School of Computer Science
University of St Andrews
2021-22
CS4402
Constraint Programming
Practical 1: Solar Power Company Problem

This Practical comprises 50% of the coursework component of CS4402. It is due 26th October at 21:00.

The deliverables consist of:

- A report, the contents of which are specified below.
- The Essence Prime constraint model for the basic specification.
- The Essence Prime constraint model(s), and possibly additional instances, for the extension part.

The practical will be marked following the standard mark descriptors as given in the Student Handbook (see link below). Please read this document *carefully*. There are a number of requirements for the problem being modelled, and the way in which your solution and experiments should be reported.

The Solar Power Company Problem

The Solar Panel Placement Problem (SPCP) is to decide on *two* main components:

- How to allocate solar panels to buildings to best meet the power demands of each building.
- How to allocate workers to install the solar panels.

The company has a set of solar panels in the warehouse, each of which has two attributes: an area and an average power output. For each building, we are given its roof area and its average power demand. Areas are measured in m^2 and power in kW. Note that each panel is unique. i.e. it cannot be installed twice!

Each solar panel must be placed on one building.

The total area of panels on a building cannot be greater than the building's roof area.

The company has a limited number of days which they can use to complete the installations they agreed upon, by sending workers to install the different solar panels.

Each worker has a limited "output": they can only install up to a certain number of square meters of solar panels per day. Some workers are more efficient than others, and therefore they might be able to install more square meters per day, but they also might ask for a better pay.

The installation of a panel is a very intricate process, and therefore each panel is installed by one and only one worker. The installation of a panel must also be finished the same day it is started.

Irrespectively if a worker is installing either a few or lots of solar panels in a given day, they must be paid their full amount that day.

The company has a limited budget per day for paying their workers. The amount spent for each day in worker salaries cannot surpass the daily budget.

Buildings are all connected to the power grid. Building owners can purchase power from the grid to make up a shortfall (where the power generated by the panels is less than the power demand of the building). Power can also be sold to the grid by building owners if there is surplus power generated by the panels (i.e. the power generated by the panels is greater than the power demand of the building). However, both buying and selling power is inefficient because of transmission losses in the power grid.

In addition to the constraints described above, SPCP has an optimisation function: The goal is to minimise *both* the power purchased from the grid *and* the power sold to the grid, in total for the set of buildings.

We want also to penalise strong outliers: that is, installations in buildings that are far from their ideal power demand. We are required to do it by squaring the difference between the generated and the required power for each building. More precisely, the goal is to minimise, for each building, the squared difference of its power produced and power required.

Example

Suppose we have **2 days** to install the 4 solar panels:

Solar panel	**Power Generated**	**Area**
1	13	4
2	20	5
3	12	6
4	15	3

On two buildings:

Building	**Power Demand**	**Roof Area**
1	32	11
2	30	9

Using two workers and a budget of 60 per day:

Worker	**Area installed per day**	**Salary per day**
1	8	20
2	9	30

In a solution, each one of the panels must be placed on one of the two buildings without exceeding the building's roof area. One solution would place panels 1 and 3 on building 1, and the others on building 2:

Building	**Panels**	**Power Generated**	**Power Demand**	**Power Shortfall**
1	1, 3	25	32	7
2	2, 4	35	30	-5

Worker 1 would install panels 1,2 and 4, while worker 2 would install panel 3:

Worker	**Panels day 1**	**Panels day 2**	**Salary day 1**	**Salary day 2**
1	1	2,4	20	20
2	3		30	0

Note that the sum of salaries for both days (30+20 and 20 respectively) is less than the budget of 60 per day. Also, worker 1 is installing panels 2 and 4 on the second day. He can install up to 9 square meters per day and panels 2 and 4 have a total area of 5+3, so he is still able to manage.

This solution has a large shortfall in power for building 1, and a large surplus for building 2. The total power bought from or sold to the grid is 12 kW. It is possible to find a much better solution:

Placing panels 2 and 3 on building 1, and the others on building 2, is much more efficient:

Building	**Panels**	**Power Generated**	**Power Demand**	**Power Shortfall**
1	2, 3	32	32	0
2	1, 4	28	30	2

In this solution, the total power bought from or sold to the grid is only 2 kW. In fact this is an optimal solution. Note that we can also do a better assignment regarding the workers, but we don't need to optimise that.

Parameter Format and Modelling

For simplicity we will standardise the format of the parameters, for which we shall use the following identifiers:

```
language ESSENCE' 1.0

letting numPanels = 4

letting panelPower=[13,20,12,15]

letting panelArea=[4,5,6,3]

letting numBuilds = 2

letting buildPower=[32,30]

letting buildArea=[11,9]

letting numWorkers = 2

letting workercapacity = [8,9]

letting workercost = [20,30]

letting budgetPerDay = 50

letting numDays = 2
```

The identifiers `numPanels, panelPower and panelArea` identify the number of panels and their attributes (power generated and squared meters).

`numBuilds, buildPower and buildArea` describe the buildings that we need to supply, giving the number of buildings, their power requirements and their available squared meters on the roof.

Similarly, `numWorkers, workercapacity` and `workercost` describe the number of workers we have available, their squared meters per day and their salary per day.

Finally `budgetPerDay` describes how much we can spend on salaries per day and `numDays` the number of days that we can use to make all the installations.

As a hint and example, you can use the `numPanels` and `panelPower` identifiers in your model as follows:

```
given numPanels:int(1..)
given panelPower : matrix indexed by [int(1..numPanels)] of int(1..)
```

From here, you will now be able to use the identifier `numPanels` to refer to the number of panels and, for example, `panelPower[3]` to access the power generated by the third panel in the rest of your model.

Along with this practical specification, you will find a set of .param files with which to test your model – see the section on Empirical Evaluation below.

General advice: debugging constraint models can be difficult because a conflicting set of constraints will cause the solver to return no solution without there being an obvious cause. For this reason, you are encouraged to build up the set of variables and constraints in your model **incrementally**, continually testing them on small instances.

Empirical Evaluation

Evaluate the performance of your model on the provided instances supplied with this practical specification. In doing so use the default settings of Savile Row, such as:

```
./savilerow -run-solver spcp.eprime X.param
```

You should, at least, record in your report for each instance: the `SolverNodes, SolverSolveTime, Savile Row TotalTime`. As discussed in Tutorial 1, these are available in the `.info` file after your model has been run, irrespective of whether the instance is solvable.

There are some ideas for a more extensive empirical evaluation in the Extensions section below.

Report

Your report should at least have an **Introduction** and **Conclusion** sections. In addition to that, the report should include:

- **Variables and Domains**: Describe how you chose the variables and domains to represent the problem.
- **Constraints**: Describe each of the (sets of) constraints you have added to your model, and their purpose.
- **Additional constraints**: If you have added any implied or symmetry breaking constraints, explain them here.
- **Example Solution**: Demonstrate that your model is correct by showing the solution it produces for the supplied satisfiable instance named **spcp2.param**. The style used in the example given above is fine for this purpose.
- **Empirical Evaluation**: Describe your experimental setup, and record and analyse the output of the empirical evaluation described above. Discuss the differences in performance of both viewpoints.
- **Extensions**: Describe here the extensions you have attempted, if any. Empirical work should be reported as above.

Extensions

Some ideas for extension activities follow. These are not required, but attempting at least one extension activity is required to gain a mark above 17.
You may wish to explore and explain:

- Giving minion a variable order for the search (SR manual Section 2.5) and checking other processing Optimisation Levels that Savile Row provides (SR manual Section 4.5.3).
- Try different solver backends (SR manual Section 1.2).
- Formulate a second model of SPCP with a different viewpoint compared to your first model.
- What is the most challenging instance you can create with 10 panels, 5 buildings and 3 workers? Discuss why you think it is challenging for your model.

Marking

The practical will be marked following the standard mark descriptors as given in the Student Handbook (see link below). Further guidance as to what is expected follows:

- To achieve a mark of 7 or higher: A rudimentary attempt at a model, incomplete or with several errors. Adequate evaluation and report.
- To achieve a mark of 11 or higher: A reasonable attempt at a model, mostly complete, or complete and with a few flaws. Reasonably well evaluated and reported.
- To achieve a mark of 14 or higher: A good attempt at a model with only minor flaws, well evaluated and reported.
- To achieve a mark of 17: A fully correct model, very well evaluated and reported.
- To achieve a mark greater than 17: In addition to the requirements for a mark of 17, an attempt at some suggested extension activities.

Pointers

Your attention is drawn to the following:

- Mark Descriptors:
  `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html`
- Lateness:
  `https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html`
- Good Academic Practice:
  `https://info.cs.st-andrews.ac.uk/student-handbook/academic/gap.html`