50.021 – Artificial Intelligence

Kwan Hui

Week 3 - Constraint Satisfaction Problems

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources. ]

Due: 16th Feb, 11:59pm

Submission: via eDimension

**Name: Sharryl Seto 1005523**

**Class: Cl03**

# Pure backtracking

Let’s look at a simple map coloring problem and explore the behavior of constraint propagation, backtracking and forward checking in this context. Here is the relevant problem specification:

**V1**

**V2**

**V3**

**V4**

**{**

**R, G, B**

**}**

**{**

**G, B**

**}**

**{**

**R, G**

**}**

**{**

**G**

**}**

Assume we use pure backtracking to search for a solution to this problem. We use a fixed variable ordering in the search (V1, V2, V3 then V4) and the values are considered in the order shown in the picture above. Then, show the order in which individual variable assignments are considered by backtracking (this is analogous to the order in which nodes are expanded by depth-first search).

Show all assignments even if it is immediately found to be inconsistent upon testing.

Write each variable assignment per line as a number from 1 to 4 (indicating the variable) followed by a letter, drawn from R, G, B. The first four assignments are given below, complete the rest.

V1 R

V2 G

V3 R (inconsistent assignment with V1, go to other possible value)

V3 G

V4 G (inconsistent assignment with V3, backtrack to V1 as no other possible values for V3)

V1 G

V2 G (inconsistent assignment with V1, go to other possible value)

V2 B

V3 R

V4 G

Solution: V1 G, V2 B, V3 R, V4 G

# Backtracking with forward checking

Repeat the previous question but using backtracking with forward checking (BT-FC) to search for a solution to this problem. We use the same variable ordering and value ordering as before. Show the order in which assignments are considered by BT-FC. Whenever propagating after an assignment causes a domain to become empty, that causes the search to backtrack. Write all assignments as before.

**V1**

**V2**

**V3**

**V4**

**{**

**R, G, B**

**}**

**{**

**G, B**

**}**

**{**

**R, G**

**}**

**{**

**G**

**}**

V1 R (delete inconsistent values R from neighbours. Current domain: V2 {G,B}, V3 {G})

V2 G (delete inconsistent values G from neighbours. Current domain: V1 {R,B}, V4 becomes empty, backtrack to V2)

V2 B (delete inconsistent values B from neighbours. Current domain: V1 {R}, V4 {G})

V3 G (delete inconsistent values G from neighbours. Current domain: V1 {R}, V4 becomes empty, backtrack to V1 as no other values of V2)

V1 G (delete inconsistent values G from neighbours. Current domain: V2 {B}, V3 {R})

V2 B (delete inconsistent values B from neighbours. Current domain: V1 {R,G}, V4 {G})

V3 R (delete inconsistent values R from neighbours. Current domain: V1 {G}, V4 {G})

V4 G (delete inconsistent values G from neighbours. Current domain: V2 {B}, V3 {R})

Solution: V1 G, V2 B, V3 R, V4 G

# Arc consistency

Apply the AC-3 algorithm on the same constraint graph. Write the values in each of the indicated variable domains after any changes required to achieve arc consistency for just that arc. Then, assume that the following arcs are done sequentially, with the effects on the domains propagating. Write domains as a sequence of letters, for example, R B. If there are no values left in a domain, write None.

X → Y is consistent iff for every value x of X there is some allowed y, need to check all edges in all directions

Queue: V1-2-4-3-1-3-4-2-1 (clockwise, then anti-clockwise)

V1 --> V2: D1=RGB, D2=GB (don’t delete as there is allowance for all R, G, B in V1)

V2 🡪 V4: D2=GB , D4=G (delete G from V2, add incoming edges to V2)

V4 🡪 V3: D4=G , D3=RG (OK)

V3 🡪 V1: D3=RG , D1=RGB (OK)

V1 🡪 V3: D1=RGB, D3=RG (OK)

V3 🡪 V4: D3=RG , D4=G (delete G from V3, add incoming edges to V3)

V4 🡪 V2: D4=G , D2=B (OK)

V2 🡪 V1: D2=B , D1=RGB (OK)

Original queue done, now do the additional edges we added

- Incoming edges to V2, ignore repeated V4 🡪 V2 (completed above)

V1 🡪 V2: D1=RGB, D2=B (delete B from V1, add incoming edges to V1)

- Incoming edges to V3

V1 --> V3: D1=RG, D3=R (delete R from V1, add incoming edges to V1)

V4 🡪 V3: D4=G , D3=R (OK)

- Incoming edges to V1

V2 🡪 V1: D2=B , D1=G (OK)

V3 -> V1: D3=R , D1=G (OK)

Solution:

D1=G, D2=B, D3=R, D4=G