

50.021 - AI

Alex

Week 10: CSP

[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.]

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:

MRV:

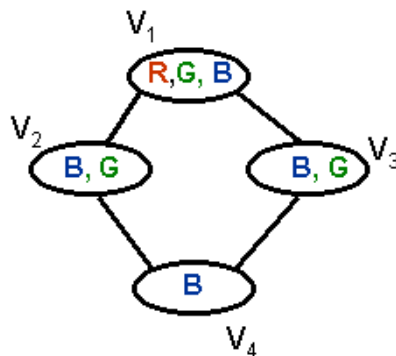
4B 2B x

4B 2G

4B 2G 3B x

4B 2G 3G

4B 2G 3G 1R ✓
1 - pure backtracking



Assume we use pure backtracking to search for a solution to this problem. We use a fixed variable ordering in the search (V_1, V_2, V_3 and V_4) 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 the first 10 assignment even if it is immediately found to be inconsistent upon testing. Assume that the search continues even after finding a valid solution. 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, e.g. 1 R.

2 - backtracking with forward checking

Repeat assuming we use backtracking with forward checking to search for a solution to this problem. We use the same variable or-

1 R

1 R 2 B

1 R 2 B 3 B

1 R 2 B 3 B 4 B x

1 R 2 B 3 G

1 R 2 B 3 G 4 B x

1 R 2 G

1 R 2 G 3 B

1 R 2 G 3 B 4 B

1 R 2 G 3 G

1 R 2 G 3 G 4 B ✓

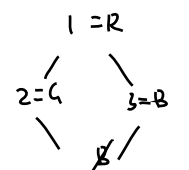
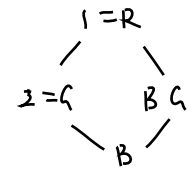
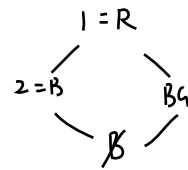
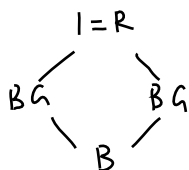
Back track

x

x

Back track

...



1 R

1 R 2 B

BACKTRACK

1 R 2 G

1 R 2 G 3 B

BACKTRACK

dering 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 backup in the search. Assume that the search continues even after finding a valid solution. Write each assignment (up to 10) as before.

3 - Arc consistency

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.

aB

V1 - V2: D1= \hat{R} D2= B G
 V1 - V3: D1= \hat{R} D3= B G
 V2 - V4: D2= G D4= B
 V3 - V4: D3= G D4= B
 V1 - V2: D1=RB D2= G
 V1 - V3: D1=RB D3= G

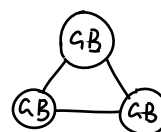
Domain changed,
 changed set : D2, D3 } check their neighbors again

4 - Thinking of CSP

Indicate whether the following statements are true or false.

1. Constraint propagation without search is sufficient to find a satisfying set of assignments in most problems. **F**

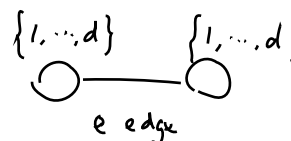
2. If constraint propagation leaves all variables with non-empty domains, there is at least one solution. **F** Count :



3. If constraint propagation leaves some variable with an empty domain, there is no solution. **T**

4. If constraint propagation leaves all variables with singleton domains, there is a unique solution. **T**

5. Constraint propagation takes on the order of ed^2 arc tests for one pass through all the arcs in a constraint graph with e edges and d values per domain. **T**



6. When doing backtracking with Forward Checking (BT-FC), then Forward checking during backtracking requires verifying that the newly assigned value for a variable is consistent with all previously assigned values of variables.

The value assignment algorithm checks, consistency check,
 not forward checking (which is for
 removing unsatisfying domain values — prune)

F

7. Backtracking search with full constraint propagation (until arc-consistency is reached) generally results in an answer using fewer arc-tests than search with forward-checking only. *→ Lots of arc-tests whenever value changes*
8. Backtracking with Forward Checking (BT-FC) never explores more possible variable values than pure backtracking. *→ will prune domains* **F** **T**
9. BT-FC never tests more constraint arcs than pure backtracking. **F**
10. Dynamic ordering of variables based on domain size (e.g. minimum remaining values) is generally a very effective strategy to speed up solving CSP (whether or not all the answers are desired). **T** *need to enumerate all, ordering does not help*
11. Dynamic ordering of values based on impact on neighboring variables (e.g. least constraining value) is generally a very effective strategy to speed up solving CSP when all possible answers are desired and not just one. **F**

5 - Semi-Magic Square

Consider a 3×3 array, each of whose entries can be either 1, 2 or 3. We want to find an assignment to each of the entries so that the entries in each row, in each column and in one of the diagonals are different. Note that this will also ensure that these row, column and diagonals add up to 6 ($1 + 2 + 3$). But, note that the "adding to 6" constraint is not a "binary constraint", that is, it involves more than three variables. However, the constraint that each pair of values in the row, column or diagonal be different is a "binary constraint".

You can find a description of the Semi-Magic Square problem in these slides https://www.autonlab.org/_media/tutorials/constraint05.pdf, in particular in Slide 14 (reproduced here).

A New CSP (where fancier propagation is possible)

- The semi magic square
- Each variable can have value 1, 2 or 3

V_1	V_2	V_3	⬇ This row must sum to 6
V_4	V_5	V_6	⬇ This row must sum to 6
V_7	V_8	V_9	⬇ This row must sum to 6
⬆ This column must sum to 6	⬆ This column must sum to 6	⬆ This column must sum to 6	⬆ This diagonal must sum to 6

Slide 14

Implement a CSP that captures this problem. Use the file `semi-magic.py` in the code distribution; it imports `csp.py`.

You should use the variable names in the image above in your CSP formulation.

Experiment solving this problem with the various solution methods, ranging from pure backtracking, variable and value orderings, and inference methods (forward_checking and mac). Look at the number of assignments attempted by each algorithm, that should give you some idea of the effectiveness of the methods on this problem. Make sure that you understand the results.

6 - Scheduling a Job Shop

You can find a description of Job Shop Scheduling in these slides https://www.autonlab.org/_media/tutorials/constraint05.pdf , in particular Slides 39-44.

Implement a CSP that captures this type of constraints. The file `jobs.py` has several sample problems. Note that, in general, we want the shortest possible schedule, that is, the smallest deadline for completion that is feasible. We have given you deadline values that work for each of the problems.

Experiment solving these problems with the various solution methods, ranging from pure backtracking, variable and value orderings, and inference methods (forward_checking and mac). What seems to matter most?

Try some infeasible values for the deadline and report what happens.