

Constraint Satisfaction Problems

Chapter 6 AIMA 3rd Edition

4 Feb 2004CS 3243 - Constraint Satisfaction1

Outline

- Constraint Satisfaction Problems (CSP)
- Backtracking search for CSPs
- Local search for CSPs


4 Feb 2004CS 3243 - Constraint Satisfaction2

Constraint satisfaction problems (CSPs)

- **Standard search problem:**
 - **state** is a "black box" – any data structure that supports successor function, heuristic function, and goal test
- **CSP:**
 - **state** is defined by **variables X_i** , with **values** from **domain D_i**
 - **goal test** is a set of **constraints** specifying allowable combinations of values for subsets of variables
- Simple example of a **formal representation language**
- Allows useful **general-purpose** algorithms with more power than standard search algorithms

4 Feb 2004CS 3243 - Constraint Satisfaction3

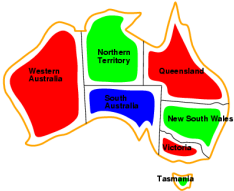
Example: Map-Coloring



- **Variables** WA, NT, Q, NSW, V, SA, T
- **Domains** $D_i = \{red, green, blue\}$
- **Constraints:** adjacent regions must have different colors
- e.g., $WA \neq NT$, or $(WA, NT) \in \{(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)\}$

4 Feb 2004CS 3243 - Constraint Satisfaction4

Example: Map-Coloring

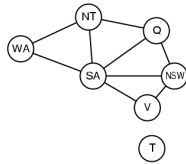


- **Solutions** are **complete** and **consistent** assignments, e.g., $WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = green$

4 Feb 2004CS 3243 - Constraint Satisfaction5

Constraint graph

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



4 Feb 2004CS 3243 - Constraint Satisfaction6

Varieties of CSPs

- Discrete variables
 - finite domains:
 - n variables, domain size $d \rightarrow O(d^n)$ complete assignments
 - e.g., Boolean CSPs, incl. \sim Boolean satisfiability (NP-complete)
 - infinite domains:
 - integers, strings, etc.
 - e.g., job scheduling, variables are start/end days for each job
 - need a constraint language, e.g., $StartJob_1 + 5 \leq StartJob_2$
- Continuous variables
 - e.g., start/end times for Hubble Space Telescope observations
 - linear constraints solvable in polynomial time by linear programming

4 Feb 2004 CS 3243 - Constraint Satisfaction 7

Varieties of constraints

- Unary constraints involve a single variable,
 - e.g., $SA \neq \text{green}$
- Binary constraints involve pairs of variables,
 - e.g., $SA \neq WA$
- N-ary constraints involve n variables,

4 Feb 2004 CS 3243 - Constraint Satisfaction 8

Real-world CSPs

- Assignment problems (e.g., who teaches what class)
- Timetabling problems (e.g., which class, when, where?)
- Transportation scheduling
- Factory scheduling

4 Feb 2004 CS 3243 - Constraint Satisfaction 9

Standard search formulation (incremental)

States are defined by the values assigned so far

- Initial state: the empty assignment $\{ \}$
- Successor function: assign a value to an unassigned variable that does not conflict with current assignment
 - \rightarrow fail if no legal assignments
- Goal test: the current assignment is complete

- This is the same for all CSPs
- Every solution appears at depth n with n variables
 - \rightarrow use depth-first search

4 Feb 2004 CS 3243 - Constraint Satisfaction 10

Backtracking search

- Variable assignments are **commutative**, i.e., $[WA = \text{red then } NT = \text{green}]$ same as $[NT = \text{green then } WA = \text{red}]$
- Only need to consider assignments to a single variable at each node
 - $\rightarrow b = d$ and there are d^n leaves
- Depth-first search for CSPs with single-variable assignments is called **backtracking search**
- Backtracking search is the basic uninformed algorithm for CSPs
- Can solve n -queens for $n \approx 25$

4 Feb 2004 CS 3243 - Constraint Satisfaction 11

Backtracking search


```

function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING( $\{ \}$ , csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment according to Constraints[csp] then
      add { var = value } to assignment
      result  $\leftarrow$  RECURSIVE-BACKTRACKING(assignment, csp)
      if result  $\neq$  failure then return result
      remove { var = value } from assignment
  return failure
  
```


4 Feb 2004 CS 3243 - Constraint Satisfaction 12

Backtracking example



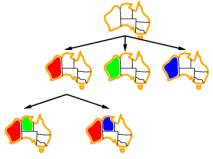
4 Feb 2004 CS 3243 - Constraint Satisfaction 13

Backtracking example



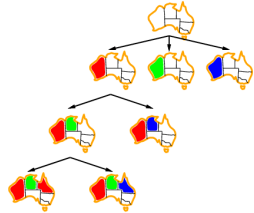
4 Feb 2004 CS 3243 - Constraint Satisfaction 14

Backtracking example



4 Feb 2004 CS 3243 - Constraint Satisfaction 15

Backtracking example



4 Feb 2004 CS 3243 - Constraint Satisfaction 16


Improving backtracking efficiency

- **General-purpose** methods 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?

4 Feb 2004 CS 3243 - Constraint Satisfaction 17

Most constrained variable

- **Most constrained variable:** choose the variable with the fewest legal values



- **a.k.a. minimum remaining values (MRV) heuristic**

4 Feb 2004 CS 3243 - Constraint Satisfaction 18

Most constraining variable

- Tie-breaker among most constrained variables
- Most constraining variable:
 - choose the variable with the most

4 Feb 2004 CS 3243 - Constraint Satisfaction 19

Least constraining value

- Given a variable, choose the least constraining value:
 - the one that rules out the fewest values in the remaining variables

- ... heuristics make "1000 queens" feasible

4 Feb 2004 CS 3243 - Constraint Satisfaction 20

Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables
 - Terminate search when any variable has no legal values

4 Feb 2004 CS 3243 - Constraint Satisfaction 21

Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables
 - Terminate search when any variable has no legal values

4 Feb 2004 CS 3243 - Constraint Satisfaction 22

Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables
 - Terminate search when any variable has no legal values

4 Feb 2004 CS 3243 - Constraint Satisfaction 23

Forward checking

- Idea:
 - Keep track of remaining legal values for unassigned variables
 - Terminate search when any variable has no legal values

4 Feb 2004 CS 3243 - Constraint Satisfaction 24

Constraint propagation

- Forward checking propagates information from assigned to unassigned variables, but doesn't provide early detection for all failures:

WA	NT	Q	NSW	V	SA	T
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue

- NT and SA cannot both be blue!
- Constraint propagation repeatedly enforces constraints locally

4 Feb 2004 CS 3243 - Constraint Satisfaction 25

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff

for every value x of X there is some allowed y

WA	NT	Q	NSW	V	SA	T
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue

4 Feb 2004 CS 3243 - Constraint Satisfaction 26

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff

for every value x of X there is some allowed y

WA	NT	Q	NSW	V	SA	T
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue

4 Feb 2004 CS 3243 - Constraint Satisfaction 27

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff

for every

WA	NT	Q	NSW	V	SA	T
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue

- If X loses a value, neighbors of X need to be rechecked

4 Feb 2004 CS 3243 - Constraint Satisfaction 28

Arc consistency

- Simplest form of propagation makes each arc consistent
- $X \rightarrow Y$ is consistent iff

for every value x of X there is some allowed v

WA	NT	Q	NSW	V	SA	T
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue
Red	Blue	Green	Blue	Blue	Blue	Blue

- If X loses a value, neighbors of X need to be rechecked
- Arc consistency detects failure earlier than forward checking

4 Feb 2004 CS 3243 - Constraint Satisfaction 29

Arc consistency algorithm AC-3

```

function AC-3(csp) returns the CSP, possibly with reduced domains
inputs: csp, a binary CSP with variables  $\{X_1, X_2, \dots, X_n\}$ 
local variables: queue, a queue of arcs, initially all the arcs in csp
while queue is not empty do
   $(X_i, X_j) \leftarrow \text{REMOVE-FIRST}(\text{queue})$ 
  if RM-INCONSISTENT-VALUES( $X_i, X_j$ ) then
    for each  $X_k$  in NEIGHBORS[ $X_i$ ] do
      add  $(X_k, X_i)$  to queue

function RM-INCONSISTENT-VALUES( $X_i, X_j$ ) returns true iff remove a value
removed ← false
for each  $x$  in DOMAIN[ $X_i$ ] do
  if no value  $y$  in DOMAIN[ $X_j$ ] allows  $(x, y)$  to satisfy constraint( $X_i, X_j$ )
  then delete  $x$  from DOMAIN[ $X_i$ ]; removed ← true
return removed
    
```

- Time complexity: $O(n^2d^3)$

4 Feb 2004 CS 3243 - Constraint Satisfaction 30

Local search for CSPs

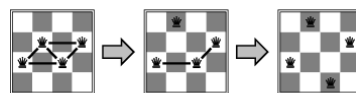
- Hill-climbing, simulated annealing typically work with "complete" states, i.e., all variables assigned
- To apply to CSPs:
 - allow states with unsatisfied constraints
 - operators **reassign** variable values
- Variable selection: randomly select any conflicted variable

4 Feb 2004 CS 3243 - Constraint Satisfaction 31

Value selection by **min-conflicts** heuristic

Example: 4-Queens

- States: 4 queens in 4 columns ($4^4 = 256$ states)
- Actions: move queen in column
- Goal test: no attacks
- Evaluation: $h(n) =$ number of attacks



- Given random initial state, can solve n queens in almost constant time for arbitrary n with high probability (e.g., $n = 10,000,000$)

4 Feb 2004 CS 3243 - Constraint Satisfaction 32

Summary

- CSPs are a special kind of problem:
 - states defined by values of a fixed set of variables
 - goal test defined by constraints on variable values
- Backtracking = depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that guarantee later failure
- Constraint propagation (e.g., arc consistency) does additional work to constrain values and detect inconsistencies
- Iterative min-conflicts is usually effective in practice

4 Feb 2004 CS 3243 - Constraint Satisfaction 33