ALC 2018/2019
3^{rd} Project – Job-Flow Scheduling Problem with CSP

20/Nov/2018, version 1.0

Overview

The 3rd ALC project is to develop a software tool for solving the Job-Flow Scheduling (JFS) problem [1]. In order to solve this problem, students must use the minizinc CSP framework.

Problem Specification

The Job-Flow Scheduling problem (JFS) is a particular case of the open Job-Shop Scheduling problem, commonly used in planning industrial production and management.

The Job-Shop Scheduling problem can be defined as follows. Let J define a set of n jobs $\{J_1, J_2, \ldots, J_n\}$ and let M define a set of m machines $\{M_1, M_2, \ldots, M_m\}$. For each job J_j there is a sequence $T = \{t_{j,1}, t_{j,2}, \ldots, t_{j,t}\}$ of t tasks to be completed. The tasks must be executed in sequence and task $t_{j,k}$ cannot start before task $t_{j,k-1}$ is completed. Moreover, each task $t_{j,k}$ occupies one and only one machine in M and it needs $d_{j,k}$ units of time to be completed. **Finally, a task can be interrupted. However, a machine cannot be used by two tasks at the same time.** The most common goal in Job-Shop Scheduling is to find the smallest makespan U such that all jobs are executed between starting time 0 and U. Hence, the time interval $[0, U]$ defines the smallest interval where all jobs can be completed.

In the particular case of the Job-Flow Scheduling problem, M defines a sequence of machines that is always fixed for all jobs. Hence, the first task must be carried out at machine M_1, the second at machine M_2, etc.

Table 1 illustrates a Job-Flow Scheduling problem with 2 jobs and 3 machines. For this example, the optimal makespan is 9 and figure 1 provides an optimal schedule for these tasks. Observe that if task 2 of job 2 could not be interrupted, then the optimal makespan would be 10.

Project Goals

You are to implement a tool, or optionally a set of tools, invoked with command `proj3`. This set of tools should use minizinc solver to compute the schedule of a Job-Flow problem.

Your tool does not take any command-line arguments. The problem instance is to be read from the standard input.

$d_{i,j}$	Machine 1	Machine 2	Machine 3
Job 1	2	1	4
Job 2	1	6	1

Table 1: Job-Flow Scheduling Problem

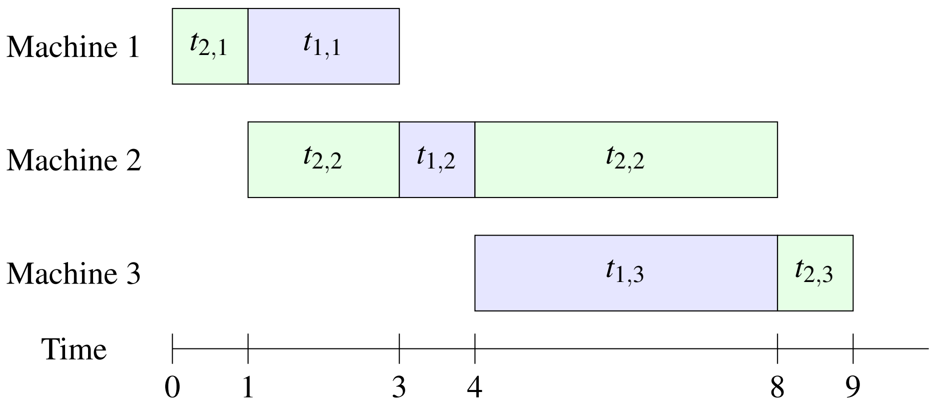

Figure 1: Job-Flow Scheduling Solution

Consider an instance file named `job.jfp`. The tool is expected to be executed as follows:

```
proj3 < job.jfp > solution.txt
```

The tool must write the solution to the standard output, which can then be redirected to a file (e.g., `solution.txt`).

The programming languages to be used are only C/C++, Java or Python. The formats of the files used by the tool are described below.

File Formats

You can assume that all input files follow the description provided in this document. There is no need to check if the input file is correct. Additionally, all lines (input or output) must terminate with the end-of-line character.

Input Format

The input file representing a problem instance is a text file that follows the same format as projects 1 and 2:

- One line with two integers n and m defining the number of jobs and machines.

- A sequence of n lines where the j^{th} line describes the tasks for job j. Each line describing a job contains the following:
  - An integer k denoting the number of machines to be used.
  - A sequence of k pairs of integers separated by the character : denoting the machine identification (an integer between 1 and m) and the duration of the task on that machine. The duration of a task is an integer greater or equal to 1.

Observe that in our formulation, each job does not need to use all m machines. Nevertheless, for a given job J_j, the identification of machine used in task $t_{j,k}$ is always greater than the identification of machine used in task $t_{j,k-1}$.

Output Format

The output of the program representing an optimal solution to the problem instance must comply with the following format:

- One line with an integer U defining the makespan of the optimal solution;
- One line with two integers n and m defining the number of jobs and machines;
- A sequence of n lines where each line contains the schedule for each job. Hence, for the j^{th} line in the sequence:
  - An integer k denoting the number of machines used in job j;
  - A sequence of k **triples** of integers separated by the character : denoting the machine identification, the starting time of the task and **the time duration of the machine's occupancy. This sequence must be sorted first by the machine identification and then by the starting time of the task's fragment.**

Important: The final version to be submitted for evaluation must comply with the described output. Project submissions that do not comply will be severely penalized, since each incorrect output will be considered as a wrong answer. An application that verifies if the output complies with the description is available on the course's website.

Example

The file describing the problem in Table 1 is as follows:

```
2 3
3 1:2 2:1 3:4
3 1:1 2:6 3:1
```

The optimal solution corresponding to Figure 1 would be:

```
9
2 3
3 1:1:2 2:3:1 3:4:4
3 1:0:1 2:1:2 2:4:4 3:8:1
```

Additional Information

The project is to be implemented in groups of one or two students.

The project is to be submitted through the course website. Jointly with your code, you should submit a short text file describing the main features of your project.

The evaluation will be made taking into account correctness given a reasonable amount of CPU time (80%) and efficiency (20%).

The input and output formats described in this document must be strictly followed.

Project Dates

- Project published: 20/11/2018.
- Project due: 30/11/2018 at 23:59.

Omissions & Errors

Any detected omissions or errors will be added to future versions of this document. Any required clarifications will be made available through the course's official website.

Versions

20/11/2018, version 1.0: Original version.

References

[1] P. Brucker, Y. N. Sotskov, and F. Werner. Complexity of shop-scheduling problems with fixed number of jobs: a survey. *Mathematical Methods of Operations Research*, 65(3):461–481, Jun 2007.