Discrete Optimization Specialization: Workshop 10
Preparing Alchemical Potions

1 Introduction

Shennong has resolved the curse on the village so that no new boys and girls will become sick, and prepared the herbs needed. He now needs to create the potions from the herbs to heal the children who are already sick. Some potions are prerequisite ingredients for other potions, and need to be created before the potions for which they are ingredients. Shennong will need help from alchemists to prepare the potions. He can prepare at most a maximum number of potions each day, since he has limited equipment. He also has to prepare a minimum number to keep the alchemists busy. Each potion needs a certain number of alchemists to prepare. Shennong wants to deploy at least a certain number of alchemists each day to get his money's worth, but the capacity of the workplace imposes a maximum number on the alchemists who can work there on a day. He has to hire the alchemists for the entire period, so the aim is to minimize the maximum number of alchemists required for each day across all days.

Preparing Potions — `alchemy.mzn`

The data for the problem is encoded as, a number of potions that need to be created, with the number of alchemists each requires, the number of days, a number of precedences among the potions, and a minimum and maximum number of potions to be made each day, and a minimum and maximum number of alchemists to deploy each day.

An example data file is

```
n_potions = 10;
alchemy = [6, 3, 5, 3, 7, 3, 1, 2, 4, 3 ];
n_days = 6;
prereqs = [|3,1 |4,1 |5,1 |6,1 |7,1 |6,2 |8,2 |];
min_potions = 1;
max_potions = 4;
min_alchemists = 2;
max_alchemists = 12;
```

For example, potion 3 needs to be completed before potion 1.

We are given a model for the problem `alchemy.mzn`. The model has three different representations of the decisions: `day` to represent which day each potion is made, `potions` to represent the set of potions made on each day, and `potion_day` are Booleans which hold if a given potion is made on a given day. Each of the constraints for the model are given in three versions: using the three different decisions. The model also has channelling constraints between the decisions of each pair of representations. The model also has three alternate output statements. You can use any one of them.

The aim of this workshop is to find the best representation of the model and the best programmed search strategy you can for the given data files. The rules are that you can comment out

parts of the model, but not add any new parts except to define search strategies. You should find the right representation to keep for each constraint to give the best possible model. Note that the model includes some redundant constraints that are marked as such. You can omit them entirely if you desire.

Experiment with which variables to search over `day`, `potions`, `potion_day` and the variable selections:

> `input_order`, `first_fail`, `smallest`, `largest`, `dom_w_deg`

and value selections:

> `indomain_min`, `indomain_max`, `outdomain_min`, `outdomain_max`, `indomain_median`, `indomain_random`, `indomain_split`, `indomain_reverse_split`.

You should also investigate restarting to improve the search, using the search annotations

> `restart_luby`, `restart_geometric`, `restart_linear` or `restart_constant`

to make use of one of the restart strategies.

Use the `-s` statistics flag (or check box in the IDE) to compare how much search each search strategy uses. Use the command line

> `--time-limit` X

(or the time limit in the solver configuration part of check box in the IDE) to limit the execution time to X milliseconds.

Compare strategies in terms of

- *completeness*: how often it proves the optimal solution (in the time you are prepared to wait)
- *search-space*: how many failures to prove the optimal solution
- *robustness*: how good a solution it finds in some fixed time (1 minute say)

Note that data files from `alchemy-*.dzn` are easier than the `alchemy12-*.dzn` and considerably more varied. You should concentrate on the first class, and use the second class to verify your experiments. What do you think is the best search strategy overall? Justify your answer.

2 Technical Requirements

For completing the workshop you will need MiniZinc 2.2.x (`http://www.minizinc.org/software.html`).