Constraint satisfaction problems

- Describe states implicitly in terms of variables and constraints
- More search algorithms:
  - backtracking search
  - local search

Constraint satisfaction problem (CSP)

- A CSP is defined by:
  - a set of variables $\{x_1, \ldots, x_n\}$
  - a set of values for each variable $dom(x_1), \ldots, dom(x_n)$
  - a set of constraints $\{C_1, \ldots, C_m\}$
- A solution to a CSP is a complete assignment to all the variables that satisfies the constraints

Given a CSP

- Determine whether it has a solution or not
- Find one solution
- Find all solutions
- Find an optimal solution, given some cost function

Example domains and constraints

- Reals, linear constraints
  - $3x + 4y \leq 7$, $5x - 3y + z = 2$
  - Guassian elimination, linear programming
- Integers, linear constraints
  - integer linear programming, branch-and-bound
- Boolean, propositional sentences
- Here:
  - finite domains
  - expressive constraint languages

Constraint languages

- Usual arithmetic operators:
  - $=, \leq, \geq, <, >, \neq, +, -, *, /$, absolute value, exponentiation
  - e.g., $3x + 4y \leq 7$, $5x^3 - x*y = 9$
- Usual logical operators:
  - $\wedge, \vee, \neg, \Rightarrow$ (or “if … then”)
  - e.g., if $x = 1$ then $y = 2$, $\neg x \vee y \vee z$, $(3x + 4y \leq 7) \vee (x*y = z)$
- Global constraints:
  - can be specified over an arbitrary number of variables
  - e.g., alldifferent($x_1, \ldots, x_n$) — pairwise different
- Table constraints

Alldifferent constraint

- Consists of:
  - set of variables $\{x_1, \ldots, x_n\}$
- Satisfied iff:
  - each of the variables is assigned a different value

Example: Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Each Sudoku has a unique solution that can be reached logically without guessing.

Enter digits from 1 to 9 into the blank spaces. Every row must contain one of each digit. So must every column, as must every 3x3 square.

Constraint model for Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	…			
x_{19}	x_{20}	x_{21}	…					
x_{28}	…							
x_{37}	…							
x_{46}	…							
x_{55}	…							
x_{64}	…							
x_{73}	…							

Constraint model for Sudoku

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

dom$(x_i) = \{1, \ldots, 9\}$, for all $i = 1, \ldots, 81$

alldifferent$(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)$

…

alldifferent$(x_1, x_{10}, x_{19}, x_{28}, x_{37}, x_{46}, x_{55}, x_{64}, x_{73})$

…

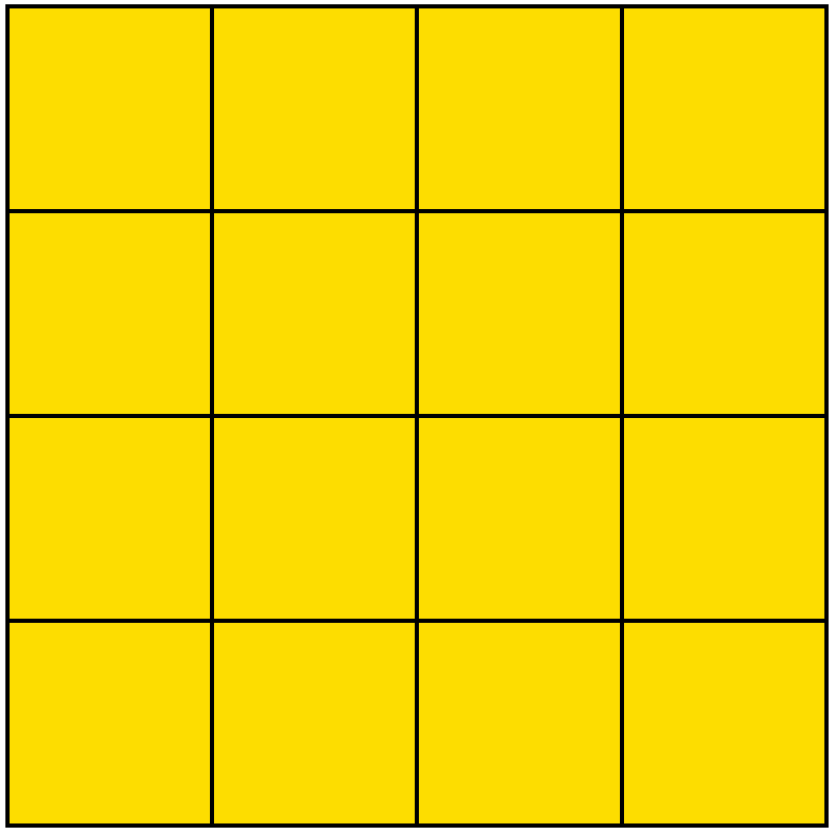
alldifferent$(x_1, x_2, x_3, x_{10}, x_{11}, x_{12}, x_{19}, x_{20}, x_{21})$

…

$x_1 = 5, x_2 = 3, x_5 = 7, \ldots, x_{81} = 9$

Example: *n*-queens

Place *n*-queens on an $n \times n$ board so that no pair of queens attacks each other



Constraint model

variables:

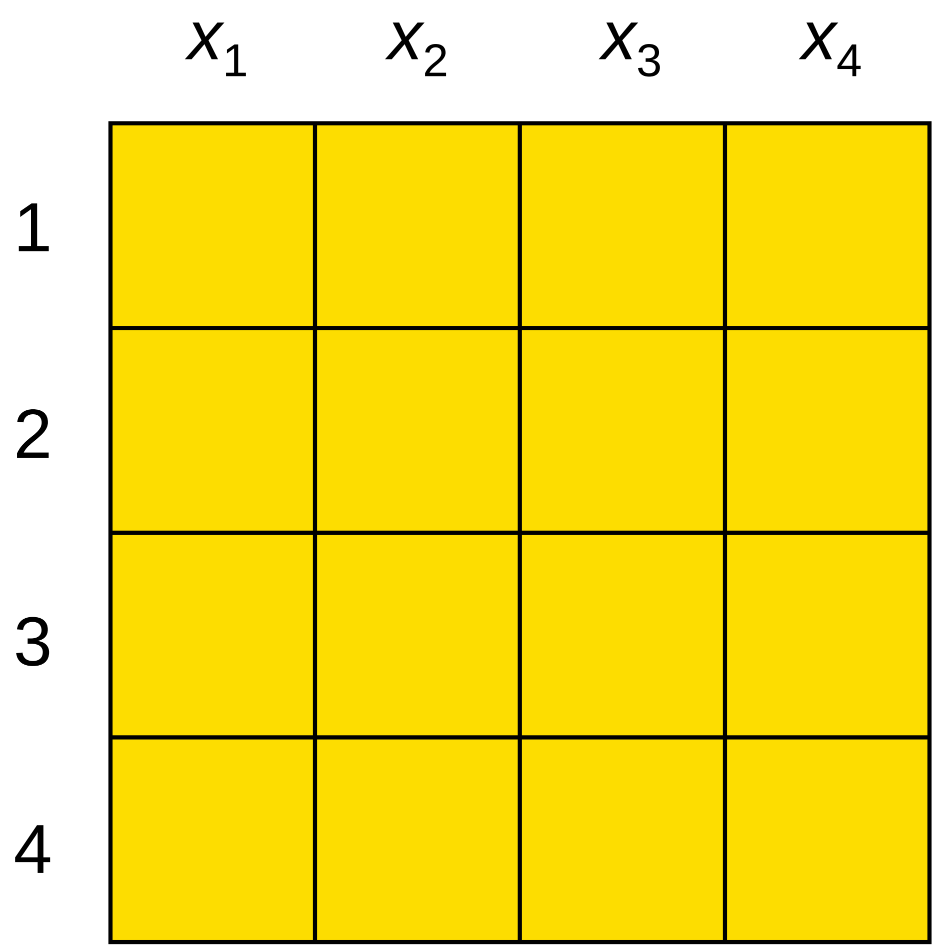
x_1, x_2, x_3, x_4

domains:

$\{1, 2, 3, 4\}$

constraints:

$$x_1 \neq x_2 \;\wedge\; |x_1 - x_2| \neq 1$$
$$x_1 \neq x_3 \;\wedge\; |x_1 - x_3| \neq 2$$
$$x_1 \neq x_4 \;\wedge\; |x_1 - x_4| \neq 3$$
$$x_2 \neq x_3 \;\wedge\; |x_2 - x_3| \neq 1$$
$$x_2 \neq x_4 \;\wedge\; |x_2 - x_4| \neq 2$$
$$x_3 \neq x_4 \;\wedge\; |x_3 - x_4| \neq 1$$



Example: 4-queens

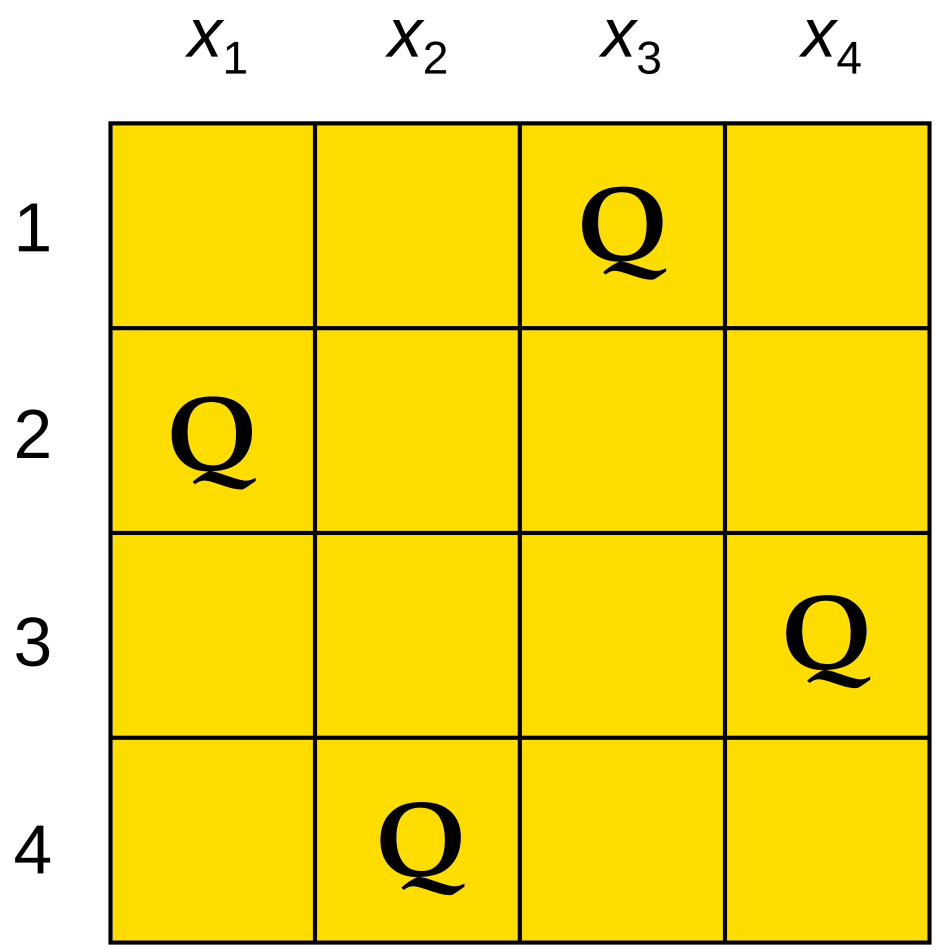
A solution

$x_1 = 2$

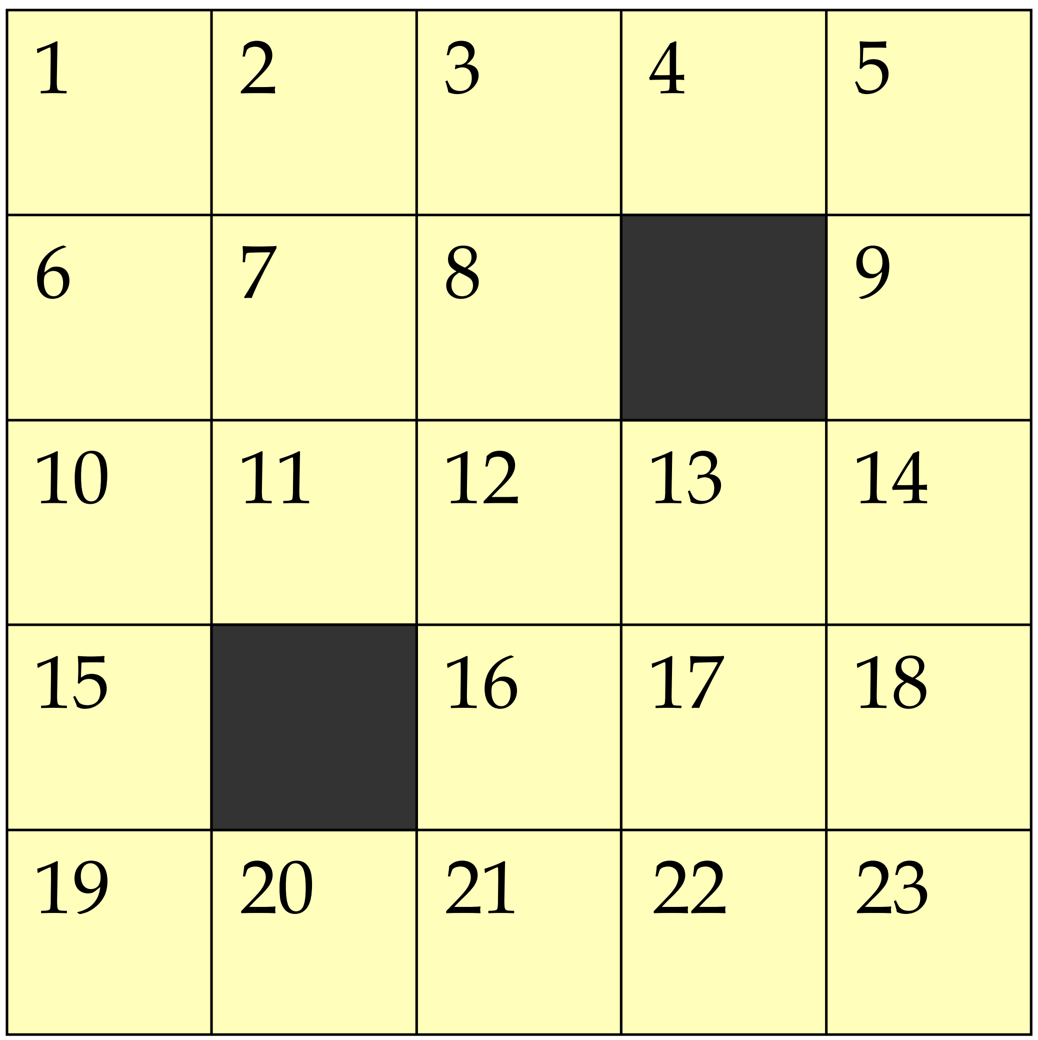
$x_2 = 4$

$x_3 = 1$

$x_4 = 3$



Example: crossword puzzles



a
aardvark
aback
abacus
abaft
abalone
abandon
...

…
monarch
monarchy
monarda
...
zymurgy
zyrian
zythum

A closer look at constraints

- An *assignment* (also called an *instantiation*)
  - $x = a$, where $a \in dom(x)$,
- A *tuple* t over an ordered set of variables $\{x_1, \ldots, x_k\}$ is an ordered list of values $(a_1, \ldots, a_k)$ such that $a_i \in dom(x_i)$, $i = 1, \ldots, k$
  - can be viewed as a set of assignments $\{x_1 = a_1, \ldots, x_k = a_k\}$
- Given a tuple t, notation $t[x_i]$ selects out the value for variable x_i; i.e. $t[x_i] = a_i$

A closer look at constraints

- Each constraint C is a relation
  - a set of tuples over some ordered subset of the variables, denoted by *vars*(C)
  - specifies the allowed combinations of values for the variables in *vars*(C)
- The set *vars*(C) is called the *scope* (or *scheme*) of the constraint
- The size of *vars*(C) is known as the *arity* of the constraint
  - a unary constraint has an arity of 1
  - a binary constraint has an arity of 2
  - a non-binary constraint has arity greater than 2
- A *binary* CSP is a CSP where all constraints are unary or binary

Example

- Let
  - $dom(x_1) = \{1, 2, 3, 4\}$,
  - $dom(x_2) = \{1, 2, 3, 4\}$
  - C be the constraint $x_1 \neq x_2 \ \wedge \ | x_1 - x_2 | \neq 1$

intensional

- Then
  - $vars(C) = \{x_1, x_2\}$
  - tuples in C = {(1,3), (1,4), (2,4), (3,1), (4,1), (4,2)}
  - C is a binary constraint

x_1	x_2
1	3
1	4
2	4
3	1
4	1
4	2

if $t = (1, 3)$

$t[x_1] = 1$

$t[x_2] = 3$

extensional

(table constraint)

Application areas

- scheduling
- logistics
- planning
- supply chain management
- rostering
- timetabling
- vehicle routing
- bioinformatics
- networks
- configuration
- assembly line sequencing
- cellular frequency assignment
- airport counter and gate allocation
- airline crew scheduling
- optimize placement of transmitters for wireless
- …

Some commercial applications

Outline

- Introduction
- Constraint propagation
- Backtracking search
- Local search

Outline

- Introduction
- Constraint propagation
    - arc consistency
- Backtracking search
- Local search

Local consistency: arc consistency

Given a constraint, remove a value from the ***domain*** of a variable if it cannot be part of a solution according to that constraint

Local consistency: arc consistency

- Given a constraint C, a value $a \in dom(x)$ for a variable $x \in vars(C)$ has:
  - a *domain support* in C if there exists a $t \in C$ such that $t[x] = a$ and $t[y] \in dom(y)$, for every $y \in vars(C)$
  - i.e., there exists values for each of the other variables (from their respective domains) such that the constraint is satisfied
- A constraint C is:
  - *arc consistent* iff for each $x \in \text{vars}(C)$, each value $a \in dom(\text{x})$ has a domain support in C
- A CSP is:
  - arc consistent if every constraint is arc consistent
- A CSP can be made arc consistent by repeatedly removing unsupported values from the domains of its variables

Arc consistency algorithm

```
ac() : boolean
    Q ← all variable/constraint pairs (x, C)
    while Q ≠ {} do
        select and remove a pair (x, C) from Q
        if revise(x, C)
            if dom(x) = {}
                return false
            else
                add pairs to Q
    return true

revise(x, C) : boolean
    change ← false
    for each a ∈ dom(x) do
        if ¬∃ domain support for a in C
            remove a from dom(x)
            change ← true
    return change
```

Arc consistency algorithm

```
ac() : boolean
    Q ← all variable/constraint pairs (x, C)
    while Q ≠ {} do
    |   select and remove a pair (x, C) from Q
    |   if revise(x, C)
    |   |   if dom(x) = {}
    |   |   |__ return false
    |   |   else
    |___|___|__ add pairs to Q
    return true
```

```
revise(x, C) : boolean
    change ← false
    for each a ∈ dom(x) do
    |  if ¬∃ domain support for a in C
    |  |  remove a from dom(x)
    |__|__change ← true
    return change
```

variable	*domain*
x	{1, ~~2~~, ~~3~~}
y	{~~1~~, 2, ~~3~~}
z	{~~1~~, ~~2~~, 3}

constraints

C_1: $x < y$

C_2: $y < z$

4-queens: Is it arc consistent?

variables:

x_1, x_2, x_3, x_4

domains:

{1, 2, 3, 4}

constraints:

$$
\begin{aligned}
&x_1 \neq x_2 \;\wedge\; |x_1 - x_2| \neq 1\\
&x_1 \neq x_3 \;\wedge\; |x_1 - x_3| \neq 2\\
&x_1 \neq x_4 \;\wedge\; |x_1 - x_4| \neq 3\\
&x_2 \neq x_3 \;\wedge\; |x_2 - x_3| \neq 1\\
&x_2 \neq x_4 \;\wedge\; |x_2 - x_4| \neq 2\\
&x_3 \neq x_4 \;\wedge\; |x_3 - x_4| \neq 1
\end{aligned}
$$

	x_1	x_2	x_3	x_4
1	Q	Q		
2	Q	Q		
3	Q	Q		
4	Q	Q		