Greedy Scheduling with Dynamic Priority Elevations and Deadlines

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

We study parallel machine scheduling with priority classes and propose Dynamic Priority Elevation (DPE), a novel threshold-based mechanism that prevents low-priority task starvation while maintaining high-priority deadline guarantees. Unlike traditional static priority scheduling, DPE dynamically promotes tasks based on deadline pressure, addressing a key limitation in parallel machine environments where priority inversion can indefinitely delay lower-priority work.

2 Project Proposal

2.1 Survey

2.1.1 Motivations

Our interest in this scheduling problem stems from its direct applications in manufacturing and task allocation systems, where production lines must balance urgent orders with regular manufacturing schedules. We were particularly inspired by the Week 2 workshop problems on greedy scheduling algorithms, which introduced us to the theoretical framework of optimal ordering strategies. These exercises demonstrated how elegant greedy approaches could solve complex scheduling scenarios, motivating us to explore whether similar techniques could handle the additional complexity of priority classes and parallel machines.

2.1.2 Real-world Applications

The scheduling algorithm can be applied to cloud computing by using two deadlines: high-priority tasks (premium customers), and low-priority tasks (standard customers), ensuring both groups meet their deadlines [6]. This algorithm can also be applied to emergency department patient management with two priority classes: high-priority patients (urgent cases) and low-priority patients (non-urgent cases). This approach addresses emergency department overcrowding while ensuring timely medical services for critical patients and efficient management of routine cases[1].

2.1.3 Known Results

- Graham's study shows 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. Therefore, we expect our algorithm should achieve better than the optimal $2 \times \omega_0$ makespan[2].
- Liu and Layland's work proved that Earliest Deadline First (EDF) scheduling achieves optimal processor utilization on single-processor systems by dynamically assigning priorities based on task deadlines[5].

- Lee and Pinedo's work develops a multi-phase approach of using different greedy strategies followed by local search optimization, which directly parallels our proposed methodology for handling complex parallel machine scheduling with priority constraints[4].

2.1.4 Special Cases

- **Case 1: Equal processing times** ($p_i = p$ for all tasks i)
  - Reduces to bin packing with priority deadline constraints
  - Enables polynomial-time optimal solutions for small instances
  - Simplifies analysis of DPE threshold effects
- **Case 2: Single machines** ($m = 1$)
  - Serves as the most fundamental and achievable case for this problem
- **Case 3: Single priority class**
  - Equivalent to classical multiprocessor scheduling without priorities
  - Validates our base algorithms against well-known benchmarks
  - DPE becomes inactive (no priority elevation needed)

2.1.5 Preliminary Direction

We plan to develop a multi-phase greedy scheduling framework that systematically explores different priority-based strategies. Our approach will generate and evaluate multiple greedy solutions using various rules (such as **shortest processing time first(STF)**, **earliest deadline first(EDF)**, and load balancing strategies) applied within each priority class[3].

2.1.6 Our Contribution

Our main innovation is the Dynamic Priority Elevation (DPE) mechanism that dynamically adjusts task priorities during execution based on deadline pressure, unlike existing scheduling algorithms that use static priority assignments throughout. While traditional approaches like EDF and fixed-priority scheduling can lead to low-priority task starvation in parallel machine environments, our threshold-based elevation system prevents indefinite postponement while maintaining high-priority task guarantees.

2.2 Research Plan

2.2.1 Aims

Primary Focus: Implement established algorithms first, and then add DPE innovation as the experimental twist.

2.2.2 Plan

1. Foundation (Week 3 - 4)
    - Finalize the problem definitions to make the problem statement concise and accurate
    - Create comprehensive test suite with 20+ examples covering: single machine, multiple machine, all high-priority, all low-priority edge cases, and general cases (from simple to complex)
    - Research and design specifications for 3 baseline algorithms: SPT (Shortest Processing Time), EDF (Earliest Deadline First), and Priority-First scheduling
2. Algorithm Design I (Week 5 - 6)
    - **Implement 3 established greedy algorithms** for single machine case:
        - SPT (Shortest Processing Time First)
        - EDF (Earliest Deadline First)
        - Static Priority-First scheduling
    - Develop infeasibility detection method for all algorithms
    - Test and validate all baseline algorithms on example suite
    - **Week 6 Goal:** Working baseline implementations with performance benchmarks
3. Proof and Analysis (Week 7)
    - Prove correctness for the best-performing baseline algorithm
    - Analyze time complexity for implemented algorithms
    - Establish baseline performance metrics (makespan, deadline satisfaction, fairness)
4. Algorithm Design II (Week 8 - 9)
    - **Implement DPE (Dynamic Priority Elevation) as innovative twist** on best baseline algorithm
    - Experimental comparison: DPE-enhanced vs. all 3 baseline approaches
    - Measure and analyze when/why DPE outperforms or underperforms baselines
    - Document scenarios where innovation provides measurable improvement
5. Final documentation (Week 10) By week 10, we will have the technical report and poster presenting our comparative study of baseline algorithms vs. DPE innovation. The deliverables will include experimental results, performance analysis, and lessons learned about the effectiveness of our innovative twist.

3 Problem Statement

We consider the problem of scheduling tasks with different priority classes and deadline constraints on identical parallel machines to minimize total completion time while ensuring all deadline requirements are satisfied.

3.1 Definitions

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:

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

3.2 DPE Definition

We introduce **Dynamic Priority Elevation (DPE)** for parallel machine scheduling:

A low-priority task T_i gets temporarily elevated to high-priority when its deadline pressure exceeds threshold α:

$$\text{deadline\_pressure}(T_i) = \frac{\text{current\_time} - \text{arrival\_time}(T_i)}{D_i - \text{arrival\_time}(T_i)} > \alpha$$

where $\alpha = 0.7$ (task has consumed 70% of its deadline window).

This means: elevate low-priority tasks when they've used up 70% of their allowed time but haven't started processing yet.

3.3 Objective

Design an polynomial-time algorithm to find a feasible schedule that minimizes the makespan $C_{\max}$ while preventing starvation of low-priority tasks..

References

[1] Thiago Alves de Queiroz, Manuel Iori, Arthur Kramer, and Yong-Hong Kuo. Dynamic scheduling of patients in emergency departments. *European Journal of Operational Research*, 310(1):100–116, 2023.

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

[3] Cengiz Kahraman, Orhan Engin, İhsan Kaya, and R. Elif Öztürk. Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach. *Applied Soft Computing*, 10(4):1293–1300, 2010. Optimisation Methods Applications in Decision-Making Processes.

[4] Young Hoon Lee and Michael Pinedo. Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464–474, 1997.

[5] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, January 1973.

[6] Longxin Zhang, Liqian Zhou, and Ahmad Salah. Efficient scientific workflow scheduling for deadline-constrained parallel tasks in cloud computing environments. *Information Sciences*, 531:31–46, 2020.