References

[1] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966.

Greedy Scheduling with Deadlines and Priorities

Project Team

Jintian Wang (z5536837)
Dennis Shu (z5522609)

Mentor: Ayda Valinezhad Orang

School of Computer Science and Engineering
University of New South Wales
Sydney, Australia

Table of Contents

1 Introduction

Scheduling tasks on parallel machines is a fundamental problem in computer science with applications in cloud computing, manufacturing, and network management. We study fair scheduling algorithms that must balance efficiency (minimizing total completion time) with limited waiting time guarantees for different priority classes.

2 Project Proposal

2.1 Problem Statement

We consider the problem of scheduling tasks on parallel machines where tasks have different priority levels requirements.

Definition 1 (Problem Instance)**.** An instance consists of:

- A set of n indivisible tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$
- A set of m identical parallel machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$
- Each task T_i has:
  - Processing time $p_i > 0$
  - Priority class $c \in \{h, l\}$ where h is high priority and l is low priority
- Machine properties:
  - A machine can only process 1 task at a time, but can process arbitrarily many tasks sequentially
  - Each task is assigned to exactly one machine (indivisible - no task splitting)
  - Once the machine begins to process a task T_i, it works without interruption until completing the task.
- Deadline constraints
  - All tasks T_i have deadlines $D_h > 0$ or $D_l > 0$
  - $D_h < D_l$, where high priority tasks have earlier deadlines

Definition 2 (Completion Time)**.** For a task T_i with arrival time a_i and start time $s_i \geq a_i$, the completion time is:

$$C_i = s_i + p_i$$

Definition 3 (Feasible Schedule)**.** A schedule is *feasible* if

- For every high-priority task T_i:$C_i \leq D_h$
- For every low-priority task $T_j : C_j \leq D_l$

Definition 4 (Makespan)**.** The *makespan* of a schedule is the time when all tasks have finished, which is equivalent to the the completion time of the last task:

$$C_{\max} = \max_{i \in [n]} C_i$$

Objective: Design an algorithm to find a feasible schedule that minimizes the makespan $C_{\max}$.

2.2 Survey

2.2.1 Known Results

Graham has established a theoretical framework for multiprocessor scheduling problems. He studied scheduling n tasks on m identical machines where tasks have processing times and precedence constraints, and the goal is to minimise makespan. His results provide baseline for our problem, which is that any scheduling algorithm achieves a makespan $\omega \leq (2 - \frac{1}{m}) \times \omega_0$, where ω_0 is the optimal makespan and m is the number of identical machines [1].
Therefore, we expect our algorithm should achieve better than $2 \times \omega_0$ makespan.

2.2.2 Open Questions

2.2.3 Special Cases

Special cases where you aim to achieve or have thought about.

2.2.4 Preliminary Direction

2.3 Research Plan

A set of goals you aim to achieve by Week 7 (i.e. progression check) and a set of goals you aim to achieve by Week 10. You should aim to keep your research plan realistic to the confines of the project timeline.

References

[1] R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966.