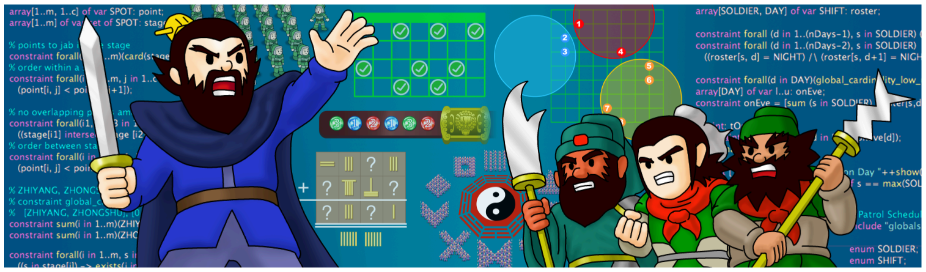
Cryptarithms for Seven Paces Verse

Basic Modeling for Discrete Optimization: Assignment 1



Problem Statement

Cao Zhi was the third son of the powerful warlord Cao Cao. He could recite more than ten thousand verses of poetry before he was ten, which made him a favourite of his father. But he was impetuous with little self-discipline. His older brother, Cao Pi was much more controlled. Eventually Cao Cao chose Cao Pi to succeed him, which did not sit well with Cao Zhi. When Cao Cao died Cao Zhi failed to turn up to his funeral, and was found drunk in his own house. He was brought bound to Cao Pi who was furious.

Cao Pi asked his brother, in order to show his worth and avoid execution, to prove his literary talent. Cao Zhi had to construct a poem about his relationship to his brother, without using the word "brother" within seven paces. Then, Cao Zhi took seven paces and recited a poem comparing them to two fighting bulls. His brother was not satisfied so demanded another poem. Without hesitation Cao Zhi recited:

> Cooking beans on a fire of beanstalks,
>
> The beans weep in the pot.
>
> Born of the same roots,
>
> Why the eagerness to destroy one another?

Cao Pi was moved to tears and had Cao Zhi released unharmed.

This assignment solves cryptarithm related to the seven paces story. A *cryptarithm* is a mathematical puzzle which requires determining the digit for each letter in an equation. The most famous cryptarithm is SEND + MORE = MONEY. That is we need to determine which digit each the letters represent so that,

```
    S E N D
  + M O R E
-----------
= M O N E Y
```

The rules of cryptarithms are:

- Each letter represents a different digit

- The first letter in each word cannot be 0 (otherwise it would not be a proper number)
- The arithmetic equation must hold.

This project will require modelling and solving cryptarithm problems.

An Example — CUHK+MELB=LORE

Consider the cryptarithm,

```
  C U H K
+ M E L B
---------
= L O R E
```

A MiniZinc model for this problem would be

```
var 1..9: C;
var 0..9: U;
var 0..9: H;
var 0..9: K;
var 1..9: M;
var 0..9: E;
var 1..9: L;
var 0..9: B;
var 0..9: O;
var 0..9: R;

constraint    1000 * C + 100 * U + 10 * H + K
            + 1000 * M + 100 * E + 10 * L + B
            = 1000 * L + 100 * O + 10 * R + E;

include "alldifferent.mzn";
constraint alldifferent([C,U,H,K,M,E,L,B,O,R]);

solve satisfy;
```

Part 1 — CUHK+MELB=LORE

Simply submit the provided `cuhkmelb.mzn` model. This will test that MiniZinc is installed and working correctly.

Part 2 — Beans are Crying

Build a MiniZinc model `beanscrying.mzn` which solves the problem,

```
    B E A N S
```

```
+   B E A N S
------------
= C R Y I N G
```

You should determine at least one solution.

Part 3 — Cao Zhi's Test

Build a MiniZinc model `caotest.mzn` which solves the problem,

```
    C A O
+   Z H I
+   C A O
+     P I
---------
= T E S T
```

with the additional constraints that `P` (for Pi) is worth 3 times `Z` (for his brother Zhi), and that `O` is equal to its partner shape 0. You should determine how many solutions there are to this problem.

Part 4 — Seven Paces Verse

Build a MiniZinc model `sevenpaces.mzn` which solves the problem,

```
  S E V E N
+ P A C E S
-----------
= V E R S E
```

Rather than simply finding a solution, your model should maximize the value of the number represented by VERSE.

Interface to the Grader

If you declare the necessary variables, for example `C`, `U`, `H`, `K`, `M`, `E`, `L`, `B`, `O`, `R` for part 1, then MiniZinc will be able to find the necessary variables to send to the grader. To visualize the result for yourself, you can also write an output item to output a value for each variable. For example, for part 1 we could use the following:

```
output
["  \(C) \(U) \(H) \(K)\n" ++
 "+ \(M) \(E) \(L) \(B)\n" ++
 "---------\n" ++
 "= \(L) \(O\) \(R) \(E)\n"];
```

Instructions

Edit the provided `mzn` model files to solve the problems described above. Your implementations can be tested locally by using the **Run + Check** icon in the MiniZinc IDE. Once you are confident that you have solved the problem, submit your solution for grading.

Technical Requirements To complete this assignment you will a new version of the MiniZinc Bundle (`http://www.minizinc.org`). Your submissions to the Coursera grader are currently checked using MiniZinc version 2.6.1.

Handin This assignment contains 4 solution submissions and 0 model submissions. For solution submissions, we will retrieve the best/last solution the solver has found using your model and check its correctness and quality. For model submissions, we will retrieve your model file (.mzn) and run it on some hidden data to perform further tests.

From the MiniZinc IDE, the **Submit to Coursera** icon can be used to submit assignment for grading. Follow the instructions to apply your MiniZinc model(s) on the various assignment parts. You can submit multiple times and your grade will be the best of all submissions.[1]. You can track the status of your submission on the **programming assignments** section of the course website.

[1]Please limit your number of submissions and test on your local machine first to avoid congestion on the Coursera servers