Feasibility in Real-Time Task Scheduling

CPSC 490 Mathematical Presentation

Jacob Earle

Yale University

April 1, 2022

Outline

Introduction

My Project

1. **Practical:** Implementing a basic real-time scheduling algorithm for an embedded operating system (`https://github.com/theseus-os/Theseus/pull/467`)
2. **Theoretical/Mathematical:** Research theoretical underpinnings of realtime scheduling theory; write a tool in R that performs feasibility tests on theoretical sets of tasks (`https://github.com/jacob-earle/RealtimeDeadlineAnalysis`)

Real-Time Systems in Theory

Real-Time Systems

Real-Time System: A computer system where tasks and jobs are associated with specific deadline constraints that must be met

1. **Hard:** All jobs must be completed before their deadlines
2. **Soft:** A certain percentage of jobs may finish late or be dropped

[4]

Real-Time Tasks: A Theoretical Definition

1. **Set of Tasks:** $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$
2. **Individual Task:** $\tau_i = (T_i, C_i, D_i)$
   1. T_i: Period of the task; each associated job released at fixed intervals of T_i
   2. C_i: Amount of compute time each associated job needs
   3. D_i: Deadline for associated job relative to its release time
3. **Utilization**
   1. **Individual Utilization:** $U_i = C_i / T_i$
   2. **System Utilization:** $U = \sum_{i=1}^{n} U_i$

[4]

Scheduling Algorithms

1. **Fixed Priority:** Each task assigned a static priority; jobs all scheduled with this priority
2. **Dynamic Priority:** Each job assigned a priority individually, may vary between other jobs from the same task

[4]

Feasibility of Task Sets: Hard Deadlines

Liu-Layland Axioms

1. All tasks are periodic
2. All tasks released at beginning of period; deadlines equal to periods
3. All tasks are independent
4. All tasks have fixed upper bound on computation times
5. No task may voluntarily suspend itself
6. All tasks are fully preemptible
7. All overheads are assumed to be 0
8. System has 1 processor

[3]

Optimum Scheduling Algorithms

- **Rate Monotonic Scheduling (RMS):** Optimum fixed-priority scheduling algorithm; tasks with the lowest periods are assigned the highest priorities
- **Earliest Deadline First (EDF):** Optimum dynamic priority scheduling algorithm; job that has the closest deadline is given the highest priority

[3]

RMS Feasibility Test (Liu-Layland 1973)

Set of tasks is feasible if

$$U \leq n(2^{1/n} - 1)$$

*** Sufficient but not necessary** [3]

Improved RMS Feasibility Test (Bini 2003)

Set of tasks is feasible if

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

*** Less pessimistic than Liu-Layland Test** [4]

EDF Feasibility Test

Set of tasks is feasible if and only if

$$U \leq 1$$

*** Sufficient AND necessary** [3]

Relaxing the Axioms (Explicit Deadlines) (1 / 3)

Allow tasks to have any deadline

$$C_i \leq D_i \leq T_i$$

1. **RMS no longer optimum!** Optimum fixed priority algorithm now *Deadline Monotonic Scheduling*
2. **EDF still optimum!**

[4]

Relaxing the Axioms (Explicit Deadlines) (2 / 3)

Hyperplanes δ-Exact Test

Set of tasks is feasible under a fixed priority algorithm if and only if:

$$C_i + W_{i-1}(D_i) \leq D_i, \quad \forall i \in \{1, ..., n\}$$

where

$$W_{i-1}(D_i) = \min_{t \in P_{i-1}(D_i)} \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil C_j + (D_i - t)$$

*** Necessary and sufficient!** [2]

Relaxing the Axioms (Explicit Deadlines) (3 / 3)

Improved EDF Test (Baruah et. al.) Let

$$L^* = \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}$$

Tasks are feasible under EDF if and only if $U < 1$ and for all L in the set of deadlines in the interval

$$0 \leq L \leq \max(D_1, ..., D_n, L^*),$$

$$\sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i \leq L$$

[4]

Relaxing the Axioms (Multiple Processors) (1 / 3)

- System has m processors
- **Rate Monotonic First Fit (RMFF):** Priorities assigned in same way as RMS
- Optimum scheduling of tasks is variation on "bin packing" (NP hard), so only sufficiency tests exist

[4]

Relaxing the Axioms (Multiple Processors) (2 / 3)

Two tests for RMFF feasibility:

1. **Oh and Baker (1998):**

$$U \leq m(2^{1/2} - 1)$$

2. **Baker (2003):** Let λ be the minimum individual task utilization. Then tasks are feasible if

$$U \leq \frac{m(1-\lambda)}{2} + \lambda$$

[4]

Relaxing the Axioms (Multiple Processors) (3 / 3)

Test for general multiprocessor algorithms (Andersson 2001):

$$U \leq \frac{m+1}{2}$$

[4]

Feasibility of Task Sets: Soft Deadlines

Varying Definitions

- Permits various alternative definitions of tasks and schedulability
- **Firm Deadlines:** Tasks that miss their deadline have failed; concerned with ensuring a certain percentage do not fail
- Alternatively, seek to minimize total lateness across all tasks

$$\alpha(x) < \beta$$

[4]

SRMS Model (Atlas/Bestavros 1998)

1. **Set of Tasks:** $\Gamma = \{\tau_1, ..., \tau_n\}$
2. **Individual Task:** $\tau_i = (P_i, f_i(x), Q_i)$
   1. P_i: Period of task
   2. $f_i(x)$: PDF of compute times of jobs of τ_i; $\mathrm{P}(x > P_i) = 0$
   3. Q_i: *Quality of Service*; long-run fraction of jobs from τ_i that do not miss deadlines
3. **Superperiod:** P_{i+1} is the period of the next shortest period task
4. **Phases:** Task τ_i will release $\frac{P_{i+1}}{P_i}$ jobs with each period
5. $e_{i,j}$: Compute time of j-th job in superperiod of τ_i
6. Task set *schedulable* if, $\forall 1 \leq i \leq n$,

$$\text{Long-run fraction of finished jobs of } \tau_i \geq Q_i$$

[1]

SRMS Algorithm

1. Tasks assigned priorities in increasing order of period
2. Assign each τ_i a utilization budget a_i, for the superperiod
3. For each task accepted, reduce a_i by $e_{i,j}$
4. If $e_{i,j}$ less than the remaining budget, reject the task to save resources
5. At the end of the superperiod P_{i+1}, reset budget to a_i

[1]

a_i Determines QoS

We can use a_i to calculate long-run fraction of accepted jobs:

$$\text{QoS}(\tau_i) = \frac{P_i}{P_{i+1}} \times \sum_{j=1}^{\frac{P_{i+1}}{P_i}} \text{P}(e_{i,j} \text{ meets its deadline})$$

[1]

Feasibility of SRMS

Assume periods are harmonic (periods are divisible). Task set is schedulable if and only if, $\forall 1 \leq i \leq n$

1. a_i is selected such that
$$\text{QoS}(\tau_i) \geq Q_i$$
2. $$\sum_{i=1}^{n} \frac{a_i}{P_{i+1}} \leq 1$$

[1]

Conclusion

Mathematical Takeaways

1. Implications of sufficient vs. necessary and sufficient tests
2. Vastly unexplored area of study
3. Discovery through relaxing axioms

Bibliography


Alia Atlas and Azer Bestavros.
Statistical rate monotonic scheduling.
1998.


Enrico Bini and Giorgio C. Buttazzo.
Schedulability analysis of periodic fixed priority systems.
2004.


C. L. Liu and James W. Layland.
Scheduling algorithms for multiprogramming in a hard-real-time environment.
1973.


Lui Sha, Tarek Abdelzaher, Karl-Erik Arzen, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius Mok.
Real time scheduling theory: A historical perspective.
Real-Time Systems, (28):101–155, 2004.