Venkata Krishna Mohan Sunkara
65528957

HOME WORK-5

1.

1. In a constraint satisfaction problem, we are to assign values to variables such that a set of constraints are satisfied, the order in which variables are ordered have a profound effort on search effort. The principle of dynamic variable ordering is 'the choice of next variable to be considered at any point depends on the current state of search.' [1]

2. The dynamic variable ordering is implemented by the following heuristics:

a. Minimum Remaining Values or Fail-First Heuristic: It is a heuristic which selects the next variable to be considered as the one, with the fewest legal values. It picks a variable which is most likely to cause a failure soon, thereby pruning the search tree. It avoids pointless searches by detecting the failure immediately. [2]

b. Degree Heuristic or Brelaz Heuristic: It is a heuristic which selects the next variable to be considered as the one, with highest degree. It attempts to reduce the branching factor on future choices by selecting the variable that is involved in the largest number of constraints on the other unassigned variables. [2]

c. Rho Heuristic: It is a heuristic which branches in to sub problem that maximizes the solution density, p. The intuition is to branch in to sub problem where the greatest fraction of states is expected to be solutions. [3]

2.

1. A CSP is said to be arc consistent if every variable is arc consistent with every other variable.

CSP-1 is arc-consistent as all variables are arc consistent with each other.

CSP-2 is not arc-consistent, because value 2 of variable 'x' doesn't have a support in the domain of variable 'y' for the constraint (x+y) is odd.

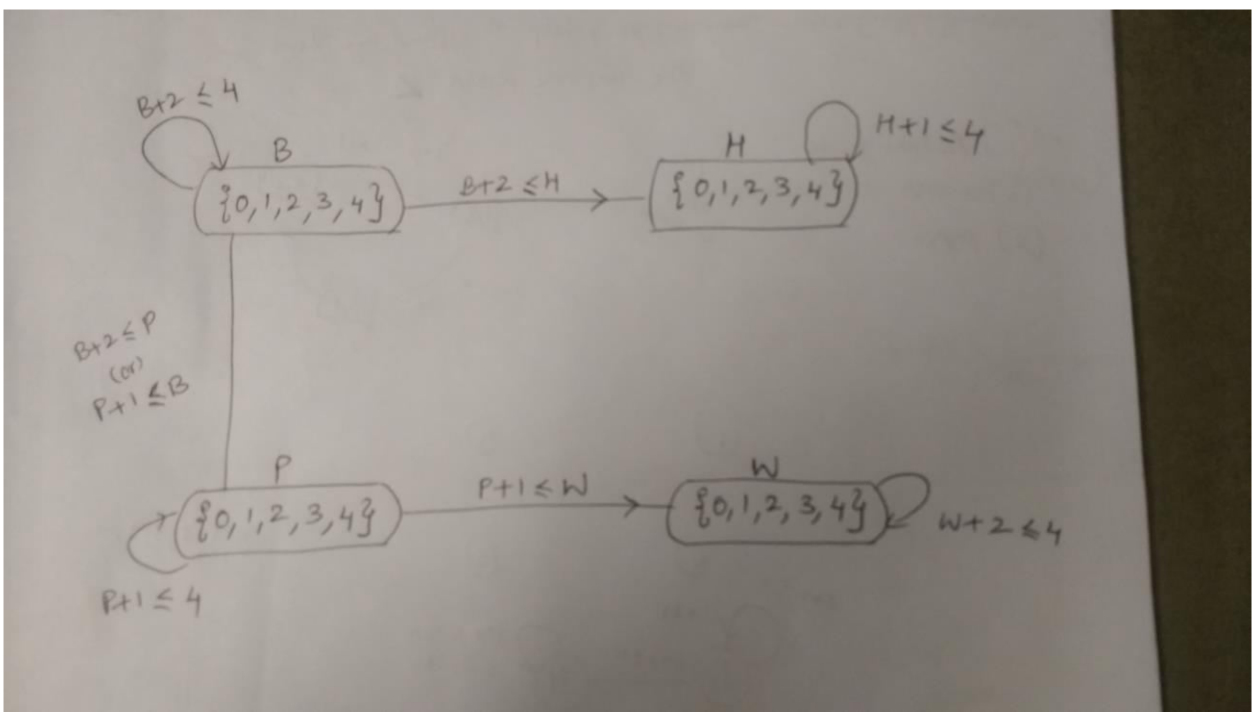
2. CSP-1 is arc consistent because for value 1 in variable 'x' has a support of value 4 in domain of variable 'y'. Similarly value 2 in variable 'x' has a support of value 3 in domain of variable 'y'.

Similarly, 1 has support 4 and 2 has support 3 in domain of variable 'z'. So, the variable 'x' is arc consistent.

Now the values in ‘y’, 3 has a support of 4 and 4 has a support of 3 in domain of variable ‘z’. So, the variables ‘y’ and ‘z’ are arc consistent.
As all variables are arc consistent, the entire CSP-1 is arc consistent.
3. CSP-2 is not arc consistent because value 2 of variable ‘x’ doesn’t have a support in the domain of variable ’y’ for the constraint (x+y) is odd. So, the entire CSP-2 is not arc-consistent.
The domain of ‘x’ must be reduced to {1} to make the node ‘x’ arc-consistent with respect to the constraint (x+y) is odd. Now we have to update the domains of all the adjacent variables to ‘x’ which share a constraint. Now update the domain of ‘z’ to {4} to make it arc- consistent with respect to the constraint x+2 != z. Now as the domain of ‘z’ is changed update the domain of ‘y’ is reduced to {2} with respect to the constraint y != z. Now again check for support in the domain of ‘x’. As, we have a support the entire CSP-2 is now arc-consistent.

3.

a.



The Job scheduling problem can be represented as CSP as shown above. Each task/job is represented as a variable and the domain of each variable is the start time of the task/job. The unary constraint on each task/job restrict the start time of a task/job given its duration. The values of domain denote the start time of a task/job. For example, if value=0 means the task/job starts at 0 time, if value=1 then the task/job starts after 1 hour. The binary constraints between two tasks/jobs represent the constraint that a task has to be completed before another task.

The binary constraint among the tasks 'B' and 'P' is that both of them cannot be performed simultaneously, but it isn't mentioned which task needs to be done first. So, the constraint among 'B' and 'P' is represented using OR condition. However, while solving the CSP, if we consider B task to be completed first then a solution doesn't exist. So, we choose to start with task P.

b. Now we have 4 variables with domains as:

B = {0,1,2,3,4}
H = {0,1,2,3,4}
P = {0,1,2,3,4}
W = {0,1,2,3,4}

Now update the domains of each variable to satisfy the unary constraints by forward checking.

B = {0,1,2}
H = {0,1,2,3}
P = {0,1,2,3}
W = {0,1,2}

Now let us select the variable ordering as P B W H.

Let us start with P=0

The updated domains of the future variables become:

B = {1,2} by constraint P+1<=B
H = {0,1,2,3}
W = {1,2} by constraint P+1<=W

Now consider expanding B

Consider B=2 // Just to show backtracking

The updated domains of future variables become:

H = {} by constraint B+2<=H. We obtained a nullified domain which means this assignment is not valid. So, backtrack to another value assignment.

Consider B =1

The updated domains of future variables become:

H = {3} by constraint B+2<=H

W = {1,2}

Now consider expanding W

Consider W =1

The updated domains of future variable become:

H = {3}

Now consider expanding H

Consider H = 3

Now we can see all the variables have been assigned and this path is a solution to the CSP.

Now consider W =2 and H=3. Even this is a solution to the CSP.

Now consider P=1 then the updated domains become:

B = {2} by constraint P+1<=B

H = {0,1,2,3}

W = {2} by constraint P+1<=W

Now consider B =2 then the updated domains become:

H = {} by constraint B+2<=H

Now backtrack to P and assign P=2 then the updated domains become:

B = {} by constraint P+1<=B

Now consider another value P=3 then the updated domains become:

B = {} by constraint P+1<=B

The complete search tree is represented as:

6.

1. 3-SAT is a satisfiability problem which consists of clauses of 3 literals each. A 3-SAT is satisfiable if every clause consists of at least one true literal. A 3-SAT can be represented as CSP as follows:

1. Represent each literal in the 3-SAT problem as a variable. For example, if literals x1, ~x1, x3 are present in 3-SAT then all the three literals are represented as variables in CSP with domains {T, F} for each of them.
2. Now represent the constraints among the variables as a condition of the clause. For example, if x1 and x2 are present in a clause then the constraint among x1 and x2 variables is x1 v x2.
3. Also represent a special constraint among xi and ~xi such that both of them are not equal. For example, if x1 and ~x1 are present as variables in CSP then add a constraint among them as !-=.
4. Now the question of 3-SAT is reduced as question to the CSP as If the 3-SAT has a satisfying truth assignment then it can be transformed as a solution to the CSP.

   For example, if the 3-SAT has 2 clauses such that c1 = (x1 v x2 v x3) and c2 = (x2 v x3) then it is transformed in to a CSP as variables x1 and x2 and x3 with domains {T, F}. Now the acceptable values of a constraint c1 are {(T T T); (T F F); (T F T); (T T F); (F T T); (F F T); (F T F)} and the acceptable values of constraint c2 are {(T T); (T F); (F T).

2. The arity of a constraint is the number of literals in the corresponding clause. For example, arity of constraint c1 = (x1 v x3) is 2.

3. The variables in the corresponding CSP for 3-SAT are c1, c2, c3, c4, ~c1, c5. The domains of all these variables are {T, F}. The constraints are (c1 v c2 v c3), (c2 v c3 v c4), (~c1 v c5) and (c1 v c4 v c5) and (c1 != ~c1).

4. The representation of SAT in CSP is similar to 3-SAT except that the constraints over the variables may not span over 3 variables but may span across many variables.