Constraint Satisfaction Problem(CSP)
Comparative Performance Analysis of Arc-Consistency Algorithms AC-1, AC-2, AC-3, AC4

Amit Roy, Roll: JH-40
Department of Computer Science and Engineering,
University of Dhaka,
Email: aroy7298@gmail.com

I. Problem Definition

Artificial intelligence tasks can be formulated as **Constraint Satisfaction Problems(CSP)**.Constraint satisfaction problem consists of three components, $\langle$ ***X,D,C***$\rangle$

X: A set of variables, $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$.

D: A set of domains, $\{\mathbf{D}_1, \ldots, \mathbf{D}_n\}$, one for each variable.

C: A set of constraints that specify allowable combinations of values.

Each domain $\mathbf{D}_i$ consists of a set of allowable values, $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ for each variable $\mathbf{X}_i$. Each constraint $\mathbf{C}_i$ consists of a relation between two variables. This type of constraint is called **binary constraint**.

An assignment that does not violate any constraints is called a **consistent** or **legal assignment**. A **complete assignment** is one in which every variable is assigned, and a solution to a CSP is a **consistent, complete assignment**.

Before making an attempt to construct a complete solution of a CSP problem, we should eliminate local inconsistencies. A variable in a CSP is **arc-consistent** if every value in its domain satisfies the variable's binary constraints.

$\mathbf{X}_i$ is arc-consistent with respect to another variable $\mathbf{X}_j$ if for every value in the current domain $\mathbf{D}_i$ there is some value in the domain $\mathbf{D}_j$ that satisfies the binary constraint on the arc $(\mathbf{X_i}, \mathbf{X_j})$. A network is arc-consistent if every variable is arc-consistent with every other variable.

In this problem we wish to implement the most popular algorithms for arc-consistency named AC-1, AC-2, AC-3, AC4.

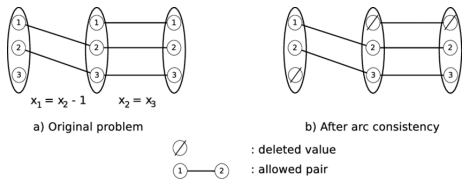

Fig. 1 Example of Binary CSP

II. Constraints

To implement the arc-consistency algorithms we need a set of constraints which will contain some relations between two variables of the consistency graph. For the sake of simplicity we are considering binary constraints between two variables and we define a set of mathematical relations as the constraints.

Constraints:

$$y \leq x \tag{1}$$

$$y \geq x \tag{2}$$

$$y = x \tag{3}$$

$$y \neq x \tag{4}$$

$$y\%x = 0, \text{ y divides x} \tag{5}$$

$$gcd(x, y) = 1 \text{ i.e. x and y are co-prime} \tag{6}$$

$$y = mx + c, \text{ m and c are constant} \tag{7}$$

$$y = ax^2 + bx + c, \text{ a,b and c are constant} \tag{8}$$

Before implementing an arc-consistency algorithm, we will generate random graph for different number of nodes(e.g 20,40,80,100) of the network using some built library(e.g. NetworkX in python). Then, for each edge of the network we will randomly choose a constraint from the predefined constraints list and then assign that constraint on that particular arc.

We use those constraints to check the consistency of the nodes. Suppose two node $\mathbf{X}_i$ and $\mathbf{X}_j$ share an arc between them in the network. While checking whether $\mathbf{X}_i$ is arc-consistent with respect to $\mathbf{X}_j$, we will check for every value x $\in \mathbf{D}_i$, whether there is a corresponding value y $\in \mathbf{D}_j$ which satisfies the constraint assigned on the arc between $\mathbf{X}_i$ and $\mathbf{X}_j$. If x $\notin \mathbf{D}_i$ then x is removed from $\mathbf{D}_i$ to make it arc-consistent with respect to $\mathbf{X}_j$

III. Domains

As we have already defined, a CSP problem has a set of domains, one for each variable. Since our constraints are mathematical equations we will choose integers within a range as the domain.

For each variable, first we will randomly choose a domain size within a range(e.g. 20-100). When the domain size of a variable is fixed, we will assign values to the domain of that variable. While assigning we can choose a range between 1 and 10000 and pick values randomly from this range as the domain of that variable.

Those domains will be necessary to determine whether two variables sharing an arc between them satisfy the constraint of that arc. In other words, the purpose of these domains of each variable is to check the arc-consistency between two variables.

IV. Comparative Performance Analysis

After generating the graph, assigning constraining on the edges and and also defining the domains of each variable we will run the **AC-1,AC-2,AC-3,AC-4** algorithms on the network. For the same number of nodes we will generate different graphs and run the algorithms several(10-20) times on them and take the average value of the performance metrics(e.g. running time or % of domain reduction per edge).

Finally we will plot a graph where the **Y axis** will be **performance metrics** and **X axis** will be **number of nodes** in the network. We can find 4 lines for four different algorithms and compare between them by analyzing the graph.