Hand-in Assignment 2

Abstract

This hand-in assignment concerns solving each task and design a linear model to describe the real systems presented in the sections. Results have to be presented in the form of a short-written report (max 6 pages). Together with the report, you are required to upload your code. As explained in the following section, your program has to be able to parse any problem instance of the Job Shop with the mentioned structure and solve it, returning the optimal value of the objective function and the running time. Your program must also be able to provide, given a directed graph, for each node, the shortest distance to any other node (using the A* algorithm)

Task I: The classical Job Shop Problem

The scheduling problem you are supposed to model is called the job shop problem (JSP) and is described as follows.

The JSP is defined by a finite set *J* of *n* jobs and a finite set *M* of *m* machines. When dealing with the standard JSP, we may assume the problem has the size *n* x *m*, that is, all jobs have the same number of operations, but in general it does not have to be so. For each job $j \in J$, we are given a list $(\sigma_1^j, \dots, \sigma_h^j, \dots, \sigma_m^j)$ of the machines which represents the processing order of *j* through the machines. Note that σ_h^j is called the *h*-th operation of job *j* and σ_m^j is the last operation of job *j*. In addition, for each job *j* and machine *i*, we are given a non-negative integer p_{ij}, which represents the processing time of *j* on *i*. Each machine can process at most one job at a time, and once a job starts on a given machine, it must complete processing on that machine without interruption. The objective is to find a schedule of *J* on *M* that minimizes the *makespan*, i.e., the maximum completion time of the last operation of any job in *J*. Makespan minimization for the JSP is a difficult problem to solve for $n \geq 3$ and $m \geq 2$.

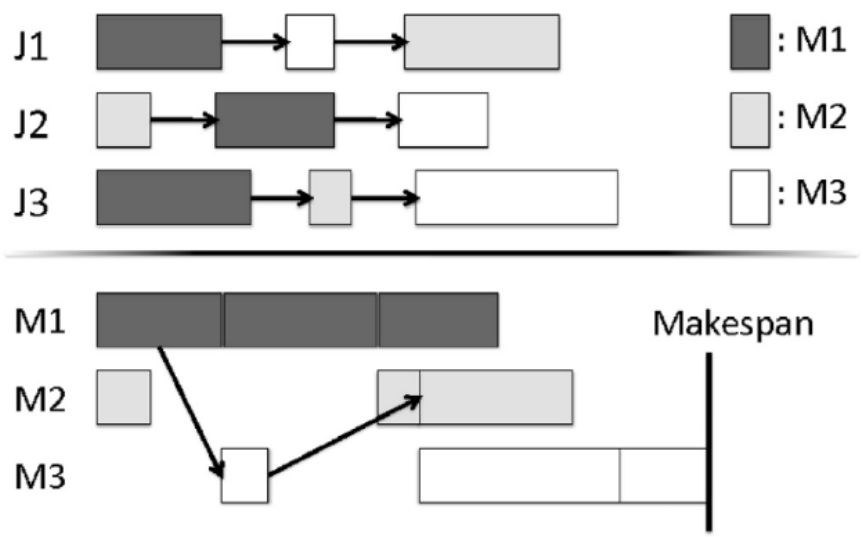

Fig. 1. Job shop scheduling problem. Three jobs J1, J2, and J3 are to be scheduled on three machines M1, M2 and M3. The graph on the top represents the precedence constraints. The Gantt Chart on the bottom displays a feasible schedule which satisfies the precedence constraints.

Example: 3 jobs, 4 machines:

Jobs	Machine Sequence	Processing Times
1	1,2,3	p_{11}=10, p_{21}=8, p_{31}=4
2	2,1,4,3	p_{22}=8, p_{12}=3, p_{42}=5, p_{32}=6
3	1,2,4	p_{13}=4, p_{23}=7, p_{43}=3

Your Task:

Given the problem description above, your job is to model the JSP using two approaches: Mixed Integer Linear Programming (MILP) formulation, and Constraint Programming formulation. You will need to create models that are scalable, i.e. can use inputs of different size. After you have created and debugged your models, you should test both of them for some input data (see the next section). You should report how fast it is to solve each model using its corresponding solver, i.e. Gurobi, or Z3. In principle, you should also get the same objective function value (not necessarily the same variable values) since the solvers use exact algorithms

You can test your code on as many instances as you like (and include the results in the report), though the mandatory ones are: ft06, ft10, la01 to la05.

For ft10 you can set up a timeout (commands are available for both Gurobi and Z3 in the documentation) of 20 minutes or so if the search is taking too long.

What you have:

1- You can find a very large set of test problems here:

https://github.com/google/or-tools/tree/master/examples/data/jobshop

The original format of each test problem is something like:

++++++++++++++++++++++++++++

instance ft06

++++++++++++++++++++++++++++

Fisher and Thompson 6x6 instance, alternate name (mt06)

6 6

2 1 0 3 1 6 3 7 5 3 4 6

1 8 2 5 4 10 5 10 0 10 3 4

2 5 3 4 5 8 0 9 1 1 4 7

1 5 0 5 2 5 3 3 4 8 5 9

2 9 1 3 4 5 5 4 0 3 3 1

1 3 3 3 5 9 0 10 4 4 2 1

In the above table, 6 6 means the problem has a 6 x 6 dimension. In the first row you see 12 numbers. Each pair of two numbers represents machine process order and processing time respectively, for each job on its corresponding line. For example, the first line corresponds to the first job, and the values are

2 1 0 3 1 6 3 7 5 3 4 6 which mean the first operation of job 1 is done on machine 2 with duration 1, the second operation is done on machine 0 with duration 3, the third operation is done on machine 1 with duration 6 and so on.

Hint: Python offers several packages to parse text files; we recommend **csv** (plenty of documentation can be found on the web). A good way to tackle the problem is to generate **lists** (or even better **dictionaries**) containing the data regarding the instance and then use them to encode the constraints through the solver.

Task II: The Extended Job Shop Problem

In the previous task, there were some implicit assumptions; one of them was that it took no time to move a part (job) from one machine to the following one. In a real plant though, this transportation time may not be negligible. In fact, let us assume that figure 1 represents the shop floor where the jobs of Fisher & Thompson 6x6 are executed. Distances are given in meters; for example, the (shortest) transportation distance from M6 to the delivery area is 28+19+28+17+8 meters long. Also, the transportation speed can be assumed to be constant (5 meters per second) and it can be assumed that processing times in the problem "ft06" are expressed in seconds).

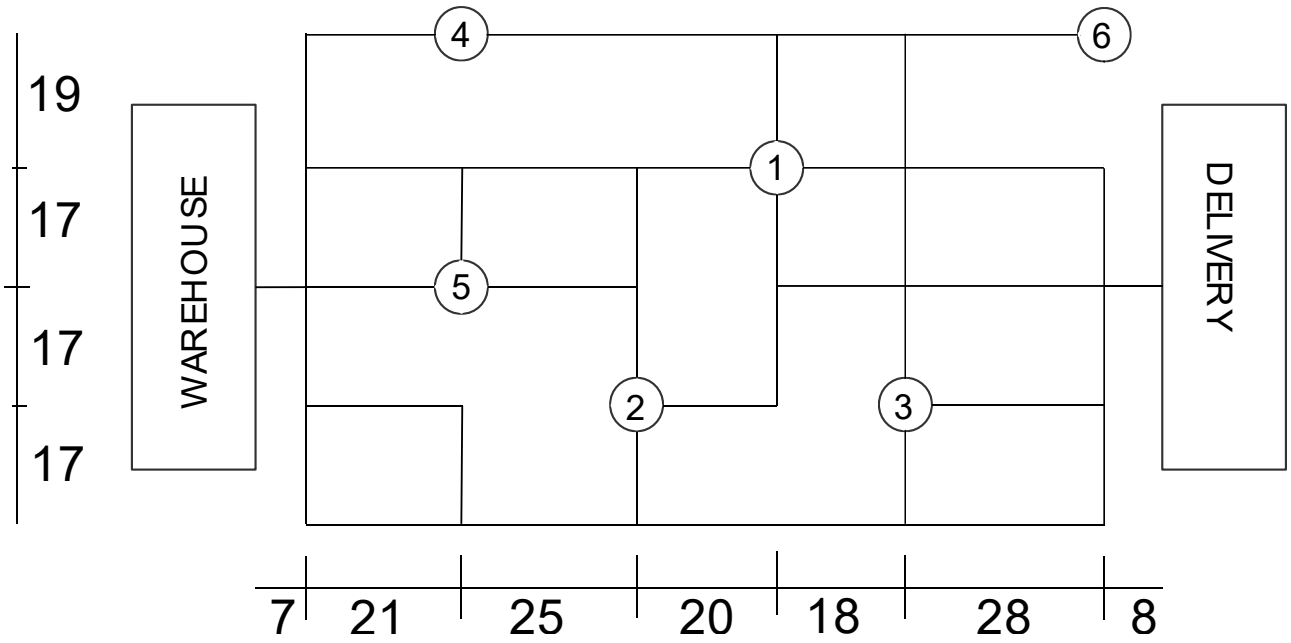

Figure 1: shop floor layout

Your Task: design a new model for the job shop problem that includes also the travelling distance between the different locations on the shop floor; all parts come from the warehouse and are stored in the delivery area after processing. In order to calculate the shortest path between any pair of locations in the plant, implement the A* algorithm and use a suitable heuristic.

In the report, present the new model for the JSP, together with a flowchart or pseudo-code for the A* algorithm. Also, present a table with the shortest paths between any pair of points in the plant. Finally, report the new objective function value for the problem "ft06".

Task III: Finding the Shortest Path using Z3

Though Dijkstra can find the shortest path in a graph very efficiently, we want to be able to define specific requirements on the paths, such as nodes we do/do not want to visit, or specific sequences of nodes. We can achieve this by modelling the problem using Boolean variables, as described in the attachment.

Your Task: implement the model described in the attachment and extend it so it can enumerate all paths for each pair of nodes in the graph. Build your model using only Boolean variables. Then use it to find *the 10 shortest paths* paths connecting “Machine 5” to “Machine 3”

Hints: unlike in the model presented in the attachment, you cannot sum variables or multiply them by constants, because they are Booleans, hence they can only be True or False. Therefore, you need to use cardinality constraints (take a look at the file **exactly_one.py** in the folder Solvers_material/code) and other “tricks” to model the problem.

As for the enumeration, a good starting point would be to find a way to keep track of the solution you find, so you can rule it out from the possible solutions.