Constraint Satisfaction Problem With Arc Consistency Algorithm

I. PROBLEM STATEMENT

Given constraint satisfaction problem <X,D,C >
-X denotes set of variables
-D denotes set of domain of the variables
-C denotes set of constraint between variables find the assignment of variables such that it satisfy all constraints.

II. SOLUTION DESCRIPTION

In constraint satisfaction problem if we reduce the doma Maof every variables using its constraint then runtime of CSP will reduce drastically. So, here I implement arc consistency algorithm for different graph to reduce the domains.

A. Constraints specification

1) $x_j > x_i \mid j > i$
2) $gcd(x_j,x_i) = 1 \mid$ i and j not prime
3) $x_j = x_i^2 \mid$ i , j even
4) $x_j \% x_i = 0 \mid j > i$ and i , j odd
5) $3x_j = x_i \mid j > 2i$

B. Domain Specification

1) D_{x_i}= [1, 2, 3, 4, 5, 6, 7, 8, 9, 10,...] where i is prime
2) D_{x_i}=[10, 11, 12, 13, 14, 15 ....] i is odd and not prime and i%5!=0
3) D_{x_i}=[2, 4, 6, 8, 10,....] where i is even and i%4!=0
4) D_{x_i}=[1, 4, 9, 16, 25, 36, 49, 64, 81, 100,...] i%4 = 0
5) D_{x_i}=[3,5,7,9,11,..] where i%5=0

I implement AC-1 , AC-2, AC-3 , AC-4 for three kinds of graph which are random graph, scale free graph, DAG.

1) first I take number of nodes and number of elements in domain. then I assign a set of domain for every node.here I used 6 domains which is distributed to all the nodes using these condition
2) Then I create a random graph using random function.
3) Now all consistency to every edges using some condition over there node number
4) After creating the CSP graph, I call Arc consistency function(AC-1 , AC-2, AC-3 , AC-4)
5) Take time interval for every function and then save it to time array Then plot a graph time vs number of nodes repeat this process for scale free graph and DAG

III. ARC CONSISTENCY ALGORITHM

AC-1: here we take every edge then check consistency using revise function. if a domain is reduced then repeat the check for all nodes

AC-2: here we use two queues . from one queue we take a edge then check revise function if any a edge revise then put all edges where second element is revised element and first element is less equal i. after checking all entry in first queue and second queue to first queue. then recheck again.

AC-3: here we use a queue . put all edges to the queue . Popping a edge from queue if the edge is revised then put all edge which second element is the revised node. repeat this until queue is empty

AC-4: At first we put all list where the consistent values for each edge are kept. if for a value there is no value that satisfy any of its constraint then remove it from the node . then check the list if any value is depended on it. it a value is not consistent with any value remove it.at a time there is no inconsistent value remaining.

IV. PERFORMANCE MEASUREMENT

A. random graph

In random graph I randomly choose two node and connected them by an edge. then impose the edge the constraint using the condition. here I run the arc-consistency algorithms for 10-200 nodes increased by 10.and plot the time vs nodes graph. I repeat the process for 25,50,75,100 domain size.

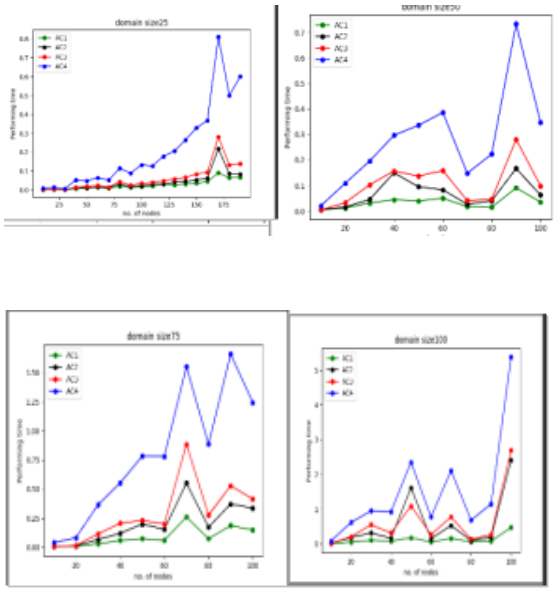

Figure 1. random graph performance measurement

B. scale free graph

create an n-node network based on the Barabasi-Albert model [6]. Starting from a connected 2-node network, we repeatedly add a new node, ran-domly connecting it to two existing nodes

here I run the arc-consistency algorithms for 10-200 nodes increased by 10.and plot the time vs nodes graph. I repeat the process for 25,50,75,100 domain size. DAG it is a acyclic

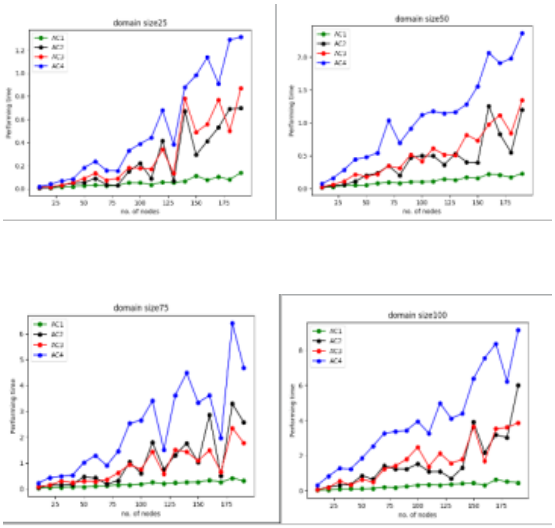

Figure 2. scale free graph performance measurement

graph. every time I connect a visited node to unvisited node. then impose constraints to the edges

here I run the arc-consistency algorithms for 10-200 nodes increased by 10.and plot the time vs nodes graph. I repeat the process for 25,50,75,100 domain size.

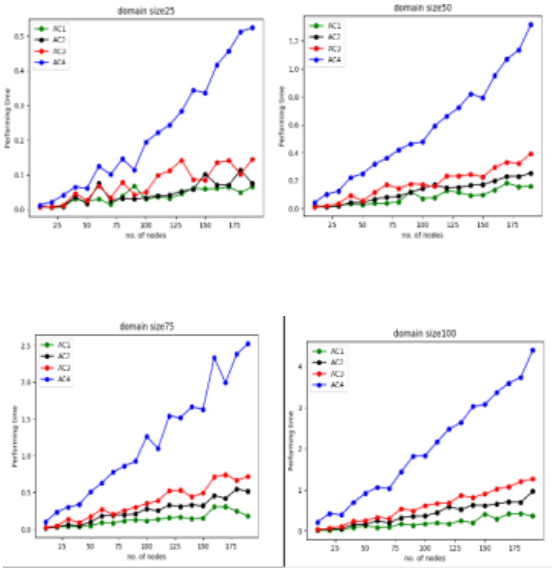

Figure 3. DAG performance measurement

V. PERFORMANCE MEASUREMENT

here we see that for every graph. we get better performance for AC-1 and worst performance for AC-4. AC-3 is better than AC-2 and AC-2 is better than AC-3 and AC-4. domain reduction here we see that for all algorithm my domain size reduced drastically. using arc-cons we increase the runtime of CSP algorithm.ioistencnstatistical Analysis here I check my one of the output to the Tukey Test. The output given below. here A,B,C,D denotes AC1, AC2,AC3,AC4 respectively.

Tukey HSD results

treatments pair	Tukey HSD Q statistic	Tukey HSD p-value	Tukey HSD inferfence
A vs B	2.5820	0.2979982	insignificant
A vs C	6.3440	0.0019122	** p<0.01
A vs D	12.0837	0.0010053	** p<0.01
B vs C	3.7620	0.0731755	insignificant
B vs D	9.5016	0.0010053	** p<0.01
C vs D	5.7396	0.0045500	** p<0.01

Figure 4. Tukey statistical measurement