CS 668 – Planning & Constraint Satisfaction – CSP Programming Assignment-Specs

Please discuss, distribute work and document the development process with weekly reviews and report. Modules I and II are necessary, III is desirable, and IV is for extra credit.

I. Representation

- Design and implement a Finite CSP representation with modules to access variables, their domains, constraints and their extensions, and the constraint graph.
- Implement modules to accept a CSP with constraints as intensions and convert the constraints to extensional forms. For example given variables X and Y and their (finite) domains and a constraint like (X<Y) to produce its extension containing the allowable pairs of values. Also cater to a special constraint "All-different" which says each variable must have a unique value (for example in cryptarithmetic problems).

II. Reasoning

- Implement program modules to implement different levels of i-consistency.
- Implement the basic procedure for Backtracking
- Implement procedures for different degrees of Lookahead search
- Implement procedures for Backjumping, Graph Backjumping, and Conflict Directed Backjumping.

III. Integration

- Implement a CSP workbench on which a user can
    - Define a new CSP or load one from a library
    - Reduce the CSP by enforcing some degree of i-consistency
    - Choose one of
        - the basic Backtracking algorithm
        - Backtracking with some degree of Lookahead
        - Backtracking with Backjumping
        - Backtracking with Lookahead and Backjumping
    - Compare two versions of solving the CSP in terms of execution time and/or some measure of complexity.

IV. Applications (optional, in increasing order of difficulty)

1. Implement Planning as CSP and use a combination of the solvers.
2. Implement a cryptarithmetic solver that does maximum amount of propagation and leaves a reasoning trace (of why values are chosen or eliminated for variables). If still unsolved, till solved, picks one value for a variable and continues propagation. May have to backtrack.
3. Implement a graphics interface for drawing trihedral figures, convert the drawing into a CSP, execute the AC-2 algorithm, and show how labels for edges and vertices are eliminated during propagation.