Universität Augsburg, LSt. Softwaretechnik, A. Schiendorfer, O. Kosak

Selbst-organisierende, adaptive Systeme (WS 16/17)

Übungsblatt 12 (Bearbeitung bis: 01.02.2017 23:59)

Diskrete Optimierung

Aufgabe 1 *(MiniZinc – Einfache Puzzles)*

Diese Aufgaben machen Sie mit der Modellierung in MiniZinc vertraut.

a) Abbot's Puzzle: Wenn 100 Scheffel Mais auf 100 Leute so verteilt wurden, dass jeder Mann 3 Scheffel, jede Frau 2 Scheffel und jedes Kind ein halbes Scheffel erhielt, wie viele Männer, Frauen und Kinder gab es, wenn wir wissen, dass es 5 Mal mehr Frauen als Männer gab? Speichern Sie Ihr Ergebnis als `abbot.mzn`.

b) Tick, Trick und Track haben etwas verbrochen und werden von ihrem Onkel befragt. Sie geben folgende Statements ab:

- Tick: Trick und Track haben gleich viel Schuld. Wenn einer schuldig ist, dann auch der andere.
- Trick: Wenn Tick schuldig ist, dann auch ich.
- Track: Trick und ich sind nicht beide schuldig.

Donald weiß, dass die drei nicht lügen. Hat er genügend Information, um die Schuldigen eindeutig zu bestimmen? Speichern Sie Ihr Ergebnis als `lying.mzn`.

c) Ritter und Schurken: Auf der Insel der Ritter und Schurken sagen Ritter stets die Wahrheit und Schurken stets die Unwahrheit. Jeder Insulaner ist entweder Ritter oder Schurke. Ein Besucher trifft drei Bewohner, die die folgenden Statements abgeben:

A: Genau einer von uns ist ein Schurke.

B: Genau zwei von uns sind Schurken.

C: Wir sind alle Schurken.

Welcher Bewohner gehört zu welchem Typ? Speichern Sie Ihr Modell als `knk1.mzn`.
Danach trifft der Besucher drei andere Bewohner mit den folgenden Statements:

A: Genau einer von uns ist ein Ritter.

B: Genau zwei von uns sind Ritter.

C: Wir sind alle Ritter.

Welche Information erhält der Besucher dadurch? Speichern Sie Ihr Modell als `knk2.mzn`.

Aufgabe 2 *(MiniZinc: Übungen)*

a) Build a MiniZinc model `xopt.mzn` with a decision variable x taking values from 0 to 10, with constraints to ensures that x is divisible by 4, which outputs the value of x that gives the minimum value of $(x-7)^2$.

Suppose you cannot use the `mod` function, how would you alternatively model that x is divisible by 4?

b) Define a MiniZinc model `array.mzn` which takes an integer parameter n defining the length of an array of numbers x taking values from 0 to 9. Constrain the array so the sum of the numbers in the array is equal to the product of the numbers in the array. Output the resulting array. Test your model using the "all solutions" setting active in the IDE. Add a constraint to ensure that the numbers in the array are non-decreasing, i.e. $x[1] \leq x[2] \leq \ldots \leq x[n]$. This should reduce the number of similar solutions. How big a number can you solve with your model? Why do you think this happens?

c) Given a group of n people, we must arrange them for a photo. The best photo is when people are next to their friends, so the aim is to arrange them so that each person is next to (to the left or right) with as many friends as possible. The data for the problem is given as

```
n = <size of problem> ;
array[1..n,1..n] of var bool: friend;
```

where `friend[f1, f2]` means `f1` and `f2` are friends. You can assume that the friend array is symmetric. You should output a list of the people in their position to maximize the number of adjacent friends. For example given the data `groupphoto1.dzn`, you should output the placement of the guests as well as the objective value, i.e.,

```
Obj = 7; [4, 3, 5, 6, 8, 7, 1, 2]
```