Constraint Satisfaction Problems (CSPs)

Russell and Norvig Chapter 6

CSP example: map coloring

Given a map of Australia, color it using three colors such that no neighboring territories have the same color.

Constraint satisfaction problems

A CSP is composed of:
- A set of variables $X_1, X_2, \ldots, X_n$ with domains (possible values) $D_1, D_2, \ldots, D_n$
- A set of constraints $C_1, C_2, \ldots, C_m$
- Each constraint $C_i$ limits the values that a subset of variables can take, e.g., $V_1 \neq V_2$

In our example:
- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: D_i = {red, green, blue}
- Constraints: adjacent regions must have different colors.
  - E.g., WA ≠ NT (if the language allows this) or
  - [WA, NT] in [(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)]
Constraint satisfaction problems

- Simple example of a formal representation language
- CSP benefits
  - Standard representation language
  - Generic goal and successor functions
  - Useful general-purpose algorithms with more power than standard search algorithms, including generic heuristics
- Applications:
  - Time table problems (exam/teaching schedules)
  - Assignment problems (who teaches what)

Varieties of CSPs

- Discrete variables
  - Finite domains of size \( d \Rightarrow O(d^n) \) complete assignments.
  - The satisfiability problem: a Boolean CSP
  - Infinite domains (integers, strings, etc.)
  - Continuous variables
    - Linear constraints solvable in poly time by linear programming methods (dealt with in the field of operations research).
- Our focus: discrete variables and finite domains

Varieties of constraints

- Unary constraints involve a single variable.
  - e.g. \( SA \neq \text{green} \)
- Binary constraints involve pairs of variables.
  - e.g. \( SA \neq WA \)
- Global constraints involve an arbitrary number of variables.
  - Preference (soft constraints) e.g. red is better than green often representable by a cost for each variable assignment; not considered here.

Constraint graph

- Binary CSP: each constraint relates two variables
- Constraint graph: nodes are variables, edges are constraints

Example: cryptarithmetic puzzles

The constraints are represented by a hypergraph

CSP as a standard search problem

- Incremental formulation
  - Initial State: the empty assignment \( \{ \} \).
  - Successor: Assign value to unassigned variable provided there is no conflict.
  - Goal test: the current assignment is complete.
- Same formulation for all CSPs !!!
- Solution is found at depth \( n \) (\( n \) variables).
- What search method would you choose?
Backtracking search

- Observation: the order of assignment doesn’t matter
  ⇒ can consider assignment of a single variable at a time.
  Results in $d^d$ leaves ($d$: number of values per variable).
- Backtracking search: DFS for CSPs with single-variable assignments (backtracks when a variable has no value that can be assigned)
- The basic uninformed algorithm for CSP

Sudoku solving

Constraints:

```
<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
<td>G</td>
<td>H</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>9</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Can be translated into constraints between pairs of variables.

Let’s see if we can figure the value of the center grid point.

Let’s see if we can figure the value of the center grid point.

Constraint propagation

- Enforce local consistency
- Propagate the implications of each constraint

http://norvig.com/sudoku.html
Arc consistency

- X → Y is arc-consistent iff for every value x of X there is some allowed value y of Y.
- Example: X and Y can take on the values 0…9 with the constraint: Y = X^2. Can use arc consistency to reduce the domains of X and Y:
  - X → Y reduce X’s domain to {0,1,2,3}
  - Y → X reduce Y’s domain to {0,1,4,9}

Path Consistency

- Looks at triples of variables
  - The set (X_i, X_j) is path-consistent with respect to X_m if for every assignment consistent with the constraints of X_i, X_j, there is an assignment to X_m that satisfies the constraints on (X_i, X_m) and (X_m, X_j).
  - The PC-2 algorithm achieves path consistency.

K-consistency

- Stronger forms of propagation can be defined using the notion of k-consistency.
  - A CSP is k-consistent if for any set of k-1 variables and for any consistent assignment to those variables, a consistent value can always be assigned to any k-th variable.
  - Not practical!
Backtracking example

Improving backtracking efficiency

- General-purpose methods/heuristics can give huge gains in speed:
  - Which variable should be assigned next?
  - In what order should its values be tried?
  - Can we detect inevitable failure early?

Most constrained variable

Choose the variable with the fewest legal values
(a.k.a. minimum remaining values (MRV) or “fail first” heuristic)

- What is the intuition behind this choice?
Most constraining variable

- Select the variable that is involved in the largest number of constraints on other unassigned variables.
- Also called the degree heuristic because that variable has the largest degree in the constraint graph.
- Often used as a tie breaker e.g. in conjunction with MRV.

Least constraining value heuristic

- Guides the choice of which value to assign next.
- Given a variable, choose the least constraining value:
  - the one that rules out the fewest values in the remaining variables
  - why?

Forward checking

- Can we detect inevitable failure early?
  - And avoid it later?
- Forward checking: keep track of remaining legal values for unassigned variables.
- Terminate search direction when a variable has no legal values.

- Assign (WA=red)
  - Effects on other variables connected by constraints with WA
    - NT can no longer be red
    - SA can no longer be red

- Assign (Q=green)
  - Effects on other variables connected by constraints with WA
    - NT can no longer be green
    - NSW can no longer be green
    - SA can no longer be green

- If V is assigned blue
  - Effects on other variables connected by constraints with WA
    - SA is empty
    - NSW can no longer be blue
    - FC has detected that partial assignment is inconsistent with the constraints and backtracking can occur.
Example: 4-Queens Problem

October 13, 2014
Example: 4-Queens Problem

\[
\begin{array}{c}
1 & 2 & 3 & 4 \\
1 & & & \\
2 & & & \\
3 & & & \\
4 & & & \\
\end{array}
\]

Example: 4-Queens Problem

\[
\begin{array}{c}
1 & 2 & 3 & 4 \\
X1 & & & \\
X2 & & & \\
X3 & & & \\
X4 & & & \\
\end{array}
\]

Example: 4-Queens Problem

\[
\begin{array}{c}
1 & 2 & 3 & 4 \\
X1 & & & \\
X2 & & & \\
X3 & & & \\
X4 & & & \\
\end{array}
\]
Forward checking

- Solving CSPs with combination of heuristics plus forward checking is more efficient than either approach alone.
- FC does not provide early detection of all failures.
  - Once WA=red and Q=green: NT and SA cannot both be blue!
- MAC (maintaining arc consistency): calls AC-3 after assigning a value (but only deals with the neighbors of a node that has been assigned a value).

Local search for CSP

- Local search methods use a “complete” state representation, i.e., all variables assigned.
- To apply to CSPs
  - Allow states with unsatisfied constraints
  - Reassign variable values
- Select a variable: random conflicted variable
- Select a value: min-conflicts heuristic
  - Value that violates the fewest constraints
- Hill-climbing like algorithm with the objective function being the number of violated constraints
  - Works surprisingly well in problems like n-Queens

Min-Conflicts

```
function MIN-CONFLICTS(csp, max_steps) returns a solution or failure
inputs: csp, a constraint satisfaction problem
        max_steps, the number of steps allowed before giving up
current ← an initial complete assignment for csp
for i = 1 to max_steps do
    if current is a solution for csp then return current
    var ← a randomly chosen conflicted variable from csp.VARIABLES
    value ← the value v for var that minimizes CONFLICTS(var, v, current, csp)
    set var = value in current
return failure
```

Problem structure

- How can the problem structure help to find a solution quickly?
- Subproblem identification is important:
  - Coloring Tasmania and mainland are independent subproblems
  - Identifiable as connected components of constraint graph.
  - Improves performance

Tree-structured CSPs

- Theorem: if the constraint graph has no loops then CSP can be solved in \(O(n d^2)\) time
- Compare with general CSP, where worst case is \(O(d^n)\)
Any tree-structured CSP can be solved in time linear in the number of variables.

Function TREE-CSP-SOLVER(csp) returns a solution or failure
inputs: csp, a CSP with components X, D, C
n ← number of variables in X
assignment ← an empty assignment
root ← any variable in X
X ← TOPOLOGICALSORT(X, root)
for i = n down to 2 do
    MAKE-ARC-CONSISTENT(PARENT(X_i), X_i)
    if it cannot be made consistent then return failure
for i = 1 to n do
    assignment[X_i] ← any consistent value from D_i
    if there is no consistent value then return failure
return assignment

Nearly tree-structured CSPs

- Can more general constraint graphs be reduced to trees?
- Two approaches:
  - Remove certain nodes
  - Collapse certain nodes

- Idea: assign values to some variables so that the remaining variables form a tree.
- Assign (SA=x) -- cycle cutset
  - Remove any values from the other variables that are inconsistent.
  - The selected value for SA could be wrong: have to try all of them

- This approach is effective if cycle cutset is small.
- Finding the smallest cycle cutset is NP-hard
  - Approximation algorithms exist
  - This approach is called cutset conditioning.
Interim class summary

- We have been studying ways for agents to solve problems.
- Search
  - Uninformed search
    - Easy solution for simple problems
    - Basis for more sophisticated solutions
  - Informed search
    - Information = problem solving power
- Adversarial search
  - αβ-search for play against optimal opponent
  - Early cut-off when necessary
- Constraint satisfaction problems
- What’s next?
  - Logic
  - Machine learning