ALC 2018/2019
2^{nd} Project – Job-Flow Scheduling Problem with SMT

02/Nov/2018, version 1.0

Overview

The 2nd 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 solvers for Satisfiability Modulo Theories (SMT).

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 during $d_{j,k}$ units of time. Finally, a task cannot be interrupted and 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 3 jobs and 2 machines. For this example, the optimal makespan is 8. Figure 1 provides an optimal schedule for these tasks.

Project Goals

You are to implement a tool, or optionally a set of tools, invoked with command `proj2`. This set of tools must use an SMT 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.

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

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

Table 1: Job-Flow Scheduling Problem

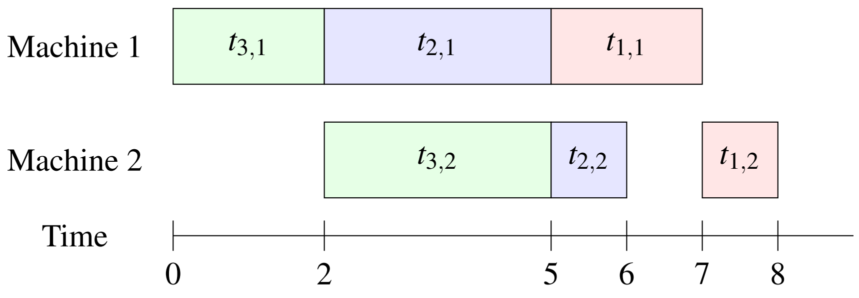

Figure 1: Job-Flow Scheduling Solution

```
proj2 < 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

In this project, the input and output formats must be strictly followed, as described in the 1st project. We refer to the 1st project for a complete description and examples.

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: 02/11/2018.
- Project due: 16/11/2018 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

02/11/2018, 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.