Discrete Optimization Specialization: Workshop 8
The Dieda Plasters

1 Introduction

Huang Gai (a general serving under Zhou Yu) proposed a ruse to win Cao Cao's trust so that Zhou Yu's fire attack plan could be carried out. Huang Gai arranged with Zhou Yu to openly challenge and insult him at a meeting to discuss battle plans, giving Zhou Yu a reason to pretend to be furious and order his execution. When this occurred the rest of Zhou Yu's generals pleaded with Zhou Yu for clemency, and he (apparently) reluctantly spared Huang Gai's life but ordered him to be flogged severely.

Huang Gai then contacted Cao Cao offering to defect. Spies for Cao Cao in Zhou Yu's camp confirmed that Haung Gai has been severely beaten. Huang Gai arranged with Cao Cao to sail across the river to Cao Cao's camp on a certain night. His ship would be loaded with highly inflammable goods, so that actually this would be the night that Zhou Yu launched the fire attack on Cao Cao's ships. Unfortunately, because of the severe injuries to his back from the flogging, Huang Gai lost consciousness before he could carry out the final step of his plan.

Zhuge Liang obtained the Dieda Plasters, a legendary set of healing plasters created by Hua Tuo. The plasters could be used to miraclously heal Huang Gai's back. However, cutting and overlapping of the plasters is forbidden or otherwise their effect can be reversed causing permanent injury or even death.

The Dieda Plasters — `plasters.mzn`

Zhuge Liang must apply the plasters to Huang Gai's back to heal him sufficiently to carry out the plan. The Dieda Plasters are very valuable, and cutting them or overlapping them will possibly reverse their effectiveness. Hence he must plan how to cover Huang Gai's wounds with plasters without overlap for the least possible cost.

The data defining the problem is: *length* the length of Huang Gai's back, *width* the width of Huang Gai's back, *wound* a 2D array of Booleans showing the positions of wounds on Huang Gai's back, *ntypes* the number of types of plaster (each type of a different size), *number* the number of each type of plaster available, *dim* a 2D array mapping each plaster type to (len, wid) where $len \geq wid$, and *price* a mapping from plaster type to price of the plaster.

The output should be for each plaster how it is used:

- *Long*: used as defined by *dim*
- *Wide*: used rotated 90 degrees (i.e. with dimensions swapped)
- *Not*: not used.

and the x and y position of each plaster. Note that the x and y positions for unused plasters can be any legitimate value. The aim is to minimize the cost of the plasters used.

The data and decisions for problem are hence

```
int: length; % length of Huang Gai's back
set of int: LENGTH = 1..length;
int: width; % width of Huang Gai's back
set of int: WIDTH = 1..width;

array[LENGTH,WIDTH] of bool: wound; % wound positions

int: ntypes;  % number of types of plaster
set of int: TYPE = 1..ntypes;
array[TYPE] of int: number;          % number of plaster available
array[TYPE,1..2] of int: dim;        % dimensions of plaster
array[TYPE] of int: price;

int: total = sum(number);
set of int: PLASTER = 1..total;
array[TYPE] of int: psum = [ sum(i in 1..t-1)(number[i]) | t in TYPE ];
array[PLASTER] of TYPE: t = [ max(t in TYPE)(t*(p > psum[t])) | p in PLASTER
                ];
array[PLASTER] of var LENGTH:  x;
array[PLASTER] of var WIDTH:   y;
enum USAGE = { Long, Wide, Not };
array[PLASTER] of var USAGE:   u;
```

An example data file is

```
length = 10;
width = 6;

wound =
[| false, false, false, false, false, false
 | false, false, false, false, false, false
 | false, false, false, true,  true,  false
 | true,  false, false, false, false, false
 | false, true,  false, false, false, false
 | false, true,  true,  false, false, false
 | false, false, false, false, false, false
 | false, false, false, false, true,  false
 | false, false, false, false, false, false
 | false, false, false, false, true,  false
|];

ntypes = 6;
number = [2,2,2,2,2,2];
dim = [| 1,1
       | 2,1
       | 3,1
```

```
        | 2,2
        | 3,3
        | 5,1 |];
price = [ 12, 5, 6, 5, 17, 12 ];
```

which shows a total of 6 wounds on Huang Gai.

A solution that could be returned is

```
x = [1, 1, 3, 1, 8, 1, 1, 1, 4, 1, 1, 1];
y = [1, 1, 4, 1, 5, 1, 1, 1, 1, 1, 1, 1];
u = [Not, Not, Wide, Not, Long, Not, Not, Not, Long, Not, Not, Not];
cost = 28;
```

which makes use of a 2 × 1 plaster in Wide format e.g. 1 × 2 at (3,4), a 3 × 1 plaster at (8,5) and a 3× 3 plaster at (4,1). Note this is not the optimal solution. An illustration of the plastering diagram is below where `#` indicates a plaster over a wound, and "`.`" indicates a plaster over a non-wound

```
    123456
 1:
 2:
 3:    ##
 4: #..
 5: .#.
 6: .##
 7:
 8:     #
 9:     .
10:     #
```

Dominance — `dominance.mzn`

If we examine the data of the plasters we can notice something. A 3 × 3 plaster costs 17, but we could always tile the same area more cheaply using a 2 × 1, 2 × 2 and 3 × 1 plaster at a cost of 16.

We can think of this as a dominance rule: as long as we have one of each of a 2 × 1, 2 × 2 and 3 × 1 plaster available we should never use a 3 × 3 plaster. If we add an array

```
array[TYPE] of var 0..max(number): used;
```

which keeps track of how many of each type of plaster are used, we can write a dominance breaking constraint like

```
constraint    used[2] <= number[2] - 1 /\ used[3] <= number[3] - 1
           /\ used[4] <= number[4] - 1 -> used[5] = 0;
```

Write a model that reads the same input format as `plaster.mzn` and prints out a dominance rule that is correct for the input data on plasters.

Use it to discover any other dominance rule in the data above.

Ideally make it print out a constraint for each dominance rule that can be directly added to the `plaster` model.

See if the dominance rules you generate actually speed up solving (this will be dependent on your model and the search that occurs).

2 Technical Requirements

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