CS 188
Fall 2018

Introduction to
Artificial Intelligence

Written HW 2

Due: Monday 9/10/2018 at 11:59pm (submit via Gradescope)

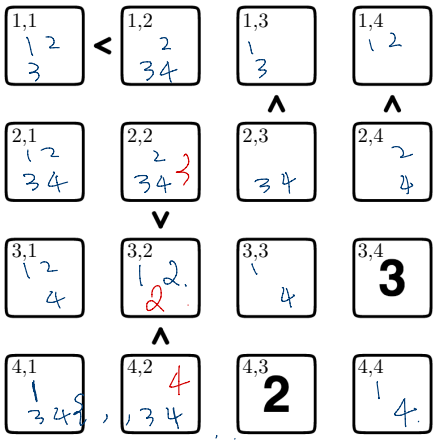
Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually

First name	Jing
Last name	Tan
SID	3034428726
Collaborators	Yitong Li

Q1. CSP Futoshiki

Futoshiki is a Japanese logic puzzle that is very simple, but can be quite challenging. You are given an n x n grid, and must place the numbers $1, \ldots n$ in the grid such that every row and column has exactly one of each. Additionally, the assignment must satisfy the inequalities placed between some adjacent squares.

To the right is an instance of this problem, for size $n = 4$. Some of the squares have known values, such that the puzzle has a unique solution. (The letters mean nothing to the puzzle, and will be used only as labels with which to refer to certain squares). Note also that inequalities apply only to the two adjacent squares, and do not directly constrain other squares in the row or column.



Let's formulate this puzzle as a CSP. We will use 4^2 variables, one for each cell, with X_{ij} as the variable for the cell in the ith row and jth column (each cell contains its i, j label in the top left corner). The only unary constraints will be those assigning the known initial values to their respective squares (e.g. $X_{34} = 3$).

(a) Complete the formulation of the CSP using only binary constraints (in addition to the unary constraints specificed above. In particular, describe the domains of the variables, and all binary constraints you think are necessary. You do not need to enumerate them all, just describe them using concise mathematical notation. You are not permitted to use n-ary constraints where $n \geq 3$.

For each variable X_{ij}, the domain is the label it ossibl has, as $\{1\ 2\ \ldots, n\}$.

Constraints: for all j,k, $X_{ij} \neq X_{ik}$ $(j \neq k)$.

for all j,k, $X_{ji} \neq X_{jk}$ $(j \neq k)$.

Missing the inequality constraints: $X_{11} < X_{12}$. $X_{13} < X_{23}$ $X_{14} < X_{24}$ $X_{22} < X_{32}$. $X_{33} < X_{43}$

(b) After enforcing unary constraints, consider the binary constraints involving X_{14} and X_{24}. Enforce arc consistency on just these constraints and state the resulting domains for the two variables.

After enforce arc-consistency on $X_{14} \to X_{24}$, domains for X_{14} becomes $\{1, 2\}$.

After enforce arc-consistenly on $X_{24} \to X_{14}$, domains for X_{24} becomes $\{2, 4\}$

correct

(c) Suppose we enforced unary constraints and ran arc consistency on this CSP, pruning the domains of all variables as much as possible. After this, what is the maximum possible domain size for any variable? [*Hint*: consider the least constrained variable(s); you should *not* have to run every step of arc consistency.]

Maximum Possible domain size is 4.

correct

(d) Suppose we enforced unary constraints and ran arc consistency on the initial CSP in the figure above. What is the maximum possible domain size for a variable adjacent to an inequality?

Maximum Possible domain size is 3.

correct

(e) By inspection of column 2, we find it is necessary that $X_{32} = 1$, despite not having found an assignment to any of the other cells in that column. Would running arc consistency find this requirement? Explain why or why not.

Reasons can be explained from the domains for X_{32}. X_{42}. X_{12} after enforcing constraints

No. Because the constraints in this algorithm are unary or binary, thus, the value of a cell cannot be influenced by cells not adjacent to it. Thus, the fact that three cells of one column shares the same domain of three values cannot influence $X_{3,2}$ through arc consistency.

and there's no single value domain after the enforcement.

correct

Q2. CSPs: Properties

(a) 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.

correct **True** **False**

(b) 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? (circle one)

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

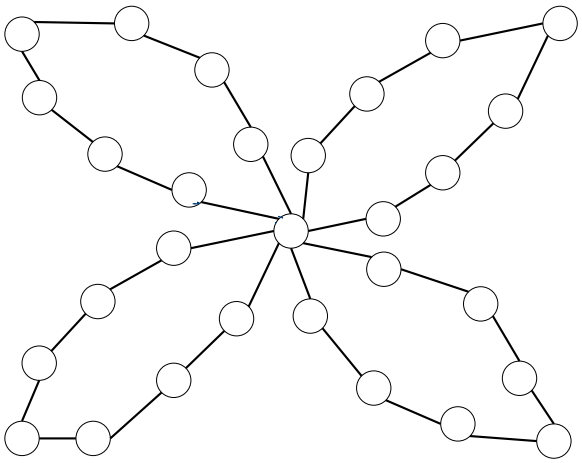
(c) What is the maximum number of times a backtracking search algorithm might have to backtrack in a general CSP, if it is running arc consistency and applying the MRV and LCV heuristics? (circle one)

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

(d) What is the maximum number of times a backtracking search algorithm might have to backtrack in a *tree-structured* CSP, if it is running arc consistency and using an optimal variable ordering? (circle one)

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

(e) **Constraint Graph** Consider the following constraint graph:



In two sentences or less, describe a strategy for efficiently solving a CSP with this constraint structure.

correct. Choose the node in the center as a cutset, instantiate this central node with all possible assignment. The residual structure are 4 trees, compute the residual CSP, and solve them.