$

Interesting, but not so practical (how do we check if it exists a point in a real interval?)

If we remember the expression of $I_i(t)$ we realize that the previous condition is equivalent to the following one, which can be better managed

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{S}_i, \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t$$

with

$$\mathcal{S}_i = \{kT_j : k \in \mathbb{N},\ 0 < kT_j < D_i,\ j < i\} \cup \{D_i\}$$

This is the set of *scheduling points*.

However the points in $\mathcal{S}_i$ can still be many and period dependent, especially when the periods of the higher priority tasks are significantly smaller than T_i.

- Can we remove points from $\mathcal{S}_i$ to reduce the complexity?
- By removing arbitrarily points we may lose necessity.

Theorem 6 (Hyperplanes Exact Test [17]) *A constrained deadline (with $D_i \leq T_i$) task set is schedulable by FP if and only if*

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t \tag{4}$$

with $\mathcal{P}_j(t)$ being a set recursively defined as

$$\begin{cases} \mathcal{P}_0(t) = \{t\} \\ \mathcal{P}_j(t) = \mathcal{P}_{j-1}\left(\left\lfloor \frac{t}{T_j} \right\rfloor T_j\right) \cup \mathcal{P}_{j-1}(t). \end{cases} \tag{5}$$

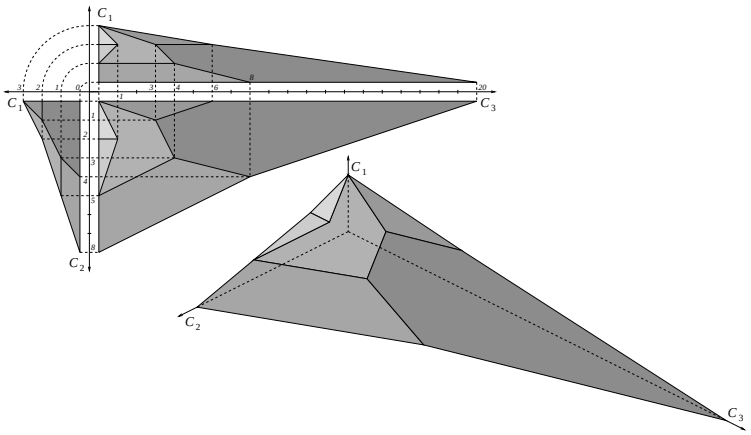
Show how $\mathcal{P}_{i-1}(D_i)$ is computed, for two tasks and U-plane.

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq t$$

If $T_1 = 3, T_2 = 8, T_3 = 20$, and $D_i = T_i$ the schedulable C_i are

5 FP: Sensitivity Analysis

- If the processor runs at speed r, then all computation times become C_i/r
- From the scheduling point condition it is not difficult [21] to find the smallest speed that guarantee FP-schedulability



$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad \frac{C_i}{r} + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil \frac{C_j}{r} \leq t$$

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad r \geq \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j}{t}$$

$$r \geq \max_{i \in \mathcal{N}} \min_{t \in \mathcal{P}_{i-1}(D_i)} \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j}{t}$$

- (there is no direct way to find it using the response time)
- Example of calculation for two tasks in the plot.

What is the maximum schedulable C_k?

- all tasks τ_i with $i < k$ are unaffected by C_k
- to ensure the schedulability of τ_k it must be

$$C_k \leq \max_{t \in \mathcal{P}_{k-1}(D_k)} (t - \sum_{j=1}^{k-1} \left\lceil \frac{t}{T_j} \right\rceil C_j)$$

  The RHS is the amount of *level-*$(k-1)$ *idle time* in $[0, D_k]$

- to ensure the schedulability of all tasks with lower priority $i > k$ it must be

$$C_k \leq \min_{i=k+1,\ldots,n} \max_{t \in \mathcal{P}_{i-1}(D_i)} \frac{t - (C_i + \sum_{\substack{j=1 \\ j \neq k}}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j)}{\left\lceil \frac{t}{T_k} \right\rceil}$$

- hence $C_k^{\max}$ is the minimum of the two RHS

Since R_k is independent of the deadline D_k, the minimum schedulable deadline is just $D_k = R_k$.

- Task periods appear in the ceiling operator. Not trivial to extract them
- More sophisticated technique needed [21].

Let $T_k^{(i)}$ denote the minimum period of τ_k such that τ_i is schedulable.

- $i < k$, meaningless
- then
$$T_k^{\min} = \max_{i \geq k} T_k^{(i)}, \tag{6}$$
- The schedulability of τ_k is not affected by T_k
  - if constrained deadline ($D_k \leq T_k$) then $T_k^{(k)} = D_k$;
  - if implicit deadline ($D_k = T_k$) then $T_k^{(k)} = R_k$.
- Computing $T_k^{(i)}$ with $i > k$ requires some efforts.

Definition 5 *Given the subset of tasks $\mathcal{M} \subseteq \mathcal{N}$, we define* level-$\mathcal{M}$ idle time *in $[0, D]$, denoted by $Y(\mathcal{M}, D)$, the amount of time in $[0, D]$ in which no task in $\mathcal{M}$ is executing, under the worst-case scenario for activations.*

Examples:

- $Y(\varnothing, D) = D$
- $Y(\{\tau_1\}, T_1) = T_1 - C_1$

  To evaluate $T_k^{(i)}$ we need $Y(\{1, \ldots, i\} \setminus \{k\}, D_i)$, because it is the time that can be consumed by τ_k, keeping τ_i schedulable.

- How do we compute $Y(\mathcal{M}, D)$?
$$Y(\mathcal{M}, D) = \max_{t \in \mathcal{P}_{|\mathcal{M}|}(D)} \left\{ t - \sum_{j \in \mathcal{M}} \left\lceil \frac{t}{T_j} \right\rceil C_j \right\}$$
- It is scaring! Let's check.
- Then the maximum number $n_k^{(i)}$ of τ_k (associated to the minimum period $T_k^{(i)}$) jobs which can preserve the schedulability of τ_i then is
$$n_k^{(i)} = \left\lfloor \frac{Y(\{1, \ldots, i\} \setminus \{k\}, D_i)}{C_k} \right\rfloor$$

- Given the maximum number of jobs $n_k^{(i)}$, what is the minimum period $T_k^{(i)}$ which preserves the schedulability of τ_i?
- $T_k^{(i)}$ is such that τ_k has $n_k^{(i)}$ jobs interfering on τ_i, and by increasing $T_k^{(i)}$ by any small amount the number of interfering jobs would increase. Hence

$$T_k^{(i)} = \frac{R_i}{n_k^{(i)}}$$

with R_i response time of τ_i with $n_k^{(i)}$ interfering jobs by τ_k

$$R_i = C_i + n_k^{(i)} C_k + \sum_{\substack{j=1 \\ j \neq k}}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Summarizing

1. For all $i > k$ we compute

$$n_k^{(i)} = \left\lfloor \frac{Y(\{1, \ldots, i\} \setminus \{k\}, D_i)}{C_k} \right\rfloor$$

$$R_i = C_i + n_k^{(i)} C_k + \sum_{\substack{j=1 \\ j \neq k}}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

$$T_k^{(i)} = \frac{R_i}{n_k^{(i)}}$$

$$T_k^{\min} = \max\{D_k, T_k^{(k+1)}, T_k^{(k+2)}, \ldots, T_k^{(n)}\} \text{ if constr dl}$$

$$T_k^{\min} = \max\{R_k, T_k^{(k+1)}, T_k^{(k+2)}, \ldots, T_k^{(n)}\} \text{ if impl dl}$$

If some pessimism is acceptable, we can gain in simplicity by using sufficient tests for the sensitivity

- Using LL bound $U_{\mathsf{LL}} = n(\sqrt[n]{2} - 1)$:
  - Min speed: $r^{\min} = \frac{\sum_i U_i}{U_{\mathsf{LL}}}$
  - Max C_k: $C_k^{\max} = T_k(U_{\mathsf{LL}} - \sum_{j \neq k} U_j)$
  - Min T_k: $T_k^{\min} = \frac{C_k}{U_{\mathsf{LL}} - \sum_{j \neq k} U_j}$

6 FP: optimal design

- 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

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 $\mathcal{X}$.

Definition 6 (Def. 1 in [10]) *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 7 (in [10]) *Exact tests of FP and EDF are sustainable.*

7 FP: optimal execution time

- $\mathcal{X} = \{C_1, \ldots, C_n\}$.
- 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

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.
- 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

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

Chung et al. [28] proposed the imprecise computation task model. In this task model, the longer a task can execute the higher accuracy can be achieved. Aydin et al [7] investigated the execution of an optional amount of computation. In their settings, the longer a job can execute, the higher reward is achieved. Hou and Kumar [36] considered the problem of maximizing the reward of tasks, which can execute chunks of code. To each chunk it is associated a reward. The problem is formulated as an LP problem with an unimodular matrix.

8 FP: optimal period

- $\mathcal{X} = \{T_1, \ldots, T_n\}$
- 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

$$\max_{T_1,\ldots,T_n} J(T_1,\ldots,T_n)$$
$$\text{s.t } \mathcal{N} \text{ is FP-schedulable}$$

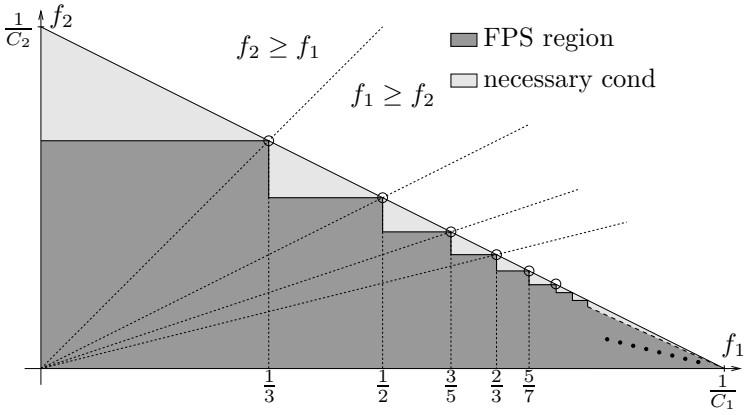
What is the geometry of the FP-schedulable periods?

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.

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.

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?

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

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

How do we enumerate the vertices?

Theorem 8 (Proposition 2.1 in [53]) *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

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

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 $k_j^{(i)} = \left\lceil \frac{t}{T_j} \right\rceil$

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

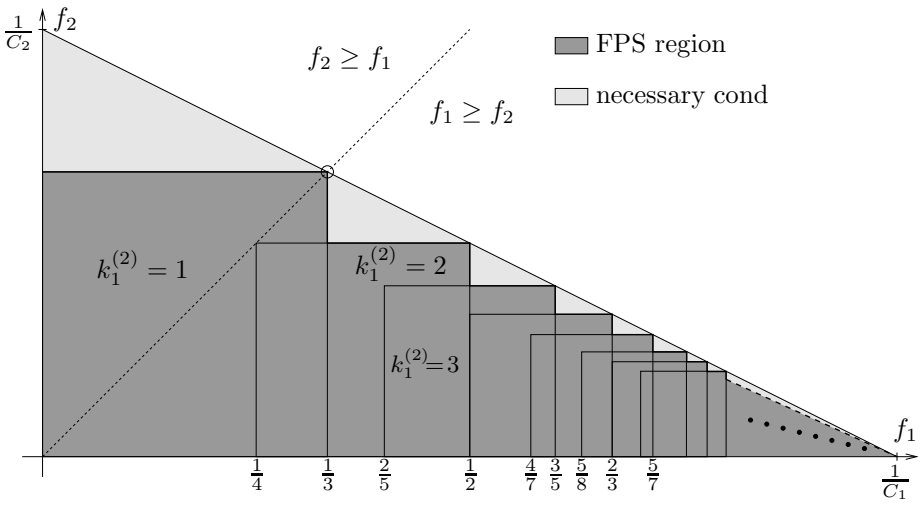
$$\forall i = 1, \ldots, n \quad \exists \mathbf{k}^{(i)} \in \mathbb{N}^{i-1}$$

such that:

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

which are the coordinates of the vertices.

With $C_1 = 1$ and $C_2 = 2$



- A branch and bound algorithm can be used to search for the optimal task frequencies.
- More details can be found in [20].

Seto et al. [54] 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

9 Beyond the period assignment

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
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
- 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?

10 Earliest Deadline First: basics

- Task model is still the same $\tau_i = (C_i, T_i, D_i)$;
- In FP, priorities are per task: all jobs of same task that have the same priority;
- In EDF, priorities are per job: jobs are prioritized according to their absolute deadline.

Theorem 9 (Liu and layland, 1973 [44]) *If a task set is feasible, then it is EDF-schedulable.*

Theorem 10 (Liu and Layland, 1973 [44]) *If $D_i = T_i$ (implicit deadline) then a task set is EDF-schedulable* ***if and only if****:*

$$\sum_{i=1}^{n} U_i \leq 1$$

- Any FP-schedulable task set is also EDF-schedulable task set.

11 EDF: demand bound function

If $D_i \neq T_i$ the schedulability condition becomes more complicated.

Theorem 11 (Lemma 3 in [12]) *The task set $\mathcal{N}$ is EDF-schedulable* ***if and only if****:*

$$\forall t \geq 0 \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t$$

The LHS is called *demand bound function* $\mathsf{dbf}(t)$ of the task set at t.

- $\mathsf{dbf}(t)$ is the maximum amount of work of jobs with both activation and deadline in $[0, t]$.
- no per-task condition: any task task may influence others
- $\max\{0, \cdot\}$ only needed for i with $D_i > T_i$
- Obviously, checking $\forall t > 0$ is not very practical
- By observing the step shape of the dbf we can check only at the points where the steps occur

Theorem 12 (Lemma 3 in [12]) *The task set $\mathcal{N}$ is EDF-schedulable* ***if and only if*** $\sum_i U_i \leq 1$ *and:*

$$\forall t \in \mathcal{D} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t$$

with

$$\mathcal{D} = \{d_{i,k} : d_{i,k} = kT_i + D_i,\ i \in \mathcal{N},\ k \in \mathbb{N},\ d_{i,k} \leq D^*\}$$

and $D^* = \mathsf{lcm}(T_1, \ldots, T_n) + \max_i\{D_i\}$.

$H = \mathsf{lcm}(T_1, \ldots, T_n)$ is often called *hyperperiod* of the task set.

As suggested by Ripoll et al [52] D^* can also be set equal to the longest busy period.

An alternative approach, quite complex though, is to compute the response time of each task as proposed by Spuri [56].

- If $U < 1$, then the for large t the condition is always true
- then D^* can be computed [12] by upper bounding $\mathsf{dbt}(t)$ with a line and we find
$$D^* = \frac{U}{1 - U} \max_i\{T_i - D_i\}$$
- what happen to D^* if $\max_i\{T_i - D_i\} \leq 0$?
- the task set is obviously EDF-schedulable,

- $\max_i\{T_i - D_i\} \leq 0 \quad \Leftrightarrow \quad \forall i,\ D_i \geq T_i$
- EDF is sustainable, hence if $D_i = T_i$ is sched then $D_i \geq T_i$ also sched
- Since $U < 1$, the task set is sched

- All deadlines in $[0, D^*]$ may be too many
- Zhang and Burns [62] proposed the Quick convergence Processor-demans Analysis (QPA)

$d_{\min} \leftarrow \min\{D_i\}$
$t \leftarrow \max\{d_{i,k} : d_{i,k} \leq D^*\}$ ▷ initial assignment
while $\mathsf{dbf}(t) \leq t \wedge \mathsf{dbf}(t) > d_{\min}$ **do**
  if $\mathsf{dbf}(t) < t$ **then**
    $t \leftarrow \mathsf{dbf}(t)$
  else
    $t \leftarrow \max\{d_{i,k} : d_{i,k} < t\}$ ▷ escape from fixed points
  end if
end while
if $\mathsf{dbf}(t) \leq d_{\min}$ **then** task set EDF-schedulable
else task set not EDF-schedulable
end if

12 EDF: sufficient tests

- By replacing T_i with the more conservative value $\min\{T_i, D_i\}$, we find

$$\sum_{i=1}^{n} \frac{C_i}{\min\{T_i, D_i\}} \leq 1$$

  the ratio $\frac{C_i}{\min\{T_i, D_i\}}$ is often called *density* of the task

Devi proposed the following sufficient test

- Assuming that tasks are sorted by non-decreasing relative deadline ($D_i \leq D_2 \leq \ldots \leq D_n$)

Theorem 13 (Theorem 1 in [30]) *The task set $\mathcal{N}$ is EDF-schedulable if:*

$$\forall k = 1, \ldots, n \quad D_k \sum_{i=1}^{k} U_i + \sum_{i=1}^{k} \frac{T_i - \min\{T_i, D_i\}}{T_i} C_i \leq D_k$$

- Proved to strictly dominate the density test
- Albers et al. [1] proposed a Fully Polynomial Time Approximation Scheme.

- The i-th term in $\mathsf{dbf}(t)$ can be upper bounded by

$$\max\{0, \left\lfloor \frac{t - D_i + T_i}{T_i} \right\rfloor\} C_i \leq \mathsf{dub}_i(k,t)$$
$$= \begin{cases} \max\{0, \left\lfloor \frac{t-D_i+T_i}{T_i} \right\rfloor\} C_i & t \leq d_{i,k} = (k-1)T_i + D_i \\ U_i(t + T_i - D_i) & t > d_{i,k} \end{cases}$$

  so that

$$\frac{k}{k+1} \sum_i \mathsf{dub}_i(k,t) \leq \mathsf{dbf}(t) \leq \sum_i \mathsf{dub}_i(k,t)$$

- This enables a quite interesting result

  Theorem 14 *If $U \leq 1$ and*

$$\forall t \in \mathcal{D}(\overline{k}) = \{d_{i,k} \in \mathbb{R}^+ : d_{i,k} = (k-1)T_i + D_i, 1 \leq k \leq \overline{k}\}$$
$$\sum_{i=1}^{n} \mathsf{dub}_i(\overline{k}, t) \leq t$$

  then the task set is schedulable.
  Otherwise it is not schedulable over a CPU with speed $\frac{\overline{k}}{\overline{k}+1}$.

- In this way only $n\overline{k}$ evaluation of the dbf are needed.
- We can trade accuracy vs. complexity. As $\overline{k} \to \infty$ it becomes necessary and sufficient.
- FPTAS

Similar results can be found for FP [34].

13 EDF: sensitivity analysis

Similarly as in the FP case we can find [16] the minimum EDF-schedulable speed as follows

$$r^{\mathsf{min}} = \max_{t \in \mathcal{D}} \frac{\sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor\right\} C_i}{t}$$

However $D^* = H + \max_i\{D_i\}$, because we are changing the speed and then altering the linear upper bound that motivates the expression

$$D^* = \frac{U}{1-U} \max_i\{T_i - D_i\}$$

The maximum EDF-schedulable $C_k^{\max}$ can be computed in a similar way as in FP

$$C_k^{\max} = \min_{t \in \mathcal{D}, t \geq D_k} \frac{t - \sum_{\substack{i=1 \\ i \neq k}}^{n} \max\left\{0, \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor\right\} C_i}{\left\lfloor \frac{t+T_k-D_k}{T_k} \right\rfloor}$$

Here too, $D^* = H + \max_i\{D_i\}$.

14 EDF: space of comp times

Theorem 15 (Lemma 3 in [12]) *$\mathcal{N}$ is EDF-schedulable **if and only if** $\sum_i U_i \leq 1$ and:*

$$\forall t \in \mathcal{D} \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t$$

with

$$\mathcal{D} = \{d_{i,k} : d_{i,k} = kT_i + D_i,\ i \in \mathcal{N},\ k \in \mathbb{N},\ d_{i,k} \leq D^*\}$$

and $D^* = \mathsf{lcm}(T_1, \ldots, T_n) + \max_i\{D_i\}$.

$H = \mathsf{lcm}(T_1, \ldots, T_n)$ is often called the *hyperperiod*.

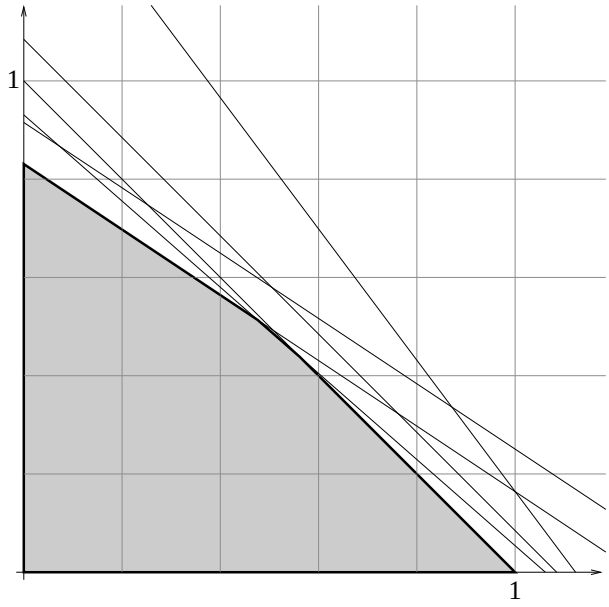
- Since we are investigating the space of C_i, no smaller D^* can be considered.
- For given T_i and D_i, the space of EDF-schedulable C_i is convex!

Let us assume $T_1 = 4, D_1 = 5$ and $T_2 = 6, D_2 = 5$.

$$\mathcal{D} = \{5, 9, 11, 13, 17\}$$

Hence equations are

$$\begin{bmatrix} 4 & 6 \\ 8 & 6 \\ 8 & 12 \\ 12 & 12 \\ 16 & 18 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 9 \\ 11 \\ 13 \\ 17 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



Hence only the equations

$$\begin{bmatrix} 4 & 6 \\ 16 & 18 \\ 1 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \leq \begin{bmatrix} 5 \\ 17 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

are needed to be checked, corresponding to $\mathcal{D} = \{5, 17\}$

- Finding a general method to reduce $\mathcal{D}$ without losing sufficiency would be an interesting contribution.
- Being the exact region convex the optimal assignment of computation times C_i does not present big difficulties.

15 EDF: space of deadlines, periods

- Example: $n = 2$, $T_1 = T_2$, $U_1 + U_2 \leq 1$
- We can reduce D_1 to C_1, but then $D_2 = C_1 + C_2$, or
- $D_2 = C_2$, but then $D_1 = C_1 + C_2$
- convex comb of these two points are not EDF-sched
- Let us assume $T_1 = T_2 = \ldots = T_n$, and $\sum_i U_1 \leq 1$.

- For any permutation $p : \mathcal{N} \to \mathcal{N}$, a "vertex" has coordinates

$$D_{p(i)} = \sum_{j=1}^{i} C_{p(j)}$$

Being the periods and deadlines into the set $\mathcal{D}$ and into the $\lfloor \cdot \rfloor$, dbf condition is unfit to show the space of feasible periods/deadline.

Theorem 16 (Theorem 1 in[18], Theorem 2 in [15]) *The task set $\mathcal{N}$ is EDF-schedulable* ***if and only if****:*

$$\forall \mathbf{k} \in \mathbb{N}^n \setminus \{\mathbf{0}\} \quad \exists i \in I_{\mathbf{k}} \quad \sum_{j=1}^{n} C_j k_j \leq (k_i - 1)T_i + D_i$$

where

$$I_{\mathbf{k}} = \{j : k_j \neq 0\}$$

is the set of non-zero indexes in $\mathbf{k}$*.*

$$\forall \mathbf{k} \in \mathbb{N}^n \setminus \{\mathbf{0}\} \quad \exists i : k_i \neq 0 \quad \sum_{j=1}^{n} C_j k_j \leq (k_i - 1)T_i + D_i$$

can be seen as the problem of finding a cover to $\mathbb{N}^n \setminus \{\mathbf{0}\}$ with halfspaces of equations

$$(C_i - T_i)k_i + \sum_{j \neq i} C_j k_j \leq -T_i + D_i$$

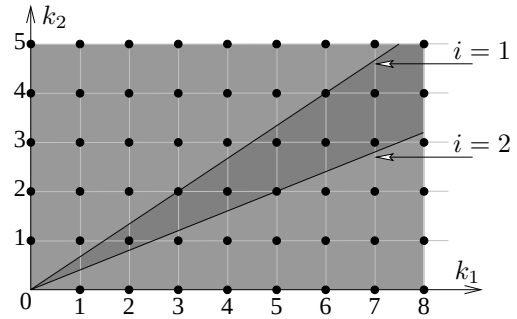
considering $(k_1, \ldots, k_n)$ as variables.

- Remember: $\exists i : k_i \neq 0$, not just $\exists i \in \mathcal{N}$
- If $U = 1$ linearly dependent.

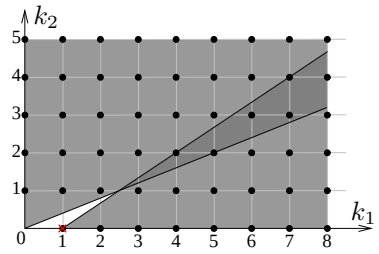
If $T_1 = D_1 = 4$, $C_1 = 2$, and $T_2 = D_2 = 8$, $C_2 = 3$, then

$$\begin{aligned} -2k_1 + 3k_2 &\leq 0 \\ 2k_1 - 5k_2 &\leq 0 \end{aligned}$$

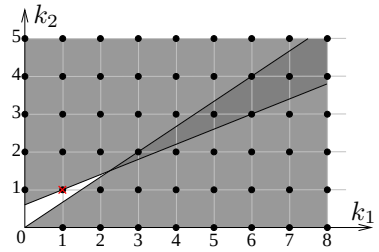
have to cover $\mathbb{N}^2 \setminus \{(0,0)\}$



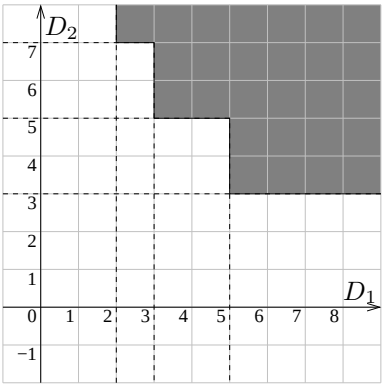
- If $U > 1$ cannot cover, ever.
- Deadline D_i appears at the RHS of the i-th inequality only
- Changing D_i means to translate boundary of the i-th halfspace



$D_1^{\min} = 2 \quad (= C_1)$



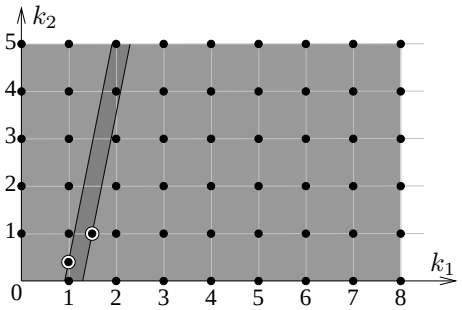
$D_2 = 5$



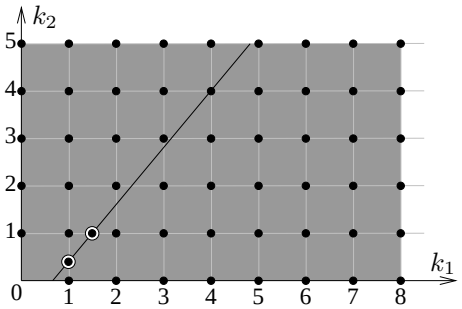
- The i-th period has an effect only on the i-th equation
- however T_i appears in the coefficients and at the RHS
- changing T_i is a rotation of the i-th halfspace
- the points in common between two equations with T_i' and T_i'' are

$$(C_i - T_i')k_i + \sum_{j \neq i} C_j k_j = -T_i' + D_i$$
$$(C_i - T_i'')k_i + \sum_{j \neq i} C_j k_j = -T_i'' + D_i$$
$$k_i = 1$$
$$\sum_{j \neq i} C_j k_j = D_i - C_i$$

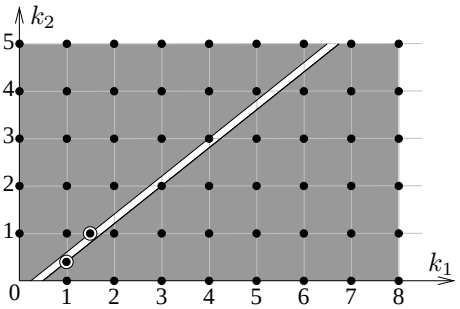
- As T_1 changes, the 1st boundary rotates around
$$k_1 = 1, \quad k_2 = \frac{D_1 - C_1}{C_2}$$
- As T_2 changes, the 2nd boundary rotates around
$$k_1 = \frac{D_2 - C_2}{C_1}, \quad k_2 = 1$$
- $C_1 = 2$, $D_1 = 4$, and $C_2 = 5$, $D_2 = 8$
- changing T_1 rotates around $(1, 0.4)$
- changing T_2 rotates around $(1.5, 1)$



- $T_1 = 27$
- $T_2 = {}^{27}/_{5}$



- $T_1 = 8$
- $T_2 = {}^{20}/_{3}$



- $T_1 = 6$
- $T_2 = {}^{15}/_{2}$

By making some more restrictive hypothesis, we can derive the following sufficient EDF-test

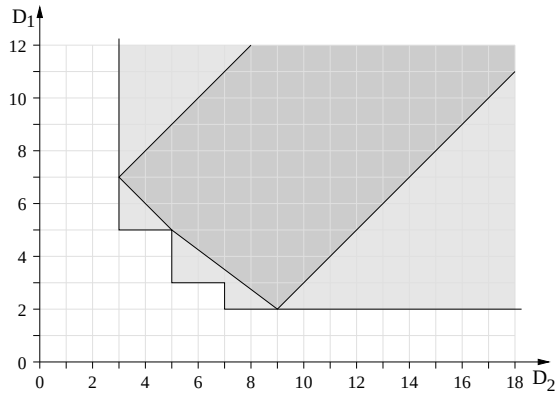
Theorem 17 (Chantem et al. [25]) *A task set is EDF-schedulable if:*

$$\begin{cases} \forall i, \quad D_i \leq T_i + D_{\min} \\ \sum_{i=1}^{n} U_i \, D_i \geq \sum_{i=1}^{n} C_i - D_{\min} \, (1 - \sum_{i=1}^{n} U_i). \end{cases}$$

with $D_{\min} = \min_i D_i$.

This condition has the advantage of being convex in the space of deadlines.

If $C_1 = 2$, $T_1 = 4$ and $C_2 = 4$, $T_2 = 7$, we have



- $\mathcal{X} = \{T_1, D_1, \ldots, T_n, D_n\}$
- Typical problem in control systems:

- given the execution times $\{C_1, \ldots, C_n\}$
- choose the sampling periods and deadlines (=delays) of all controllers
- such that a control cost (LQG?) is minimized

$$\min_{T_1, D_1, \ldots, T_n, D_n} J(T_1, D_1, \ldots, T_n, D_n)$$
$$\text{s.t } \mathcal{N} \text{ is EDF-schedulable}$$

We could try [61] to find optimal T_i and then D_i

1. Suppose $C_1 = C_2 = 1$
2. Suppose that the optimal T_i are $T_1 = 2.1$, $T_2 = 1.91$
3. How much can we reduce the deadlines?
   - very little.
4. However by choosing non-optimal periods $T_1 = T_2 = 2$, we can actually reduce one of the two deadlines by 1, possibly making the cost J smaller

- To best of my knowledge it is still unsolved
- (due to bizarre shape of the feasible region)
- solving this problem optimally would be a decent contribution

16 Hierarchical Scheduling: intro

- Applications often need to be isolated, otherwise a misbehaviour on one (such as taking longer than expected) can cause misbehaviors on other applications
- Example of high priority task executing for longer
- The execution of each application is controlled by a mechanism (aka server, resource reservation), which prevents the application from running more than planned
- Such a mechanism provides the abstraction of a *virtual resource*: a resource which is not always fully available
- How can we guarantee the timing constraints in presence of such a mechanism?

17 System model

- We model an application by a set of n tasks (C_i, T_i, D_i)
  - C_i computation time
  - T_i period
  - D_i deadline
- All tasks are scheduled by some scheduling algorithm (FP, EDF, etc.) **over a the virtual resource**. Such a scheduling alg. is often called *local scheduler* (local to the virtual resource)
- How to we check schedulability over virtual resources?
- The model of a virtual resource must capture the "not-fully-available" characteristic.

Definition 7 (Compare with [46, 42, 55, 43]) *We define the* supply bound function $\mathsf{sbf}(t)$ *of a virtual resource as the minimum amount of execution time available in any interval of length t.*

- Let us assume a virtual resource that provides execution time (to the application) in
$$\bigcup_{k\in\mathbb{Z}} [4k, 4k+1]$$
- What is its $\mathsf{sbf}(t)$?
- Remember: "any interval of length t" not necessarily $[0, t]$
- Let us now assume that the virtual resource is implemented by a periodic server with period (=deadline) P, time budget Q
- What is its $\mathsf{sbf}(t)$
- Remember: scenario of minimum possible supply must be assumed
$$\mathsf{sbf}(t) = \begin{cases} 0 & t \in [0, P-Q] \\ (k-1)Q & t \in (kP-Q, (k+1)P-2Q] \\ t-(k+1)(P-Q) & \text{otherwise} \end{cases}$$
with $k = \left\lceil \frac{t-(P-Q)}{P} \right\rceil$.
- This example explains why this is often called *hierarchical scheduling* since it is about to schedule a task set within another task
- Let us assume a virtual resource that provides execution time (to the application) in
$$[2,3] \cup [5,7] \cup [10,12] \quad \text{with period } 12$$
- Remember: the interval with the minimum supply can start at different points for different t

18 Schedulability conditions in hierarchical scheduling

Theorem 18 (FP-schedulability [42]) *A constrained deadline (with $D_i \leq T_i$) task set is FP-schedulable over a virtual resource with* $\mathsf{sbf}(t)$, ***if and only if***

$$\forall i \in \mathcal{N},\ \exists t \in \mathcal{P}_{i-1}(D_i), \quad C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j \leq \mathsf{sbf}(t)$$

Theorem 19 (EDF-schedulability [55]) *The task set $\mathcal{N}$ is EDF-schedulable over a virtual resource with* $\mathsf{sbf}(t)$, ***if and only if:***

$$\forall t \geq 0 \quad \sum_{i=1}^{n} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq \mathsf{sbt}(t)$$

- Testing with the exact $\mathsf{sbf}(t)$, for example

$$\mathsf{sbf}(t) = \begin{cases} 0 & t \in [0, P-Q] \\ (k-1)Q & t \in (kP-Q, (k+1)P-2Q] \\ t-(k+1)(P-Q) & \text{otherwise} \end{cases}$$

  with $k = \left\lceil \frac{t-(P-Q)}{P} \right\rceil$,
  may be too complicated.

- It is safe (sufficient) to make the test by using a **lower** bound to $\mathsf{sbf}(t)$

Given a supply bound function $\mathsf{sbf}(t)$, it can be lower bounded [46] by any of the following functions

$$\mathsf{lsbf}(t) = \max\{0, \alpha(t-\Delta)\}$$

with

$$\alpha \leq \lim_{t\to\infty} \frac{\mathsf{sbf}(t)}{t}$$

and

$$\Delta = \sup_{t\geq 0} \left\{ t - \frac{\mathsf{sbf}(t)}{\alpha} \right\}$$

Typically we take $\alpha = \ldots$.

- α is often called *bandwidth* of the virtual resource;
- Δ is often called *delay* of the virtual resource.
  - $\Delta \geq \sup\{t : \mathsf{sbf}(t) = 0\}$ always.

1. What is the $\mathsf{lsbf}(t)$ of a periodic (Q, P) server?

$$\alpha = \frac{Q}{P}, \qquad \Delta = 2(P - Q)$$

2. What is the $\mathsf{lsbf}(t)$ of

$$[1, 2] \cup [3, 6] \quad \text{with period } 6$$

$$\alpha = \frac{5}{12}, \qquad \Delta = 1.5 \ (> \text{longest idle time})$$

1. Can we schedule by FP the two (implicit deadline) tasks $C_1 = 2$, $T_1 = 7$ and $C_2 = 2$, $T_2 = 15$, over a virtual resource implemented by a periodic server with period $P = 4$ and budget $Q = 2$?
2. Is it FP-schedulable over the virtual resource, if it is abstracted by the lsbf?
3. Is it EDF-schedulable over the virtual resource, if abstracted by the sbf?
4. Is it EDF-schedulable over the virtual resource, if abstracted by the lsbf?

19 Designing a virtual resource

- Often the task set is given
- We just have to design the virtual platform parameters such that the application is schedulable, and
- the minimum amount of (real, physical) resource is consumed (the bandwidth α is minimal).

For any pair (t, w), $t, w \geq 0$, let us define the region of feasible parameters as

$$\begin{aligned} F_{\alpha,\Delta}(t, w) &= \{(\alpha, \Delta) \in \mathbb{R}^2 : \mathsf{lsbf}(t) \geq w, \alpha \geq 0\} \\ &= \{(\alpha, \Delta) \in \mathbb{R}^2 : \alpha(t - \Delta) \geq w, \alpha \geq 0\} \end{aligned}$$

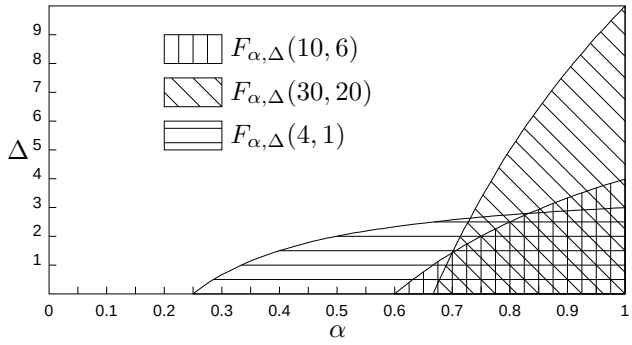
If local scheduler is EDF, then the problem is:

$$\begin{aligned} &\mathsf{minimize}\ J(\alpha, \Delta) \\ &\quad \mathsf{s.t.}\ (\alpha, \Delta) \in \bigcap_{t \in \mathcal{D}} F_{\alpha,\Delta}(t, \mathsf{dbf}(t)) \end{aligned}$$

If local scheduler is FP, then the problem is:

$$\begin{aligned} &\mathsf{minimize}\ J(\alpha, \Delta) \\ &\quad \mathsf{s.t.}\ (\alpha, \Delta) \in \bigcap_{i \in \mathcal{N}} \bigcup_{t \in \mathcal{P}_{i-1}(D_i)} F_{\alpha,\Delta}(t, C_i + I_i(t)) \end{aligned}$$

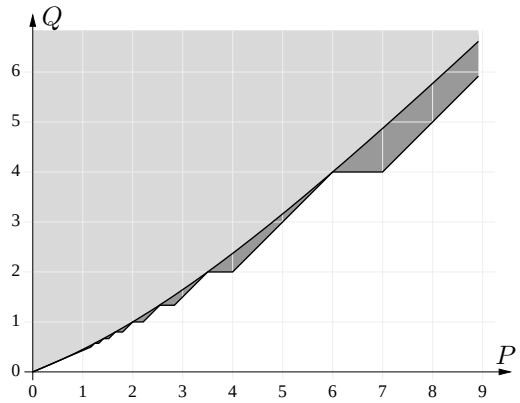
- $F_{\alpha,\Delta}(t,w)$ is convex



- If local scheduler is EDF the feasible region is convex.
- What is the cost function that it is reasonable to minimize?
- $J = \alpha$
  - good motivation (as little bandwidth as possible)
  - however, it may lead to impractical implementations
  - $J = \alpha$ implies that $\Delta^{\text{opt}} = 0$
  - $\Delta = 0$ can never be physically achieved. For example, if a periodic server is used, then
$$P = \frac{\Delta}{2(1-\alpha)}$$
- To prevent $\Delta = 0$ cases, we can consider to minimize the really consumed bandwidth
$$J = \underbrace{\alpha}_{\text{useful}} + \underbrace{\frac{c}{P}}_{\text{wasted}} = \alpha + 2c\frac{1-\alpha}{\Delta}$$
with c equal to some overhead. It is quasiconvex [24], hence good for optimization.
- lsbf is a lower bound, hence the previous analysis is not optimal. Exact analysis should consider the exact shape of the sbf

$$F_{P,Q}(t,w) = \{(P,Q) \in \mathbb{R}^2 : \mathsf{sbf}_{(Q,\,P)\text{ server}}(t) \geq w\}$$

In the figure below $F_{P,Q}(10,4)$



If local scheduler is EDF, then the problem is:

$$\begin{aligned} &\mathsf{minimize} \frac{Q}{P} + \frac{c}{P} \\ &\quad \mathsf{s.t.}\ (P,Q) \in \bigcap_{t \in \mathcal{D}} F_{P,Q}(t, \mathsf{dbf}(t)) \end{aligned}$$

If local scheduler is FP, then the problem is:

$$\begin{aligned} &\mathsf{minimize} \frac{Q}{P} + \frac{c}{P} \\ &\quad \mathsf{s.t.}\ (P,Q) \in \bigcap_{i \in \mathcal{N}} \bigcup_{t \in \mathcal{P}_{i-1}(D_i)} F_{P,Q}(t, C_i + I_i(t)) \end{aligned}$$

Level sets of the cost function are lines going through $P = 0$ and $Q = -c$.

20 Conclusions

- Hierarchical scheduling is well motivated by the need to compose application which are designed and guaranteed in isolation
- The optimal selection of the parameters of the virtual resource was discussed
  - EDF over lsbf easy because convex
  - FP over lsbf complicated, but approachable
  - EDF over sbf (exact supply) messy
  - FP over sbf messy mess
- With the simplest possible task model
- FP over uniprocessor resource
  - schedulability analysis

  - sensitivity analysis
  - optimal design of the task set
- EDF over uniprocessor resource
  - schedulability analysis
  - sensitivity analysis
  - optimal design of the task set
- FP+EDF over uniprocessor virtual resource
  - schedulability analysis
  - optimal design of the virtual resource
- If you want to publish in this area, you can complicate the task model, the scheduling algorithm, or the resource model according to your taste (possibly getting closer to the reality).

References

[1] Karsten Albers and Frank Slomka. Efficient feasibility analysis for real-time systems with edf scheduling. In *Proceedings of Design, Automation and Test in Europe*, pages 492–497, 2005.

[2] James H. Anderson and Anand Srinivasan. Mixed pfair/erfair scheduling of asynchronous periodic tasks. In *Proceedings of the* 13^{th} *Euromicro Conference on Real-Time Systems*, pages 76–85, 2001.

[3] Björn Andersson. Global static-priority preemptive multiprocessor scheduling with utilization bound 38%. In Theodore P. Baker, Alain Bui, and Sébastien Tixeuil, editors, *Principles of Distributed Systems*, volume 5401 of *Lecture Notes in Computer Science*, pages 73–88. Springer Berlin, Heidelberg, 2008.

[4] Björn Andersson, Sanjoy K. Baruah, and Jan Jonsson. Static-priority scheduling on multiprocessors. In *Proceedings of the* 22^{nd} *IEEE Real-Time Systems Symposium*, pages 193–202, London, U.K., December 2001.

[5] Björn Andersson and Eduardo Tovar. Multiprocessor scheduling with few preemptions. In *Proceedings of the* 12^{th} *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 322–331, 2006.

[6] Neil C. Audsley, Alan Burns, Mike Richardson, Ken W. Tindell, and Andy J. Wellings. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[7] Hakan Aydin, Rami Melhem, Daniel Mossé, and Pedro Mejía-Alarez. Optimal reward-based scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, 50(2):111–130, February 2001.

[8] Theodore P. Baker. An analysis of edf schedulability on a multiprocessor. *IEEE Transactions on Parallel and Distributed Systems*, 16(8):760–768, August 2005.

[9] Sanjoy K. Baruah. Dynamic- and static-priority scheduling of recurring real-time tasks. *Real-Time Systems*, 24(1):93–128, January 2003.

[10] Sanjoy K. Baruah and Alan Burns. Sustainable scheduling analysis. In *Proceedings of the* 27^{rd} *IEEE Real-Time Systems Symposium*, pages 159–168, Rio de Janerio, Brazil, December 2006.

[11] Sanjoy K. Baruah, Neil K. Cohen, Greg Plaxton, and Donald A. Varvel. Proportionate progress: a notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, June 1996.

[12] Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 182–190, Lake Buena Vista (FL), U.S.A., December 1990.

[13] Guillem Bernat, Alan Burns, and Martin Newby. Probabilistic timing analysis: An approach using copulas. *Journal of Embedded Computing*, 1(2):179–194, 2005.

[14] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari. Improved schedulability analysis of EDF on multiprocessor platforms. In *Proceedings of the* 17^{th} *Euromicro Conference on Real-Time Systems*, pages 209–218, Palma de Mallorca, Spain, July 2005.

[15] Enrico Bini. Uniprocessor EDF feasibility is an integer problem. In Jane W.-S. Liu, Rolf H. Möhring, and Kirk Pruhs, editors, *Scheduling*, volume 08071 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.

[16] Enrico Bini, Giorgio Buttazzo, and Giuseppe Lipari. Minimizing CPU energy in real-time systems with discrete speed management. *ACM Transactions on Embedded Computing Systems*, 8(4):31:1–23, July 2009.

[17] Enrico Bini and Giorgio C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53(11):1462–1473, November 2004.

[18] Enrico Bini and Giorgio C. Buttazzo. The space of EDF deadlines; the exact region and a convex approximation. *Real-Time Systems*, 41:27–51, 2009.

[19] Enrico Bini, Giorgio C. Buttazzo, and Giuseppe M. Buttazzo. Rate monotonic scheduling: The hyperbolic bound. *IEEE Transactions on Computers*, 52(7):933–942, July 2003.

[20] Enrico Bini and Marco Di Natale. Optimal task rate selection in fixed priority systems. In *Proceedings of the* 26^{th} *IEEE Real-Time Systems Symposium*, pages 399–409, Miami (FL), U.S.A., December 2005.

[21] Enrico Bini, Marco Di Natale, and Giorgio Buttazzo. Sensitivity analysis for fixed-priority real-time systems. *Real-Time Syst.*, 39(1–3):5–30, 2008.

[22] Enrico Bini, Thi Huyen Châu Nguyen, Pascal Richard, and Sanjoy K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Transactions on Computers*, 58(2):279–286, February 2009.

[23] Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, and Sebastian Stiller. A constant-approximate feasibility test formultiprocessor real-time scheduling. *Algorithmica*, 62:1034–1049, 2012.

[24] Giorgio Buttazzo, Enrico Bini, and Yifan Wu. Partitioning real-time applications over multicore reservations. *IEEE Transactions on Industrial Informatics*, 7(2):302–315, May 2011.

[25] Thidapat Chantem, Xiaobo Sharon Hu, and Michael D. Lemmon. Generalized elastic scheduling. In *Proceedings of the* 27^{th} *IEEE Real-Time Systems Symposium*, pages 236–245, Rio de Janeiro, Brazil, December 2006.

[26] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. *A review of machine scheduling: Complexity, algorithms and approximability*, chapter in Handbook of Combinatorial Optimization, pages 21–169. Kluwer Academic Publishers, London, 1998.

[27] Hyeonjoong Cho, Binoy Ravindran, and E. Douglas Jensen. An optimal real-time scheduling algorithm for multiprocessors. In *Proceedings of the* 27^{th} *IEEE Real-Time Systems Symposium*, pages 101–110, December 2006.

[28] Jen-Yao Chung, Jane W.S. Liu, and Kwei-Jay Lin. Scheduling periodic jobs that allow imprecise results. *IEEE Transactions on Computers*, 39(9):1156–1174, September 1990.

[29] Michele Cirinei and Theodore P. Baker. Edzl scheduling analysis. In *Proceedings of the* 19^{th} *Euromicro Conference on Real-Time Systems*, pages 9–18, July 2007.

[30] UmaMaheswari C. Devi. An improved schedulability test for uniprocessor periodic task systems. In *Proceedings* 15^{th} *Euromicro Conference on Real-Time Systems*, pages 23–30, July 2003.

[31] UmaMaheswari C. Devi and James H. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. *Real-Time Systems*, 38:133–189, 2008.

[32] José Luis Díaz, Daniel F. García, Kanghee Kim, Chang-Gun Lee, Lucia Lo Bello, José María López, Sang Lyul Min, and Orazio Mirabella. Stochastic analysis of periodic real-time systems. In *Proceedings of the* 23^{rd} *IEEE Real-Time Systems Symposium*, pages 289–300, 2002.

[33] Jeremy P. Erickson and James H. Anderson. Fair lateness scheduling: Reducing maximum lateness in g-edf-like scheduling. In *Proceedings of the* 24^{th} *Euromicro Conference on Real-Time Systems*, pages 3–12, July 2012.

[34] Nathan Fisher and Sanjoy Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with bounded relative deadlines. *Journal of Embedded Computing*, 2(3,4):291–299, December 2006.

[35] Shelby Funk, Joël Goossens, and Sanjoy Baruah. On-line scheduling on uniform multiprocessors. In *Proceedings of the* 22^{nd} *IEEE Real-Time Systems Symposium*, pages 183–192, London, United Kingdom, December 2001.

[36] I-Hong Hou and P.R. Kumar. Scheduling periodic real-time tasks with heterogeneous reward requirements. In *Proceedings of the* 32^{nd} *IEEE Real-Time Systems Symposium*, pages 282–291, December 2011.

[37] Kevin Jeffay and Steve Goddard. A theory of rate-based execution. In *Proceedings of the* 20^{th} *IEEE Real-Time Systems Symposium*, pages 304–314, Phoenix, AZ, USA, December 1999.

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

[39] John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadline. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 201–209, Lake Buena Vista (FL), U.S.A., December 1990.

[40] John P. Lehoczky, Lui Sha, and Ye Ding. The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica (CA), U.S.A., December 1989.

[41] Joseph Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2(4):237–250, December 1982.

[42] Giuseppe Lipari and Enrico Bini. Resource partitioning among real-time applications. In *Proceedings of the* 15^{th} *Euromicro Conference on Real-Time Systems*, pages 151–158, Porto, Portugal, July 2003.

[43] Giuseppe Lipari and Enrico Bini. A methodology for designing hierarchical scheduling systems. *Journal Embedded Computing*, 1(2):257–269, 2005.

[44] Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[45] Aloysius K. Mok and Deji Chen. A multiframe model for real-time tasks. *IEEE Transactions on Software Engineering*, 23(10):635–645, October 1997.

[46] Aloysius K. Mok, Xiang Feng, and Deji Chen. Resource partition for real-time systems. In *Proceedings of the* 7^{th} *IEEE Real-Time Technology and Applications Symposium*, pages 75–84, Taipei, Taiwan, May 2001.

[47] Michael L. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2012.

[48] Gang Quan and Xiaobo Sharon Hu. Enhanced fixed-priority scheduling with (m, k)-firm guarantee. In *Proceedings of the* 21^{st} *IEEE Real-Time Systems Symposium*, pages 79–88, 2000.

[49] Parameswaran Ramanathan. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel Distributed Systems*, 10(6):549–559, 1999.

[50] Paul Regnier, George Lima, Ernesto Massa, Greg Levin, and Scott Brandt. Run: Optimal multiprocessor real-time scheduling via reduction to uniprocessor. In *Proceedings of the* 32^{nd} *IEEE Real-Time Systems Symposium*, pages 104–115, December 2011.

[51] Kai Richter and Rolf Ernst. Event model interfaces for heterogeneous system analysis. In *Design, Automation and Test in Europe (DATE)*, pages 506–513, Paris, France, March 2002.

[52] Ismael Ripoll, Alfons Crespo, and Aloysius K. Mok. Improvement in feasibility testing for real-time tasks. *Real-Time Systems*, 11(1):19–39, 1996.

[53] Danbing Seto, John P. Lehoczky, and Lui Sha. Task period selection and schedulability in real-time systems. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 188–198, Madrid, Spain, December 1998.

[54] Danbing Seto, John P. Lehoczky, Lui Sha, and Kang G. Shin. On task schedulability in real-time control systems. In *Proceedings of the* 17^{th} *IEEE Real-Time Systems Symposium*, pages 13–21, Washington, DC, USA, December 1996.

[55] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Proceedings of the* 24^{th} *Real-Time Systems Symposium*, pages 2–13, Cancun, Mexico, December 2003.

[56] Marco Spuri. Analysis of deadline scheduled real-time systems. Technical Report RR-2772, INRIA, France, January 1996.

[57] Anand Srinivasan and Sanjoy Baruah. Deadline-based scheduling of periodic task systems on multiprocessors. *Information Processing Letters*, 84(2):93–98, 2002.

[58] Martin Stigge, Pontus Ekberg, Nan Guan, and Wang Yi. The digraph real-time task model. In *Proceeding of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 71–80, Chicago, IL, USA, April 2011.

[59] Noel Tchidjo Moyo, Eric Nicollet, Frederic Lafaye, and Christophe Moy. On schedulability analysis of non-cyclic generalized multiframe tasks. In *Proceedings of the 22nd Euromicro Conference on Real-Time Systems*, pages 271–278, Bruxelles, Belgium, July 2010.

[60] Manel Velasco, Pau Martí, and Enrico Bini. Control-driven tasks: Modeling and analysis. In *Proceedings of the* 29^{th} *IEEE Real-Time Systems Symposium*, Barcelona, Spain, December 2008.

[61] Yifan Wu, Giorgio Buttazzo, Enrico Bini, and Anton Cervin. Parameter selection for real-time controllers in resource-constrained systems. *IEEE Transactions on Industrial Informatics*, 6(4):610–620, 2010.

[62] Fengxiang Zhang and Alan Burns. Schedulability analysis for real-time systems with edf scheduling. *IEEE Transactions on Computers*, 58(9):1250–1258, September 2009.