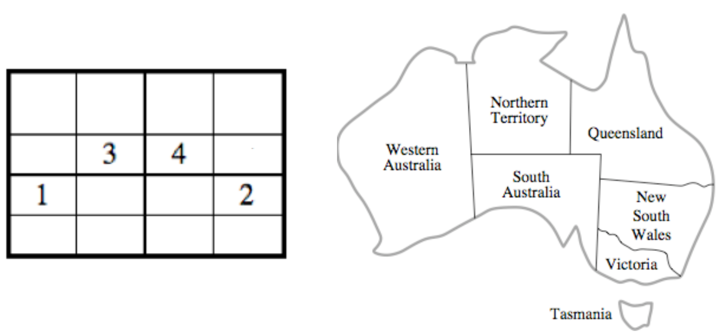
COMP 102: Excursions in Computer Science
Lecture 20: Constraint Satisfaction Problems

Instructor: Joelle Pineau (jpineau@cs.mcgill.ca)

Class web page: www.cs.mcgill.ca/~jpineau/comp102

Similar problems?

What do colouring maps and solving Sudokus have in common?



Both are instances of constraint satisfaction problems.

Today's plan

- What is a Constraint Satisfaction Problem (CSP)?
- Common examples of CSPs
- How can we solve CSPs?
  - A constructive approach.
  - An iterative approach.

Constraint satisfaction problems (CSPs)

- A CSP is defined by:
  - Set of variables V_i, that can take values from domain D_i
  - Set of constraints specifying what combinations of values are allowed (for subsets of variables)
  - Constraints can be represented:
    - Explicitly, as a list of allowable values (E.g. C_1=red)
    - Implicitly, in terms of other variables (E.g. $C_1=C_2$)
- A CSP solution is an assignment of values to variables such that all the constraints are true.
  - Want to find *any solution* or find that there is *no solution*.

Example: Map coloring

- Color a map so that no adjacent countries have the same color, using only the colors Red, Blue and Green.
  - Variables:
  - Domains:
  - Constraints:

- How should we solve this problem? Is it tractable?

Example: Sudoku puzzle

Rule: Each number {1, 2, 3, 4} must appear once (and only once) in every row, in every column, and in every 2x2 square.

	3	4	
1			2

- Variables:
- Domains:
- Constraints:

How should we solve this problem? Is it tractable?

Example: Satisfying boolean expression

- Find an assignment (*True* or *False*) for each variable $x_1, x_2, \ldots, x_n$ such that the boolean expression evaluates to **True**.

  E.g. Boolean expression =

  $(x_3 \text{ OR } x_2 \text{ OR } x_4) \text{ AND } (x_5 \text{ OR } (\text{NOT } x_2) \text{ OR } x_1) \text{ AND } (x_4 \text{ OR } x_5 \text{ OR } x_3)$

  – Variables:
  – Domains:
  – Constraints:

  How should we solve this problem? Is it tractable?

Varieties of variables

- Boolean variables (e.g. satisfiability)
- Finite domain, discrete variables (e.g. colouring)
- Infinite domain, discrete variables (e.g. start/end of operation in scheduling)
- Continuous variables.

Problems range from solvable in poly-time (using linear programming) to NP-complete to undecidable

Varieties of constraints

- Unary: involve one variable and one constraint.
- Binary.
- Higher-order (involve 3 or more variables)

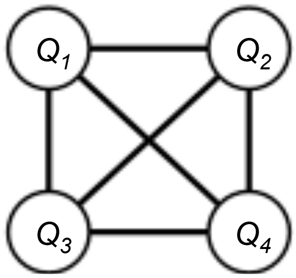
- Preferences (soft constraints): can be represented using costs and lead to constrained optimization problems.

Real-world CSPs

- Assignment problem (e.g. who teaches what class)
- Timetable problems (e.g. which class is offered when and where)
- Hardware configuration
- Transportation scheduling
- Factory scheduling
- Floor planning

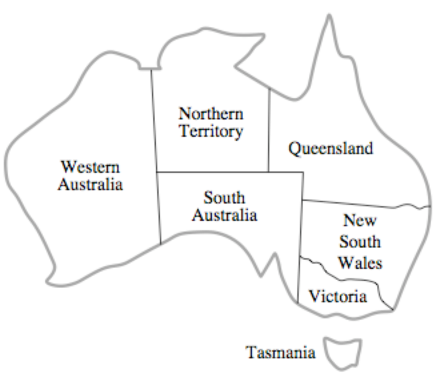
Constraint graph

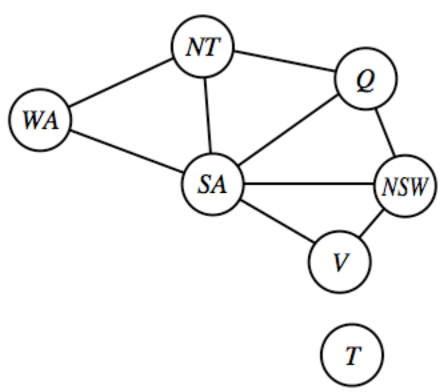
- Binary CSP: each constraint relates at most two variables.
- Constraint graph: nodes are variables, arcs show constraints.



- The structure of the graph can be exploited to provide problem solutions.

Constraint graph example





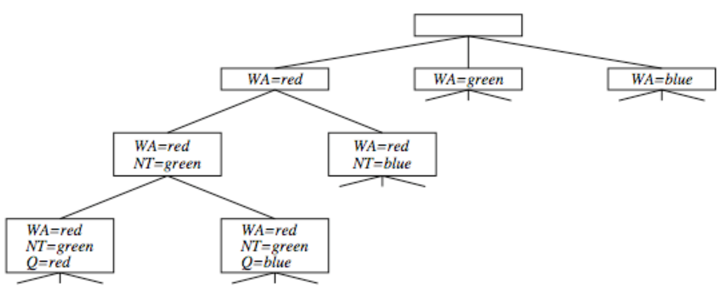
Applying standard search

- Assume a constructive approach. Start by defining:
  - States: defined by the set of values assigned so far.
  - An initial state: all variables are unassigned.
  - Operators: assign a value to an unassigned variable.
  - Goal test: all variables assigned, no constraint violated.

- Build a search tree, continue until you find a path to the goal.

This is a general purpose algorithm that works for all CSPs!

Standard search applied to map coloring



- Is this a practical approach?

Analysis of the simple approach

- Maximum search depth = number of variables
  - Each variable has to get a value.
- Number of branches per node in the tree = $\Sigma_i \mid D_i \mid$, where D_i is the size of the domain for the i^{th} variable.

  This can be a big search! Often requires lots of backtracking!

BUT: Here are a few useful observations

- Order in which variables are assigned is irrelevant -> Many paths are equivalent!
- Adding assignments cannot correct a violated constraint!

Heuristics for CSPs

- What is a heuristic?
  - A simple guide that helps in solving a hard problem.
- How does this help us solve CSPs?
  - It guides our choice of:
    - which value to choose for which variable.
    - which variable to assign next.

Heuristics for CSPs

E.g. Map coloring

- Say WA = **red**, NT = **blue**
- Choose which variable next?
- Choose **SA**

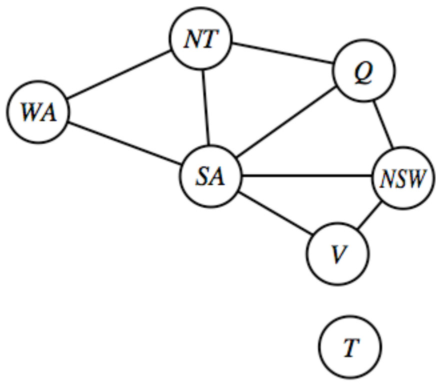
(most constrained variable!)

- Assign which value?
- Let **SA** = **green**

(least constraining value!)

- What next?
- Choose Q.
- Let Q = **red**

Etc.



Summary of heuristics for CFP

- Most Constrained Variable

  Choose the variable which has the least possible number of choices of value.

- Least Constraining Value

  Assign the value which leaves the greatest number of choices for other variables.

Note: For both of these heuristics, it is useful to keep track of the possible choices of value at each variable.

Another way to solve CSPs

Iterative improvement method:

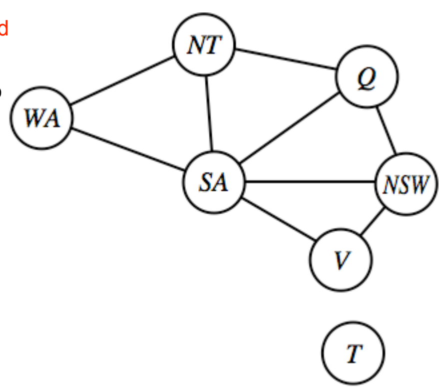
- Start with a broken but complete assignment of values to variables.
  - Broken = some variables may be assigned values that don't satisfy some constraints.
  - Complete = each variable is assigned a value.
- Repeat until all constraints are satisfied:
  - Pick a broken constraint.
  - Randomly select one of the variables involved in this constraint.
  - Re-assign the value of that variable using the Min-conflicts heuristic.
    - Min-conflicts heuristic = choose value that violates the fewest constraints.

Iterative improvement example

E.g. Map coloring

- Let WA=red, NT=blue, Q=red, SA=blue, NSW=red, V=blue, T=red
- Where are the conflicts? (Useful to look at the constraint graph for this.)

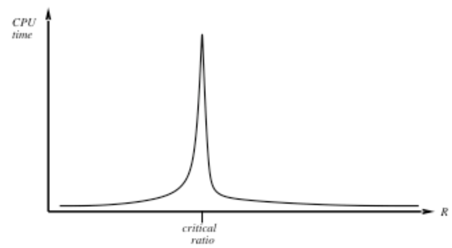
  NT≠SA, Q≠NSW, SA≠V
- How can we apply the min-conflict heuristic to resolve those?
- Randomly pick constraint NT≠SA. Randomly pick variable NT. Change it to green to satisfy min-conflict heuristic.



Performance of min-conflicts heuristic

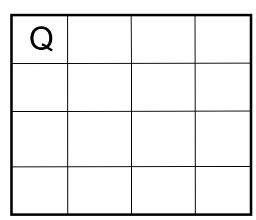
- Given random initial state, works very well for many large CSP problems (almost constant time).
- This holds true for any randomly-generated CSP <u>except</u> in a narrow range of the ratio:

$$R = \frac{\text{number of constraints}}{\text{number of variables}}$$



N-Queens problem

- Place N queens on an NxN chess board so that no queen can attack another.
  - Variables?
  - Domain?
  - Constraints?



- Assume we try a <u>constructive</u> approach:
  - What variable should we select next?
  - What value should we assign it?
  - What next?

N-Queens problem

- Place N queens on an NxN chess board so that no queen can attack another.

Q			

- Assume we try an <u>iterative</u> approach:
  - First randomly fill in all the missing values.
  - What constraint should we pick?
  - What variable should we work on?
  - How should we fix it?

Take-home message

- CSPs are everywhere!
- CSPs can be solved using either <u>constructive methods</u> or <u>iterative improvement methods</u>.
- Heuristics are useful guides to focus the search. You should understand the basic heuristics.
- Iterative improvement methods with min-conflicts heuristic are very general, and often work best.