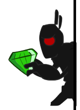
CSE 473: Artificial Intelligence
Winter 2017

Constraint Satisfaction Problems - Part 1 of 2

Steve Tanimoto



With slides from :
Dieter Fox, Dan Weld, Dan Klein, Stuart Russell, Andrew Moore, Luke Zettlemoyer

1

Previously

- Formulating problems as search
- Blind search algorithms
  - Depth first
  - Breadth first (uniform cost)
  - Iterative deepening
- Heuristic Search
  - Best first
    - Beam (Hill climbing)
    - A*
    - IDA*
- Heuristic generation
  - Exact soln to a relaxed problem
  - Pattern databases
- Local Search
  - Hill climbing, random moves, random restarts, simulated annealing

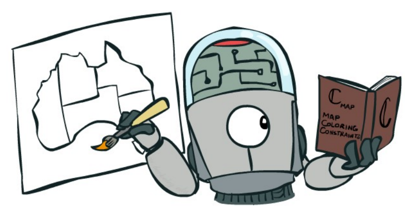
2

What is Search For?

- **Planning:** sequences of actions
  - The ***path to the goal*** is the important thing
  - Paths have various costs, depths
  - Assume little about problem structure
- **Identification:** assignments to variables
  - The ***goal itself*** is important, ***not the path***
  - All paths at the same depth (for some formulations)

3

Constraint Satisfaction Problems

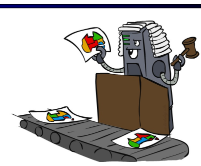


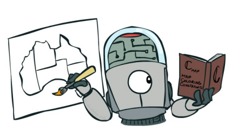
CSPs are *structured* (factored) identification problems

4

Constraint Satisfaction Problems

- Standard search problems:
  - State is a "black box": arbitrary data structure
  - Goal test can be any function over states
  - Successor function can also be anything
- Constraint satisfaction problems (CSPs):
  - A special subset of search problems
  - State is defined by variables X_i with values from a domain D (sometimes D depends on i)
  - Goal test is a set of constraints specifying allowable combinations of values for subsets of variables
- Making use of CSP formulation allows for optimized algorithms
  - Typical example of trading generality for utility (in this case, speed)



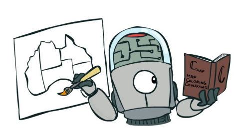


5

Constraint Satisfaction Problems

- "Factoring" the state space
- Representing the state space in a knowledge representation

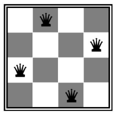
- Constraint satisfaction problems (CSPs):
  - A special subset of search problems
  - State is defined by variables X_i with values from a domain D (sometimes D depends on i)
  - Goal test is a set of constraints specifying allowable combinations of values for subsets of variables

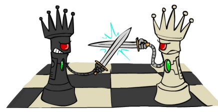


6

CSP Example: N-Queens

Is there a queen at X_{ij}?





- Formulation 1:
  - Variables: X_{ij}
  - Domains: $\{0, 1\}$
  - Constraints

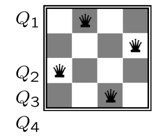
$$\forall i,j,k \ \ (X_{ij}, X_{ik}) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall i,j,k \ \ (X_{ij}, X_{kj}) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall i,j,k \ \ (X_{ij}, X_{i+k,j+k}) \in \{(0,0),(0,1),(1,0)\}$$
$$\forall i,j,k \ \ (X_{ij}, X_{i+k,j-k}) \in \{(0,0),(0,1),(1,0)\}$$

$$\sum_{i,j} X_{ij} = N$$

7

CSP Example: N-Queens

What column is the queen on for row k?



- Formulation 2:
  - Variables: Q_k
  - Domains: $\{1, 2, 3, \ldots N\}$
  - Constraints:

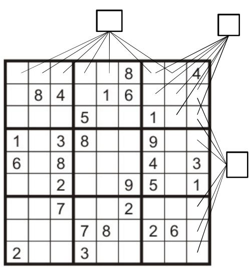
Implicit: $\forall i,j$ non-threatening(Q_i, Q_j)

Explicit: $(Q_1, Q_2) \in \{(1,3),(1,4),\ldots\}$

...

8

CSP Example: Sudoku



- Variables:
  - Each (open) square
- Domains:
  - {1,2,…,9}
- Constraints:

9-way alldiff for each column

9-way alldiff for each row

9-way alldiff for each region

(or can have a bunch of pairwise inequality constraints)

9

Propositional Logic

$$((p \leftrightarrow q) \land r) \lor (p \land q \land \sim r)$$

- Variables: propositional variables
- Domains: {T, F}
- Constraints: logical formula

10

CSP Example: Map Coloring

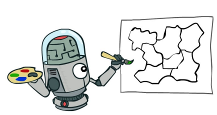
- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: $D = \{\text{red}, \text{green}, \text{blue}\}$
- Constraints: adjacent regions must have different colors

Implicit: $\text{WA} \neq \text{NT}$

Explicit: $(\text{WA}, \text{NT}) \in \{(\text{red}, \text{green}), (\text{red}, \text{blue}), \ldots\}$

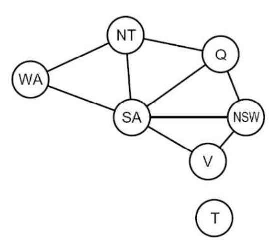
- Solutions are assignments satisfying all constraints, e.g.:

{WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}



11

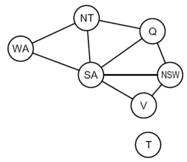
Constraint Graphs



12

Constraint Graphs

- Binary CSP: each constraint relates (at most) two variables
- Binary constraint graph: nodes are variables, arcs show constraints
- General-purpose CSP algorithms use the graph structure to speed up search. E.g., Tasmania is an independent subproblem!



13

Example: Cryptarithmetic

- Variables:

  $F\ T\ U\ W\ R\ O\ X_1\ X_2\ X_3$
- Domains:

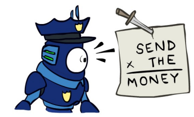
  $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Constraints:

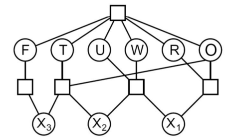
  alldiff(F, T, U, W, R, O)

  $O + O = R + 10 \cdot X_1$

  $\cdots$

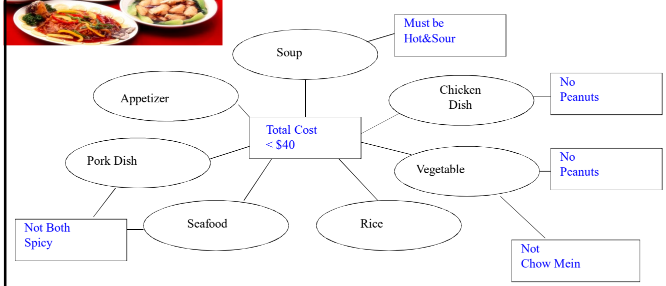
```
  T W O
+ T W O
F O U R
```





15

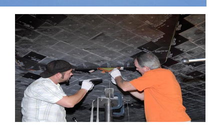
Chinese Constraint Network



16

Real-World CSPs

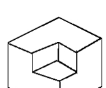
- Assignment problems: e.g., who teaches what class
- Timetabling problems: e.g., which class is offered when and where?
- Hardware configuration
- Gate assignment in airports
- Space Shuttle Repair
- Transportation scheduling
- Factory scheduling
- … lots more!

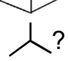


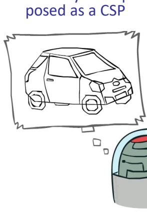
17

Example: The Waltz Algorithm

- The Waltz algorithm is for interpreting line drawings of solid polyhedra as 3D objects
- An early example of an AI computation posed as a CSP



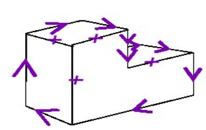




19

Waltz on Simple Scenes

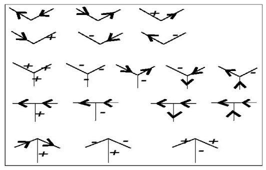
- Assume all objects:
  - Have no shadows or cracks
  - Three-faced vertices
  - "General position": no junctions change with small movements of the eye.
- Then each line on image is one of the following:
  - Boundary line (edge of an object) (>) with right hand of arrow denoting "solid" and left hand denoting "space"
  - Interior convex edge (+)
  - Interior concave edge (-)

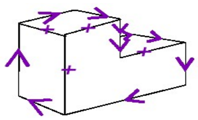


21

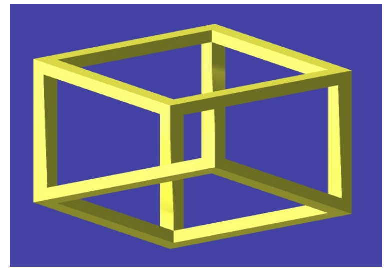
Legal Junctions

- Only certain junctions are physically possible
- How can we formulate a CSP to label an image?
- **Variables:** edges
- **Domains:** >, <, +, -
- **Constraints:** legal junction types





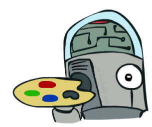
Slight Problem: Local *vs* Global Consistency



Varieties of CSPs

Varieties of CSP Variables

- Discrete Variables
  - Finite domains
    - Size d means $O(d^n)$ complete assignments
    - E.g., Boolean CSPs, including Boolean satisfiability (NP-complete)
  - Infinite domains (integers, strings, etc.)
    - E.g., job scheduling, variables are start/end times for each job
    - Linear constraints solvable, nonlinear undecidable



- Continuous variables
  - E.g., start/end times for Hubble Telescope observations
  - Linear constraints solvable in polynomial time by linear program methods (see CSE 521 for a bit of LP theory)

Varieties of CSP Constraints

- Varieties of Constraints
  - Unary constraints involve a single variable (equivalent to reducing domains), e.g.:

    $$SA \neq green$$

  - Binary constraints involve pairs of variables, e.g.:

    $$SA \neq WA$$

  - Higher-order constraints involve 3 or more variables: e.g., cryptarithmetic column constraints

- Preferences (soft constraints):
  - E.g., red is better than green
  - Often representable by a cost for each variable assignment
  - Gives constrained optimization problems
  - (We'll ignore these until we get to Bayes' nets)

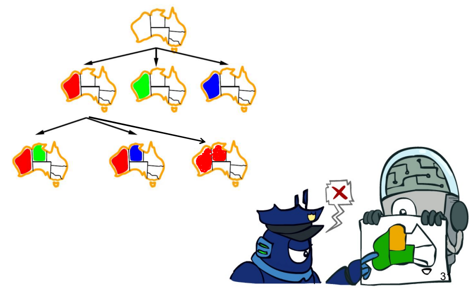
Solving CSPs

CSP as Search

- States
- Operators
- Initial State
- Goal State

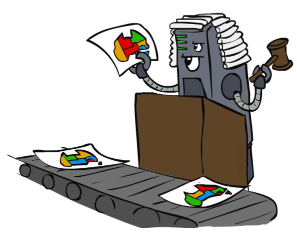
29

Standard Depth First Search



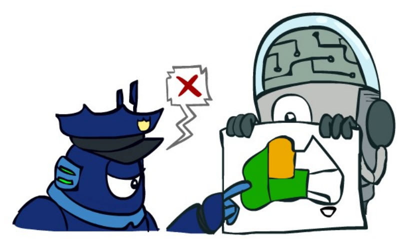
Standard Search Formulation

- Standard search formulation of CSPs
- States defined by the values assigned so far (partial assignments)
  - Initial state: the empty assignment, {}
  - Successor function: assign a value to an unassigned variable
  - **Goal test: the current assignment is complete and satisfies all constraints**
- We'll start with the straightforward, naïve approach, then improve it



32

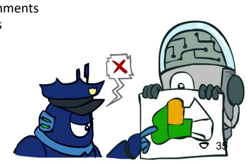
Backtracking Search



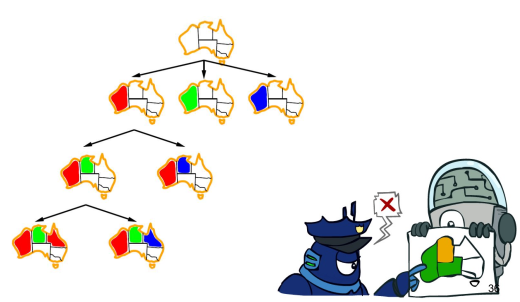
34

Backtracking Search

- Backtracking search is the basic uninformed algorithm for solving CSPs
- Idea 1: One variable at a time
  - Variable assignments are commutative, so fix ordering
  - I.e., [WA = red then NT = green] same as [NT = green then WA = red]
  - Only need to consider assignments to a single variable at each step
- Idea 2: Check constraints as you go
  - I.e. consider only values which do not conflict previous assignments
  - Might have to do some computation to check the constraints
  - "Incremental goal test"
- Depth-first search with these two improvements is called *backtracking search*
- Can solve n-queens for n ≈ 25



Backtracking Example



Backtracking Search

```
function BACKTRACKING-SEARCH(csp) returns solution/failure
    return RECURSIVE-BACKTRACKING({ }, csp)

function RECURSIVE-BACKTRACKING(assignment, csp) returns soln/failure
    if assignment is complete then return assignment
    var ← SELECT-UNASSIGNED-VARIABLE(VARIABLES[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment given CONSTRAINTS[csp] then
            add {var = value} to assignment
            result ← RECURSIVE-BACKTRACKING(assignment, csp)
            if result ≠ failure then return result
            remove {var = value} from assignment
    return failure
```

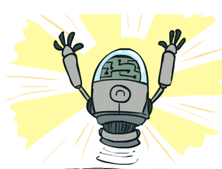
- What are the choice points?

[Demo: coloring -- backtracking]

Backtracking Search

- Kind of depth first search
- Is it ***complete***?

Improving Backtracking



- General-purpose ideas give huge gains in speed
- Ordering:
  - Which variable should be assigned next?
  - In what order should its values be tried?
- Filtering: Can we detect inevitable failure early?
- Structure: Can we exploit the problem structure?

Next: Constraint Satisfaction Problems - Part 2