Homework CP-2: Task Scheduling

Assigned: 15.12.2016 – Due: 12.01.2017 23:59

1 Problem Definition

You are the new director of a computing center and need to write a program to assign tasks to machines. The tasks are given in a single batch and the goal is to compute all of the tasks as quickly as possible (makespan minimization). You are given a set of machines M, a set of tasks T and a graph of precedence constraints $G = (T, A)$ where an arc $(i, j) \in A$ between two tasks $i, j \in T$ means that task i must finish before task j can start. Furthermore, each task has a duration d_{tm} that is machine specific. Note that task j may start at time $\text{start}(i) + d_{i,\text{machine}(i)}$, i.e., the job ends some epsilon before the next job would start. There is a set of resources, R, and each task has a resource consumption given by u_{tr} for resource r. Each machine has a maximum usage g_{mr} that may not be exceeded at any point in time. The machines in your computing center are particularly strange, because after they compute k_m tasks they become tired and shut down, and are unable to compute any more tasks in the planning period. Assume all values given are integers and that the starting time of tasks is always an integer value. The planning period is at most t^{max} units long.

Your task is to construct a constraint programming model for the given problem, using global constraints whenever possible, and to implement the model in the MiniZinc framework.

2 Tasks

This homework consists of three tasks:

1. Write down a CP model for the above problem, formalizing all components of the model and listing them clearly.

2. Implement the model in MiniZinc

3. Try out at least three different search annotations and analyze the runtime performance of your model under different annotations, including these results/analysis in the final document.

4. Investigate at least three different CP solvers and present results showing which has the best performance on our instance set.

3 Technical requirements

3.1 Execution

All code must be command line executable with the command:

<minizinc solver> hw2.mzn <instance.dzn>

3.2 Input Format

The input format is in the MiniZinc dzn format:

machines = $|M|$;
tasks = $|T|$;
resources = $|R|$;
precedes = [|0, δ_{12}, ... $\delta_{1|T|}$,
|δ_{21}, 0, ... $\delta_{2|T|}$,
|$\delta_{|T|1}$, $\delta_{|T|2}$, ... 0|];
duration = [|d_{11}, d_{12}, ... $d_{1|M|}$,
|d_{21}, d_{22}, ... $d_{2|M|}$,
|$d_{|T|1}$, $d_{|T|2}$, ... $d_{|T||M|}$|];
rconsumption = [|u_{11}, u_{12}, ... $u_{1|R|}$,
|u_{21}, u_{22}, ... $u_{2|R|}$,
|$u_{|T|1}$, $u_{|T|2}$, ... $u_{|T||R|}$|];
rmax = [|g_{11}, g_{12}, ... $g_{1|R|}$,
|g_{21}, g_{22}, ... $g_{2|R|}$,
|$g_{|M|1}$, $g_{|M|2}$, ... $g_{|M||R|}$|];
tired = [k_1, k_2, ..., $k_{|M|}$];
maxtime = t^{max};

Note that the precedence matrix is provided as a dense matrix where each entry δ_{ij} is equal to 1 if i must precede j and 0 otherwise. Be assured that the problem sizes will not be large enough to use lots of memory.

3.3 Output Format

Your program has to output a solution in the following form, if one was found:

Solution: $f(s)$
Task 1: machine_number
⋮
Task $|T|$: Machine machine_number; Start start_time

We will restrict the amount of time your program is allowed to solve using the *nix timeout command from GNU coreutils. It is recommended that you use this command as well when testing your program under linux or OS X. In Windows, you can either use timeout within a cygwin shell or use an equivalent for the Windows command line.

3.3.1 Example Instance

```
machines = 2;
tasks = 5;
resources = 2;
precedes =  [|0,  1,  0,  1,  0,
            |0,   0,  1,  0,  0,
            |0,   0,  0,  0,  0,
            |0,   1,  0,  0,  1,
            |0,   0,  0,  0,  0|];
duration =  [|3,  2,  2,  1,  4,
            |2,   1,  4,  2,  3|];
rconsumption =  [|1,  4,
                |3,   1,
                |2,   2,
                |2,   1,
                |1,   3|];
rmax =  [|4,  3,
        |3,   4|];
tired = [2, 4];
maxtime = 15;
```

```
Solution:  8
Task 1:  Machine 2; Start 1 Task 2:  Machine 2; Start 4 Task 3:  Machine
1; Start 5 Task 4:  Machine 1; Start 3 Task 5:  Machine 2; Start
5
```

4 Miscellaneous

If the output of your program does not match the output format described in Section 3.3, you will be receive 0 points without any exception.

4.1 Submission

The solution (implementation) of this assignment must be handed before Thursday, 12.01.2017, 23:59 (midnight). Please combine your .mzn and model description as a pdf in a single .zip or .tar.gz and upload them into Koala.

4.2 Collaboration policy

This project is not a group project and as such there may be no sharing of code between students. Discussion of high level concepts is, however, allowed and encouraged. Students may not share their mathematical model with other students.

Code re-use policy

Standard libraries as well as libraries/code snippets from the internet may be used as long as the fulfill all of the following:

1. They are credited in the submission document, as well as in the submitted code,
2. They do not solve the given task directly (i.e., a solver accepting a model is OK)
3. The code license permits its use in 3rd party works.

In other words, third party code can be used as part of the instance reading, to assist in objective evaluation, or can be used within neighborhood/initial solution/termination heuristics, but cannot completely replace

these components. The overall search strategy itself, however, must be completely the own work of the student.

5 Grading

Homeworks will be evaluated based on the following criteria:

- Correctness, quality, presentation and tightness of the mathematical model(s)
- Correctness and quality of the model implementation
- Performance of the model