CSCE220301- Analysis and Design of Algorithm Lab

Fall'24

Final Lab project

Topic: Scheduling Factory Interdependent Production Tasks: multiple tasks depend on each other and need scheduling to minimize downtime.

Maha Shakshuki - 900225906
Dr. May Shalaby

Content :

- **Topic :**

Scheduling Factory Interdependent Production Tasks: multiple tasks depend on each other and need scheduling to minimize downtime.

- **The significance of the problem :**

This problem is widespread in the work section, especially in manufacturing, where multiple tasks depend on one another to function. Delays in on-task tasks can cause bottlenecks in the pipeline (the production status), which can lead to wasted resources and financial losses.

- **Examples :**

1- Electronics Manufacturing: The factory produces CPUs and circuit boards goes through steps in order to ship the board and announce the CPUs.

- If delays occur, testing will be postponed and will affect the company financially and reputationally (as Apple in each September announces a new version for the iPhone, and Mac)

2- Software Pipelines: It has to go through different stages: compilation, testing, debugging, integrating, and finally running the code. If one stage is late, this one will be late, too.

- **Complexity :**

1- Optimization: the schedule must minimize total downtime and at the same time, maximize machine utilization.
2- Constraints: task duration and machine availability.
3- Task dependencies: For some tasks, in order to start it has to have predecessor tasks are completed.

- **Why it's worth solving :**

Efficient task scheduling will minimize downtime and boost productivity and resource utilization. Developing optimal schedules is important to reduce operational costs.

- **Key components of the problem :**

1—List of tasks; each task has a specific duration. List of prerequisite tasks. Each task is defined with a different operation number or ID.
2-Machines; specific number of machines available for each task
3- Dependencies: relationship with different variables. (for example in order to access Task 2: it has prerequisites Task1, etc)

- **Inputs** :

1- Array of Task operation numbers
2- Duration for each task
3- Prerequisites for each task
4-Number of machines available for each task

- **Output :**

1- Start and end time for each task.
2- Machine availability for each task

- **Constrain :**

1- Minimize the overall time for completion of tasks
2- Tasks have to fulfill prerequisite requirements (a task completed before the new task)
3- Each machine handles only one task at a specified time

- **Example For the problem :**

(To build on Electronics Manufacturing)
Problem statement :
Scheduling tasks across CPUs in a manufacturing simulation for circuit boards
The process :
Task (1) : Etch the board (Duration: 2 hours)
Task(2) : Solder components (Duration: 6 hours) - Prerequisite : Task 1
Task (3) : Test the board (Duration : 3 hours) - Prerequisite : Task 2
Task (4) : Package the board (Duration 2 hours) - Prerequisite : Task 3

- 2 machines available for executing tasks

- **Alghortim approach :**

1- Greedy :

Code : Attached to file

Rationale :
The idea behind greedy in this problem is to assign tasks to the machine that becomes available earliest. This minimizes time for machines and ensures the satisfaction of prerequisites.

Trade-offs:
Greedy approach can result in an imbalance in the workload because it prioritizes minimizing the time. For example, in some cases; some tasks can be completed by one machine and the other by another different machine which causes an imbalance in the workload because the algorithm picks the machine that becomes available first in this way we can't guarantee a balanced workload.

Efficiency :

- **Time Complexity**: O(N), where N is the number of tasks. Each task is processed once
- **Space Complexity**: O(N), to store task completion statuses and prerequisites.
-

Accuracy:
It schedules tasks while respecting prerequisites but doesn’t ensure balanced workload distribution.

Suitability:
ideal for scenarios where minimizing time is important.

2- Backtracking :

Code : Attached to file

Rationale:
BackTracking will explore possible solutions and will test different assignments of tasks to the machine and adjust the start time based on the availability of each machine. For example, if we started with Machine 1 and we were able to generate all tasks within our constraints; task dependencies, and machine availability, the answer would be for Machine 1 because we made sure that no solution would be missed, else it would be for Machine 2 and all tasks will be generated by one of them after checking the constraints.

Trade-offs:
It guarantees to explore all solutions and ensure that the constraints are respected, however, the time complexity will be exponential (depending on the n tasks) as well as

when the N.O. of tasks increases the algorithm might be inefficient because it will take a longer time to explore all the options.

Efficiency :

- **Time complexity:** o(n!) , where n! is the number of n tasks.
- **Space complexity:** o(n), when n depends only on the data structure(vectors in this case)

Accuracy:
BackTracking approach ensures that our constraints are met (dependencies and machine constraints). It also generates a valid answer. However, it might not be the most optimal solution (in case we have big tasks), there are better algorithms for the optimal solutions (DP and greedy)

Suitability:
For a small number of tasks, backtracking will be suitable for these kinds of problems because it will guarantee the optimal solution.

3- DP:

Code : Attached to file

Rationale: DP can solve this problem because it calculates optimal solutions for overlapping subproblems (smaller problems: tasks on machines) as it will ensure dependencies are respected and avoid redundant computations. DP will optimize time and space and will handle the constraints in an efficient way.

Trade-offs: will guarantee the optimal solution: exploring all possible options.
Also, the space complexity might increase because it depends on the DP table size

Efficiency :

- **Time complexity:** O(n*m) , where n is the number of tasks and m is the number of machines involved to handle the tasks.
- **Space complexity:** O(n*m), where n is the number of tasks and m is the number of machines involved to handle the tasks.

Accuracy:
It will produce high accuracy because all the constraints are handled in the main code to ensure every task meets its prerequisite. Also, it is assigned to machines based on minimizing the time required to implement all the tasks.

Suitability:
DP approach is suitable for medium problems in order to ensure optimal time and space complexity as well as provides an accurate time because it compares to the minimum in each time.

- **Comparison between algorithms :**

Category	**Dynamic programming**	**Backtracking**	**Greedy Algorithm**
Strengths	- Optimal solution - Ensure handling Task dependencies	- Handle constraints in an effective way - Fast for small problems	- Straight forward - Runs effectively for all types of large inputs
Weakness	- Hard to implement - Not cheap; consumes too much memory (High Space complexity)	- Slow for large problems - Exponential complexity - Explicitly checking constraints after each iteration -> more complexity	- An optimal solution is not guaranteed - Greedy decisions may ignore handling dependencies.
Cases Where One Might Outperform the Others	- (Best) Most efficient and guarantees optimal solution -> optimal substructure	- Suitable for problems when all the solutions must be explored - Suitable for problems that support flexibility and exhaustiveness as it supports	- Ideal for large problems as it prioritizes speed. - Suitable for simple constraints

		complex constraints	

References :

- https://www.smartfactorymom.com/blog/production-planning-and-scheduling-in-manufacturing/
- https://www.geeksforgeeks.org/job-sequencing-problem/
- https://leetcode.com/discuss/interview-question?currentPage=1&orderBy=most_relevant&query=Task%20scheduling%20problem%20c%2B%2B
- https://www.geeksforgeeks.org/find-the-ordering-of-tasks-from-given-dependencies/
- Analysis and Design of Algorithms Lab - Dr May Slides (Canvas)
- Analysis and Design of Algorithms course - Dr Nouri Slides (Google Drive)