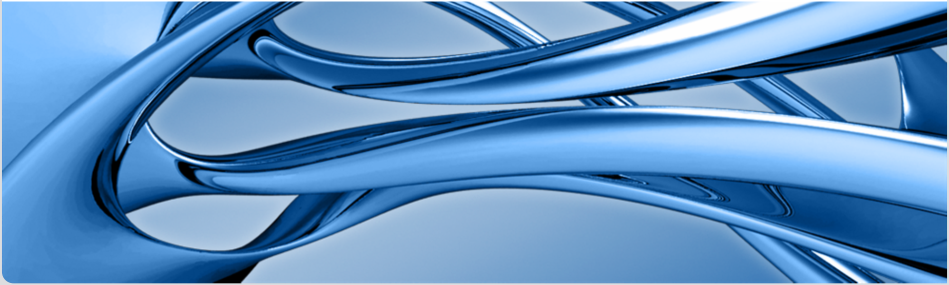
Karlsruhe Institute of Technology

Task Set Synthesis with Cost Minimization for Sporadic Real-Time Tasks

Juniorprofessor Dr. Jian-Jia Chen

Karlsruhe Institute of Technology (KIT), Germany



Outline

Introduction

Task Set Synthesis Problem

Proposed Combinatorial Algorithms

Conclusion

Decomposition of the Analysis

- Typical analysis and optimizations in real-time systems are decomposed into two phases
  - Phase 1: Worst-case execution time (WCET) of a stand-alone program
    - using WCET analyzers such as aiT or Chronous.
  - Phase 2: Worst-case response time of a periodic/sporadic task by considering the competition with the other tasks
    - analyzing the worst-case interference from the other tasks
    - many techniques such as utilization-based tests, response time analysis, busy-interval techniques, real-time calculus, max-plus algebra, etc.

Decomposition of the Analysis

- Typical analysis and optimizations in real-time systems are decomposed into two phases
  - Phase 1: Worst-case execution time (WCET) of a stand-alone program
    - using WCET analyzers such as aiT or Chronous.
  - Phase 2: Worst-case response time of a periodic/sporadic task by considering the competition with the other tasks
    - analyzing the worst-case interference from the other tasks
    - many techniques such as utilization-based tests, response time analysis, busy-interval techniques, real-time calculus, max-plus algebra, etc.

Decomposition of the Analysis

- Typical analysis and optimizations in real-time systems are decomposed into two phases
  - Phase 1: Worst-case execution time (WCET) of a stand-alone program
    - using WCET analyzers such as aiT or Chronous.
  - Phase 2: Worst-case response time of a periodic/sporadic task by considering the competition with the other tasks
    - analyzing the worst-case interference from the other tasks
    - many techniques such as utilization-based tests, response time analysis, busy-interval techniques, real-time calculus, max-plus algebra, etc.

Sporadic Task Models

Sporadic Task τ_i:

- T_i is the minimal time between any two consecutive job releases
- A relative deadline D_i for each job from task τ_i
- (C_i, T_i, D_i) is the specification of sporadic task τ_i, where C_i is the worst-case execution time.

- implicit deadline: $D_i = T_i$, for every task τ_i,
- constrained deadline: $D_i \leq T_i$, for every task τ_i
- arbitrary deadline: otherwise

Cost-Dependent WCET

- Deriving WCET is not a simple problem
- By spending more cost, the WCET may be reduced
  - Using more SRAM in the system or larger cache size
  - Using code redundancy or execution reordering to improve the reliability
- By reducing the quality of computation, the WCET may be reduced
  - Imprecise computation
  - Multiple versions of execution plans with different qualities

QRAM Model (Rajkumar et al. RTSS'97)

QRAM model: maximizing the system quality by choosing proper versions to meet the timing constraints of real-time tasks.

Cost-Dependent WCET

- Deriving WCET is not a simple problem
- By spending more cost, the WCET may be reduced
  - Using more SRAM in the system or larger cache size
  - Using code redundancy or execution reordering to improve the reliability
- By reducing the quality of computation, the WCET may be reduced
  - Imprecise computation
  - Multiple versions of execution plans with different qualities

QRAM Model (Rajkumar et al. RTSS'97)

QRAM model: maximizing the system quality by choosing proper versions to meet the timing constraints of real-time tasks.

Cost-Dependent WCET

- Deriving WCET is not a simple problem
- By spending more cost, the WCET may be reduced
  - Using more SRAM in the system or larger cache size
  - Using code redundancy or execution reordering to improve the reliability
- By reducing the quality of computation, the WCET may be reduced
  - Imprecise computation
  - Multiple versions of execution plans with different qualities

QRAM Model (Rajkumar et al. RTSS'97)

QRAM model: maximizing the system quality by choosing proper versions to meet the timing constraints of real-time tasks.

Minimum Cost Synthesis Problem

Input:

- A sporadic real-time task set $\mathcal{T}$
- Each task $\tau_i \in \mathcal{T}$ has
  - T_i: minimum inter-arrival time
  - D_i: relative deadline
- τ_i has $w_i \geq 1$ different versions with different costs
  - $\theta_i(k)$ is the cost for the k-th version of task τ_i
  - $C_i^{\theta_i(k)}$ is the corresponding WCET
  - $U_i^{\theta_i(k)} = \frac{C_i^{\theta_i(k)}}{T_i}$ as the utilization
- Without loss of generality, $\theta_i(1) < \theta_i(2) < \cdots < \theta_i(w_i)$

Output:

Select one version m_i for task τ_i such that $\mathcal{T}$ be feasibly scheduled and the system cost $\sum_{\tau_i \in \mathcal{T}} \theta_i(m_i)$ is minimized.

Minimum Cost Synthesis Problem

Input:

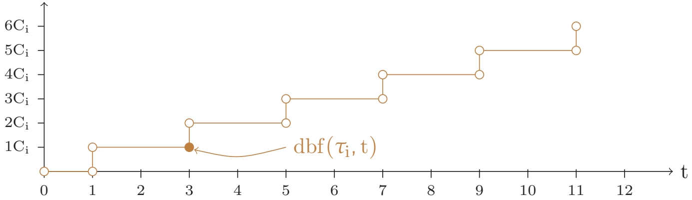
- A sporadic real-time task set $\mathcal{T}$
- Each task $\tau_i \in \mathcal{T}$ has
  - T_i: minimum inter-arrival time
  - D_i: relative deadline
- τ_i has $w_i \geq 1$ different versions with different costs
  - $\theta_i(k)$ is the cost for the k-th version of task τ_i
  - $C_i^{\theta_i(k)}$ is the corresponding WCET
  - $U_i^{\theta_i(k)} = \frac{C_i^{\theta_i(k)}}{T_i}$ as the utilization
- Without loss of generality, $\theta_i(1) < \theta_i(2) < \cdots < \theta_i(w_i)$

Output:

Select one version m_i for task τ_i such that $\mathcal{T}$ be feasibly scheduled and the system cost $\sum_{\tau_i \in \mathcal{T}} \theta_i(m_i)$ is minimized.

Schedulability of EDF

- Implicit deadlines: EDF is feasible if and only if the total utilization $U = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i}$ is at most 100%.
- Constrained/arbitrary deadlines: demand bound testing is required

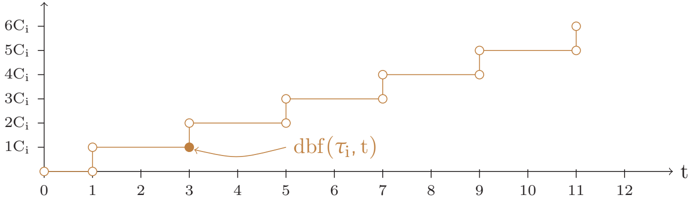


Baruah et al. [RTSS 1990]: A task set $\mathcal{T}$ can be feasibly scheduled (under EDF) on one processor if and only if

$$\forall t \geq 0, \sum_{\tau_i \in \mathcal{T}} \mathrm{dbf}(\tau_i, t) = \sum_{i=1}^{n} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor C_i \leq t.$$

Schedulability of EDF

- Implicit deadlines: EDF is feasible if and only if the total utilization $U = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i}$ is at most 100%.
- Constrained/arbitrary deadlines: demand bound testing is required



Baruah et al. [RTSS 1990]: A task set $\mathcal{T}$ can be feasibly scheduled (under EDF) on one processor if and only if

$$\forall t \geq 0, \sum_{\tau_i \in \mathcal{T}} dbf(\tau_i, t) = \sum_{i=1}^{n} \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor C_i \leq t.$$

Rate-Monotonic (RM) Scheduling

Priority Definition: A task with a smaller period has higher priority, in which ties are broken arbitrarily, i.e., $T_i \leq T_j$ if $i \leq j$.

- Least utilization upper bound for implicit deadlines: $U = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$ for n tasks
- If the following condition holds, the task set is schedulable under RM:

$$\forall \tau_i \in \mathcal{T} \; \exists t \text{ with } 0 < t \leq D_i \text{ and } W_i(t) \leq t,$$

where $W_i(t)$ of task τ_i is defined as follows:

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

Rate-Monotonic (RM) Scheduling

Priority Definition: A task with a smaller period has higher priority, in which ties are broken arbitrarily, i.e., $T_i \leq T_j$ if $i \leq j$.

- Least utilization upper bound for implicit deadlines:
  $U = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$ for n tasks
- If the following condition holds, the task set is schedulable under RM:

$$\forall \tau_i \in \mathcal{T} \; \exists t \text{ with } 0 < t \leq D_i \; \text{ and } W_i(t) \leq t,$$

where $W_i(t)$ of task τ_i is defined as follows:

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

Rate-Monotonic (RM) Scheduling

Priority Definition: A task with a smaller period has higher priority, in which ties are broken arbitrarily, i.e., $T_i \leq T_j$ if $i \leq j$.

- Least utilization upper bound for implicit deadlines: $U = \sum_{\tau_i \in \mathcal{T}} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$ for n tasks
- If the following condition holds, the task set is schedulable under RM:

$$\forall \tau_i \in \mathcal{T} \; \exists t \text{ with } 0 < t \leq D_i \;\text{ and } W_i(t) \leq t,$$

where $W_i(t)$ of task τ_i is defined as follows:

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

Optimality of RM and EDF

- For uniprocessor scheduling, if there exists a feasible schedule, scheduling jobs by using EDF is also feasible.
  - EDF scheduling algorithm is optimal
- If a set of n implicit-deadline tasks, can be feasibly scheduled on a processor with a fixed-priority assignment, assigning tasks by using rate monotonic scheduling also leads to a feasible schedule.
  - RM scheduling algorithm is optimal for fixed-priority scheduling
  - Deadline Monotonic (DM) scheduling algorithm is optimal for fixed-priority scheduling

Schedulability

- The issue for uniprocessor scheduling is on how to analyze the schedulability.
- Verifying the schedulability is $\mathcal{N}$P-hard or co$\mathcal{N}$P-hard
- Approximations are possible, but what do we approximate when only binary decisions (Yes or No) have to be made?
  - Answers like probabilistic guarantee are unlikely preferred, e.g., the task set is 99% schedulable.
  - Resource augmentation by : requires a faster platform
  - Resource augmentation by : requires a better platform

Schedulability

- The issue for uniprocessor scheduling is on how to analyze the schedulability.
- Verifying the schedulability is $\mathcal{N}$P-hard or co$\mathcal{N}$P-hard
- Approximations are possible, but what do we approximate when only binary decisions (Yes or No) have to be made?
  - Answers like probabilistic guarantee are unlikely preferred, e.g., the task set is 99% schedulable.
  - Resource augmentation by speeding up: requires a faster platform
  - Resource augmentation by allocating more processors: requires a better platform

Schedulability

- The issue for uniprocessor scheduling is on how to analyze the schedulability.
- Verifying the schedulability is $\mathcal{N}$P-hard or co$\mathcal{N}$P-hard
- Approximations are possible, but what do we approximate when only binary decisions (Yes or No) have to be made?
  - Answers like probabilistic guarantee are unlikely preferred, e.g., the task set is 99% schedulable.
  - Resource augmentation by speeding up: requires a faster platform
  - Resource augmentation by allocating more processors: requires a better platform

Resource Augmentation

For an algorithm A with a β resource augmentation factor, it guarantees that

$\Rightarrow$

if the task set (system) is schedulable (feasible), Algorithm A will also returns a schedulable (feasible) answer by speeding up the system by a factor β, or

$\Leftarrow$

if Algorithm A does not return a schedulable (feasible) answer, the system is also unschedulable (infeasible) by slowing down by a factor β.

Resource Augmentation

For an algorithm A with a β resource augmentation factor, it guarantees that

$\Rightarrow$

if the task set (system) is schedulable (feasible), Algorithm A will also returns a schedulable (feasible) answer by speeding up the system by a factor β, or

$\Leftarrow$

if Algorithm A does not return a schedulable (feasible) answer, the system is also unschedulable (infeasible) by slowing down by a factor β.

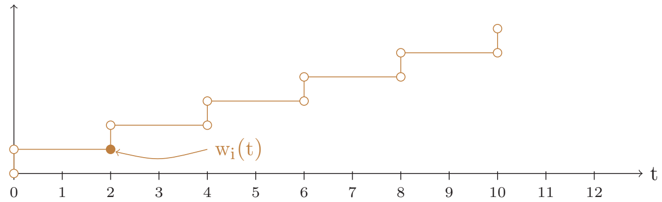
Time Demand Function Revisit for RM

Let $w_i(t)$ of the task τ_i be defined as follows

$$w_i(t) = \left\lceil \frac{t}{T_i} \right\rceil C_i.$$

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

Schedulable if for each τ_i $\exists t$ with $W_i(t) \leq t$.

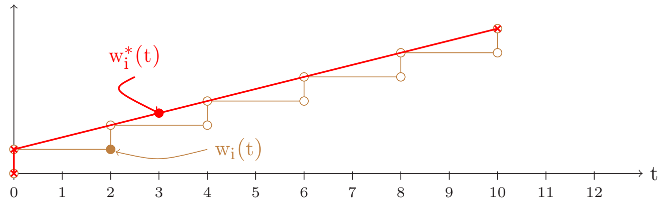


Time Demand Function Revisit for RM

Let $w_i(t)$ of the task τ_i be defined as follows

$$w_i(t) = \left\lceil \frac{t}{T_i} \right\rceil C_i.$$

$W_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j$
Schedulable if for each τ_i $\exists t$ with $W_i(t) \leq t$.



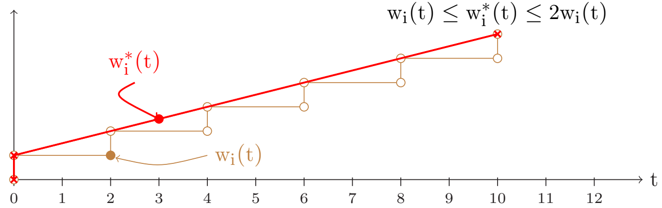
Time Demand Function Revisit for RM

Let $w_i(t)$ of the task τ_i be defined as follows

$$w_i(t) = \left\lceil \frac{t}{T_i} \right\rceil C_i.$$

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

Schedulable if for each τ_i $\exists t$ with $W_i(t) \leq t$.



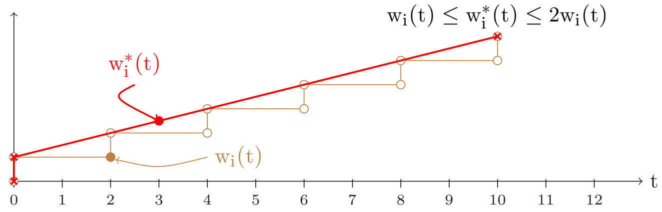
Time Demand Function Revisit for RM

Let $w_i(t)$ of the task τ_i be defined as follows

$$w_i(t) = \left\lceil \frac{t}{T_i} \right\rceil C_i.$$

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

Schedulable if for each τ_i $\exists t$ with $W_i(t) \leq t$.

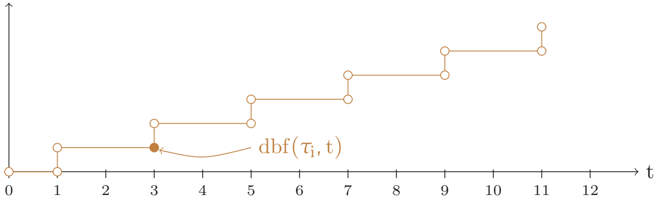


- The linear approximation makes the schedulability test easier
  - The test can be done in $O(n^2)$
  - The resource augmentation factor is 2. [Albers and Slomka ECRTS'04]

Demand Bound Function Revisit for EDF

Define demand bound function $\mathrm{dbf}(\tau_i, t)$ as

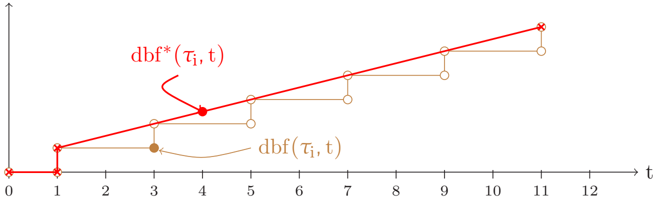
$$\mathrm{dbf}(\tau_i, t) = \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i = \max\left\{0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right\} C_i.$$



Demand Bound Function Revisit for EDF

Define demand bound function $\mathrm{dbf}(\tau_i, t)$ as

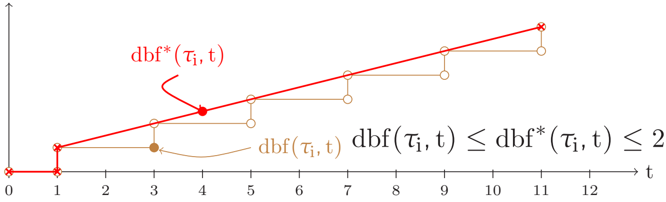
$$\mathrm{dbf}(\tau_i, t) = \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i = \max\left\{0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right\} C_i.$$



Demand Bound Function Revisit for EDF

Define demand bound function $\mathrm{dbf}(\tau_i, t)$ as

$$\mathrm{dbf}(\tau_i, t) = \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i = \max\left\{0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right\} C_i.$$

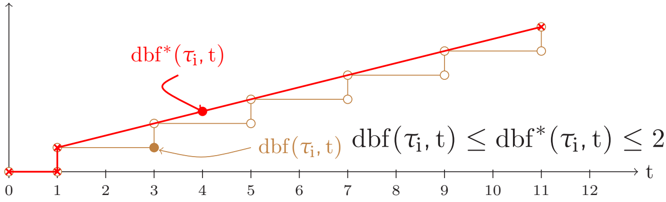


$$\mathrm{dbf}(\tau_i, t) \leq \mathrm{dbf}^*(\tau_i, t) \leq 2\mathrm{dbf}(\tau_i, t)$$

Demand Bound Function Revisit for EDF

Define demand bound function $\mathrm{dbf}(\tau_i, t)$ as

$$\mathrm{dbf}(\tau_i, t) = \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor \right\} C_i = \max\left\{0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right\} C_i.$$



$$\mathrm{dbf}(\tau_i, t) \leq \mathrm{dbf}^*(\tau_i, t) \leq 2\mathrm{dbf}(\tau_i, t)$$

- The linear approximation makes the schedulability test easier
  - The test can be done in $O(n^2)$
  - The resource augmentation factor is 1.6322. [Chen and Chakraborty, RTSS'11]

Outline

Minimum Cost Synthesis Problem

Input:

- A sporadic real-time task set $\mathcal{T}$
- Each task $\tau_i \in \mathcal{T}$ has
  - T_i: minimum inter-arrival time
  - D_i: relative deadline
- τ_i has $w_i \geq 1$ different versions with different costs
  - $\theta_i(k)$ is the cost for the k-th version of task τ_i
  - $C_i^{\theta_i(k)}$ is the corresponding WCET
  - $U_i^{\theta_i(k)} = \frac{C_i^{\theta_i(k)}}{T_i}$ as the utilization
- Without loss of generality, $\theta_i(1) < \theta_i(2) < \cdots < \theta_i(w_i)$

Output:

Select one version m_i for task τ_i such that $\mathcal{T}$ be feasibly scheduled and the system cost $\sum_{\tau_i \in \mathcal{T}} \theta_i(m_i)$ is minimized.

Special Case for Implicit Deadlines with EDF

- The utilization bound 100% is a necessary and sufficient test.
- The problem is equivalent to the minimum multiple-choice knapsack problem
  - Given a set of items, each with w_i versions and each version has a weight and a value, the objective is to choose one version in each item such that the total weight is no more than a given limit and the total value is as small as possible.
- Many results are already known.
  - O. H. Ibarra and C. E. Kim. "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems." In: J. ACM (1975), pp. 463-468.
  - E. L. Lawler. "Fast Approximation Algorithms for Knapsack Problems". In: Math. Oper. Res. 4.4 (1979), pp. 339-356.

(α, β)-Approximation

- Suppose the optimal system cost is $B^*(I)$ for an input instance I.
- An algorithm has an α-approximation if it guarantees to have at most $\alpha \cdot B^*(I)$ system cost for any input instance I

- An (α, β)-approximation guarantees to have at most $\alpha \cdot B^*(I)$ system cost by using β speed augmentation factor.
  - With respect to speeding-up: the derived solution is a feasible solution by speeding up the platform to β, and has an α-approximation in the system cost with respect to the original instance.
  - With respect to slowing-down: the derived solution is α-approximation with respect to a problem instance by slowing down the platform to $\frac{1}{\beta}$.

An optimal algorithm for the minimum multi-choice knapsack problem:

- $(1, 1)$ for EDF with implicit deadlines
- $(1, \frac{1}{\ln 2})$ for RM with implicit deadlines

(α, β)-Approximation

- Suppose the optimal system cost is $\mathrm{B}^*(\mathrm{I})$ for an input instance I.
- An algorithm has an α-approximation if it guarantees to have at most $\alpha \cdot \mathrm{B}^*(\mathrm{I})$ system cost for any input instance I

- An (α, β)-approximation guarantees to have at most $\alpha \cdot \mathrm{B}^*(\mathrm{I})$ system cost by using β speed augmentation factor.
  - With respect to speeding-up: the derived solution is a feasible solution by speeding up the platform to β, and has an α-approximation in the system cost with respect to the original instance.
  - With respect to slowing-down: the derived solution is α-approximation with respect to a problem instance by slowing down the platform to $\frac{1}{\beta}$.

An optimal algorithm for the minimum multi-choice knapsack problem:

- $(1, 1)$ for EDF with implicit deadlines
- $(1, \frac{1}{\ln 2})$ for RM with implicit deadlines

Hardness of Approximation

Theorem

Unless $\mathcal{P} = \mathcal{NP}$, there is no polynomial-time $(\alpha, 1)$-approximation algorithm for the minimum cost synthesis problem for any $\alpha \geq 1$ when considering EDF or FP scheduling.

Proof

It is based on an L-reduction from the uniprocessor schedulability problem for sporadic real-time tasks:

- Each task has two versions
- The one with cost equals to 1 has small execution time, and another one is with "very high" cost with 50% reduction of the execution time.

Outline

Introduction

Task Set Synthesis Problem

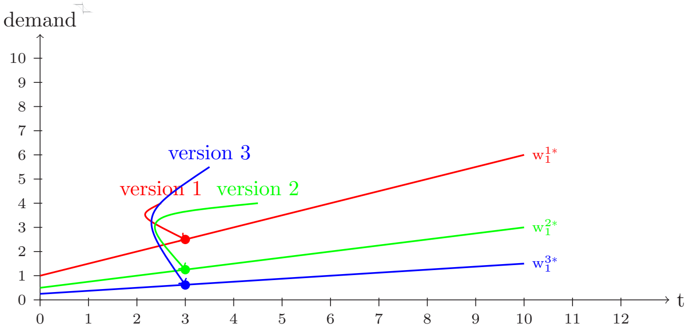
Proposed Combinatorial Algorithms

Conclusion

DM Scheduling for Constrained Deadlines

almost all the equations are different from the paper for simplicity

Deadline Monotonic (DM) is optimal when $D_i \leq T_i$.

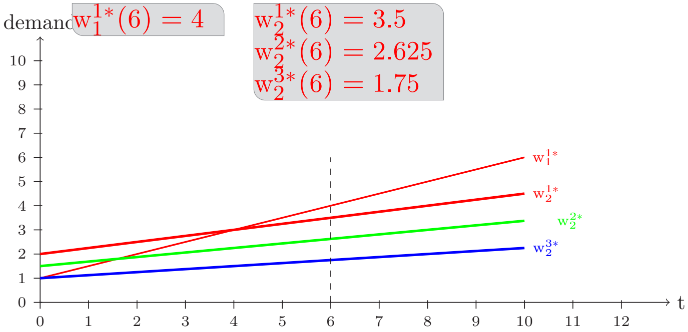


- $C_1^1 = 1$, $C_1^2 = 0.5$, $C_1^3 = 0.25$, $T_1 = 2$, $D_1 = 2$.
- $C_2^1 = 2$, $C_2^2 = 1.5$, $C_2^3 = 1$, $T_2 = 8$, $D_2 = 6$.

DM Scheduling for Constrained Deadlines

almost all the equations are different from the paper for simplicity

Deadline Monotonic (DM) is optimal when $D_i \leq T_i$.



- $C_1^1 = 1$, $C_1^2 = 0.5$, $C_1^3 = 0.25$, $T_1 = 2$, $D_1 = 2$.
- $C_2^1 = 2$, $C_2^2 = 1.5$, $C_2^3 = 1$, $T_2 = 8$, $D_2 = 6$.

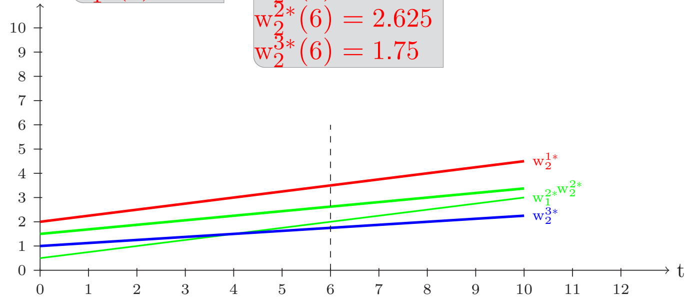
DM Scheduling for Constrained Deadlines

almost all the equations are different from the paper for simplicity

Deadline Monotonic (DM) is optimal when $D_i \leq T_i$.

$w_1^{2*}(6) = 2$

$w_2^{1*}(6) = 3.5$
$w_2^{2*}(6) = 2.625$
$w_2^{3*}(6) = 1.75$

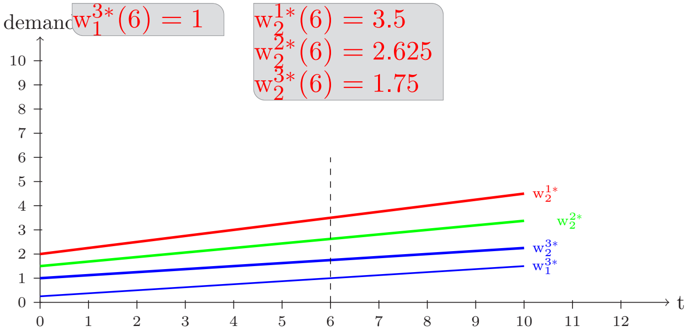


- $C_1^1 = 1$, $C_1^2 = 0.5$, $C_1^3 = 0.25$, $T_1 = 2$, $D_1 = 2$.
- $C_2^1 = 2$, $C_2^2 = 1.5$, $C_2^3 = 1$, $T_2 = 8$, $D_2 = 6$.

DM Scheduling for Constrained Deadlines

almost all the equations are different from the paper for simplicity

Deadline Monotonic (DM) is optimal when $D_i \leq T_i$.



- $C_1^1 = 1$, $C_1^2 = 0.5$, $C_1^3 = 0.25$, $T_1 = 2$, $D_1 = 2$.
- $C_2^1 = 2$, $C_2^2 = 1.5$, $C_2^3 = 1$, $T_2 = 8$, $D_2 = 6$.

Schedulability for DM

Deadline Monotonic (DM) is optimal when $D_i \leq T_i$.

For a given selection of versions $(m_1, m_2, \ldots, m_i)$, task τ_i can be feasibly scheduled by the DM scheduling if

$$\sum_{j=1}^{i} C_j^{\theta_j(m_j)} + D_i \cdot \sum_{j=1}^{i} U_j^{\theta_j(m_j)} \leq D_i$$

$$\Rightarrow \quad \left(\sum_{j=1}^{i-1} C_j^{\theta_j(m_j)} + D_{i-1} \cdot \sum_{j=1}^{i-1} U_j^{\theta_j(m_j)}\right) + \left(C_i^{\theta_i(m_i)} + D_i \cdot U_i^{\theta_j(m_j)} + (D_i - D_{i-1}) \sum_{j=1}^{i-1} U_j^{\theta_j(m_j)}\right) \leq D_i$$

Schedulability for DM

Deadline Monotonic (DM) is optimal when $D_i \leq T_i$.

For a given selection of versions $(m_1, m_2, \ldots, m_i)$, task τ_i can be feasibly scheduled by the DM scheduling if

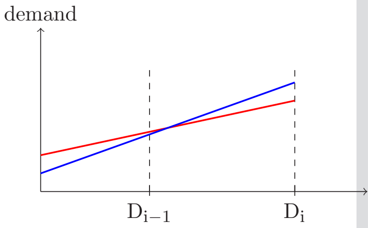
$$\sum_{j=1}^{i} C_j^{\theta_j(m_j)} + D_i \cdot \sum_{j=1}^{i} U_j^{\theta_j(m_j)} \leq D_i$$

$\Rightarrow$ demand for the first $i-1$ tasks at time D_{i-1}

$$+ \left(C_i^{\theta_i(m_i)} + D_i \cdot U_i^{\theta_j(m_j)} + (D_i - D_{i-1}) \sum_{j=1}^{i-1} U_j^{\theta_j(m_j)} \right) \leq D_i$$

Dynamic Programming

- What is the minimum cost to be feasible?
- What is the minimum cost to be feasible for the first i tasks under the approximation?
  - Two terms matter: the (sub-)demand $D_i \cdot \sum_{j=1}^{i} U_j$ and (sub-)demand $\sum_{j=1}^{i} C_j^{\theta_j(m_j)}$.

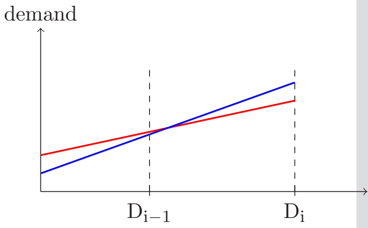


Suppose that $G(i, \delta, u)$ is the minimum system cost, represented by a version selection $m_1, m_2, \ldots, m_i$, for the first i tasks such that

- the total utilization for the first i tasks is equal to u,
- the total execution time for the first i tasks is equal to $\delta \cdot D_i$, and
- $\frac{\sum_{j=1}^{k} C_j^{\theta_j(m_j)}}{D_k} + \sum_{j=1}^{k} U_j^{\theta_j(m_j)} \leq 1$ for any $k = 1, 2, \ldots, i$.

Dynamic Programming

- What is the minimum cost to be feasible?
- What is the minimum cost to be feasible for the first i tasks under the approximation?
  - Two terms matter: the (sub-)demand $D_i \cdot \sum_{j=1}^{i} U_j$ and (sub-)demand $\sum_{j=1}^{i} C_j^{\theta_j(m_j)}$.



Suppose that $G(i, \delta, u)$ is the minimum system cost, represented by a version selection $m_1, m_2, \ldots, m_i$, for the first i tasks such that

- the total utilization for the first i tasks is equal to u,
- the total execution time for the first i tasks is equal to $\delta \cdot D_i$, and
- $\frac{\sum_{j=1}^{k} C_j^{\theta_j(m_j)}}{D_k} + \sum_{j=1}^{k} U_j^{\theta_j(m_j)} \leq 1$ for any $k = 1, 2, \ldots, i$.

Dynamic Programming (cont.)

- Constructing $G(i, \delta, u)$ can be done by using the standard dynamic programming approach.
  - Details [tighter definition and recursion] are in the paper
  - The minimum $G(N, \delta, u)$ for $0 \le \delta \le 1$ and $0 \le u \le 1$ has a $(1, 2)$-approximation factor for N tasks.
  - The solution is optimal on a slow-down platform with speed $\frac{1}{2}$.
- It takes pseudo-polynomial time/space for building the table

- Instead of building $G(i, \delta, u)$ for all possible values of δ and u
  - Approximate the values of interests
  - Build the table by a user-specified granularity σ
  - Lose some accuracy but earn the efficiency
  - $(1, \frac{2}{1-\eta})$-approximation with time complexity $O((\frac{N}{\eta})^2 \sum_{i=1}^{N} w_i)$ under the DM scheduling by setting σ to $\frac{1}{\lceil \frac{3N}{\eta} \rceil}$ for any given η with $0 < \eta < 1$

Dynamic Programming (cont.)

- Constructing $G(i, \delta, u)$ can be done by using the standard dynamic programming approach.
  - Details [tighter definition and recursion] are in the paper
  - The minimum $G(N, \delta, u)$ for $0 \leq \delta \leq 1$ and $0 \leq u \leq 1$ has a $(1, 2)$-approximation factor for N tasks.
  - The solution is optimal on a slow-down platform with speed $\frac{1}{2}$.
- It takes pseudo-polynomial time/space for building the table

- Instead of building $G(i, \delta, u)$ for all possible values of δ and u
  - Approximate the values of interests
  - Build the table by a user-specified granularity σ
  - Lose some accuracy but earn the efficiency
  - $(1, \frac{2}{1-\eta})$-approximation with time complexity $O((\frac{N}{\eta})^2 \sum_{i=1}^{N} w_i)$ under the DM scheduling by setting σ to $\frac{1}{\lceil \frac{3N}{\eta} \rceil}$ for any given η with $0 < \eta < 1$

Dynamic Programming (cont.)

- Constructing $G(i, \delta, u)$ can be done by using the standard dynamic programming approach.
  - Details [tighter definition and recursion] are in the paper
  - The minimum $G(N, \delta, u)$ for $0 \leq \delta \leq 1$ and $0 \leq u \leq 1$ has a $(1, 2)$-approximation factor for N tasks.
  - The solution is optimal on a slow-down platform with speed $\frac{1}{2}$.
- It takes pseudo-polynomial time/space for building the table

EDF

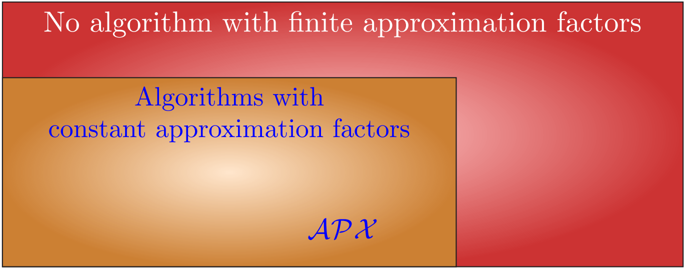
The some design philosophy also works for EDF scheduling (for arbitrary deadlines) with some minor changes.

  - Build the table by a user-specified granularity σ
  - Lose some accuracy but earn the efficiency
  - $(1, \frac{2}{1-\eta})$-approximation with time complexity $O((\frac{N}{\eta})^2 \sum_{i=1}^{N} w_i)$ under the DM scheduling by setting σ to $\frac{1}{\lceil \frac{3N}{\eta} \rceil}$ for any given η with $0 < \eta < 1$

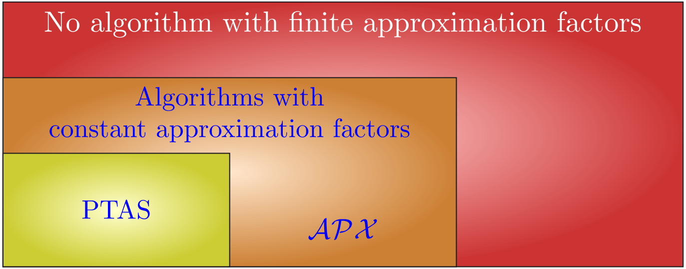
Classification of $\mathcal{NP}$-Hard Problems

No algorithm with finite approximation factors

Classification of $\mathcal{NP}$-Hard Problems



Classification of $\mathcal{NP}$-Hard Problems



PTAS (Polynomial-time Approximation Scheme): For each constant $\epsilon > 0$, a polynomial-time partitioning algorithm A_ϵ, with approximation factor $(1 + \epsilon)$.

- complexity depends on $\frac{1}{\epsilon}$, which is assumed to be a constant, e.g., $O(n^{\frac{2}{\epsilon}})$
- allows for a trade-off of run-time versus accuracy

d-Dimensional Representative Vector Set (Chen and Chakraborty, ECRTS'12)

Among $t \in (0, \infty]$, choose $t_1, \ldots, t_d$ for density values $\frac{dbf(\tau_i, t_j)}{t_j}$ for $j = 1, \ldots, d$.

- Representative — For the accuracy
- Constant dimensions — For complexity

A d-dimensional representative vector set $\mathcal{V}$ for the given task set $\mathcal{T}$ under a user-defined tolerable error $0 < \eta$:

- for any configuration $\mathcal{T}$ and the corresponding vector set $\mathcal{V}$

$$\gamma(\mathcal{T}) \geq \max_{j=1,2,\ldots,d} \sum_{v_i \in \mathcal{V}} q_{i,j} \geq \left(\frac{1}{1+\eta}\right)\gamma(\mathcal{T}),$$

Less sampling points

Bounded error

where $\gamma(\mathcal{T})$ is the maximum density of $\mathcal{T}$.

d-Dimensional Representative Vector Set (Chen and Chakraborty, ECRTS'12)

Among $t \in (0, \infty]$, choose $t_1, \ldots, t_d$ for density values $\frac{dbf(\tau_i, t_j)}{t_j}$ for $j = 1, \ldots, d$.

- Representative — For the accuracy
- Constant dimensions — For complexity

A d-dimensional representative vector set $\mathcal{V}$ for the given task set $\mathcal{T}$ under a user-defined tolerable error $0 < \eta$:

- for any configuration $\mathcal{T}$ and the corresponding vector set $\mathcal{V}$

$$\gamma(\mathcal{T}) \geq \max_{j=1,2,\ldots,d} \sum_{v_i \in \mathcal{V}} q_{i,j} \geq \left(\frac{1}{1+\eta}\right)\gamma(\mathcal{T}),$$

Less sampling points

Bounded error

where $\gamma(\mathcal{T})$ is the maximum density of $\mathcal{T}$.

$(1+\epsilon, 1+\eta)$-Approximation for EDF

- Chen and Chakraborty, ECRTS'12: when $\frac{D_{max}}{D_{min}}$ is a constant, the number of representative dimensions is a constant.
- How do we achieve $(1+\epsilon, 1+\eta)$-Approximation?
  - Build a d-dimensional representative vector set $\mathcal{V}$ for $1+\eta$ speed-up guarantee
  - The problem is reduced to a minimum-cost multiple choice multiple-dimension knapsack problem
    - Set $Z = \min\{N, \lceil \frac{d}{\epsilon} \rceil\}$, bounded by a constant
    - Enumerate the combinations to select the versions for Z large tasks
    - Select the versions of the other $N - Z$ light tasks by using linear programming
    - Round the fractional variables to yield a feasible solution
    - Return the best found feasible solution as the result

Conclusion

- (α, β)-approximation for combinatorial optimization problems in RTS
  - With respect to speeding-up: the platform is speeded up to β to ensure the feasibility and optimality
  - With respect to slowing-down: the derived solution is α-approximation with respect to a problem instance by slowing down the platform to $\frac{1}{\beta}$.

	EDF	DM ($\mathrm{D_i} \leq \mathrm{T_i}$)
pseudo-polynomial	(1,1.6322)	(1,2)
polynomial	$(1, \frac{1.6322}{1-\eta})$	$(1, \frac{2}{1-\eta})$
polynomial	$(\frac{1}{1-\epsilon}, \frac{1}{1-\eta})$ $\frac{\mathrm{D_{max}}}{\mathrm{D_{min}}}$ is a constant	???????