COMP 102: Computers and Computing

Lecture 19: Constraint Satisfaction Problems

Instructor: Kaleem Siddiqi (siddiqi@cim.mcgill.ca)

Class web page: www.cim.mcgill.ca/~siddiqi/102.html

Overview

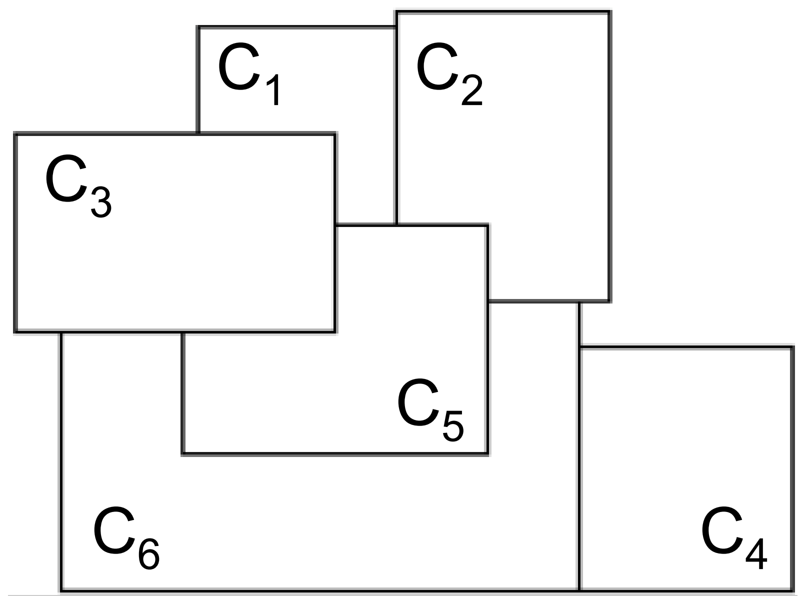
- What is a Constraint Satisfaction Problem (CSP)?
- Common examples of CSPs
- How can we solve CSPs?
  - A constructive approach.
  - An iterative approach.

Example #1: Map coloring

- Color a map so that no adjacent countries have the same color.

How should we do this?

What do we know of this problem?



Example #2: 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 =

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

How should we solve this problem? Is it tractable?

Example #3: Sudoku puzzle

- A simple puzzle:

	3	4	
1			2

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

How should we solve this problem? Is it tractable?

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: Sudoku puzzle

- A simple puzzle:

	3	4	
1			2

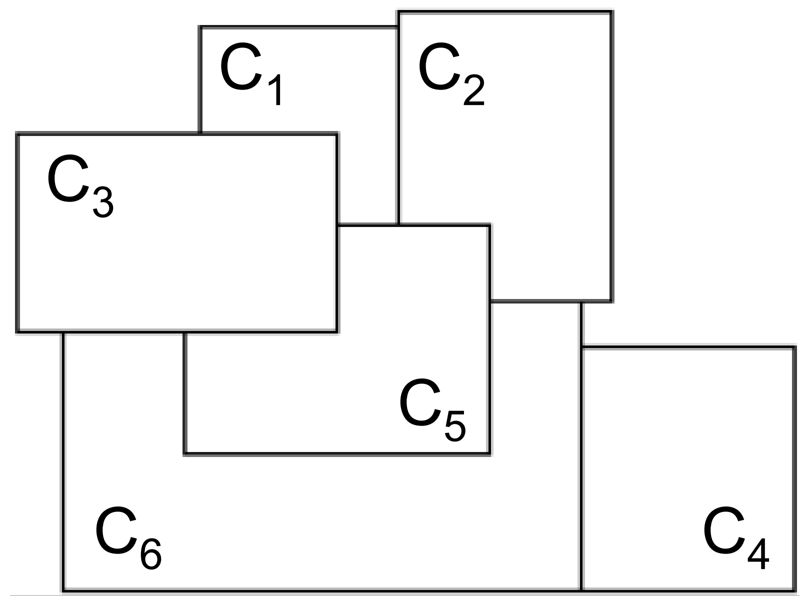
  – Variables:

  – Domains:

  – Constraints:

Example: Map coloring

- Color a map so that no adjacent countries have the same color.
  - Variables:
  - Domains:
  - Constraints:



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)

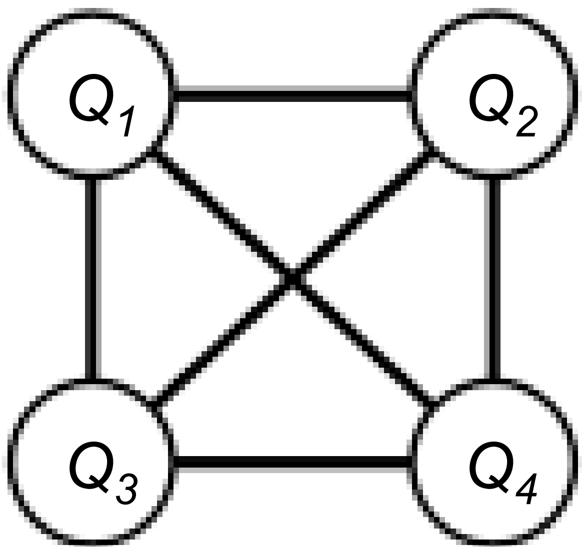
- Preferences (soft constraints): can be represented using costs and lead to constrained optimization problems.

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
- Office space allocation

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.

Applying standard search

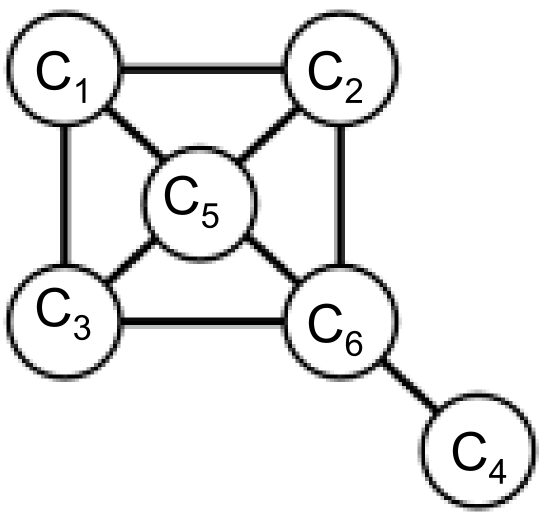
- Assume a constructive approach:
  - State: defined by set of values assigned so far.
  - Initial state: all variables are unassigned.
  - Operators: assign a value to an unassigned variable.
  - Goal test: all variables assigned, no constraint violated.

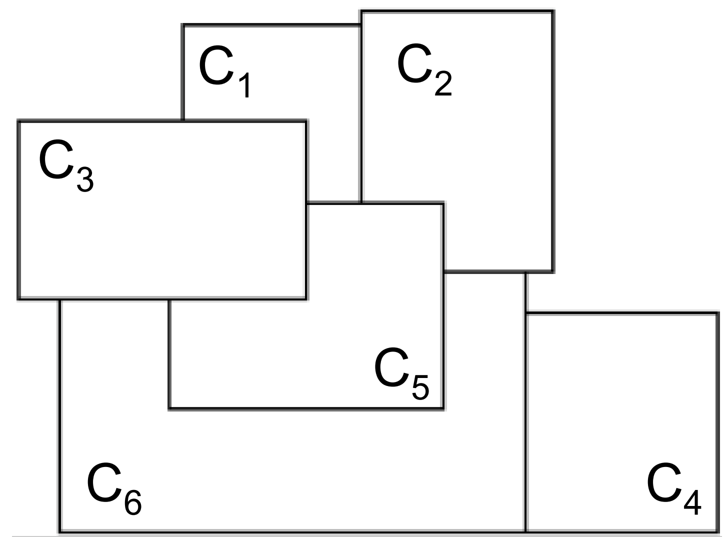
- Build a <u>search tree</u>, continue until you find a path to the goal.

- This is a general purpose algorithm which works for all CSPs!

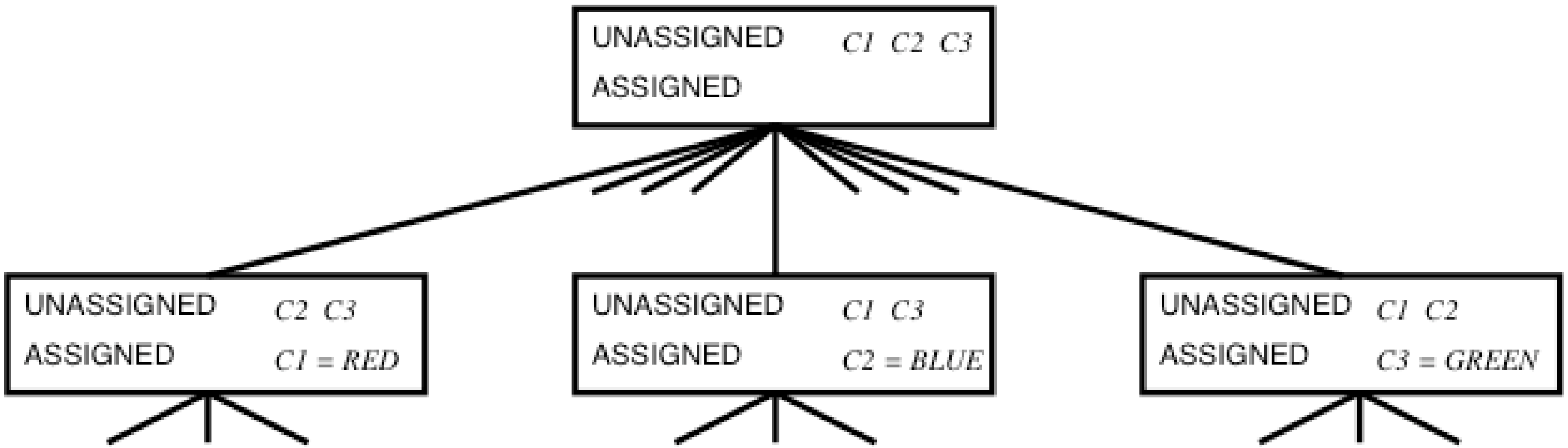
Example: map coloring

- Color a map so that no adjacent countries have the same color.
  - Variables: Countries C_i
  - Domains: {Red, Blue, Green}
  - Constraints: $\{C_1 \neq C_2, C_1 \neq C_5, \ldots\}$
- <u>Constraint graph:</u>





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 in the tree = $\sum_i |D_i|$

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 C_1 = **red**, C_2 = **blue**
- Choose which variable next?
- Choose **C_5**

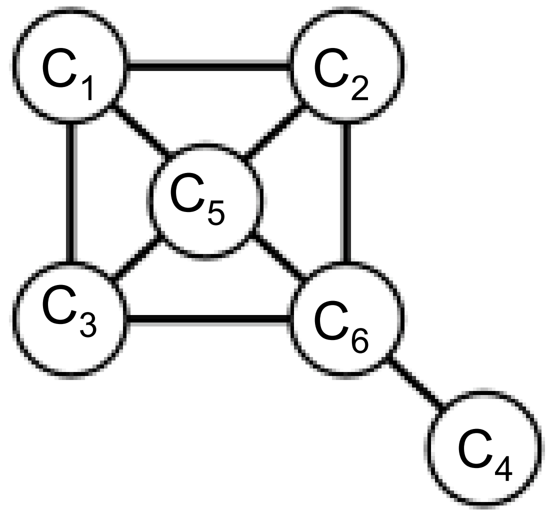
(most constrained variable!)

- Assign which value?
- Let C_5 = **green**

(least constraining value!)

- Choose C_3 (alternately, could also choose C_6).
- Let C_3 = **blue**

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.

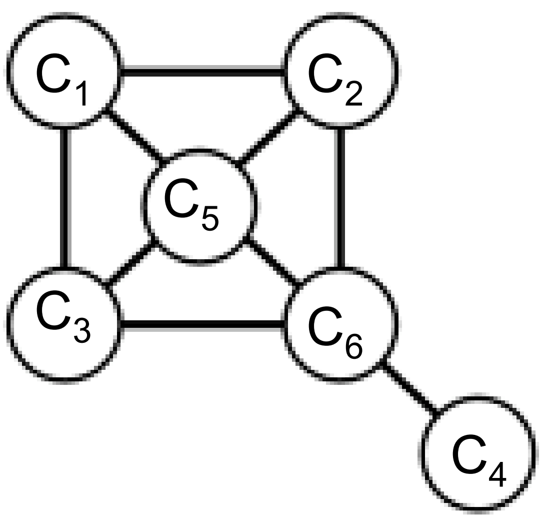
Iterative improvement example

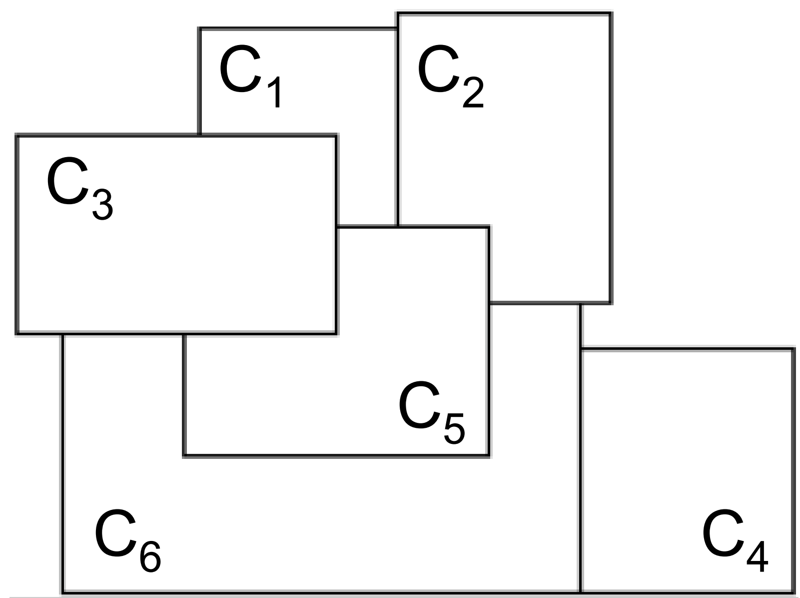
E.g. Map coloring

- Let C_1=red, C_2=green, C_3=blue, C_4=red, C_5=green, C_6=red
- Where are the conflicts? (Useful to look at the constraint graph for this.)

  $C_2 \neq C_5$ $C_4 \neq C_6$

- How can we apply the min-conflict heuristic to resolve those?

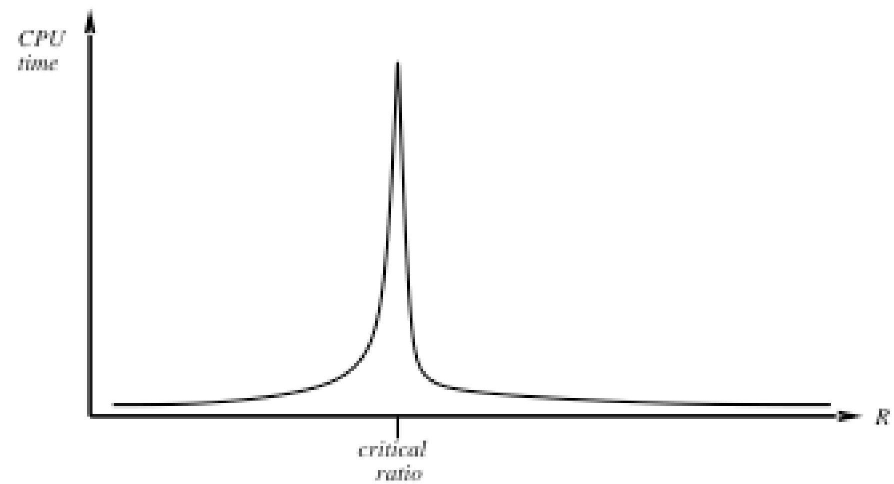




Performance of min-conflicts heuristic

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

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



Solving our Sudoku puzzle

	3	4	
1			2

Assume we try a <u>constructive</u> approach:

- What variable should we select next?
- What value should we assign it?
- What next?

Now you know how to become a Sudoku master!

The Pentagon Problem

- Imagine that you have a 5-sided polygon where each side has a node in its middle. There is also a node at each corner.
- Therefore, there are 10 nodes in all.
- Can you place the digits 0,1,2,3,4,5,6,7,8,9 in the nodes, with each digit appearing exactly once, and only once, such that the numbers on each side add up to 13?
- How about the same problem, but with the numbers totaling to 11?
- This is an example of a problem where SCP ideas could be applied.

Take-home message

- CSPs are everywhere!
- CSPs can be solved using either constructive methods or iterative improvement methods.
- 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.