COMP3620/COMP6320 Artificial Intelligence
Tutorial 4: Constraint Satisfaction

Answers

April 24–27, 2018

Exercise 1

Consider the problem of scheduling five tasks

$$t_1,\ t_2,\ t_3,\ t_4,\ t_5$$

each of which takes exactly one hour to complete. The tasks may start at 1 pm, 2 pm or 3 pm. Tasks can be executed simultaneously, subject to the restrictions that:

1. t_1 must start after t_3;
2. t_3 must start before t_4 and after t_5;
3. t_2 cannot be executed at the same time as either t_1 or t_4;
4. t_4 cannot start at 2 pm.

Formulate this problem as a constraint network $\gamma = (V, D, C)$, defining variables, domains and constraints following the notation used in the lectures. Constraints can be defined either explicitly—specifying the allowed pairs of values—or in some more compact notation that summarises the allowed pairs of values.

Solution

The constraint network $\gamma = (V, D, C)$ is defined as follows:

V: the set $\{start(t_i) \mid i = 1, \ldots, 5\}$

D: the set $\{1, 2, 3\}$ for all $v \in V$

C:
- $C_{start(t_1),start(t_3)} \equiv start(t_1) > start(t_3)$,
- $C_{start(t_3),start(t_4)} \equiv start(t_3) < start(t_4)$,
- $C_{start(t_3),start(t_5)} \equiv start(t_3) > start(t_5)$,
- $C_{start(t_2),start(t_1)} \equiv start(t_2) \neq start(t_1)$,
- $C_{start(t_2),start(t_4)} \equiv start(t_2) \neq start(t_4)$,
- $C_{start(t_4)} \equiv start(t_4) \neq 2$.

Exercise 2

Consider the following constraint network $\gamma = (V, D, C)$:

- Variables: $V = \{a, b, c, d\}$.
- Domains: For all $v \in V$: $D_v = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$.
- Constraints: $|a - b^2| \leq 1$; $b \leq c - 7$; $c < d - 1$.

Run the AC-3(γ) algorithm, as specified in the lecture. Precisely, for each iteration of the while-loop, give the content of M at the start of the iteration, give the pair (u, v) removed from M, give the domain of u after the call to Revise(γ, u, v), and give the pairs (w, u) added into M.

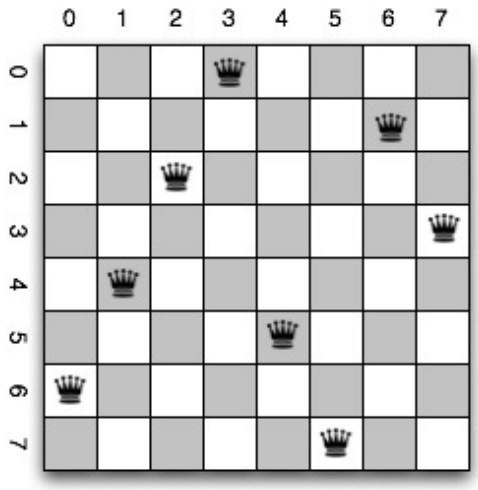
Note: Initialize M as a lexicographically ordered list (i.e., (a,b) would be before (a,c), both before (b,a) etc., if any of those exist). Furthermore, use M as a FIFO queue, i.e., always remove the element at the front and add new elements at the back.

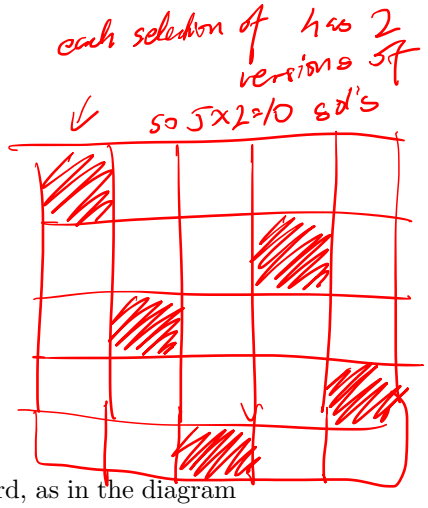
Solution

Here is the required information for the iterations of the while-loop as executed. In this case, we always removed the element at the front first, and added new elements to the end.

1. M content: $\{(a, b), (b, a), (b, c), (c, b), (c, d), (d, c)\}$; pair selected: (a, b); $D_a = \{1, 2, 3, 4, 5, 8, 9, 10\}$. No candidates to be inserted into M.
2. M content: $\{(b, a), (b, c), (c, b), (c, d), (d, c)\}$; pair selected: (b, a); $D_b = \{1, 2, 3\}$. Candidate (c, b) already present in M.
3. M content: $\{(b, c), (c, b), (c, d), (d, c)\}$; pair selected: (b, c); $D_b = \{1, 2, 3\}$. No change, so nothing added into M.
4. M content: $\{(c, b), (c, d), (d, c)\}$; pair selected: (c, b); $D_c = \{8, 9, 10\}$. Candidate (d, c) already present in M.
5. M content: $\{(c, d), (d, c)\}$; pair selected: (c, d); $D_c = \{8\}$. Candidate (b, c) added into M.
6. M content: $\{(d, c), (b, c)\}$; pair selected: (d, c); $D_d = \{10\}$. No candidates to be inserted into M.
7. M content: $\{(b, c)\}$; pair selected: (b, c); $D_b = \{1\}$. Candidate (a, b) inserted into M.
8. M content: $\{(a, b)\}$; pair selected: (a, b); $D_a = \{1, 2\}$. No candidates to be inserted into M.
9. M empty; return modified γ with reduced domains.

Exercise 3





The Queens Problem is usually stated in terms of a standard 8×8 chessboard, as in the diagram above, probably because a board of that size looks familiar. However, it can be formulated similarly for boards of other sizes, including smaller ones.

How many solutions are there to the 5 Queens Problem (i.e. find 5 squares of a 5×5 chessboard on which queens could be placed without any queen attacking any other)? You should run backtracking search using forward checking to answer this question. Can you use any symmetries of the problem to make the enumeration of solutions easier?

If time permits, you may look similarly at the 6 Queens Problem.

Solution

Start putting queens on the board. It is quite a good idea to put a queen on the first column, or in the first row if you prefer, and proceed with the others, using the Minimum Remaining Values heuristic to select variables. Not much backtracking happens.

Forward checking simply makes unavailable the squares directly attacked by one of the queens which has been placed. This can be done with a pen on paper quite easily.

Each choice of square for a queen in the first column can be extended to exactly two solutions, so there are 10 solutions altogether.

Given that the board has the symmetries of the square, we could for instance reflect it about the middle row. Then the two solutions we find with the first queen (i.e. the queen in the leftmost column) on the bottom two rows mean there are mirror-image solutions with the queen on the top two rows. When she is on the middle row, the possible positions of the queen in the next column are symmetric in the same way: once we find one we know about the other. So this symmetry halves the work.

The 6 Queens problem is more tightly constrained than the 5 Queens, counter-intuitive as that may seem. There is no solution in which there is a queen on either of the long diagonals, and only 4 solutions altogether. All four solutions are isomorphic in the sense that they can be turned into each other by applying symmetries of the square such as rotation through 90 degrees or reflecting about the centre line.