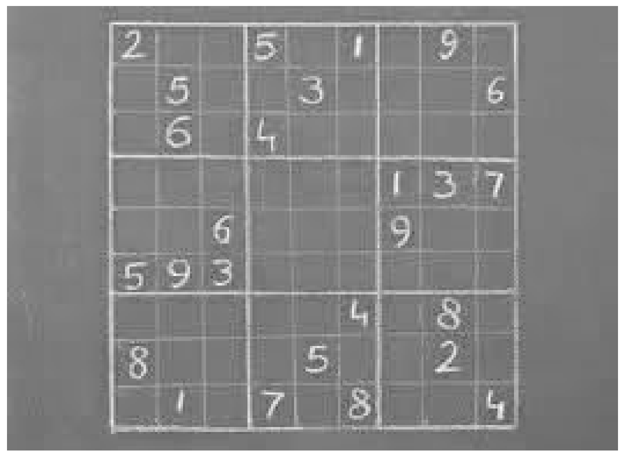
Constraint Satisfaction Problems (CSPs)



Lecture

CSPs: Overview

CSPs: Definitions

CSPs: Examples

Course plan

Constraint satisfaction problems

Search problems

Markov decision processes

Markov networks

Adversarial games

Bayesian networks

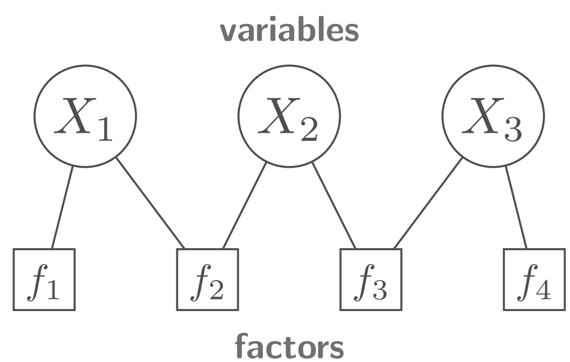
Reflex **States** **Variables** **Logic**

Low-level

High-level

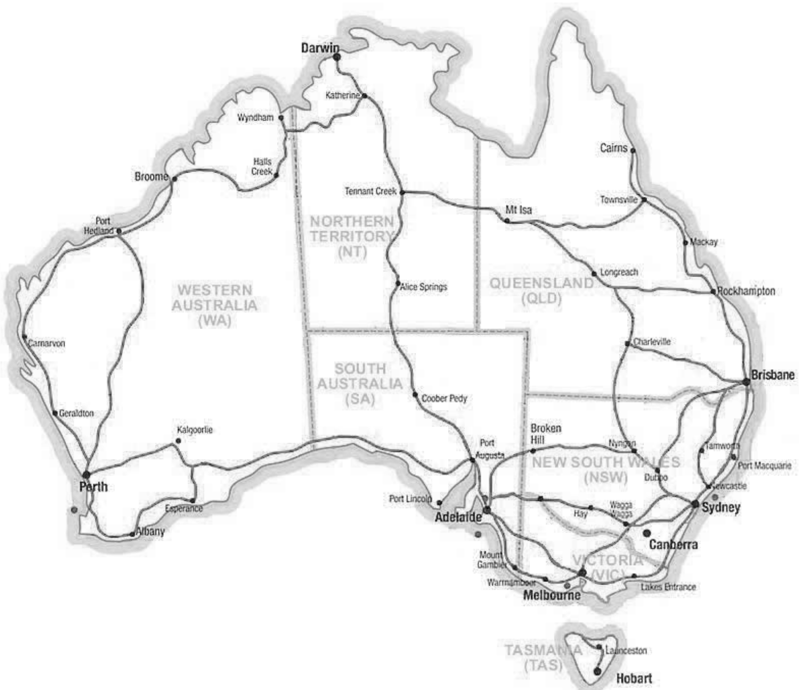
Machine learning

Factor graphs



Objective: find the best assignment of values to the variables

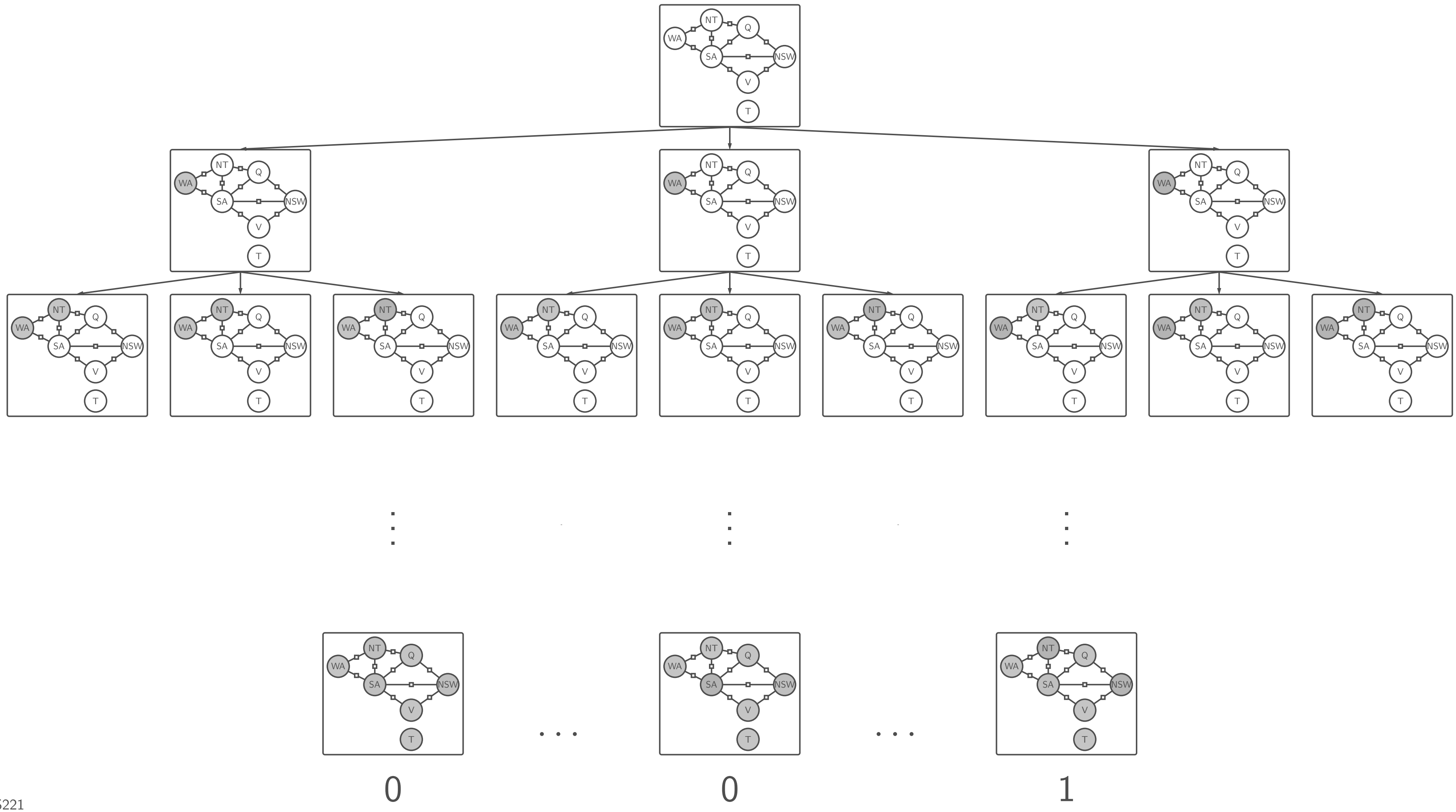
Map coloring



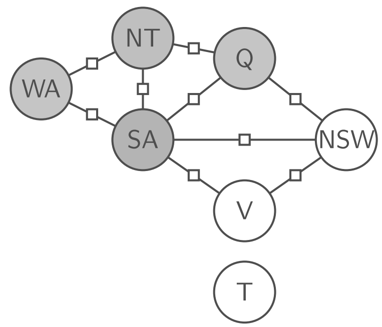
Question: how can we color each of the 7 provinces {red,green,blue} so that no two neighboring provinces have the same color?

Map coloring

(one possible solution)


WA
NT
SA
Q
NSW
V
T
0
0
1

As a search problem



- State: partial assignment of colors to provinces
- Action: assign next uncolored province a compatible color

What's missing? There's more problem structure!

- Variable ordering doesn't affect correctness, can optimize
- Variables are interdependent in a local way, can decompose

Variable-based models

Special cases:

- Constraint satisfaction problems
- Markov networks
- Bayesian networks

Key idea: variables

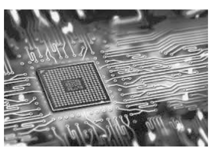
- Solutions to problems $\Rightarrow$ assignments to variables (modeling).
- Decisions about variable ordering, etc. chosen by inference.

Higher-level modeling language than state-based models

Applications

Delivery/routing: how to assign packages to trucks to deliver to customers

Sports scheduling: when to schedule pairs of teams to minimize travel



Formal verification: ensure circuit/program works on all inputs

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

Arc consistency

Approximate search

Beam search

Local search

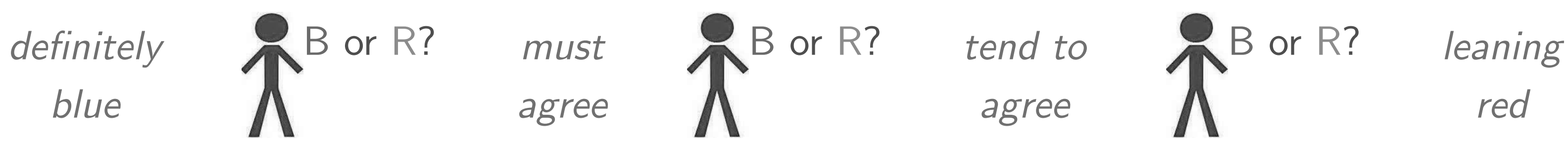
Lecture

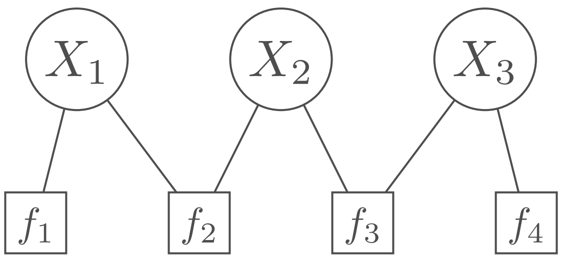
CSPs: Overview

CSPs: Definitions

CSPs: Examples

Factor graph example: voting





x_1	$f_1(x_1)$
R	0
B	1

$f_1(x_1) = [x_1 = \mathsf{B}]$

x_1	x_2	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

$f_2(x_1, x_2) = [x_1 = x_2]$

x_2	x_3	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

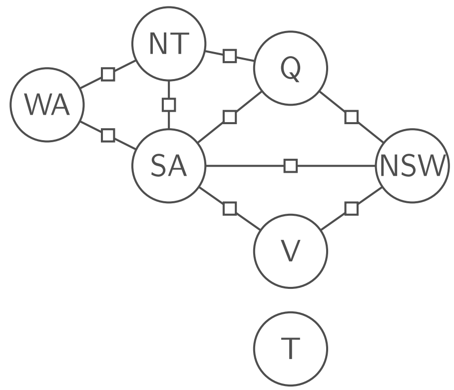
$f_3(x_2, x_3) = [x_2 = x_3] + 2$

x_3	$f_4(x_3)$
R	2
B	1

$f_4(x_3) = [x_3 = \mathsf{R}] + 1$

[demo]

Example: map coloring



Variables:

$$X = (\mathsf{WA}, \mathsf{NT}, \mathsf{SA}, \mathsf{Q}, \mathsf{NSW}, \mathsf{V}, \mathsf{T})$$

$$\mathsf{Domain}_i \in \{\mathsf{R}, \mathsf{G}, \mathsf{B}\}$$

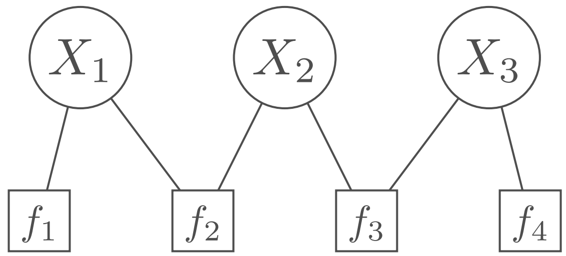
Factors:

$$f_1(X) = [\mathsf{WA} \neq \mathsf{NT}]$$

$$f_2(X) = [\mathsf{NT} \neq \mathsf{Q}]$$

...

Factor graph



Definition: factor graph

Variables:

$X = (X_1, \dots, X_n)$, where $X_i \in \text{Domain}_i$

Factors:

$f_1, \dots, f_m$, with each $f_j(X) \geq 0$

Factors

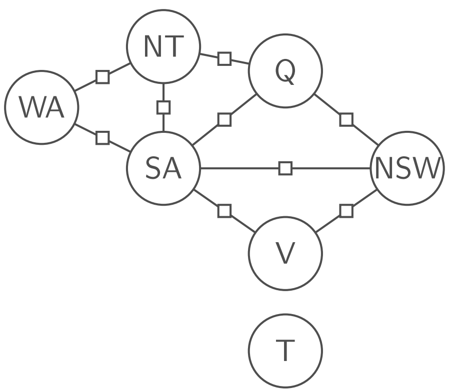
Definition: scope and arity

Scope of a factor f_j: set of variables it depends on.

Arity of f_j is the number of variables in the scope.

Unary factors (arity 1); **Binary** factors (arity 2).

Constraints are factors that return 0 or 1.



Example: map coloring

Scope of $f_1(X) = [\text{WA} \neq \text{NT}]$ is $\{\text{WA}, \text{NT}\}$

f_1 is a binary constraint

Assignment weights example: voting

x_1	$f_1(x_1)$
R	0
B	1

x_1	x_2	$f_2(x_1, x_2)$
R	R	1
R	B	0
B	R	0
B	B	1

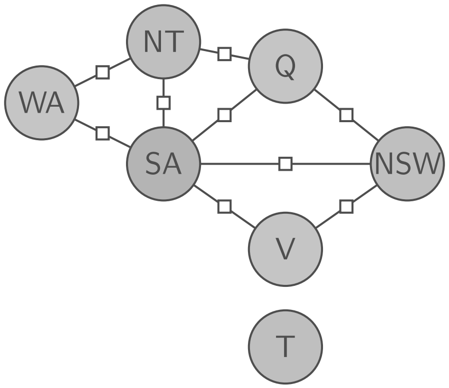
x_2	x_3	$f_3(x_2, x_3)$
R	R	3
R	B	2
B	R	2
B	B	3

x_3	$f_4(x_3)$
R	2
B	1

x_1	x_2	x_3	Weight
R	R	R	$0 \cdot 1 \cdot 3 \cdot 2 = 0$
R	R	B	$0 \cdot 1 \cdot 2 \cdot 1 = 0$
R	B	R	$0 \cdot 0 \cdot 2 \cdot 2 = 0$
R	B	B	$0 \cdot 0 \cdot 3 \cdot 1 = 0$
B	R	R	$1 \cdot 0 \cdot 3 \cdot 2 = 0$
B	R	B	$1 \cdot 0 \cdot 2 \cdot 1 = 0$
B	B	R	$1 \cdot 1 \cdot 2 \cdot 2 = 4$
B	B	B	$1 \cdot 1 \cdot 3 \cdot 1 = 3$

[demo]

Example: map coloring



Assignment:

$$x = \{\mathsf{WA} : \mathsf{R}, \mathsf{NT} : \mathsf{G}, \mathsf{SA} : \mathsf{B}, \mathsf{Q} : \mathsf{R}, \mathsf{NSW} : \mathsf{G}, \mathsf{V} : \mathsf{R}, \mathsf{T} : \mathsf{G}\}$$

Weight:

$$\mathsf{Weight}(x) = 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 1$$

Assignment:

$$x' = \{\mathsf{WA} : \mathsf{R}, \mathsf{NT} : \mathsf{R}, \mathsf{SA} : \mathsf{B}, \mathsf{Q} : \mathsf{R}, \mathsf{NSW} : \mathsf{G}, \mathsf{V} : \mathsf{R}, \mathsf{T} : \mathsf{G}\}$$

Weight:

$$\mathsf{Weight}(x') = 0 \cdot 0 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 \cdot 1 = 0$$

Assignment weights

Definition: assignment weight

Each **assignment** $x = (x_1, \dots, x_n)$ has a **weight**:

$$\text{Weight}(x) = \prod_{j=1}^{m} f_j(x)$$

An assignment is **consistent** if $\text{Weight}(x) > 0$.

Objective: find the maximum weight assignment

$$\arg\max_x \text{Weight}(x)$$

A CSP is **satisfiable** if $\max_x \text{Weight}(x) > 0$.

Constraint satisfaction problems

Boolean satisfiability (SAT):

variables are booleans, factors are logical formulas $[X_1 \vee \neg X_2 \vee X_5]$

Linear programming (LP):

variables are reals, factors are linear inequalities $[X_2 + 3X_5 \leq 1]$

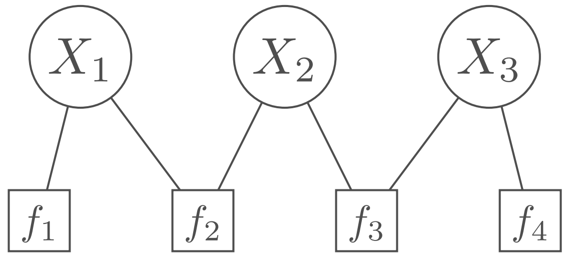
Integer linear programming (ILP):

variables are integers, factors are linear inequalities

Mixed integer programming (MIP):

variables are reals and integers, factors are linear inequalities

Summary



Variables, factors: specify locally

$$\text{Weight}(\{X_1 : \mathsf{B}, X_2 : \mathsf{B}, X_3 : \mathsf{R}\}) = 1 \cdot 1 \cdot 2 \cdot 2 = 4$$

Assignments, weights: optimize globally

Lecture

CSPs: Overview

CSPs: Definitions

CSPs: Examples

Example: LSAT question

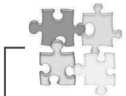
Three sculptures (A, B, C) are to be exhibited in rooms 1, 2 of an art gallery.

The exhibition must satisfy the following conditions:

- Sculptures A and B cannot be in the same room.
- Sculptures B and C must be in the same room.
- Room 2 can only hold one sculpture.

[demo]

Example: object tracking

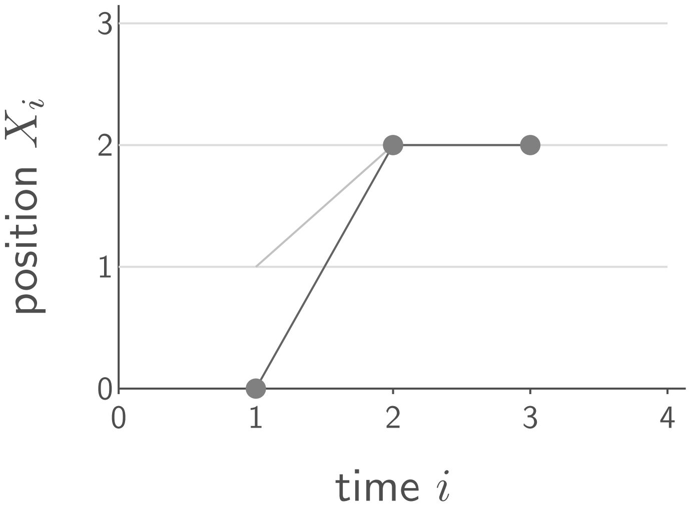


Problem: object tracking

(O) Noisy sensors report positions: $0, 2, 2$.

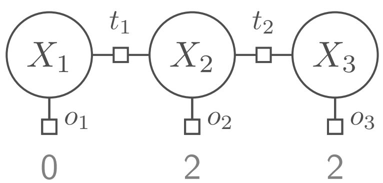
(T) Objects can't teleport.

What trajectory did the object take?



Example: object tracking CSP

Factor graph:



x_1	$o_1(x_1)$
0	2
1	1
2	0

x_2	$o_2(x_2)$
0	0
1	1
2	2

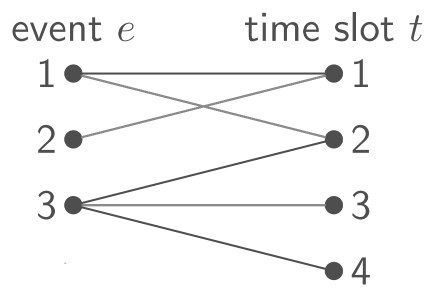
x_3	$o_3(x_3)$
0	0
1	1
2	2

$\lvert x_i - x_{i+1} \rvert$	$t_i(x_i, x_{i+1})$
0	2
1	1
2	0

[demo]

- Variables $X_i \in \{0, 1, 2\}$: position of object at time i
- Observation factors $o_i(x_i)$: noisy information compatible with position
- Transition factors $t_i(x_i, x_{i+1})$: object positions can't change too much

Example: event scheduling



Problem: Event scheduling

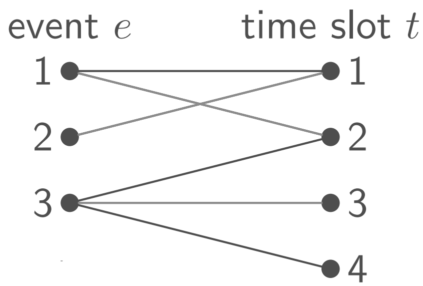
Have E events and T time slots

(C1) Each event e must be put in **exactly one** time slot

(C2) Each time slot t can have **at most one** event

(C3) Event e allowed in time slot t only if $(e, t) \in A$

Example: event scheduling (formulation 1)



Problem: Event scheduling

Have E events and T time slots

(C1) Each event e must be put in **exactly one** time slot

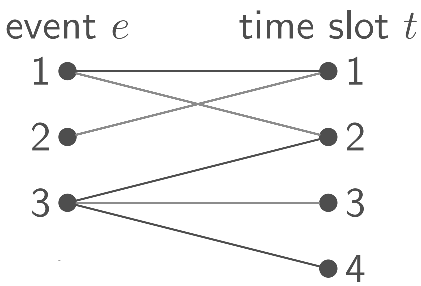
(C2) Each time slot t can have **at most one** event

(C3) Event e allowed in time slot t only if $(e, t) \in A$

CSP formulation 1:

- Variables: for each event e, $X_e \in \{1, \ldots, T\}$; satisfies (C1)
- Constraints (only one event per time slot): for each pair of events $e \neq e'$, enforce $[X_e \neq X_{e'}]$; satisfies (C2)
- Constraints (only scheduled allowed times): for each event e, enforce $[(e, X_e) \in A]$; satisfies (C3)

Example: event scheduling (formulation 2)



Problem: Event scheduling

Have E events and T time slots

(C1) Each event e must be put in **exactly one** time slot

(C2) Each time slot t can have **at most one** event

(C3) Event e allowed in time slot t only if $(e, t) \in A$

CSP formulation 2:

- Variables: for each time slot t, $Y_t \in \{1, \dots, E\} \cup \{\emptyset\}$; satisfies (C2)
- Constraints (each event is scheduled exactly once): for each event e, enforce $[Y_t = e \text{ for exactly one } t]$; satisfies (C1)
- Constraints (only schedule allowed times): for each time slot t, enforce $[Y_t = \emptyset \text{ or } (Y_t, t) \in A]$; satisfies (C3)

Example: program verification

```
def foo(x, y):
   a = x * x
   b = a + y * y
   c = b - 2 * x * y
   return c
```

Specification: `c >= 0` for all `x` and `y`

CSP formulation:

- Variables: x, y, a, b, c
- Constraints (program statements): $[a = x^2]$, $[b = a + y^2]$, $[c = b - 2xy]$

Note: program ($=$ is assignment), CSP ($=$ is mathematical equality)

- Constraint (negation of specification): $[c < 0]$

Program satisfies specification iff CSP has no consistent assignment

Summary

- Decide on variables and domains
- Translate each desideratum into a set of factors
- Try to keep CSP small (variables, factors, domains, arities)
- When implementing each factor, think in terms of checking a solution rather than computing the solution

Overall Summary

- Constraint satisfaction problems as Factor graphs
- Definitions: variables factors, assignments, weights
- Examples: tracking, scheduling, program verification
- Next: Solving CSPs