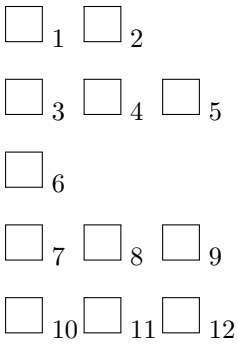
CS 188
Fall 2019

Final Review CSPs Solutions

Q1. Encrypted Knowledge Base

We have a propositional logic knowledge base, but unfortunately, it is encrypted. The only information we have is that:

- Each of the following 12 boxes contains a propositional logic symbol (A, B, C, D, or E) or a propositional logic operator and
- Each line is a valid propositional logic sentence.



(a) We are going to implement a constraint satisfaction problem solver to find a valid assignment to each box from the domain $\{A, B, C, D, E, \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow\}$.

Propositional logic syntax imposes constraints on what can go in each box. What values are in the domain of boxes 1-6 after enforcing the unary syntax constraints?

Box	Remaining Values
1	$\neg$
2	A B C D E
3	A B C D E $\neg$
4	$\wedge$ $\vee$ $\neg$ $\Rightarrow$ $\Leftrightarrow$
5	A B C D E
6	A B C D E

(b) You are given the following assignment as a solution to the knowledge base CSP on the previous page:

$$\neg A$$
$$B \Rightarrow A$$
$$D$$
$$C \vee B$$
$$D \vee E$$

Now that the encryption CSP is solved, we have an entirely new CSP to work on: finding a model. In this new CSP the variables are the symbols $\{A, B, C, D, E\}$ and each variable could be assigned to $true$ or $false$.

We are going to run CSP backtracking search with forward checking to find a propositional logic model M that makes all of the sentences in this knowledge base true.

After choosing to assign C to false, what values are removed by running forward checking? On the table of remaining values below, cross off the values that were removed.

Symbol	Remaining Values
A	F
B	T ~~F~~
C	F
D	T
E	T F

Forward checking removes the value false from the domain of B. Forward checking does not continue on to make any other arcs consistent.

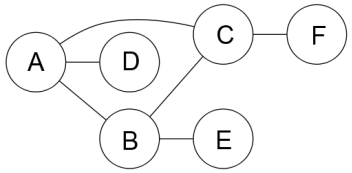
(c) We eventually arrive at the model $M = \{A = False, B = False, C = True, D = True, E = True\}$ that causes all of the knowledge base sentences to be true. We have a query sentence α specific as $(A \vee C) \Rightarrow E$. Our model M also causes α to be true. Can we say that the knowledge base entails α? Explain briefly (in one sentence) why or why not.

No, the knowledge base does not entail α. There are other models for which the knowledge base could be true and the query be false. Specifically $\{A = False, B = False, C = True, D = True, E = False\}$ satisfies the knowledge base but causes the query α to be false.

Q2. CSPs

(a) The graph below is a constraint graph for a CSP that has only binary constraints. Initially, no variables have been assigned.

For each of the following scenarios, mark all variables for which the specified filtering might result in their domain being changed.



(i) A value is assigned to A. Which domains might be changed as a result of running forward checking for A?

○ A ● B ● C ● D ○ E ○ F

Forward checking for A only considers arcs where A is the head. This includes $B \to A$, $C \to A$, $D \to A$. Enforcing these arcs can change the domains of the tails.

(ii) A value is assigned to A, and then forward checking is run for A. Then a value is assigned to B. Which domains might be changed as a result of running forward checking for B?

○ A ○ B ● C ○ D ● E ○ F

Similar to the previous part, forward checking for B enforces the arcs $A \to B$, $C \to B$, and $E \to B$. However, because A has been assigned, and a value is assigned to B, which is consistent with A or else no value would have been assigned, the domain of A will not change.

(iii) A value is assigned to A. Which domains might be changed as a result of enforcing arc consistency after this assignment?

○ A ● B ● C ● D ● E ● F

Enforcing arc consistency can affect any unassigned variable in the graph that has a path to the assigned variable. This is because a change to the domain of X results in enforcing all arcs where X is the head, so changes propagate through the graph. Note that the only time in which the domain for A changes is if any domain becomes empty, in which case the arc consistency algorithm usually returns immediately and backtracking is required, so it does not really make sense to consider new domains in this case.

(iv) A value is assigned to A, and then arc consistency is enforced. Then a value is assigned to B. Which domains might be changed as a result of enforcing arc consistency after the assignment to B?

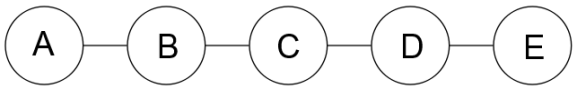
○ A ○ B ● C ○ D ● E ● F

After assigning a value to A, and enforcing arc consistency, future assignments and enforcing arc consistency will not result in a change to As domain. This means that Ds domain won't change because the only arc that might cause a change, $D \to A$ will never be enforced.

(b) You decide to try a new approach to using arc consistency in which you initially enforce arc consistency, and then enforce arc consistency every time you have assigned an even number of variables.

You have to backtrack if, after a value has been assigned to a variable, X, the recursion returns at X without a solution. Concretely, this means that for a single variable with d values remaining, it is possible to backtrack up to d times. For each of the following constraint graphs, if each variable has a domain of size d, how many times would you have to backtrack in the worst case for each of the specified orderings?

(i)



A-B-C-D-E: 0

A-E-B-D-C: $2d$

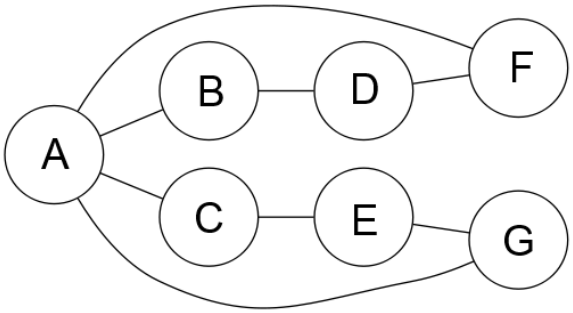
C-B-D-E-A: 0

If no solution containing the current assignment exists on a tree structured CSP, then enforcing arc consistency will always result in an empty domain. This means that running arc consistency on a tree structured CSP will immediately tell you whether or not the current assignment is part of a valid solution, so you can immediately start backtracking without further assignments.

$A - B - C - D - E$ and $C - B - D - E - A$ are both linear orderings of the variables in the tree, which is essentially the same as running the two pass algorithm, which will solve a tree structured CSP with no backtracking.

$A - E - B - D - C$ is not a linear ordering, so while the odd assignments are guaranteed to be part of a valid solution, the even assignments are not (because arc consistency was not enforced after assigning the odd variables). This means that you may have to backtrack on every even assignment, specifically E and D. Note that because you know whether or not the assignment to E is valid immediately after assigning it, the backtracking behavior is not nested (meaning you backtrack on E up to d times without assigning further variables). The same is true for D, so the overall behavior is backtracking $2d$ times.

(ii)



A-B-C-D-E-F-G: d^2

F-D-B-A-C-G-E: $d^4 + d$

C-A-F-E-B-G-D: d^2

$A - B - C - D - E - F - G$: The initial assignment of A, B might require backtracking on both variables, because there is no guarantee that the initial assignment to A is a valid solution. Because A is a cutset for this graph, the resulting graph consists of two trees, so enforcing arc consistency immediately returns whether the assignments to A and B are part of a solution, and you can begin backtracking without further assignments.

$F - D - B - A - C - G - E$: Until A is assigned, there is no guarantee that any of the previous values assigned are part of a valid solution. This means that you may be required to backtrack on all of them, resulting in d^4 times. Furthermore, the remaining tree is not assigned in a linear order, so further backtracking may be required on G (similar to the second ordering above) resulting in a total of $d^4 + d$.

$C - A - F - E - B - G - D$: This ordering is similar to the first one. except that the resulting trees are not being assigned in linear order. However, because each tree only has a single value assigned in between each run of arc consistency, no backtracking will be required (you can think of each variable as being the root of the tree, and the assignment creating a new tree or two where arc consistency has been enforced), resulting in a total of d^2 times.