Constraint Satisfaction Problem - Part1

Submitted buy -

Paritosh Goel

Abstract

Summary of the problem

Constraint satisfaction problems are mathematical questions defined as a set of objects whose state must satisfy a number of constraints or limitations. CSPs represent the entities in a problem as a homogeneous collection of finite constraints over variables, which is solved by constraint satisfaction methods.

Solution Summary

This project aims to support Constraint Satisfaction Problem in general and specific domain will be chosen in part-2 of the project. We devise the solution by using strategies for Variable Selection (Minimum Remaining Values, Degree Heuristics), domain value selection by using Least constraining value. We also use preprocessing by using AC3 and constraint propogation as we move forwards in the Search Tree.

Note: The program is written in Java language.

Instructions

Compilation -

1. Provided a compiled version for ease.
2. Otherwise the source code can be compiled in Java runnable jar format, by importing the source code in an IDE and exporting the project as Runnable jar from there.

Execution -

1. After the project is compiled, place the runnable jar in a suitable directory.
2. Place the input files in the same directory
3. Run command Java -jar {Jar-name} {file-name}

Command -
java -jar ConstraintSatisfactionProblem.jar input_file_7.txt

Approach

The overall approach is explained as below -

1. **File parsing**
    a. First the input file is parsed.
    b. Created Variables and Domain objects in Java with relevant information.
    c. Parse Constraints (All three types) and represent them as constraint Matrices and in CSP.
    d. Create Domains of each variable .
    e. Parse the unary constraints and reduce the domains accordingly.
    f. Now created a Constraint Satisfaction Problem by having variables and domains with Constraints specified.
    g. Create a empty assignment.
2. **PreProcessing**
    a. Apply AC3 algorithm to above approach to satisfy the binary constraints.
    b. Note - Unary constraints have already been used to reduce the domains.
    c. I use only binary constraints in AC3 algorithm.
3. **AC3 Algorithm**
    a. Create a set of arcs representing all the Binary Constraints.
    b. Push all these arcs in the queue.
    c. Revise the domain and update the queue.
    d. Repeat - till queue is not empty.
4. **Backtracking Search**
    a. Once the Preprocessing is done, we start the problem solving by using the Backtrack search algorithm.
    b. We pass the initial empty assignment and the csp to the Search
    c. The Search -
        i. Select-Variable according to the minimum remaining values and if clash is found then use degree heuristic. Note: Degree heuristics will be compared between
        ii. Order-domain-values - After the variable is being selected, select a list of domain values to check with.

iii. Keep a copy of current assignment, so that in case of failure, we can backtrack to that copy.
iv. Try to assign the domain value to variable and infer if it fails or succeeds by checking the length of task (which should not exceed the deadline) and the constraints.
v. When we infer, we also propogate our constraints and check weather we have valid values left for the unassigned variables.
vi. Incase out infer steps rejects the proposal, we backtrack and try to assign the different value.
vii. If we iterate over all the possible values, we backtrack to the previous assignment and try to change the value of domain selected for previous assignment.
viii. If we are exausted with all possible values, we will return the incomplete assignment and the main function will know the incompleted assignment and report failures.
ix. In case of success, we will display the assignment with all relevant information.

Results

Below executions were made to test the program -

1. **Successfully tried my program on 7 different Processor - Task problems (All the files and their results are included in the end of the report)**
2. **Successfully tried my program on Map Colouring Problem with below mentioned input file and output.**

   ##### - variables
   W 1
   N 1
   A 1
   Q 1
   S 1
   V 1
   T 1
   ##### - values
   r
   g
   b
   ##### - deadline constraint
   22
   ##### - unary inclusive
   ##### - unary exclusive
   ##### - binary equals
   ##### - binary not equals
   W N
   W A
   N A
   N Q
   A Q
   A S
   Q S
   A V
   S V
   ##### - binary not simultaneous

Output -

{W(1)=2, V(1)=2, T(1)=0, Q(1)=3, A(1)=5, S(1)=3, N(1)=3}
Initial Domain of each Variable by considering UNARY CONSTRAINTS ONLY
{W(1)=[r , g , b], N(1)=[r , g , b], A(1)=[r , g , b], Q(1)=[r , g , b], S(1)=[r , g , b], V(1)=[r , g , b], T(1)=[r , g , b]}

Initiating Preprocessing by Using AC3...
AC3 applied!
Domain after AC3 == > {W(1)=[r , g , b], N(1)=[r , g , b], A(1)=[r , g , b], Q(1)=[r , g , b], S(1)=[r , g , b], V(1)=[r , g , b], T(1)=[r , g , b]}

Initiating backtrack search ...

Now displaying intermediate assignments while back track search ==>
Assigned{}
######
Assigned{A(1)= r }
######
Assigned{A(1)= r , S(1)= g }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b , T(1)= r }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b , T(1)= g }
######
Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b , T(1)= b }
######
Assignment Failed - S(1) b to Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b , T(1)= b }
Assignment Failed - A(1) g to Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b , T(1)= b }
Assignment Failed - A(1) b to Assigned{A(1)= r , S(1)= g , Q(1)= b , N(1)= g , V(1)= b , W(1)= b , T(1)= b }

##
CSP Problem succeeded!!
Below is the final list of Processors, assigned with tasks and their total time-
b [W(1)] 1
g [S(1)] 1
r [A(1)] 1
b [V(1)] 1
b [Q(1)] 1
g [N(1)] 1
b [T(1)] 1
Maximum length - 1

Strengths

1. Uses AC3 algorithm as pre-processing.
2. Program is modular enough in Java to handle extensions (Each logic is separated).
3. Program is flexible enough to accomodate new type of complex domain and variable objects.
4. Program uses Minimum Value Remaining and Degree Heuristics both for variable selection.
5. Program is easy enough to change Domain Selection Strategy.
6. Effective Constraint Propogation by reducing the domain of the variables on the go
7. Logging is effective for debugging purposes.
8. Program runs very fast.
9. Even though a tree is formed, we are not replicating the unnecessary nodes in the memory. So memory consumption is less.

Weakness

1. Do not handle the extra credit use case of cost consideration.

Input Files and Output

- variables
C 6
D 3
E 5
F 8
G 15
H 12
I 7
J 19
K 4
L 9
- values
p
q
r
x
y
z
- deadline constraint
22
- unary inclusive
D q r
G p r y z
- unary exclusive
C q r
E z
L y p
- binary equals
C D
E F
- binary not equals
C G
- binary not simultaneous
E G p z
{C(6)=2, H(12)=0, I(7)=0, K(4)=0, J(19)=0, F(8)=1, E(5)=2, L(9)=0, G(15)=2, D(3)=1}
Initial Domain of each Variable by considering UNARY CONSTRAINTS ONLY
{C(6)=[p , x , y , z], D(3)=[q , r], E(5)=[p , q , r , x , y], F(8)=[p , q , r , x , y , z], G(15)=[p , r , y , z], H(12)=[p , q , r , x , y , z], I(7)=[p , q , r , x , y , z], J(19)=[p , q , r , x , y , z], K(4)=[p , q , r , x , y , z], L(9)=[q , r , x , z]}

Initiating Preprocessing by Using AC3...
AC3 applied!
AC3 algorithm failed, indicating no such solution possible!

2.
- variables
C 6
D 3
E 5
F 8
G 15
H 12
I 7
J 19
K 4
L 9
- values
p
q
r
x
y
z
- deadline constraint
22
- unary inclusive
D q r
- unary exclusive
- binary equals
C D
E F
- binary not equals
C G
- binary not simultaneous
CSP Problem succeeded!!
Below is the final list of Processors, assigned with tasks and their total time-
q [F(8), E(5)] 13
x [K(4)] 4
z [H(12)] 12
r [G(15)] 15
q [C(6), D(3)] 9
z [I(7)] 7
x [L(9)] 9
y [J(19)] 19
Maximum length - 19

3.

- variables
C 6
D 3
E 5
F 8
G 15
H 12
I 7
J 24
K 4
L 9
- values
p
q
r
x
y
z
- deadline constraint
24
- unary inclusive
D q r
- unary exclusive
- binary equals
C D
E F
C E
- binary not equals
C G
L G
- binary not simultaneous
L C x q

##
CSP Problem succeeded!!
Below is the final list of Processors, assigned with tasks and their total time-
x [G(15)] 15
y [J(24)] 24
r [L(9)] 9
q [C(6), E(5), F(8), D(3)] 22
z [H(12)] 12
z [K(4)] 4
z [I(7)] 7
Maximum length - 24

4.
- variables
C 6
D 12
E 6
F 24
- values
p
q
r
s
- deadline constraint
24
- unary inclusive
C p
- unary exclusive
F q
- binary equals
D E
- binary not equals
C D
- binary not simultaneous
E C q p

##
CSP Problem succeeded!!
Below is the final list of Processors, assigned with tasks and their total time-
r [D(12), E(6)] 18
s [F(24)] 24
p [C(6)] 6
Maximum length - 24

5.

- variables
C 6
D 3
E 5
F 8
G 15
H 12
I 7
J 19
K 4
L 9
- values
p
q
r
x
y
z
- deadline constraint
22
- unary inclusive
D q r
L p
- unary exclusive
- binary equals
C D
E F
- binary not equals
C G
- binary not simultaneous

###
CSP Problem succeeded!!
Below is the final list of Processors, assigned with tasks and their total time-
x [G(15)] 15
x [K(4)] 4
z [H(12)] 12
q [C(6), D(3)] 9
p [L(9)] 9
z [I(7)] 7
p [F(8), E(5)] 13
y [J(19)] 19
Maximum length - 19

6.
- variables
C 6
D 3
E 5
F 8
G 15
H 12
I 7
J 19
K 4
L 9
- values
p
q
r
x
y
z
- deadline constraint
22
- unary inclusive
D q r
L p
J q
- unary exclusive
- binary equals
C D
E F
- binary not equals
C G
- binary not simultaneous

NO such assignment is possible

7.

- variables
C 6
D 3
E 5
F 8
G 15
H 12
I 7
J 19
K 4
L 9
- values
p
q
r
x
y
z
- deadline constraint
22
- unary inclusive
D q r
J z
I y
- unary exclusive
- binary equals
C D
E F
- binary not equals
C G
J G
- binary not simultaneous

##
CSP Problem succeeded!!
Below is the final list of Processors, assigned with tasks and their total time-
z [J(19)] 19
y [I(7)] 7

y [H(12)] 12
q [C(6), D(3)] 9
r [G(15)] 15
x [K(4)] 4
q [F(8), E(5)] 13
x [L(9)] 9
Maximum length - 19