Constraint Satisfaction Problems

(Source: https://aimacode.github.io/aima-exercises/, accessed in Nov 2022)

Exercise 01

Solution (textbook manual)

1. **Rectilinear floor-planning**: find non-overlapping places in a large rectangle for several smaller rectangles.

   Variables: each of the small rectangles, with the value of each variable being a 4-tuple consisting of the x and y coordinates of the upper left and lower right corners of the place where the rectangle will be located.

   The domain of each variable is the set of 4-tuples that are the right size for the corresponding small rectangle and that fit within the large rectangle.

   Constraints say that no two rectangles can overlap; for example, if the value of variable R1 is [0, 0, 5, 8], then no other variable can take on a value that overlaps with the 0, 0 to 5, 8 rectangle.

2. **Class scheduling**: There is a fixed number of professors and classrooms, a list of classes to be offered, and a list of possible time slots for classes. Each professor has a set of classes that he or she can teach.

   Variables; three for each class, one with times for values (e.g. MWF8:00, TuTh8:00, MWF9:00, ...), one with classrooms for values (e.g. Wheeler110, Evans330, ...) and one with instructors for values (e.g. Abelson, Bibel, Canny, ...).

   Constraints say that only one class can be in the same classroom at the same time, and an instructor can only teach one class at a time. There may be other constraints as well (e.g. an instructor should not have two consecutive classes).

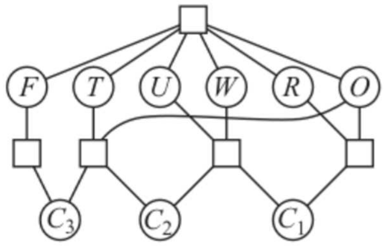
3. **Hamiltonian tour**: given a network of cities connected by roads, choose an order to visit all cities in a country without repeating any.

   Variable: one for each stop on the tour, with binary constraints requiring neighboring cities to be connected by roads, and an *AllDiff* constraint that all variables have a different value

Exercise 02

Solve the cryptarithmetic problem in the Figure by hand, using the strategy of backtracking with **forward checking** and the **MRV** and least-constraining-value heuristics.

$$\begin{array}{r} T\ W\ O \\ +\ T\ W\ O \\ \hline F\ O\ U\ R \end{array}$$



The exact steps depend on each on choices:

a) Choose the C3 variable. Its domain is {0, 1}.

b) Choose the value 1 for C3. We can't choose 0, because it would force F to be 0 (forward checking), and the leading digit of the sum must be non-zero.)

c) Choose F, because it has only one remaining value.

d) Choose the value 1 for F.

e) Now C2 and C1 are tied for minimum remaining values at 2; let's choose C2.

f) Either value survives forward checking, let's choose 0 for C2.

g) Now C1 has the minimum remaining values.

h) Again, arbitrarily choose 0 for the value of C1.

i) The variable O must be an even number, because it is the sum of T + T less than 5 (because O +O = R+ 10 × 0). That makes it most constrained.

j) Arbitrarily choose 4 as the value of O.

k) R now has only 1 remaining value.

l) Choose the value 8 for R.

m) T now has only 1 remaining value.

n) Choose the value 7 for T.

o) U must be an even number less than 9; choose U.

p) The only value for U that survives forward checking is 6.

q) The only variable left is W.

r) The only value left for W is 3.

s) This is a solution.

This is a rather easy (under-constrained) puzzle, so it is not surprising that we arrive at a solution with no backtracking (given that we are allowed to use forward checking).

Exercise 03

Consider the graph with 8 nodes $A_1, A_2, A_3, A_4, H, T, F_1, F_2$. A_i is connected to A_{i+1} for all i, each A_i is connected to H, H is connected to T, and T is connected to each F_i. Find a 3-coloring of this graph by hand using the following strategy: backtracking with conflict-directed back jumping, the variable order $A_1, H, A_4, F_1, A_2, F_2, A_3, T$, and the value order R, G, B.

Solution

a) *A1 = R.*

b) *H = R conflicts with A1.*

c) *H = G.*

d) *A4 = R.*

e) *F1 = R.*

f) *A2 = R conflicts with A1, A2 = G conflicts with H, so A2 = B.*

g) *F2 = R.*

h) *A3 = R conflicts with A4, A3 = G conflicts with H, A3 = B conflicts with A2, so backtrack. Conflict set is {A2,H, A4}, so jump to A2. Add {H, A4} to A2's conflict set.*

i) *A2 has no more values, so backtrack.*
Conflict set is {A1,H, A4} so jump back to A4. Add {A1,H} to A4's conflict set.

j) *A4 = G conflicts with H, so A4 = B.*

k) *F1 = R*

l) *A2 = R conflicts with A1, A2 = G conflicts with H, so A2 = B.*

m) *F2 = R*

n) *A3 = R.*

o) *T = R conflicts with F1 and F2, T = G conflicts with H, so T = B.*

p) *Success.*

Exercise 04

Consider the problem of completely tiling a surface with n dominoes (2×1 rectangles). The surface is an arbitrary edge-connected, i.e., adjacent along an edge, collection of $2n$ 1×1 squares (e.g., a checkerboard, a checkerboard with some squares missing, a 10×1 row of squares, etc.).

1. Formulate this problem precisely as a CSP where the dominoes are the variables.
2. Formulate this problem precisely as a CSP where the squares are the variables, keeping the state space as small as possible. (Hint: does it matter which domino goes on a given pair of squares?)

Solution

1. **Variable** domains: all pairs of adjacent squares. You might want to avoid having the same pair of squares appearing twice in different orders.
   Constraints: every pair of variables is connected by a constraint stating that their values may not overlap (i.e., they cannot share any square).
2. **Variable** domains: the set of (up to four) adjacent squares. The idea is that the domino covering this square can choose exactly one of the adjacent squares to cover too.
   Constraints: between every pair of adjacent squares A and B. A can have value B iff B has value A.