CS 188
Fall 2023

Midterm Review CSPs Solutions

Q1. CSPs

In this question, you are trying to find a four-digit number satisfying the following conditions:

1. the number is odd,
2. the number only contains the digits 1, 2, 3, 4, and 5,
3. each digit (except the leftmost) is strictly larger than the digit to its left.

(a) CSPs

We will model this as a CSP where the variables are the four digits of our number, and the domains are the five digits we can choose from. The last variable only has 1, 3, and 5 in its domain since the number must be odd. The constraints are defined to reflect the third condition above. Thus before we start executing any algorithms, the domains are

1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

(i) Before assigning anything, enforce arc consistency. Write the values remaining in the domain of each variable after arc consistency is enforced.

1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

(ii) With the domains you wrote in the previous part, which variable will the MRV (Minimum Remaining Value) heuristic choose to assign a value to first? If there is a tie, choose the leftmost variable.

- ○ The first digit (leftmost)
- ○ The second digit
- ○ The third digit
- ● The fourth digit (rightmost)

(iii) Now suppose we assign to the leftmost digit first. Assuming we will continue filtering by enforcing arc consistency, which value will LCV (Least Constraining Value) choose to assign to the leftmost digit? **Break ties from large (5) to small (1).**

- ■ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5

(iv) Now suppose we are running min-conflicts to try to solve this CSP. If we start with the number 1332, what will our number be after one interation of min-conflicts? Break variable selection ties from left to right, and **break value selection ties from small (1) to large (5)**.

1232

(b) The following questions are completely unrelated to the above parts. Assume for these following questions, there are only binary constraints unless otherwise specified.

(i) [*true* or *false*] When enforcing arc consistency in a CSP, the set of values which remain when the algorithm terminates does not depend on the order in which arcs are processed from the queue.

(ii) [*true* or *false*] Once arc consistency is enforced as a pre-processing step, forward checking can be used during backtracking search to maintain arc consistency for all variables.

False. Forward checking makes the current variable arc-consistent, but doesn?t look ahead and make all the other variables arc-consistent.

(iii) In a general CSP with n variables, each taking d possible values, what is the worst case time complexity of enforcing arc consistency using the AC-3 method discussed in class?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ● $O(n^2d^3)$ ○ $O(d^n)$ ○ ∞

$O(n^2d^3)$. There are up to n^2 constraints. There are d^2 comparisons for enforcing arc consistency per each constraint, and each constraint can be inserted to the queue up to d times because each variable has at most d values to delete.

(iv) In a general CSP with n variables, each taking d possible values, what is the maximum number of times a backtracking search algorithm might have to backtrack (i.e. the number of the times it generates an assignment, partial or complete, that violates the constraints) before finding a solution or concluding that none exists?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ● $O(d^n)$ ○ ∞

$O(d^n)$. In general, the search might have to examine all possible assignments.

(v) What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics?

○ 0 ○ $O(1)$ ○ $O(nd^2)$ ○ $O(n^2d^3)$ ● $O(d^n)$ ○ ∞

$O(d^n)$. The MRV and LCV heuristics are often helpful to guide the search, but are not guaranteed to reduce backtracking in the worst case.

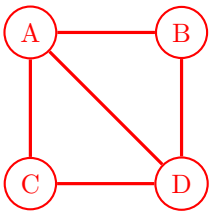
Q2. CSPs: Apartments

Four people, A, B, C, and D, are all looking to rent space in an apartment building. There are three floors in the building, 1, 2, and 3 (where 1 is the lowest floor and 3 is the highest). Each person must be assigned to some floor, but it's ok if more than one person is living on a floor. We have the following constraints on assignments:

- A and B must not live together on the same floor.
- If A and C live on *the same* floor, they must both be living on floor 2.
- If A and C live on *different* floors, one of them must be living on floor 3.
- D must not live on the same floor as anyone else.
- D must live on a higher floor than C.

We will formulate this as a CSP, where each person has a variable and the variable values are floors.

(a) Draw the edges for the constraint graph representing this problem. Use binary constraints only. You do not need to label the edges.



(b) Suppose we have assigned C = 2. Apply forward checking to the CSP, filling in the boxes next to the values for each variable that are eliminated:

A	■ 1	☐ 2	☐ 3
B	☐ 1	☐ 2	☐ 3
C		☐ 2	
D	■ 1	■ 2	☐ 3

(c) Starting from the original CSP with full domains (i.e. without assigning any variables or doing the forward checking in the previous part), enforce arc consistency for the entire CSP graph, filling in the boxes next to the values that are eliminated for each variable:

A	■ 1	☐ 2	☐ 3
B	☐ 1	☐ 2	☐ 3
C	☐ 1	☐ 2	■ 3
D	■ 1	☐ 2	☐ 3

(d) Suppose that we were running local search with the min-conflicts algorithm for this CSP, and currently have the following variable assignments.

A	3
B	1
C	2
D	3

Which variable would be reassigned, and which value would it be reassigned to? Assume that any ties are broken alphabetically for variables and in numerical order for values.

A is reassigned value 2.