CSE-4111 Artificial Intelligence Lab

Constraint Satisfaction Problem : Arc Consistency Algorithms

Saif Mahmud

4th Year 1st Semester, Roll : SH - 054

February 9, 2019

Abstract

Constraint Satisfaction Problem (CSP) is defined as a triple $P = (X, D, C)$ where $X = \{1, ..., n\}$ is a set of variables. Each variable $X_i \in X$ has a finite domain $D_i \in D$ of values that can be assigned to it. Arc consistency is defined as X_i is arc-consistent with respect to another variable X_j if for every value in the current domain D_i there is some value in the domain D_j that satisfies the binary constraint on the arc (X_i, X_j). A CSP is arc consistent (AC) if and only if every variable is arc consistent. Arc Consistency is achieved through removing arc inconsistent values until complete consistency or a failure is obtained. In this regard, there are 4 (four) algorithms within the scope of this experiment attaining the goal of removing arc inconsistency from CSPs and they are namely *AC - 1, AC - 2, AC - 3, and AC - 4*.

Problem Formulation

This experiment will incorporate the implementation of AC algorithms and measuring the performance of the algorithms against defined evaluation metrics. The problem for this experiment is designed in a manner which is analogous to optimization through linear programming. In case of linear programming, an optimal value for each variable is obtained within the bar of given constraints. Here, I have merged the idea of removing inconsistency with finding optimal solution for given variables. Arc Consistency algorithms will eliminate the inconsistent values with respect to the constraints and thus will yield a reduced domain for each variable. If the size of reduced domain is 1 then it is the optimal solution and otherwise I have to search through the reduced domain for obtaining optimal value in reduced search space for linear programming.

Input Specification

All four algorithms will take the CSP as the argument. Here, the nodes and edges of parameter CSP will be formulated with the following principles :

- Constraint graphs will be generated with the variables for optimization in linear programming as nodes and binary constraints incorporating two variables as edges between the nodes.
- The domain size will be varied in the range $[5, 25]$ (suppose) in order to measure the performance of the algorithms in proportion to domain size.
- Each variable will be assigned a domain set with discrete values ranging $[a, b]$. In this regard, the experiment will include intersecting and non-intersecting domain values to detect the behavior of specific algorithm. Here, the value of a and b will be chosen for a specific variable like $a = -50$ and $b = 150$ and the values of a and b will be varied for each variable. In fact, each variable will have a domain of bounded range. This formulation of domain values will incorporate bounds propagation within the scope of binary constraints. Therefore, the algorithms will reduce the range of values for each domain and this domain will indicate the optimal solution for variables in linear programming.

Constraint Definition

In order to maintain arc consistency, binary constraints can be imposed between the edge of two variables. Moreover, unary global constraints will be used to obtain node consistency. The following is an illustration of the constraints used in optimization technique for linear programming :

Global Constraints (Not always used)

- All variables must assume positive value ($X_i > 0$)
- *Alldiff* : All variable will assume different values at the same time.

Binary Constraints (Let, X and Y are arbitrary variables in CSP)

- $X \pm Y \geq val$ where val is natural number within range $[0, 100]$
- $X \pm Y \leq val$ where val is natural number within range $[0, 100]$
- $X \pm Y = val$ where val is natural number within range $[0, 100]$

Evaluation Metric

The implementation of algorithms will be evaluated with the following metric in order to show a comparison among them :

- CPU running time on the same input graph
- Change in performance with a gradual change in the size of input CSP graphs
- Time and Space Complexity of individual implementation of algorithms
- Mean constraint check and revision of nodes (in number) for each algorithm