LAUPIES Raphaël

Non-Preemptive Job Scheduling - Final Assignment

Aéro 4 - Class of 2026

Mr Jasdeep Singh
April 2025

Contents

1 Introduction

This project addresses a non-preemptive scheduling problem at job-level granularity. The goal is to optimize the execution order of tasks to minimize the total waiting time across all jobs while respecting all deadlines, unless otherwise stated.

2 Problem Definition

Given a periodic task set defined by computation times and periods, the objectives are:

- Guarantee that no job misses its deadline (except when allowed for τ_5).
- Minimize the total waiting time across all jobs.
- Maximize processor idle time by scheduling efficiently.

3 Task Set and Hyperperiod

The task set used is as follows:

Task	Computation Time (C)	Period (T)
τ_1	2	10
τ_2	3	10
τ_3	2	20
τ_4	2	20
τ_5	2	40
τ_6	2	40
τ_7	3	80

The hyperperiod is the least common multiple (LCM) of all periods, which results in:

$$\text{Hyperperiod} = 80$$

4 Scheduling Methodology

The implemented scheduling strategy is exhaustive and global:

- At each scheduling decision point, all ready jobs are considered.
- All permutations of the ready jobs are simulated.
- The schedule minimizing the total waiting time is selected, provided all deadlines are met.
- If no job is ready, the processor idles until the next arrival.

5 Schedulability Analysis

Each job's response time is analyzed to ensure it does not exceed its deadline. The schedulability was verified for all jobs under strict scheduling (without allowing τ_5 to miss).

6 Relaxed Scheduling (Allowing τ_5 to Miss Deadline)

In a second simulation, τ_5 was allowed to miss its deadline. This enables more flexible scheduling and a potentially lower overall waiting time, at the cost of a single task's timeliness.

Discussion on τ_5 Relaxed Deadline

In the relaxed scheduling simulation, τ_5 was allowed to miss its deadline without triggering a scheduling exception. However, it is important to note that τ_5's arrival time and original deadline remained unchanged. The scheduling algorithm still attempted to minimize the total waiting time and respected all other deadlines.

As a result, unless there was a strong scheduling conflict, τ_5 typically did not miss its deadline, because the permutation-based optimizer naturally continued to prioritize feasible execution sequences. Thus, the resulting Gantt charts for strict and relaxed scheduling appear nearly identical.

Allowing τ_5 to miss its deadline does not artificially extend its deadline or relax its timing constraints numerically (e.g., doubling the period or deadline). It simply means that missing the deadline would no longer cause the simulation to fail. In future implementations, if the goal were to explicitly favor other tasks at the cost of τ_5, a priority adjustment mechanism should be introduced during scheduling decisions.

7 Computational Complexity

The complexity of this scheduling method is factorial:

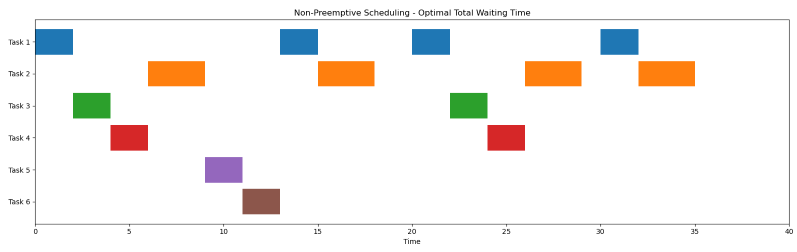
$$O(n!)$$

where n is the number of ready jobs at a given scheduling point. This factorial complexity results from the need to evaluate all possible orderings (permutations) at each decision point.

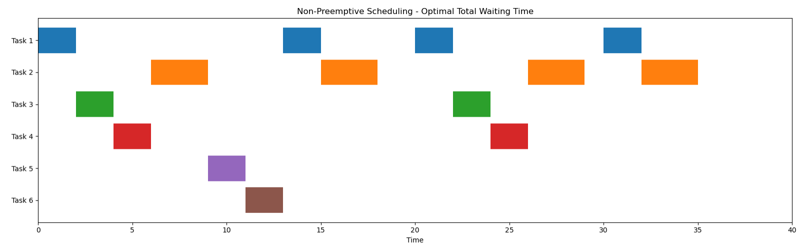
8 Results

Gantt charts were generated for both cases:

- **Strict scheduling** (no missed deadlines): `figures/gantt_strict.png`



- **Relaxed scheduling** (allowing τ_5 to miss): `figures/gantt_task5_late.png`



An extract of the job log is provided below (example for strict scheduling):

- Task 1 Job 1: Arrival = 0, Start = 0, End = 2, Deadline = 10
- Task 2 Job 1: Arrival = 0, Start = 2, End = 5, Deadline = 10
- ...

Limitations due to Computational Complexity

Initially, an attempt was made to include all seven tasks in the exhaustive permutation-based scheduling algorithm. However, the computational complexity of the method is factorial with respect to the number of ready jobs, i.e., $O(n!)$. When considering seven tasks, the number of permutations became prohibitively large, leading to extremely long computation times that made the approach impractical.

As a result, the decision was made to limit the task set to the first six tasks (τ_1 to τ_6) for both Gantt chart simulations (strict scheduling and relaxed scheduling allowing τ_5 to miss its deadline). This compromise allowed the scheduling to be completed in a reasonable amount of time while preserving the overall methodology and objectives of the project.

9 Conclusion

The exhaustive permutation-based scheduling strategy effectively minimizes total waiting time while ensuring no missed deadlines (under strict scheduling). Allowing τ_5 to miss a deadline improves flexibility but must be carefully justified in safety-critical systems.

10 Repository Link

The project repository containing code, instructions, and output results is available at:
`https://github.com/raphlp/Schedule-search`