

# CS 188: Assignment 2: Part B Sample Solutions

CS 188 Staff

## 1 Walking through a CSP

a) See figure 1 for the constraint graph. Constraints take the explicit form:

- $(A, B) \in \{(1, 2), (1, 3), (2, 3), (2, 1), (3, 1), (3, 2)\}$
- $(A, C) \in \{(1, 2), (1, 3), (2, 3), (2, 1), (3, 1), (3, 2)\}$
- $(B, C) \in \{(2, 1), (3, 1), (3, 2)\}$
- $(B, D) \in \{(1, 2), (1, 3), (2, 3)\}$

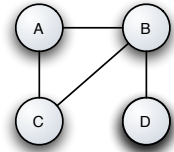


Figure 1: The constraint graph for question (1a).

b)

Step	A	B	C	D
0	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}
1	1	{2, 3}	{2, 3}	{1, 2, 3}
2	1	2	{}	{3}
3	1	3	{2}	{}
4	2	{1, 3}	{1, 3}	{1, 2, 3}
5	2	1	{}	{2, 3}
6	2	3	{1}	{}
7	3	{1, 2}	{1, 2}	{1, 2, 3}
8	3	1	{}	{2, 3}
9	3	2	{1}	{3}
10	3	2	1	{3}
11	3	2	1	3

c)

Step	A	B	C	D
0	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}
1	{2, 3}	1	{}	{2, 3}
2	{1, 3}	2	{1}	{3}
3	{3}	2	1	{3}
4	3	2	1	{3}
5	3	2	1	3

As expected, this heuristically guided approach is much faster. In particular, notice that in (b) forward-checking was unable to detect poor values of  $A$  which caused significant backtracking. With the degree heuristic guiding the search in (c), these poor values for  $A$  were never attempted.

d) Conditioning on  $C = 1$  and performing DFS backtracking on the remainder of the problem:

Step	A	B	C	D
0	{2, 3}	{2, 3}	1	{1, 2, 3}
1	2	{3}	1	{1, 2, 3}
1	2	3	1	{}
2	3	{2}	1	{3}
3	3	2	1	{3}
4	3	2	1	3

Conditioning on  $C = 2$  and  $C = 3$  would yield CSPs with no solutions.

The cutset  $C$  leaves the remainder of the problem as a tree-structured graph, which can in general be solved in  $O(n \cdot d^2)$  time.  $A$  would have the same effect, while  $B$  would leave us with an even easier problem: two independent sub-problems that each have at worst a chain structure. Using  $D$  as a cutset does not yield such beneficial properties and would make a poor cutset. Choosing more than one variable in the cutset would lead to a larger branching factor (9) for the conditioning, which would hurt efficiency.

e) First, we perform arc consistency on the initial state through the following steps:

Step	A	B	C	D
0	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}	{1, 2, 3}
1	{1, 2, 3}	{1, 2, 3}	{1, 2}	{1, 2, 3}
2	{1, 2, 3}	{1, 2, 3}	{1, 2}	{2, 3}
3	{1, 2, 3}	{2, 3}	{1, 2}	{2, 3}
4	{1, 2, 3}	{2}	{1, 2}	{2, 3}
5	{1, 2, 3}	{2}	{1}	{2, 3}
6	{1, 2, 3}	{2}	{1}	{3}
7	{2, 3}	{2}	{1}	{3}
8	{3}	{2}	{1}	{3}

Now, the domain of each variable is reduced to a singleton, so we can use backtracking search trivially to solve the CSP.

Step	A	B	C	D
0	{3}	{2}	{1}	{3}
1	{3}	2	{1}	{3}
2	3	2	{1}	{3}
3	3	2	1	{3}
4	3	2	1	3

## 2 Converting Constraints

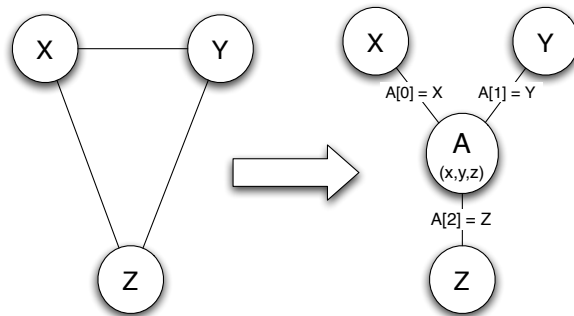


Figure 2: Removing ternary constraints from the constraint graph

a) Consider a ternary constraint,  $T$ , between variables  $X, Y$ , and  $Z$  (e.g.  $X + Y = Z$ ) that specifies a legal set of assignments to  $X, Y$ , and  $Z$ . Introduce a new auxiliary variable  $A$  whose domain is the assignments to  $X, Y$ , and  $Z$  allowed by  $T$ . More formally,  $D(A) = \{(x, y, z) : (x, y, z) \in T\}$ . Introduce a new binary constraint between  $A$  and  $X$  specifying that  $A[0]$ <sup>1</sup> is the same value as  $X$ . Do the same for  $Y$  and  $Z$ . See Figure 2 for an illustration.

b) The procedure above is easily generalized to an  $n$ -ary constraint as follows. Suppose we have an  $n$ -ary constraint among  $X_1, X_2, \dots, X_n$ . Create an auxiliary variable  $A$  whose domain is the set of assignment-tuples for  $(X_1, \dots, X_n)$  which are allowed by the  $n$ -ary constraint. Introduce a binary constraint between  $A$  and each  $X_i$  asserting that  $A[i] = X_i$ .

c) Transforming an  $n$ -ary constraint to a set of binary ones involves the introduction of a new variable whose domain is  $O(d^n)$ , assuming the size of each variable's domain is of size  $d$ . We also introduce  $n$  new binary constraints.

## 3 Minesweeper

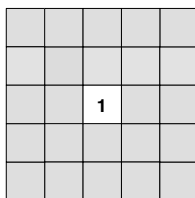


Figure 3:

---

<sup>1</sup>The first element of the tuple

a) CSP formulation:

**Variables:**  $X_{i,j}$  for each unrevealed board cell adjacent to a revealed cell

**Domain:**  $\{0, 1\}$ , where 0 corresponds to the cell being clear and 1 to a mine

**Constraints:** For each revealed position, let  $k$  be the number of adjacent cells with mines.<sup>2</sup> Then  $\sum_{k \in \{-1, 0, 1\}} \sum_{\ell \in \{-1, 0, 1\}} X_{i+k, j+\ell} = k$ .<sup>3</sup>

	1	
1	2	

Figure 4:

b) See figure 3 for an illustration. Any of the hidden squares surrounding the revealed one could be the one with the bomb, and there's no way to know which it could be.

c) See figure 4 for an illustration. The bottom right corner is guaranteed to be safe but we aren't sure of the location of the mine on either side.

d) Because we need to choose a square which is safe under all possible placements of the mine. So it isn't enough to pick a cell which is safe in a single satisfying solution, but in *all* of them. Therefore, we can't decide where to safely go based upon the single satisfying assignment found by a CSP solver.

e) We must alter the backtracking CSP solver to return all solutions to the problem. This can be accomplished by altering the algorithm to store a set of solutions as it goes and to backtrack when it finds a solution and add it to its set of solutions, instead of just quitting. Once we have all solutions, any square which is clear in all the solutions is safe to choose to expand next.

f) We can still use forward checking. When we backtrack upon finding a solution, there is only one unassigned variable and therefore, we didn't cross any values off from forward checking. We *can* use arc-consistency, but in general arc-consistency is unnecessary when the domains of our variables are all of size two(why?) and we are already using forward-checking.

---

<sup>2</sup>If a revealed cell doesn't have a number in it, we know it doesn't have any adjacent mines

<sup>3</sup>If some  $X_{i+k, j+\ell}$  is off the board, it is 0.

g) Suppose that in any given configuration that the values of the hidden squares are chosen uniformly among the possible solutions (note that the assignment of all the hidden squares is generated at once, not independently). Then for each cell there is a marginal probability that the cell is clear given by the fraction of possible solutions where the cell is clear. We chose the cell that has the highest marginal probability of being clear. This is the best choice we can make assuming that possible assignments are uniform.