

## CSE 473: Artificial Intelligence Autumn 2018

### 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

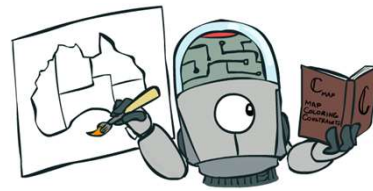
## What is Search For?

- **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  $X_{ij}$ ?

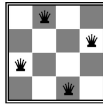
### Formulation 1:

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

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

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

7



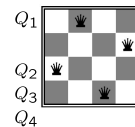
## CSP Example: N-Queens

### Formulation 2:

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

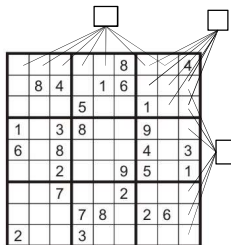
Implicit:  $\forall i, j \quad \text{non-threatening}(Q_i, Q_j)$

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



8

## CSP Example: Sudoku



- Variables:
  - Each (open) square
- Domains:
  - $\{1, 2, \dots, 9\}$
- 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

$$((p \leftrightarrow q) \wedge r) \vee (p \wedge q \wedge \sim r)$$

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

10

## CSP Example: Map Coloring

- Variables: WA, NT, Q, NSW, V, SA, T

- Domains:  $D = \{\text{red, green, blue}\}$

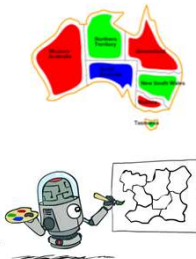
- Constraints: adjacent regions must have different colors

Implicit:  $WA \neq NT$

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

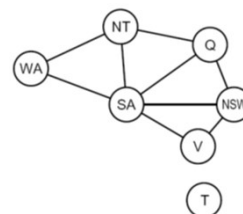
- Solutions are assignments satisfying all constraints, e.g.:

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



11

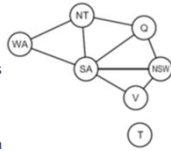
## Constraint Graphs



12

## Constraint Graphs

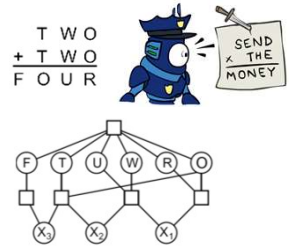
- Binary CSP: each constraint relates (at most) two variables
- Binary constraint graph: nodes are variables, arcs show constraints
- General-purpose CSP algorithms use the graph structure to speed up search. E.g., Tasmania is an independent subproblem!



13

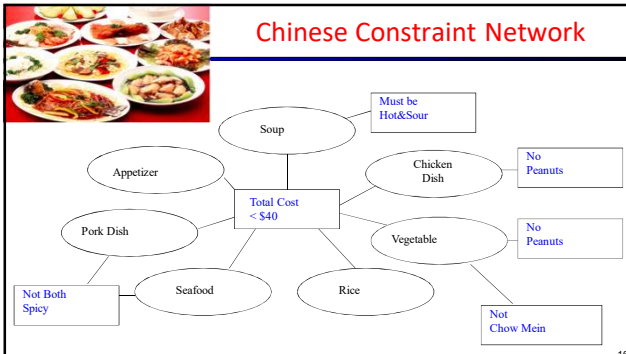
## Example: Cryptarithmic

- 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:  
 $\text{alldiff}(F, T, U, W, R, O)$   
 $O + O = R + 10 \cdot X_1$   
 $\dots$



15

## Chinese Constraint Network



16

## Real-World CSPs

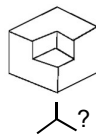
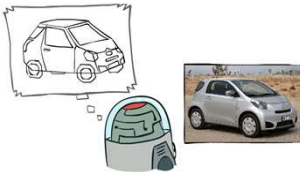
- Assignment problems: e.g., who teaches what class
- Timetabling problems: e.g., which class is offered when and where?
- Hardware configuration
- Gate assignment in airports
- Space Shuttle Repair
- Transportation scheduling
- Factory scheduling
- ... lots more!



17

## Example: The Waltz Algorithm

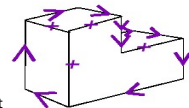
- The Waltz algorithm is for interpreting line drawings of solid polyhedra as 3D objects
- An early example of an AI computation posed as a CSP



19

## Waltz on Simple Scenes

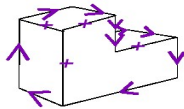
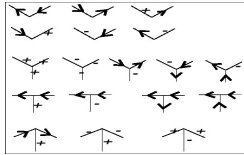
- Assume all objects:
  - Have no shadows or cracks
  - Three-faced vertices
  - "General position": no junctions change with small movements of the eye.
- Then each line on image is one of the following:
  - Boundary line (edge of an object) ( $\rightarrow$ ) with right hand of arrow denoting "solid"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



21

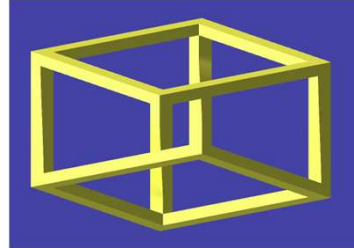
## Legal Junctions

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



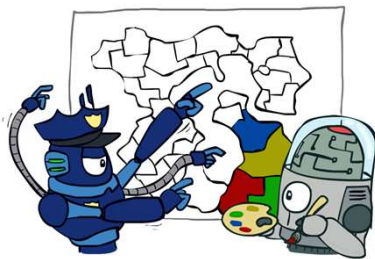
22

## Slight Problem: Local vs Global Consistency



23

## Varieties of CSPs



24

## 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)



25

## Varieties of CSP Constraints

- Varieties of Constraints**
  - Unary constraints involve a single variable (equivalent reducing domains), e.g.:
    - $SA \neq \text{green}$
  - Binary constraints involve pairs of variables, e.g.:
    - $SA \neq WA$
  - Higher-order constraints involve 3 or more variables: e.g., cryptarithmic 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)



26

## Solving CSPs



28

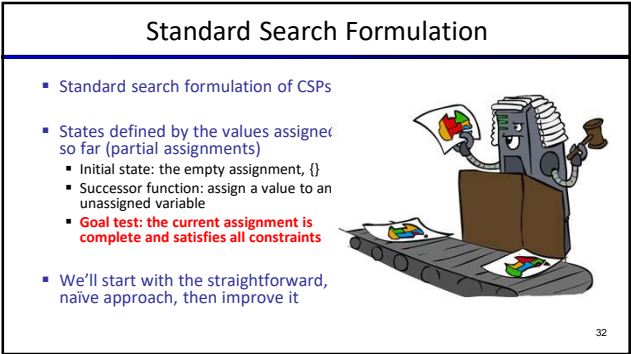
# CSP as Search

- States
- Operators
- Initial State
- Goal State

29

- 29

# Standard Depth First Search



- 32

# Backtracking Search




A cartoon illustration featuring two robots. On the left is a blue robot with a yellow visor and a blue collar. On the right is a grey robot with a white visor and a grey collar. They are both looking at a white document held by the grey robot. The document has a green shape on it, and a red 'X' is drawn over it. A speech bubble with a red 'X' is coming from the blue robot, indicating a failed attempt or a dead end in the search process.

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 make some computation to check the con:
    - "Incremental goal test"
- Depth-first search with these two improvements is called *backtracking search*
- Can solve *n*-queens for  $n \approx 25$



- ## 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]

37

## Backtracking Search

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

38

## 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?



40

## Next: Constraint Satisfaction Problems - Part 2

41