CSCI5240 - Combinatorial Search and Optimization with Constraints Fall 2015

Teacher: Prof. Jimmy Lee

Assignment 2

Due: 9 November, 2015 (Monday)

1. (a) Give an example of a CSP that is node-consistent but not arc-consistent.
   (b) Give an example of a CSP that is arc-consistent but not node-consistent.
   (c) Give an example of a CSP that is satisfiable but not node-consistent.
   (d) Give an example of a CSP that is unsatisfiable but node-consistent.
   (e) Give an example of a CSP that is satisfiable but not arc-consistent.
   (f) Give an example of a CSP that is unsatisfiable but arc-consistent.
   (g) Give an example of a CSP that is satisfiable but not node-consistent and not arc-consistent.
   (h) Give an example of a CSP that is unsatisfiable but node-consistent and arc-consistent.

2. *The n-Queens Problem and the Peaceably Co-existing Armies of Queens*
   Recall the n-Queens Problem: place n queens on an $n \times n$ chessboard so that no two queens attack each other.

   (a) Now, we rotate the board clockwisely by 45 degrees, and uses one variable to denote the queen position (if any) for each row (a diagonal in the original board). Thus the variables are $y_{1-n}, y_{2-n}, \ldots, y_0, \ldots, y_{n-2}, y_{n-1}$. Figure 1(a) gives the scenario for the 4-Queens problem. While each variable takes values according to the new coordinate system, we use the value $-n$ to denote that there are no queens on a particular row. The domain of each variable y_i is thus $D_i = \{|i|+1, |i|+3, \ldots, 2n-|i|-1\} \cup \{-n\}$ for $i \in [1-n, n-1]$. The set of constraints are given as follows:
      - for $i \in [1-n, n-1] : \sum_i (y_i \neq -n) = n$
      - for $i \in [1-n, n-1]$ and $j \in [1, 2n-1] : \sum_i (y_i = j) \leq 1$
      - for $i, j \in [1-n, n-1]$ and $i \neq j : |y_i - y_j| \neq |i-j|$

      Implement the first two constraints with counting constraints, the last constraint using the three possible ways provided by Gecode, i.e. arithmetic constraint, tuple sets and DFA. You should **not** hardcode the tuple sets and the DFAs. Give the DFA graph for each DFA and find *all* the possible solutions for the n-Queens Problem with $n = 5$. Compare the runtime and numbers of fails of the three methods.

   (b) One variant of *the n-Queens Problem* is *the Peaceably Co-existing Armies of Queens*. We are still given an $n \times n$ chessboard while the queens to be positioned are of two colours. *i.e.*, *black* and *white*. Queens of the same colour form an army and they do not attack one another, regardless of their positions on the chessboard. Now we are required to place *as many queens as possible*, such that

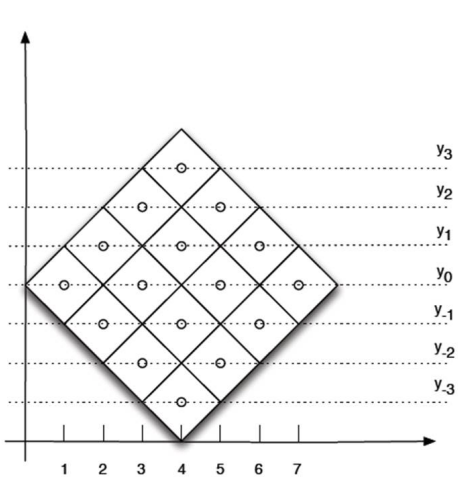
      - two armies are of the same size, and
      - no queen from one army can be attacked by any queen from the other army.

      Thus two armies can co-exist. Figure 1(b) shows two peaceably co-existing armies of queens in a 7×7 chessboard, each consisting of 5 queens. Yet it is not optimal.

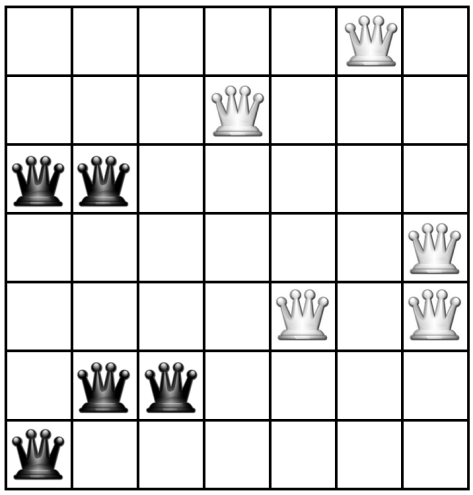
      Give a CSP model of *the Peaceably Co-existing Armies of Queens*. Implement your model in Gecode and find one solution for the case $n = 7$.

3. *The Knight's Tour*

   A Knight's tour is a sequence of moves where a Knight visits every cell of an $n \times n$ chessboard exactly once in n^2 moves. If the last cell that the Knight visits is reachable from the starting cell by one Knight's move, such a tour is called a *closed* one. Otherwise,



(a) A new coordinate system for the 4-Queens problem



(b) Two peaceably co-existing armies of queens in a 7×7 chessboard

Figure 1: The n-Queens Problem and the Peaceably Co-existing Armies of Queens

it is an *open* one. Given a particular cell on an $n \times n$ chessboard, we aim at finding a closed Knight's tour that starts from the cell.

A possible CSP model of the problem is as follows. We label each cell using numbers from 1 to n^2, and we have n^2 variables x_0, x_1, ... x_{n^2-1} with the identical domain $[1..n^2]$, where x_i indicates the number of the cell the Knight visits in the i-th move. Obviously, all variables should take different values, and the values of two consecutive variables should form a legal Knight's move. To model that, we define a new constraint $move(x, y, n)$, which returns true if x and y forms a legal Knight's move in an $n \times n$ chessboard and false otherwise.

Figure 2 shows a partially completed Knight's tour in a 6×6 chessboard starting from the top left corner, where the number i appearing in a cell means the Knight will reach the cell at the i-th move.

Design and implement the *move* constraint as a user-defined constraint by implementing a propagator that ensures the *move* constraint is *arc consistent* after propagation. Implement the model using *only* the *all-different* constraint and the *move* constraint. Find all possible Knight's tours that coincide with the configuration in Figure 2.

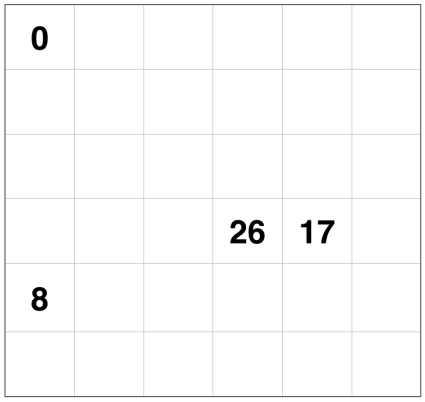


Figure 2: An instance of *the Knight's Tour Problem*

4. All non-binary CSPs can be transformed into binary CSPs with finite domains. Given the

following CSP (Z, D, C): $Z = \{x, y, z\}, D_x = D_y = D_z = \{1, 2, 3, 4\}$, and $C = \{c_{x,y,z}\}$, where $c_{x,y,z}$ is "$x + y = z$". We list the tuples satisfying $c_{x,y,x}$ as follows:

x	$+$	y	$=$	z
1		1		2
1		2		3
1		3		4
2		1		3
2		2		4
3		1		4

Show how you can transform (Z, D, C) into an "equivalent" binary CSP in such a way that there is a one-one correspondence between solution tuples of the new and the old CSPs. Compare the size of your new and old CSPs.

Repeat the same transformation for the following CSP (Z, D, C): $Z = \{u, v, x, y\}$, $D_u = D_v = D_x = D_y = \{1, ..., 100\}$, and $C = \{c_{u,v,x,y}\}$, where $C = \{c_{u,v,x,y}\}$ is "$u+v = x+y$". What conclusion, if any, can you draw?