GIVEN:
An arbitrary workflow, i.e. directed acyclic graph (DAG) G=(V, E), where each node v is a job and each directed edge (u, v) shows that the output data of job u is transferred as the input data of job v, K homogeneous machines, the execution time t(v) of each job running individually on a single machine, and the communication time t(u, v) of each data transfer between jobs u and v running in different machines.

QUESTION:
Find the minimum execution time of the entire workflow along with a feasible schedule mapping all the jobs onto no more than K machines

REQUIREMENT:
Comment on the difficulty (i.e., the problem's computational complexity) of the above problem, design an efficient algorithm, analyze your algorithm's time complexity, implement the solution, (which may be an exact, approximation, or heuristic algorithm), and show the execution results. (C/C++ in Linux and Makefile are required.)

Answers:

Problem's Computational Complexity Analysis:
I think this problem is in the NP-hard class of problems because we cannot verify whether a schedule is the optimal schedule or not. Computationally, it is very difficult to find the best schedule with the shortest execution time, and it is not easy to check.

This problem has several complexities. First, job dependencies require our jobs to be in a particular order. Second, different data transfer times between machines and assigning machines to do a job make our problem more complex. The last complexity is finding a schedule with the minimum total time, which is an optimization part. For this part, we cannot easily check a solution, especially with the communication time for dependent jobs on different machines, which makes this even harder for us.

Algorithm Design:
Because it is an optimization problem and very complex, it needs an algorithm to explore a large space and find a near-optimal or optimal solution in a reasonable time. I know two algorithms to solve these kinds of problems: one is a genetic algorithm, and the other is simulated annealing. I prefer to use the genetic algorithm.

A genetic algorithm is inspired by the process of natural selection. We should go through several steps as outlined below to find our solution.

1. Initialization: Generate an initial population of random schedules. Each schedule assigns jobs to machines while respecting the topological order.

2. Fitness Calculation: Evaluate each schedule based on the total execution time, including job execution and communication times.
3.Selection: Select the top half of the population based on fitness to be parents for the next generation.
4.Crossover: Create new schedules by combining parts of two parent schedules.
5. Mutation: Introduce random changes to some schedules to maintain genetic diversity.
6. Iteration: Repeat the selection, crossover, and mutation steps for a predefined number of generations or until convergence.

Time Complexity Analysis:

Initialization:
O(P * N), where P is the population size and N is the number of jobs. Initializing each schedule involves assigning jobs to machines randomly.

Fitness Calculation:
O(P * N * D), where D is the average number of dependencies per job. For each schedule, calculating the fitness requires checking dependencies and communication times.

Selection:
O(P log P). Sorting the population by fitness to select the top half.

Crossover:
O(P * N). Creating new schedules by combining parent schedules.

Mutation:
O(P * N). Randomly changing the assignment of jobs to machines.

Overall:
The overall time complexity for each generation is O(P * N * D + P log P). For G generations, the total time complexity is O(G * (P * N * D + P log P)).

Execution Results:

```
In Generation 1, The Best Completion Time is 137
In Generation 2, The Best Completion Time is 131
In Generation 3, The Best Completion Time is 131
In Generation 4, The Best Completion Time is 131
In Generation 5, The Best Completion Time is 131
In Generation 6, The Best Completion Time is 126
In Generation 7, The Best Completion Time is 122
In Generation 8, The Best Completion Time is 122
In Generation 9, The Best Completion Time is 119
In Generation 10, The Best Completion Time is 118
In Generation 11, The Best Completion Time is 118
In Generation 12, The Best Completion Time is 118
In Generation 13, The Best Completion Time is 118
In Generation 14, The Best Completion Time is 118
In Generation 15, The Best Completion Time is 118
Final Best Completion Time is 118
Best Schedule That We Have Found:
Machine 1: Job0 Job1 Job3
Machine 2: Job2 Job4 Job5 Job6 Job7 Job8 Job9 Job10 Job11 Job12 Job13 Job14 Job15 Job16
Job17 Job18 Job19
Machine 3:
[Finished in 22.6s]
```

We had 20 jobs with dependencies and this was my parameters:
K = 3 = number of machines
Population Size = 80000
Generations = 15 = number of generations