V= variables
D= domains
C= constraints

COMP3620/COMP6320 Artificial Intelligence
Tutorial 4: Constraint Satisfaction

April 24–27, 2018

C: $C_{start(t_1),start(t_3)} \equiv start(t_1) > start(t_3)$

$C_{start(t_3),start(t_4),start(t_5)} \equiv (start(t_3) < start(t_5)) \wedge (start(t_5) < start(t_4))$

$C_{start(t_2),\dots,start(t_4)} \equiv (start(t_2) \neq start(t_5)) \vee (start(t_2) \neq start(t_4))$

$C_{start(t_2)} \equiv start(t_2) \neq 2$

拆分成两个.

$\{start(t_i) \mid i = 1,2,3,4,5\}$

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.

$V = \{start(t_1), start(t_2), \dots, start(t_5)\}$ or $s_1, s_2, \dots, s_5$ don't treat t_i as a variable

$D = \{1, 2, 3\}\ \forall\ v \in V$

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.

Exercise 2

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

- 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: $|a - b^2| \leq 1$; $b \leq c - 7$; $c < d - 1$.

① dequeue
② reduce the domain
③ add back to queue if domain changed.

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.

$M = \{(a,b), (b,a), (b,c), (c,b), (c,d), (d,c)\}$ dequeue

Goal: reduce the domains of variables

$D_a = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
$D_b = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
$D_c = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$
$D_d = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

(a,b) ↑ reduce domain

we know $|a - b^2| \leq 1$
So if $b=1$, $|a-1| \leq 1 \Rightarrow a = 1, 2$
$b=2$, $|a-4| \leq 1 \Rightarrow a = 3, 4, 5$
For each of b, there are some values a can take.
$b=3$, $|a-9| \leq 1 \Rightarrow a = 8, 9, 10$

we did changed the domain, add back $(b,a), (b,b), (b,c), (b,d)$ back to M
(already there) (nothing) (already there) (no relation)
So we add NOTHING

dequeue (b,a)
reduce domain $D_b = \{1,2,3\}$ → add (c,b), (d,b)
already in there
don't do reverse
not related
ADD NOTHING AGAIN

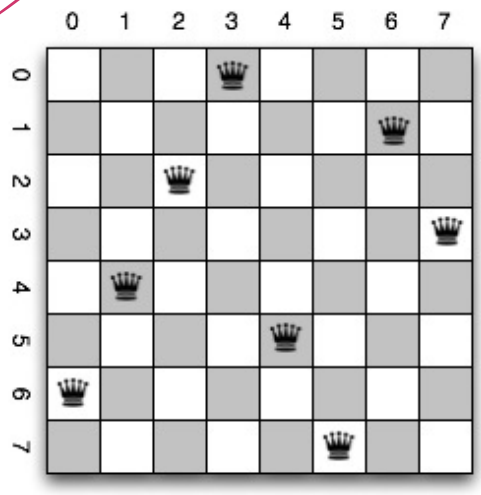
dequeue (b,c) reduce domain
$b \le c-7 \Rightarrow b+7 \le c$
1 8
2 9
3 10
→ add (a,c), (b,c), (d,c)
ADD NOTHING BACK

add (b,c) only → (b,c)
c = 8
b ≤ 8-7
b = 1

(c,d)
c < d+1
9
8
9
10
d = {b}

a = 1,2
So $D_a = \{1,2\}$
$D_b = \{1\}$
$D_c = \{8\}$
$D_d = \{b\}$

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.