National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 7: Constraint Satisfaction Problems (Solutions)

Issued: June 9, 2020 Discussion in: Week 5, Tuesday Session

1. Consider the following constraint satisfaction problem:

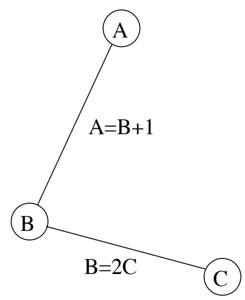
   Variables:
   $$A, B, C$$

   Domains:
   $$D_A = D_B = D_C = \{0, 1, 2, 3, 4\}$$

   Constraints:
   $$\begin{aligned} A &= B + 1 \\ B &= 2C \end{aligned}$$

   Construct a constraint graph for this problem. Show a trace of the `AC-3` algorithm on this problem. Assume that initially, the arcs in queue are in the order $\{(A, B), (B, A), (B, C), (C, B)\}$.

Solution: Constraint Graph:



Original domains:

$$D_A = D_B = D_C = \{0, 1, 2, 3, 4\}$$

Content of queue and domain of variables at the end of each iteration:

Revised Domain	Queue
	(A,B) (B,A) (B,C) (C,B)
$D_A = \{1, 2, 3, 4\}$	(B,A) (B,C) (C,B)
$D_B = \{0, 1, 2, 3\}$	(B,C) (C,B)
$D_B = \{0, 2\}$	(C,B) (A,B)
$D_C = \{0, 1\}$	(A,B)
$D_A = \{1, 3\}$	

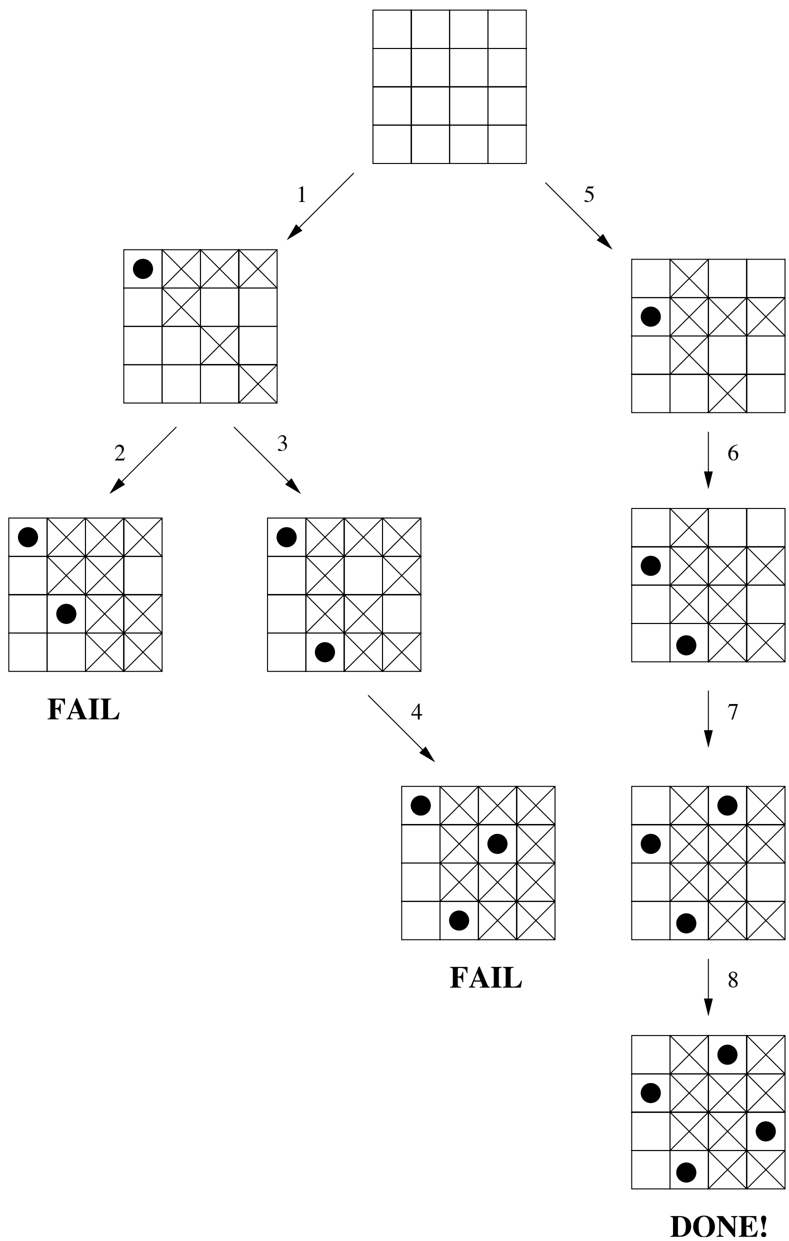
Allowable domain values:

$$\begin{aligned} D_A &= \{1, 3\} \\ D_B &= \{0, 2\} \\ D_C &= \{0, 1\} \end{aligned}$$

Note that of course not all of these values are compatible: we need to choose one of them and then the rest will be implied (e.g. $A = 1 \rightarrow B = 0 \rightarrow C = 0$).

2. Consider the 4-queens problem on a 4×4 chess board. Suppose the leftmost column is column 1, and the topmost row is row 1. Let Q_i denote the row number of the queen in column i, $i = 1, 2, 3, 4$. Assume that variables are assigned in the order Q_1, Q_2, Q_3, Q_4, and the domain values of Q_i are tried in the order $1, 2, 3, 4$. Show a trace of the backtracking algorithm with forward checking to solve the 4-queens problem.

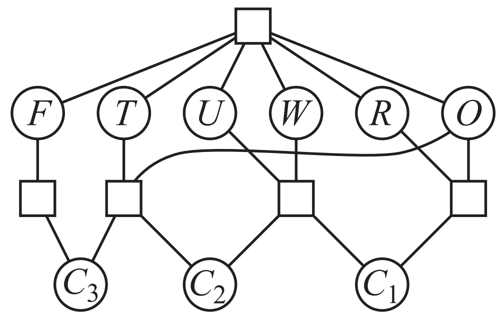
   Solution: The following is the trace of the search tree:



3. Show a trace of the backtracking algorithm with forward checking to solve the cryptarithmetic problem shown in Figure 1. Use the most constrained variable heuristic, and assume that the domain values (digits) are tried in ascending order (i.e., 0, 1, 2, $\cdots$).

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

(a)



(b)

Figure 1: Cryptarithmetic puzzle.

Solution: The following is the trace:

Assignments									Remarks
C_1=0	C_2=0	C_3=0							C_1, C_2, C_3 are the most constrained variables. C_3=0 forces F=0, which is not possible.
		C_3=1	F=1	T=2					T is the most constrained variable, with domain $\{2,\ldots,9\}$, whereas the remaining variables have domain $\{0,2,\ldots,9\}$. T=2 results in no satisfying value for O.
				T=3					T=3 results in no satisfying value for O.
				T=4					T=4 results in no satisfying value for O.
				T=5	O=0				T=5 uniquely determines the value for O. Fail since R has no satisfying value.
				T=6	O=2	R=4	W=0		T=6 uniquely determines the value for O and R. Fail since there is no satisfying value for U.
							W=3		Fail since there is no satisfying value for U.
							W=5		Fail since there is no satisfying value for U.
							W=7		Fail since there is no satisfying value for U.
							W=8		Fail since there is no satisfying value for U.
							W=9		Fail since there is no satisfying value for U.
				T=7	O=4	R=8	W=0		T=7 uniquely determines the value for O and R. Fail since there is no satisfying value for U.
							W=2		Fail since there is no satisfying value for U.
							W=3	U= 6	Succeeds!

4. Consider the *item allocation problem*. We have a group of people $N = \{1, \dots, n\}$, and a group of items $G = \{g_1, \dots, g_m\}$. Each person $i \in N$ has a utility function $u_i : G \to \mathbb{R}_+$. The constraint is that every person is assigned *at most one item*, and each item is assigned to *at most one person*. An allocation simply says which person gets which item (if any).

   In what follows, you *must* use *only* the binary variables $x_{i,j} \in \{0, 1\}$, where $x_{i,j} = 1$ if person i receives the good g_j, and is 0 otherwise.

   (a) Write out the constraints: 'each person receives no more than item' and 'each item goes to at most one person', using only the $x_{i,j}$ variables[1].

   (b) Suppose that people are divided into *disjoint types* $N_1, \dots, N_k$ (think of, say, genders or ethnicities), and items are divided into *disjoint blocks* $G_1, \dots, G_\ell$. We further require that each N_p only be allowed to take no more than λ_{pq} items from block G_q. Write out this constraint using the $x_{i,j}$ variables. (Note that each N_i corresponds to the set of people who are of that person type.)

   (c) We say that player i *envies* player i' if the utility that player i has from their assigned item is strictly lower than the utility that player i has from the item assigned to player i'. Write out the constraints that ensure that in the allocation, no player envies any other player. You may assume that the validity constraints from (a) hold.

Solution:

(a)

$$\forall i \in N : \sum_{g_j \in G} x_{i,j} \leq 1$$
$$\forall g_j \in G : \sum_{i \in N} x_{i,j} \leq 1$$

(b)

$$\forall p \in [k], q \in [\ell] : \sum_{i \in N_p} \sum_{g_j \in G_q} x_{i,j} \leq \lambda_{pq}$$

[1] You may use simple algebraic functions $-, +, \times, \div$, and numbers

(c) Note that for this constraint the definition requires that the allocation is valid, so you need to add the constraints from (a) to make either definition below meaningful.

$$\forall i, i' \in N, \forall g_j, g_{j'} \in G : (x_{i,j} \wedge x_{i',j'}) \implies u_i(g_j) \geq u_i(g_{j'})$$

OR

$$\forall i, i' \in N : \left(\left(\sum_{g_j \in G} x_{i,j} u_i(g_j) \right) \geq \left(\sum_{g_j \in G} x_{i',j} u_i(g_j) \right) \right) \wedge \left(\left(\sum_{g_j \in G} x_{i,j} u_i(g_j) \right) > 0 \right)$$