CS 188 Spring 2019 Midterm Review CSPs Solutions

Q1. CSPs

(a) Pacman's new house

After years of struggling through mazes, Pacman has finally made peace with the ghosts, Blinky, Pinky, Inky, and Clyde, and invited them to live with him and Ms. Pacman. The move has forced Pacman to change the rooming assignments in his house, which has 6 rooms. He has decided to figure out the new assignments with a CSP in which the variables are Pacman **(P)**, Ms. Pacman **(M)**, Blinky **(B)**, Pinky **(K)**, Inky **(I)**, and Clyde **(C)**, the values are which room they will stay in, from 1-6, and the constraints are:

i) No two agents can stay in the same room
ii) **P** > 3
iii) **K** is less than **P**
iv) **M** is either 5 or 6
v) **P** $>$ **M**
vi) **B** is even
vii) **I** is not 1 or 6
viii) $|$**I**-**C**$| = 1$
ix) $|$**P**-**B**$| = 2$

(i) Unary constraints On the grid below cross out the values from each domain that are eliminated by enforcing unary constraints.

P	~~1~~	~~2~~	~~3~~	4	5	6
B	~~1~~	2	~~3~~	4	~~5~~	6
C	1	2	3	4	5	6
K	1	2	3	4	5	6
I	~~1~~	2	3	4	5	~~6~~
M	~~1~~	~~2~~	~~3~~	~~4~~	5	6

The unary constraints are ii, iv, vi, and vii. ii crosses out 1,2, and 3 for P. iv crosses out 1,2,3,4 for M. vi crosses out 1,3, and 5 for B. vii crosses out 1 and 6 for I. K and C have no unary constraints, so their domains remain the same.

(ii) MRV According to the Minimum Remaining Value (MRV) heuristic, which variable should be assigned to first?

○ P ○ B ○ C ○ K ○ I ● M

M has the fewest value remaining in its domain (2), so it should be selected first for assignment.

(iii) Forward Checking For the purposes of decoupling this problem from your solution to the previous problem, assume we choose to assign P first, and assign it the value 6. What are the resulting domains after enforcing unary constraints (from part i) and running forward checking for this assignment?

P						6
B	~~1~~	~~2~~	~~3~~	4	~~5~~	~~6~~
C	1	2	3	4	5	~~6~~
K	1	2	3	4	5	~~6~~
I	~~1~~	2	3	4	5	~~6~~
M	~~1~~	~~2~~	~~3~~	~~4~~	5	~~6~~

In addition to enforcing the unary constraints from part i, the domains are further constrained by all constraints involving P. This includes constraints i, iii, v, and ix. i removes 6 from the domains

of all variables. iii removes 6 from the domain of K (already removed by constraint i). v removes 6 from the domain of M (also already removed by i). ix removes 2 and 6 from the domain of B.

(iv) Iterative Improvement Instead of running backtracking search, you decide to start over and run iterative improvement with the min-conflicts heuristic for value selection. Starting with the following assignment:

P:6, B:4, C:3, K:2, I:1, M:5

First, for each variable write down how many constraints it violates in the table below.
Then, in the table on the right, for all variables that could be selected for assignment, put an x in any box that corresponds to a possible value that could be assigned to that variable according to min-conflicts. When marking next values a variable could take on, only mark values different from the current one.

Variable	# violated
P	0
B	0
C	1
K	0
I	2
M	0

	1	2	3	4	5	6
P						
B						
C		x				
K						
I		x		x		
M						

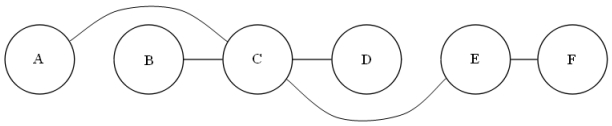
Both I and C violate constraint viii, because |I-C|=2. I also violates constraint vii. No other variables violate any constraints. According to iterative improvement, any conflicted variable could be selected for assignment, in this case I and C. According to min-conflicts, the values that those variables can take on are the values that minimize the number of constraints violated by the variable. Assigning 2 or 4 to I causes it to violate constraint i, because other variables already have the values 2 and 4. Assigning 2 to C also only causes C to violate 1 constraint.

(b) Variable ordering

We say that a variable X is backtracked if, after a value has been assigned to X, the recursion returns at X without a solution, and a different value must be assigned to X.
For this problem, consider the following three algorithms:

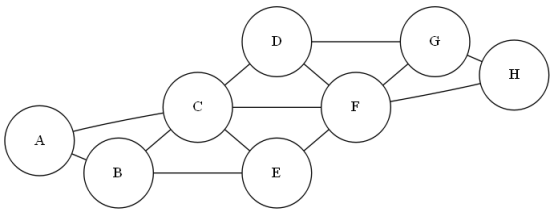
1. Run backtracking search with no filtering
2. Initially enforce arc consistency, then run backtracking search with no filtering
3. Initially enforce arc consistency, then run backtracking search while enforcing arc consistency after each assignment

(i) For each algorithm, circle all orderings of variable assignments that guarantee that no backtracking will be necessary when finding a solution to the CSP represented by the following constraint graph.
Algorithm 1:
No filtering means that there are no guarantees that an assignment to one variable has consistent assignments in any other variable, so backtracking may be necessary.
Algorithm 2:
This algorithm is very similar to the tree-structured CSP algorithm presented in class, in which arcs are enforced from one right to left, and then variables are assigned from left to right. The arcs enforced in that algorithm are a subset of all arcs enforced when enforcing arc consistency. Thus, any linear ordering of variables in which each variable is assigned before all of its children in the tree will guarantee no backtracking.
Algorithm 3:
Any first assignment can be the root of a tree, which, from class, we know is consistent and will not require backtracking. This assignment can then be viewed as conditioning the graph on that variable, and after re-running arc consistency, it can be removed from the graph. This results in either one or two tree-structured graphs that are also arc consistent, and the process can be repeated.



Algorithm 1	Algorithm 2	Algorithm 3
A-B-C-D-E-F	A-B-C-D-E-F	A-B-C-D-E-F
F-E-D-C-B-A	F-E-D-C-B-A	F-E-D-C-B-A
C-A-B-D-E-F	C-A-B-D-E-F	C-A-B-D-E-F
B-D-A-F-E-C	B-D-A-F-E-C	B-D-A-F-E-C
D-E-F-C-B-A	D-E-F-C-B-A	D-E-F-C-B-A
B-C-D-A-E-F	B-C-D-A-E-F	B-C-D-A-E-F

(ii) For each algorithm, circle all orderings of variable assignments that guarantee that no more than two variables will be backtracked when finding a solution to the CSP represented by the following constraint graph.



Algorithm 1	Algorithm 2	Algorithm 3
C-F-A-B-E-D-G-H	C-F-A-B-E-D-G-H	C-F-A-B-E-D-G-H
F-C-A-H-E-B-D-G	F-C-A-H-E-B-D-G	F-C-A-H-E-B-D-G
A-B-C-E-D-F-G-H	A-B-C-E-D-F-G-H	A-B-C-E-D-F-G-H
G-C-H-F-B-D-E-A	G-C-H-F-B-D-E-A	G-C-H-F-B-D-E-A
A-B-E-D-G-H-C-F	A-B-E-D-G-H-C-F	A-B-E-D-G-H-C-F
A-D-B-G-E-H-C-F	A-D-B-G-E-H-C-F	A-D-B-G-E-H-C-F

Algorithm 1:
This might backtrack for the same reason as algorithm 1 for the previous problem.
Algorithm 2:
If the first two assignments are not a cutset (C-F, C-G, or F-B), the graph will still contain cycles, for which there is no guarantee that backtracking will not be necessary. If the first two assignments are a cutset, the remaining graph will be a tree. However, because arc consistency was not enforced after the assignment, there is no guarantee against further backtracking. To see this, consider the sub-graph A,B,C,E, with domains {1,2,3}, and constraints A=C, B≥A, E=C+2, B≥C, E=B. If this is assigned in the order C-A-B-E, then by assigning 1 to C and A, assigning either 1 or 2 to B would result in an empty domain for E and cause B to backtrack.
Algorithm 3:
After assigning the cutset, the remaining graph is a tree, which guarantees no further backtracking with algorithm 3 as seen in the previous problem.

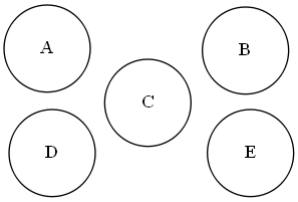
(c) All Satisfying Assignments Now consider a modified CSP in which we wish to find every possible satisfying assignment, rather than just one such assignment as in normal CSPs. In order to solve this new problem, consider a new algorithm which is the same as the normal backtracking search algorithm, except that when it sees a solution, instead of returning it, the solution gets added to a list, and the algorithm backtracks. Once there are no variables remaining to backtrack on, the algorithm returns the list of solutions it has found.

For each graph below, select whether or not using the MRV and/or LCV heuristics could affect the

number of nodes expanded in the search tree in this new situation.

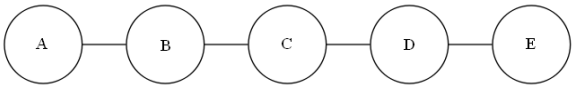
The remaining parts all have a similar reasoning. Since every value has to be checked regardless of the outcome of previous assignments, the order in which the values are checked does not matter, so LCV has no effect.
In the general case, in which there are constraints between variables, the size of each domain can vary based on the order in which variables are assigned, so MRV can still have an effect on the number of nodes expanded for the new "find all solutions" task.
The one time that MRV is guaranteed to not have any effect is when the constraint graph is completely disconnected, as is the case for part i. In this case, the domains of each variable do not depend on any other variable's assignment. Thus, the ordering of variables does not matter, and MRV cannot have any effect on the number of nodes expanded.

(i)



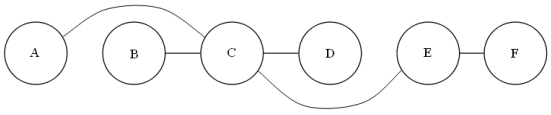
- [●] Neither MRV nor LCV can have an effect.
- [ ] Only MRV can have an effect.
- [ ] Only LCV can have an effect .
- [ ] Both MRV and LCV can have an effect.
- [ ] Neither MRV nor LCV can have an effect.

(ii)



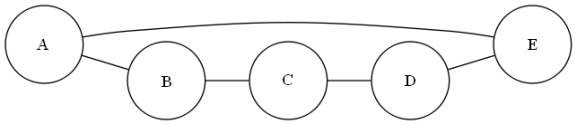
- [●] Only MRV can have an effect.
- [ ] Only LCV can have an effect .
- [ ] Both MRV and LCV can have an effect.
- [ ] Neither MRV nor LCV can have an effect.

(iii)



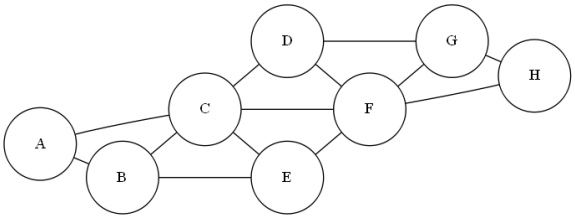
- [●] Only MRV can have an effect.
- [ ] Only LCV can have an effect .
- [ ] Both MRV and LCV can have an effect.
- [ ] Neither MRV nor LCV can have an effect.

(iv)



- [●] Only MRV can have an effect.
- [ ] Only LCV can have an effect .
- [ ] Both MRV and LCV can have an effect.
- [ ] Neither MRV nor LCV can have an effect.

(v)



- [●] Only MRV can have an effect.
- [ ] Only LCV can have an effect .
- [ ] Both MRV and LCV can have an effect.
- [ ] Neither MRV nor LCV can have an effect.

Q2. Time Management

Two of our GSIs, Arjun and Woody, are making their schedules for a busy morning. There are five tasks to be carried out:

(F) Pick up food for the group's research seminar, which, sadly, takes one precious hour.
(H) Prepare homework questions, which takes 2 consecutive hours.
(P) Prepare the PR2 robot for a group of preschoolers' visit, which takes one hour.
(S) Lead the research seminar, which takes one hour.
(T) Teach the preschoolers about the PR2 robot, which takes 2 consecutive hours.

The schedule consists of one-hour slots: 8am-9am, 9am-10am, 10am-11am, 11am-12pm. The requirements for the schedule are as follows:

(a) In any given time slot each GSI can do at most one task (F, H, P, S, T).

(b) The PR2 preparation (P) should happen before teaching the preschoolers (T).

(c) The food should be picked up (F) before the seminar (S).

(d) The seminar (S) should be finished by 10am.

(e) Arjun is going to deal with food pick up (F) since he has a car.

(f) The GSI not leading the seminar (S) should still attend, and hence cannot perform another task (F, T, P, H) during the seminar.

(g) The seminar (S) leader does not teach the preschoolers (T).

(h) The GSI who teaches the preschoolers (T) must also prepare the PR2 robot (P).

(i) Preparing homework questions (H) takes 2 consecutive hours, and hence should start at or before 10am.

(j) Teaching the preschoolers (T) takes 2 consecutive hours, and hence should start at or before 10am.

To formalize this problem as a CSP, use the variables F, H, P, S and T. The values they take on indicate the GSI responsible for it, and the starting time slot during which the task is carried out (for a task that spans 2 hours, the variable represents the starting time, but keep in mind that the GSI will be occupied for the next hour also - make sure you enforce constraint (a)!). Hence there are eight possible values for each variable, which we will denote by A8, A9, A10, A11, W8, W9, W10, W11, where the letter corresponds to the GSI and the number corresponds to the time slot. For example, assigning the value of A8 to a variables means that this task is carried about by Arjun from 8am to 9am.

(a) (2 pt) What is the size of the state space for this CSP?

8^5 since every task variable has 8 values, 4 time slots $\times$ 2 GSIs to carry them out, and there are 5 such tasks.

(b) (2 pt) Which of the statements above include unary constraints?

A unary constraint constrains the domain of a single variable. (d), (e), (i), (j) are unary constraints. (i) and (j) express both unary constraints (on the time of the tasks) and binary constraints (the length of tasks excludes other assignments during their time).

(c) **(4 pt)** In the table below, enforce all unary constraints by crossing out values in the table on the left below. If you made a mistake, cross out the whole table and use the right one.

F	A8	A9	A10	A11	~~W8~~	~~W9~~	~~W10~~	~~W11~~
H	A8	A9	A10	~~A11~~	W8	W9	W10	~~W11~~
P	A8	A9	A10	A11	W8	W9	W10	W11
S	A8	A9	~~A10~~	~~A11~~	W8	W9	~~W10~~	~~W11~~
T	A8	A9	A10	~~A11~~	W8	W9	W10	~~W11~~

(d) **(3 pt)** Start from the table above, select the variable S and assign the value A9 to it. Perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

F	A8	~~A9~~	~~A10~~	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
H	~~A8~~	~~A9~~	A10	~~A11~~	~~W8~~	~~W9~~	W10	~~W11~~
P	A8	~~A9~~	A10	A11	W8	~~W9~~	W10	W11
S	~~A8~~	A9	~~A10~~	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
T	~~A8~~	~~A9~~	~~A10~~	~~A11~~	~~W8~~	~~W9~~	W10	~~W11~~

Forward checking prunes the variables' domains of values inconsistent with S = A9, including: other choices for S, conflicting time slots for A9 (a), the choices of F that do not precede 9 (c), W from working during 9 (f), and A teaching the preschoolers (g). Furthermore, we cross out H:A8 H:W8 because preparing homework questions takes 2 hours and neither A nor W can work at 8 because they are at the seminar at 9. We also cross out T:W8 for this reason.

(e) **(3 pt)** Based on the result of (d), what variable will we choose to assign next based on the MRV heuristic (breaking ties alphabetically)? Assign the first possible value to this variable, and perform forward checking by crossing out values in the table below. Again the table on the right is for you to use in case you believe you made a mistake.

Variable F is selected and gets assigned value A8.

F	A8	~~A9~~	~~A10~~	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
H	~~A8~~	~~A9~~	A10	~~A11~~	~~W8~~	~~W9~~	W10	~~W11~~
P	~~A8~~	~~A9~~	A10	A11	W8	~~W9~~	W10	W11
S	~~A8~~	A9	~~A10~~	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
T	~~A8~~	~~A9~~	~~A10~~	~~A11~~	~~W8~~	~~W9~~	W10	~~W11~~

Have we arrived at a dead end (i.e., has any of the domains become empty)?

No.

(f) **(4 pt)** We return to the result from enforcing just the unary constraints, which we did in (c). Select the variable S and assign the value A9. Enforce arc consistency by crossing out values in the table below.

F	A8	~~A9~~	~~A10~~	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
H	~~A8~~	~~A9~~	A10	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
P	~~A8~~	~~A9~~	~~A10~~	~~A11~~	W8	~~W9~~	~~W10~~	~~W11~~
S	~~A8~~	A9	~~A10~~	~~A11~~	~~W8~~	~~W9~~	~~W10~~	~~W11~~
T	~~A8~~	~~A9~~	~~A10~~	~~A11~~	~~W8~~	~~W9~~	W10	~~W11~~

(g) **(2 pt)** Compare your answers to (d) and to (f). Does arc consistency remove more values or less values than forward checking does? Explain why.

Arc consistency removes more values by checking more relationships between variables: AC checks consistency between every pair of variables, and re-checks after domain pruning, while FC only checks between assigned and unassigned variables.

(h) **(1 pt)** Check your answer to (f). Without backtracking, does any solution exist along this path? Provide the solution(s) or state that there is none.

AC along this path gives 1 solution: F: A8 H: A10 P: W8 S: A9 T: W10
Backtracking is unnecessary since the constraints have been enforced by arc consistency and only single values remained in each domain.