CS 4100: Foundations of Artificial Intelligence (Fall 2020) Chris Amato
Student: Matthew Schanzlin Due October 06, 2020

CSPs

1 Constraint Satisfaction Problems

AIMA 6.2. Consider the problem of placing k knights on an n x n chessboard such that no two knights are attacking each other, where k is given and $k < n^2$.

(a) Choose a CSP formulation. In your formulation, what are the variables?

There are n^2 variables T, for the board tiles that the knights can be placed on.

(b) What are the possible values of each variable?

The possible values of each variable is 1 or 0/true or false/yes or no for a knight being placed on that particular tile.

(c) What sets of variables are constrained, and how?

Pairs of tiles which correspond to valid chess knight moves are constrained via a logical " NAND" constraint method. In other words, sets of variable pairs that have 1s/trues/yes for each variable would fail, but all other combinations would be allowed. For example, one such variables set could be (T1, T2), where T1 is 0,0 and T2 is 1,2. In this case, either T1 could be 1/true/yes, or T2 could be 1/true/yes, but not both because then each of these knights would be attacking each other. They both however could be 0/false/no since this would mean that neither has knights that can attack the other. This is why the constraint for this problem can't be a simple "alldiff." Additionally, there is a global constraint on squares occupied by a knight being <= k (in number).

AIMA 6.3. Consider the problem of constructing (not solving) crossword puzzles for fitting words into a rectangular grid. The grid, which is given as part of the problem, specifies which squares are blank and which are shaded. Assume that a list of words (i.e., a dictionary) is provided and that the task is to fill in the blank squares by using any subset of the list. Formulate this problem precisely in two ways:

(a) As a general search problem. Choose an appropriate search algorithm and specify a heuristic function. Is it better to fill in blanks one letter at a time or one word at a time?

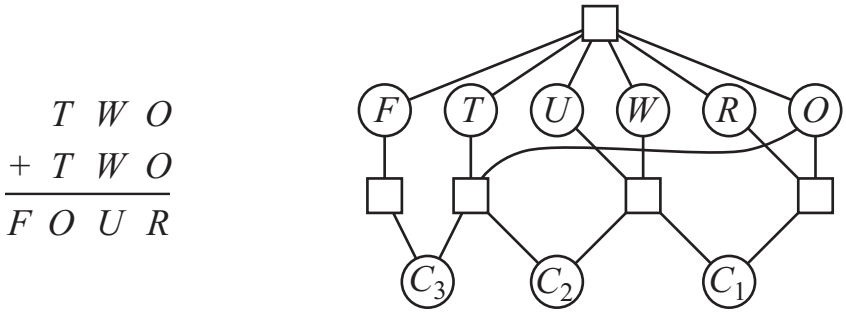
My formulation would use depth first search and a heuristic could be a function that indicates how many (or how few) complete words there are with the current substring of letters (IE, if the letters filled in so far for a 5 letter space are ma, magic would have a higher heuristic value than apple. I believe that it would be better to fill in the blanks one letter at a time because we could start with the spaces that are overlapped. Our heuristic would have us choose letters that enable the highest number of potential words to be selected from our dictionary. Once we placed all of our intersection letters we would have a greatly reduced number of potential words.

(b) As a constraint satisfaction problem. Should the variables be words or letters?

It seems more natural to have the variables be words in a constraint satisfaction model of the problem. This is because individual letters can't be constrained the way words can, and the resulting domains will be far larger. IE, the letters surrounding an overlapped square can be any letter, but many such letter possibilities will be words that aren't in our dictionary. If we choose our variables as words, then we can constrain future overlapping words more naturally as the set of words that have the matching letter at the overlapping index.

Which formulation do you think will be better? Why?

I believe the constraint satisfaction problem formulation will be better because the domain of potential choices for the variables will be much smaller/pre shrunk.



(a) (b)

Figure 1: (a) A cryptarithmetic problem. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeroes are allowed. (to The constraint hypergraph for the cryptaritlunetic problem, showing the Alldiff constraint (square box at the top) as well as the column addition constraints (four square boxes in the middle). The variables C1, C2, and C3 represent the carry digits for the three columns.

AIMA 6.5. Solve the cryptarithmetic problem in Figure 1 (6.2 from AIMA) by hand, using the strategy of backtracking with forward checking and the MRV and least-constraining-value heuristics.

The constraints for this problem are as follows:
O + O = R + 10 * C10
C10 + W + W = U + 10 * C100
C100 + T + T = O + 10 * C1000
C1000 = F
F != 0 (No Leading zeros)
T != 0 (No leading zeros)

Initial possible values:
F = {1}, O, W, U, R = {0, 2, 3, 4, 5, 6, 7, 8, 9} T = {2, 3, 4, 5, 6, 7, 8, 9}
MRV dictates we assign a value to T since it has the least number of possible values. T can't be less than 5 since T needs to produce a carry 1 for F. If W + W doesnt produce a carry, then T = 5 makes O = 0 which can't work. So LCV dictates we select from 6, 7, 8, or 9 and if we choose 7 then O must be 4 or 5. 4 would be with no carry from W, and 5 would be with a carry. If we choose 4 then than means R can only be 8, and W + W must produce no carry. This means W must be less than 5, and the only remaining values less than 5 are 2 and 3. However, 2 won't work since that would make U = 4 which is O. So we choose 3 and this leaves 6 for U. This solution in total is F=1, W=3, O=4, U=6, T=7, R=8.

AIMA 6.9 Explain why it is a good heuristic to choose the variable that is most constrained but the value that is least constraining in a CSP search.

The reason why it is good to choose the most constrained variable but the least constrained value is because this approach will minimize the number of nodes in the search tree by pruning larger parts of the tree earlier. Since values can be thought of as our solution, we want to find the solution that is the most likely to "work" and thus we choose the one that is least constraining.