Problems Problemas Problem

#### Constraint Satisfaction Problems - CSP

March 23, 2017

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 の�?

# **Examples of Applications**

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 の�?

- 8-puzzle
- blocks world (e.g.: Hanoi tower)
- n-queens
- cryptarithmetic
- missionaires and cannibals
- solitaire
- many others...

#### n-queens

- Problem: place n queens in a board  $n \times n$  so that no queens attack each other in the rows, columns or diagonals
- 2 ways of solving the problem: incremental and complete
- Possible states (a) and moves (b):
  - 1. Incremental 1

a) any arrangement of 0 to n queens in the boardb) add 1 queen to any position in the board

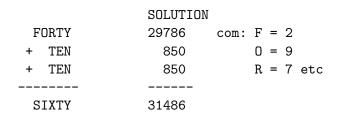
- 2. Incremental 2
  - a) arrangements of 0 to n queens with no attacks
     b) add 1 queen to the next leftmost empty column such that this does not attack the others already placed
- 3. Completo

 a) arrangements of n queens, 1 em cada coluna
 b) mover qq rainha atacada para outra posição na mesma coluna

ション ふゆ マ キャット しょう くしゃ

Problems Problems Problemas Problema

Cryptarithmtic



▲□▶ ▲圖▶ ▲臣▶ ▲臣▶ ―臣 – のへで

# Cryptarithmetic

- states: set of letters that are replaced by digits
- operators (moves): replace all occurrences of a letter by a digit that was never used before
- possible rules to select digits: numerical order to avoid repetitions, more strict obeying the mathematical properties of the problem
- final state: solution contains only digits and represent a correct sum

うして ふゆう ふほう ふほう ふしつ

• cost of the solution: zero

#### **Missionaires and Cannibals**

- 3 missionaires and 3 cannibals are by in a river bank with a boat that fits only 2 people
- Problem: find a way of moving all people from one river bank to the other
- : Constraint: we can never leave a number of cannibals larger than missionaires in a river bank :)



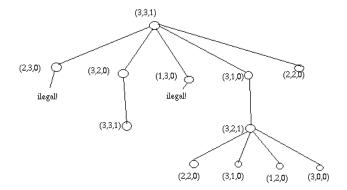
・ロト ・ 日 ・ モー・ モー・ うへぐ

## **Missionaires and Cannibals**

- 3 missionaires and 3 cannibals are by in a river bank with a boat that fits only 2 people
- Problem: find a way of moving all people from one river bank to the other
- : Constraint: we can never leave a number of cannibals larger than missionaires in a river bank :)
- states: any ordered sequence of 3 numbers representing the missionaires, the cannibals and the boat
- initial state: (3,3,1)
- final state: (0,0,0)
- cost: number of river traversals
- rules (moves): take a missionaire, take a cannibal, take 2 cannibals etc...

・ロト ・ 日 ・ モ ・ ト ・ モ ・ うへぐ

#### **Missionaires and Cannibals**



#### Other Approach and Representation

- Constraint Satisfaction
  - special type of problem that satisfies structural properties besides the basic requirements for general search problems
  - states: set of variables
  - variables can take values from a domain: set of possible values a variable can take (discrete/continuous, finite/infinite)
  - ▶ initial state: all variables with possible initial values from the domain
  - ▶ final state: final assignment of variable-value that does not violate the problem constraints

・ロト ・ 日 ・ モ ・ ト ・ モ ・ うへぐ

- ▶ maximum search tree depth: number of variables
- Explicit representation for constraints (Portuguese translation: restrições)

# Example CSP: Map-Coloring

- Variables: WA, NT, Q, NSW, V, SA, T
- Domain: D=red,green,blue
- Constraints: adjacent regions must have different colors
  - WA  $\neq$  NT
  - ▶ (WA,NT)  $\in$  { (red,green),(red,blue),(green,red),...}
- Solutions are assignments satisfying all constraints, e.g: {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}

うして ふゆう ふほう ふほう ふしつ

#### Example CSP: N-queens

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

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

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

◆□▶ ◆□▶ ◆三≯ ◆三≯ → □ ◆ ��や

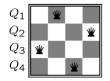
Problems Problems Problemas Problema

#### Example CSP: N-queens

- Formulation 2:
  - Variables:  $Q_k$
  - Domains:  $\{1, 2, 3, ...\}$
  - ► Constraints:

 $\forall i, j \text{ non-threatening}(Q_i, Q_j) or$  $(Q1, Q2) \in \{(1, 3), (1, 4), \ldots\}$ 

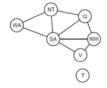
. . .



ション ふゆ マ キャット しょう くしゃ

# **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

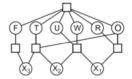


• Solutions are assignments satisfying all constraints, e.g: {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}

# Example CSP: Cryptarithmetic

- Variables (circles): F T U W R O X1 X2 X3
  - TWO +TWO
  - FOUR
- Domains:  $\{0,1,2,3,4,5,6,7,8,9\}$
- Constraints (boxes):

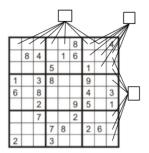
```
alldiff(F,T,U,W,R,O)
O + O = R + 10 . X1
```



ション ふゆ マ キャット しょう くしゃ

# Example CSP: 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



▲ロト ▲周ト ▲ヨト ▲ヨト ヨー のく⊙

# Varieties of CSPs

- 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

うして ふゆう ふほう ふほう ふしつ

- Linear constraints solvable, nonlinear undecidable
- Continuous variables
  - E.g., start-end state of a robot
  - Linear constraints solvable in polynomial time by LP methods

# Varieties of Constraints

- Variables of Constraints
  - ► Unary constraints involve a single variable (equiv. to shrinking domains): SA ≠ green
  - ▶ Binary constraints involve pairs of variables:  $SA \neq WA$
  - ► Higher-order constraint 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

### **Standard Search Formulation**

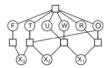
- Standard search formulation of CSPs (incremental)
- Let's start with the straightforward, dumb approach, then fix it
- States are defined by the values assigned so far
  - ▶ Initial state: the empty assignment, {}
  - ▶ Successor function: assign a value to an unassigned variable
  - Goal test: the current assignment is complete and satisfies all constraints

ション ふゆ マ キャット しょう くしゃ

Problems Problemas Problem

#### Search Methods

- What does BFS do?
- What does DFS do?



◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 の�?

# Constraint Satisfaction: improving search

- n-queens
  - variables: possible positions in the board
  - ▶ constraints: no queen can attack each other
  - ▶ initial variable values:  $V_1$  in 1..n,  $V_2$  in 1..n etc, with n the width of the board

ション ふゆ マ キャット しょう くしゃ

- aloca uma nova rainha para uma nova coluna a cada nível (solução incremental)
- complexidade:  $n^n = \approx \pi D_i = D_1 \times D_2 \times \dots \times D_n$
- fator de ramificação: n
- fator máximo de ramificação:  $n \times n = \sum D_i = D_1 + D_2 + \dots + D_n$ 
  - se todas as variáveis fossem instanciadas com todos os valores possíveis no primeiro nível da árvore

うして ふゆう ふほう ふほう ふしつ

n = 8

(0,0,0,0,0,0,0,0)

Ν (1,0,0,0,0,0,0,0)(0,1,0,0,0,0,0,0)(0.0.1.0.0.0.0) .... i (2.0.0.0.0.0.0.0)(0.2.0.0.0.0.0.0)(0.0.2.0.0.0.0.0) .... v . . . . . . . . . . 1 (8,0,0,0,0,0,0,0)(0,8,0,0,0,0,0,0)(0,0,8,0,0,0,0,0) .... (1,0,1,0,0,0,0,0) .... Ν (1,1,0,0,0,0,0,0)(0,1,1,0,0,0,0,0)i (2,2,0,0,0,0,0,0)(0,2,2,0,0,0,0,0)(2,0,2,0,0,0,0,0) .... v . . . . . . . . . . (8.8,0,0,0,0,0,0)2 (0,8,8,0,0,0,0,0)(8,0,8,0,0,0,0,0) ....

▲□▶ ▲圖▶ ▲国▶ ▲国▶ - 国 - のへで

• using DFS, the tree that contains

 $(0,0,0,0,0,0,0,0) \rightarrow (1,0,0,0,0,0,0,0) \rightarrow$ 

 $(1,1,0,0,0,0,0,0) \rightarrow$ 

 $(1,1,1,0,0,0,0,0) \rightarrow (1,1,1,1,0,0,0,0) \rightarrow \dots$ 

• will be explored even knowing that (1,1,1,1,1,1,1,1) is not a solution

うして ふゆう ふほう ふほう ふしつ

• Class of algorithms "generate-and-test"

- solution: test the constraint whenever a new queen is placed in the board
- it is not the best solution either
- Suppose that we managed to place 6 queens that are safe. This placement may threat the 8th queen and the algorithm does not know!
- DFS will end up testing ALL possibilities for placing the 7th queen in the board!

ション ふゆ マ キャット しょう くしゃ

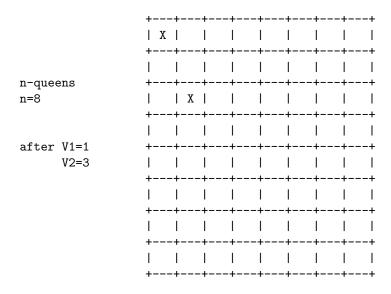
• Class of algorithms: "constrain-and-generate"

- solution: Look ahead!
- Forward Checking: checks if the other unassigned variables domains are consistent with the new partial solution, removing from the domain of other variables all values that violate the constraints
- General Lookahead (arc-consistency): besides executing forward checking, checks if the new set of domains conflict with each other

◆□ → ◆□ → ▲ □ → ▲ □ → ◆ □ → ◆ ○ ◆

#### Forward Checking: Example

#### Forward Checking: Example



◆□▶ ◆□▶ ◆三▶ ◆三▶ ・三 ・ つへぐ

Problems Problemas Problem

#### Forward Checking: Example

- Constraints:
  - $\blacktriangleright V_i = k \to V_j \neq k \quad \forall \quad j = 1, \dots, 8; j \neq i$
  - $\blacktriangleright V_i = k_i, V_j = k_j \rightarrow \mid i j \mid \neq \mid k_i k_j \mid \forall j = 1, \dots, 8; j \neq i$

◆□ → ◆□ → ▲ □ → ▲ □ → ◆ □ → ◆ ○ ◆

### Solutions to n-queens WITHOUT searching

(Explicit Solutions to the N-Queens Problem for all N, by Bo Bernhardsson)

- for n even and not of the form 6k + 2
  - (j, 2j)•  $\frac{n}{2} + j, 2j - 1$ , for  $j = 1, 2, ..., \frac{n}{2}$
- for n even, but not in the form 6k

• 
$$(j, 1 + [2(j-1) + \frac{n}{2} - 1 \mod n])$$

- $(n+1-j, n-[2(j-1)+\frac{n}{2}-1 \mod n])$ , for  $j=1, 2, \dots, \frac{n}{2}$
- for n odd

• Use any case above for n-1 and extend with 1 queen (n, n)

◆□ → ◆□ → ▲ □ → ▲ □ → ◆ □ → ◆ ○ ◆

# Solutions for n-queens WITHOUT searching

- Other references:
  - ▶ The n-Queens Problem, by Igor Rivin, Ilan Vardi, and Paul Zimmermann
  - ▶ A simplified solution of the N queens' problem, by Matthias Reichling

ション ふゆ マ キャット しょう くしゃ

• Observation: find **all** solutions for the n-queens problem is not trivial :(

#### Constraint Satisfaction: other algorithms

- for inifinite domains: linear programming, simplex, revised simplex, convex hull, Gauss elimination
- for finite domains: forward checking, lookahead, arc-consistency in general
- for finite domains, two problems to solve:
  - choice of the *variable*:
    - most-constrained: smaller domain
    - *most constraining*: constrains the domain of other variables as much as possible
  - choice of the *value* for a variable:
    - *first-fail* principle.
    - *least constraining*: value that leaves more choices open to other variables

ション ふゆ マ キャット しょう くしゃ

Problems Problems Problemas Problema

### **Constraint Satisfaction**

- most-constrained: allows to solve n-queens with n equals to 100.
- pure forward checking: solves at most 30
- least-constraining value: allows to solve n-queens with n equals to 1000.

▲□▶ ▲圖▶ ▲国▶ ▲国▶ - 国 - のへで

Problems Problems Problemas Problema

# Algorithms

- Backtracking (systematic search)
- Constraint propagation: k-consistency
- All that use heuristics to order variables and their values

ション ふゆ マ キャット しょう くしゃ

• Backjumping and dependency-directed backtracking

#### **Basic Algorithm**

```
CSP-BACKTRACKING(PartialAssignment a)
     If a is complete then return a
     X <- select an unassigned variable
     D <- select an ordering for the domain of X
     For each value v in D do
         If v is consistent with a then
              Add (X = v) to a
              result <- CSP-BACKTRACKING(a)
              If result \iff failure then return result
              Remove (X = v) from a
     Return failure
```

Initial call: CSP-BACKTRACKING({}) Solves n-queens till  $n \approx 25$ Combination of constraint propagation and heuristics can solve instances of size 1000

Random algorithms with min-conflicts plus parallelization can solve 3M queens in less than a

ション ふゆ マ キャット マックタン

minute! (see recommended reading in the discipline page)