CS322 Fall 1999
Module 6 (Constraint Satisfaction Problems)
Assignment 6

Solution.

The aim of this assignment is to learn about constraint satisfaction problems.

Question 1

In this question you will look at backtracking versus arc consistency for solving CSP problems.

Consider a scheduling problem, where there are five variables A, B, C, D, and E, each with domain $\{1, 2, 3, 4\}$. Suppose the constraints are: $E - A$ is odd, $A < D$, $D < C$, $E > B$, $A \neq B$, $E \neq C$, $E \neq D$.

(a) Show how backtracking can be used to solve this problem, using the variable ordering A, B, C, D, E. To do this you should draw the search tree generated to find all answers. Indicate clearly the satisfying assignments. How many leaves are in the search tree generated?

To indicate the search tree, write it in text form with each branch on one line. For example, suppose we had variables X, Y and Z with domains t, f, and constraints $X \neq Y$, $Y \neq Z$. The corresponding search tree can be written as:

```
X=t Y=t failure
    Y=f Z=t solution
        Z=f failure
X=f Y=t Z=t failure
        Z=f solution
    Y=f failure
```

Hint: this may be large! It may be easier to write a program to generate it (but then again, it may not be easier; try it by hand first).

(b) Is there a different variable ordering that results in a smaller tree? Give a variable ordering that results in the smallest tree. How many leaves are on this tree? Explain why you think this is the optimal ordering. (A good explanation is more important than actually finding the optimal ordering).

(c) Show how arc consistency can be used to solve this problem. To do this you need to:

i) draw the constraint graph,

ii) show which elements of a domain are deleted at each step, and which arc is responsible for removing the element,

iii) show explicitly the constraint graph after arc consistency has stopped.

iv) show how splitting domains can be used to solve this problem. Assume we split the alphabetically first variable that has more than one element in its domain for any arc-consistent network. Include all arc consistency steps.

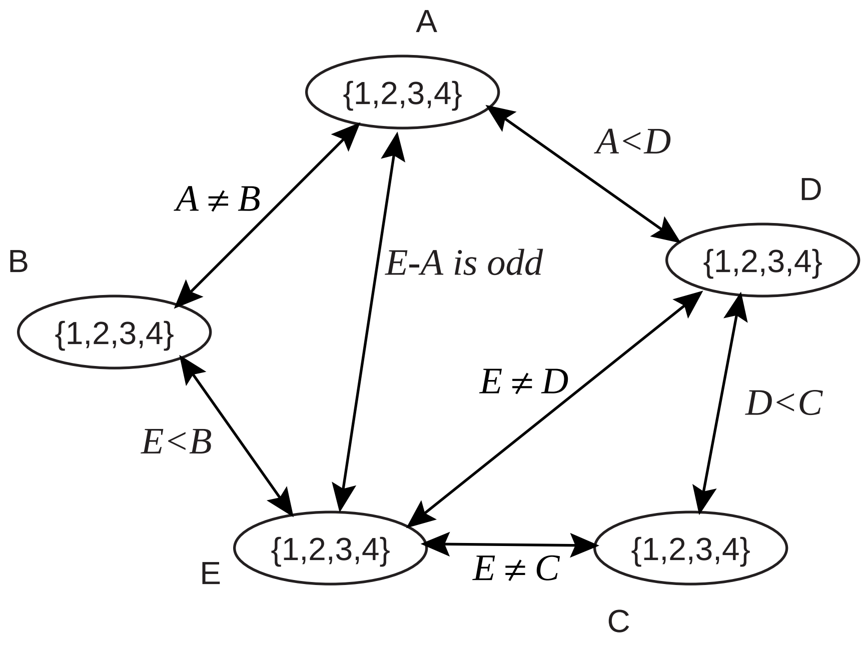

Figure 1: Constraint Graph

Solution

(a) Show how backtracking can be used to solve this problem, using the variable ordering A, B, C, D, E. To do this you should draw the search tree generated to find all answers. Indicate clearly the satisfying assignments. How many leaves are in the search tree generated?

There are 232 leaves. The tree can be generated by the following Prolog program.

(b) Is there a different variable ordering that results in a smaller tree? Give a variable ordering that results in the smallest tree. How many leaves are on this tree? Explain why you think this is the optimal ordering. (A good explanation is more important than actually finding the optimal ordering).

Look at the variables. A reasonable strategy is a greedy (hillclimbing) strategy to select the variable that cuts the space most at any stage (I.e., you want to fail as quickly as possible). For example, a good ordering is A, D, C, E, B, which results in 56 leaves (see the tree).

(c) Show how arc consistency can be used to solve this problem. To do this you need to:

i) draw the constraint graph.

See Figure 1.

ii) show which elements of a domain are deleted at each step, and which arc is responsible for removing the element,

Element Removed	Arc
E=1	$\langle E, B\rangle$
B=4	$\langle B, E\rangle$
D=4	$\langle D, C\rangle$
C=1	$\langle C, D\rangle$
A=3,A=4	$\langle A, D\rangle$
D=1	$\langle D, A\rangle$
C=2	$\langle C, D\rangle$

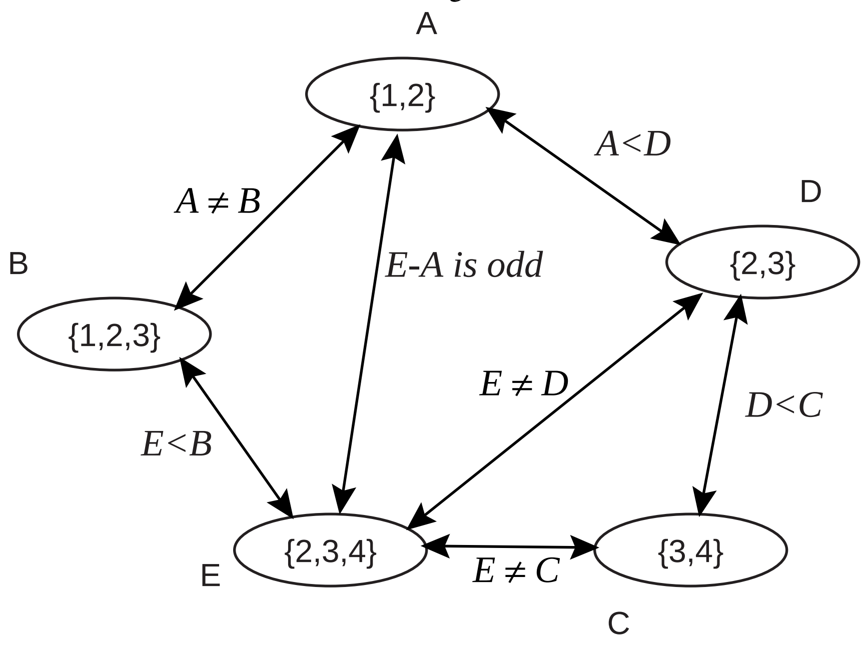

Figure 2: Arc Consistent Constraint Net

iii) show explicitly the constraint graph after arc consistency has stopped.
See Figure 2.

iv) show how splitting domains can be used to solve this problem. Assume we split the alphabetically first variable that has more than one element in its domain for any arc-consistent network. Include all arc consistency steps.
Here is the results without the arc consisteny steps:

```
A=1  B=2 solution (A=1, B=2, C=3, D=2, E=4}
     B=3 solution (A=1, B=3, C=3, D=2, E=4}
A=2 failure
```

Question 2

Consider the crossword puzzle shown in Figure 3.

Suppose that each word position is a variable, with domain a set of words. The value of the word position is the word that goes into that position.

Suppose we have the following variables and domains:

Variable	Domain
1–across	{ant, big, bus, car, has}
3–across	{book, buys, hold, lane, year}
4–across	{ant, big, bus, car, has}
1–down	{book, buys, hold, lane, year}
2–down	{browns, ginger, search, symbol, syntax}

The constraints are that the intersecting words have to have the same letter in the intersecting positions.

(a) This is not arc consistent. Give one domain value that can be pruned. Explain which arc can be used to prune it any why.

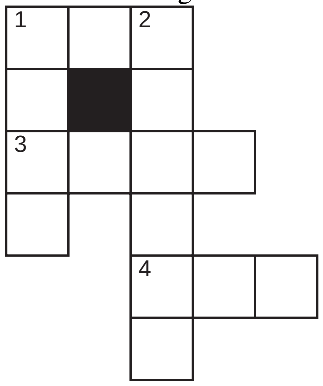

Figure 3: Simple Crossword

(b) Give the domains where arc consistency stops (without domain splitting).

(c) Show how hill climbing can be used for this problem. Suppose a neighbour is obtained by changing the value of one of the variables to the alphabetically next or previous domain element. The heuristic function to be maximized is the number of satisfied constraints, and you always choose a neighbour with the maximal heuristic value (even if it is the same as the current node).

i) Show the sequence of assignments and corresponding heuristic values when we start with the assignment: *1–across=ant, 3–across=book, 4–across=ant, 1–down=book, 2–down=brown.*

ii) Suppose Alex suggested using the domains pruned by arc consistency, but otherwise the same. Is this a good idea? Explain. Does it work well?

iii) Suppose Joe suggested picking any domain element as long as it improves the heuristic value. Why might this be better or worse than the original solution? Does it work well?

Solution

(a) This is not arc consistent. Give one domain value that can be pruned. Explain which arc can be used to prune it any why.

ant can be pruned from the domain of *1–across* using the arc ⟨*1–across,1–down*⟩ as there is no element of the domain of *1–down* that starts with "*a*".

(b) Give the domains where arc consistency stops (without domain splitting).

Variable	Domain
1–across	{bus, has}
3–across	{lane, year}
4–across	{ant, car}
1–down	{buys, hold}
2–down	{search, syntax}

(c) Show how hill climbing can be used for this problem. Suppose a neighbour is obtained by changing the value of one of the variables to the alphabetically next or previous domain element. The heuristic function to be maximized is the number of satisfied constraints, and you always choose a neighbour with the maximal heuristic value (even if it is the same as the current node).

i) Show the sequence of assignments and corresponding heuristic values when we start with the assignment: *1–across=ant, 3–across=book, 4–across=ant, 1–down=book, 2–down=browns.*

Here is one possible sequence:

1–across	3–across	4–across	1–down	2–down	h-value
ant	book	ant	book	browns	1
big	book	ant	book	browns	2
big	book	ant	book	ginger	2
big	book	ant	buys	ginger	2
big	book	big	buys	ginger	2
big	buys	big	buys	ginger	2
big	hold	big	buys	ginger	2
big	lane	big	buys	ginger	2
big	lane	big	hold	ginger	3

This reaches a foothill, and never solves the goal.

ii) Suppose Alex suggested using the domains pruned by arc consistency, but otherwise the same. Is this a good idea? Explain. Does it work well?

Yes. In fact it seems stupid not to do this. This works much better, if only because the search space is smaller, the plateaus seem smaller, and it more quickly finds better foothills. For example:

1–across	3–across	4–across	1–down	2–down	h-value
bus	lane	ant	buys	search	2
bus	lane	ant	buys	syntax	4
bus	lane	ant	hold	syntax	4
has	lane	ant	hold	syntax	5

iii) Suppose Joe suggested picking any domain element as long as it improves the heuristic value. Why might this be better or worse than the original solution? Does it work well?

This again works better, but the cost of each iteration (finding the neighbours) is more. It isn't obvious that the extra time would be better spent trying different starting points.

Here is one possible sequence:

1–across	3–across	4–across	1–down	2–down	h-value
ant	book	ant	book	browns	1
big	book	ant	book	browns	2
big	book	ant	buys	browns	2
big	year	ant	buys	browns	2
big	year	ant	buys	ginger	3
big	year	ant	buys	search	3
bus	year	ant	buys	search	4
bus	year	car	buys	search	5