Job Shop Scheduling with the SPT Rule

Algorithms and Complexity Project

konrantos

1 Problem Description

In the classic Job Shop Scheduling Problem (JSSP), we are given a set of jobs, each consisting of a sequence of operations that must be processed in a specific order on different machines. Each operation must be processed on a specific machine for a defined amount of time. The goal is to schedule all operations to minimize the total makespan, i.e., the time at which all jobs are completed.

2 Analysis and Solution

This project implements a heuristic solution using the Shortest Processing Time (SPT) rule. Each job is assigned a priority based on the sum of its operation times. Jobs with lower total processing time are scheduled earlier.

We assume non-preemptive execution and that each machine can handle only one job at a time. Machines are scheduled greedily according to the SPT job order and machine availability.

3 Python Implementation

The implementation reads job and machine data from standard dataset files (e.g., ‘la01.txt‘, ‘mt06.txt‘), which follow a specific format. For each job, both the operation durations and the machine sequence are provided.

The following Python snippet demonstrates the structure of the SPT makespan calculation:

```
def calculate_makespan_spt(num_jobs, num_machines, job_times,
    job_sequences):
    machine_schedule = [0] * num_machines
    job_completion = [0] * num_jobs
    sorted_jobs = sorted(range(num_jobs), key=lambda x: sum(job_times[x
    ]))
    for job in sorted_jobs:
        for step in range(num_machines):
            machine = job_sequences[job][step] - 1
            time = job_times[job][step]
            machine_schedule[machine] = max(machine_schedule[machine],
    job_completion[job]) + time
            job_completion[job] = machine_schedule[machine]
    return max(machine_schedule)
```

Listing 1: SPT-based makespan calculation

4 Results and Observations

Experiments were conducted using standard OR-Library datasets (‘la01‘ to ‘la05‘, ‘mt06‘, ‘mt10‘, ‘mt20‘). Below is a table showing the calculated makespan using the SPT rule versus the known optimal makespan (where available):

Instance	SPT Makespan	Optimal Makespan	Difference
la01	688	666	+22
la02	737	655	+82
la03	1110	597	+513
la04	938	590	+348
la05	1211	593	+618
mt06	1578	55	+1523
mt10	1616	930	+686
mt20	1852	1165	+687

Table 1: Comparison of SPT vs. Optimal Makespan

We observe that the SPT heuristic works better for smaller instances (e.g., ‘la01‘) and becomes increasingly suboptimal for larger or more complex instances (e.g., ‘mt10‘, ‘mt20‘).

5 Conclusions

The SPT rule offers a simple and fast scheduling strategy for job shop problems. However, its performance heavily depends on the structure of the instance. While suitable for small datasets or as a baseline, it often fails to approach the optimal solution for larger or more irregular job-machine configurations.

Further optimization methods, such as tabu search, genetic algorithms, or simulated annealing, could significantly improve the makespan.