CPSC 322: Introduction to Artificial Intelligence

Introduction to Constraint Satisfaction Problems (CSPs)

Textbook reference: [4.1]

Instructor: Varada Kolhatkar
University of British Columbia

Credit: These slides are adapted from the slides of the previous offerings of the course. Thanks to all instructors for creating and improving the teaching material and making it available!

Announcements

- Thanks for your feedback! I'll try my best to incorporate it going forward!
- Midterm time and location.
  Time: Friday, Oct 25th, from 6pm to 7pm
  Location: Woodward 2
  (Instructional Resources Centre-IRC) (WOOD) - 2
- My office hours: Fridays from 11am to noon at ICCS 185

Lecture outline

- Recap from last lecture (~2 mins) 👈🏻
- Introduction (~10 mins)
- Variables, domains, possible worlds (~20 mins)
- Constraints and CSPs (~20 mins)
- Class activity (~15 mins)

Recap: Summary of search strategies

	Method	Selection	Complete	Optimal	Time O()	Space O()
Uninformed	**DFS**	LIFO	N (Y if no cycles)	N	b^m	mb
	BFS	FIFO	Y	Y	b^m	b^m
	IDS	LIFO	Y	Y	b^m	mb
	LCFS (when arc costs available)	min cost	Y (if costs > 0)	Y (if costs ≥ 0)	b^m	b^m
informed	**BestFS (When *h* available)**	min *h*	N	N	b^m	b^m
	A* (when arc costs and *h* available)	min *f*	Y if branching factor finite, *h* is admissible, and costs > 0	Y if branching factor finite, *h* isadmissible, and costs > 0	b^m	b^m
	Branch and Bound	LIFO + pruning	N (Y UB finite)	Y	b^m	mb
	IDA*	LIFO	Y (same as A*)	Y	b^m	mb
	MBA*	min *f*	Y if enough memory	Y if enough memory and *h* is admissible	b^m	b^m

Constraint Satisfaction Problems (CSPs)

A rough CPSC 322 overview

We’ll now focus on

Representation and reasoning	Problem	Environment: Deterministic	Environment: Stochastic
Static	Constraint satisfaction	Variables + constraints Arc consistency Search	
Static	Query	Logics Search	Belief networks Variable elimination
Sequential	Planning	STRIPS Search	Decision networks Variable elimination Markov decision processes Value iteration

Today: Learning outcomes

From this lecture, students are expected to be able to:

- Define possible worlds in term of variables and their domains.
- Compute number of possible worlds on real examples
- Specify constraints to represent real world problems differentiating between: Unary and k-ary constraints, list vs. function format.
- Verify whether a possible world satisfies a set of constraints (i.e., whether it is a model, a solution)

Why study CSPs?

https://www.research.ibm.com/haifa/dept/vst/csp.shtml

← IBM Research Home

IBM R&D Labs in Israel

- Haifa Research Lab
  - Cognitive Analytics and Solutions
  - IoT and Cloud Technologies
  - Computing as a Service
  - Healthcare Informatics
  - IBM Cyber Security Center of Excellence (CCoE)
- Researchers
- ThinkLab
- Visitor Information
- Feedback
- EU-funded Programmes
- Leadership Seminars
- Seminars
- Careers at IBM R&D Labs in Israel

Constraint Satisfaction

Overview

The Constraint Satisfaction team focuses on research and development in the area of constraint satisfaction problems (CSPs). Our main asset is the constraint solver, a robust, general-purpose, state-of-the-art tool that has been used for more than a decade in modeling and solving many complex constraint problems. Our department has long-standing expertise in CSP algorithms and modeling. Our aim is to provide value to IBM through the application of constraint solving to various domains along with close interaction with the academic community. We work closely with our IBM partners to find the best constraint or optimization model for their domain and to apply the best heuristics when solving this model.

Activities

→ Data Fabrication
→ Constraint Solver
→ Verification
→ Workforce Management
→ Floorplanning
→ Pipeline Scheduling

Past Activities

→ Vehicle Configuration
→ System Engineering
→ Virtual Machine Placement

Related Links

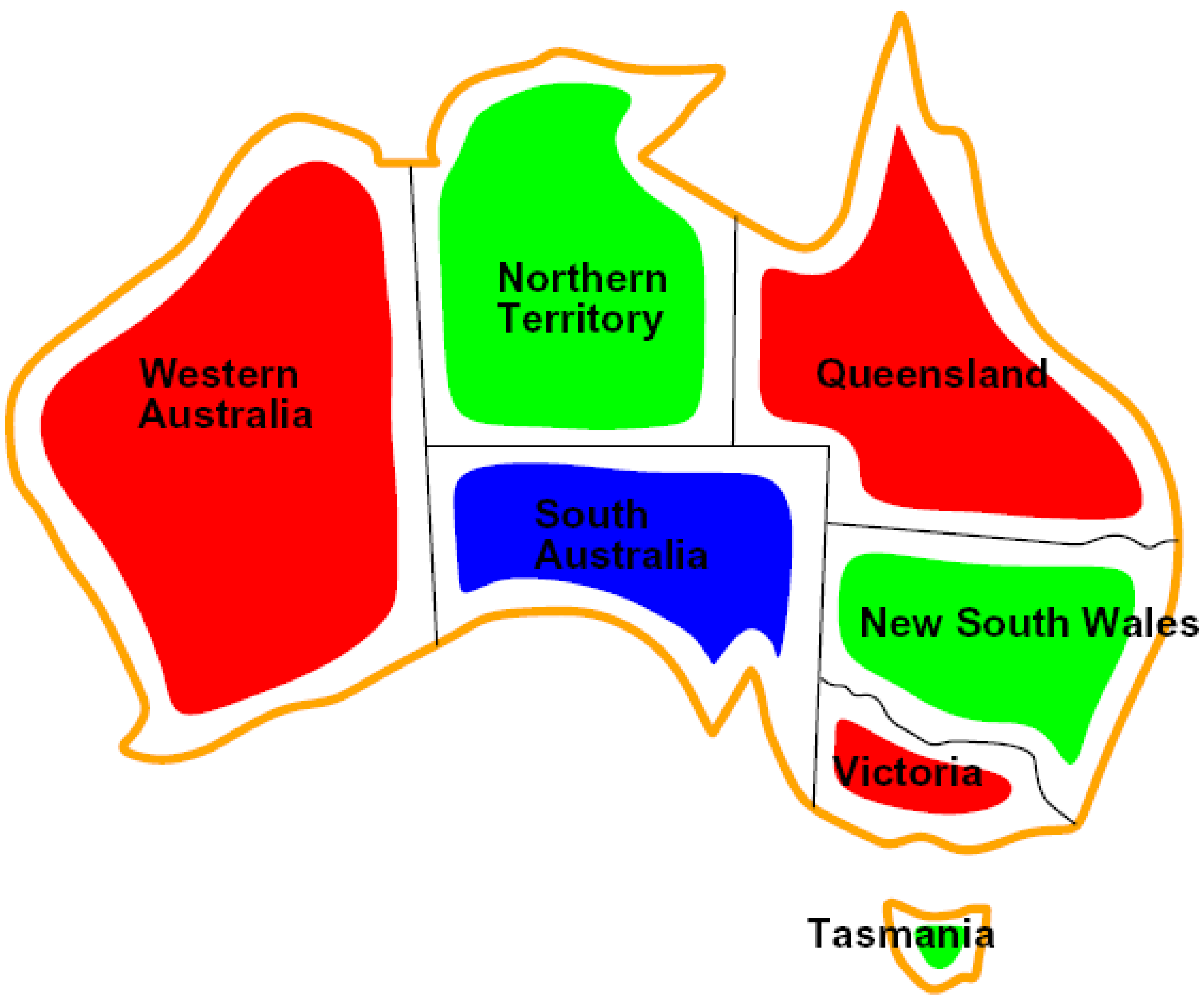
→ Publications

Example CSP problems

- Assignment problems: e.g., who teaches what class?
- Timetabling problems: e.g., which class is offered when and where?
- Airport gate allocation problem
- Hardware configuration
- Fault diagnosis …

Toy example: Graph colouring

Colour the graph so that adjacent regions must have different colours.



Standard search

- So far we have used search as the reasoning strategy for a simple goal-driven planning agent.
    - Start → path → goal
- An agent can solve a problem by searching in a space of states.
    - State is a "black box" – any arbitrary data structure that supports three problem-specific routines: neighbours(n), heuristic(n), goal(n)

Planning vs. Identification

Planning

- Sequences of actions
- The path to the goal is the important thing
- Paths have various costs, depths
- Heuristics give problem-specific guidance

Identification

- The goal itself is important not the path
- All paths at the same depth (for some formulations)
- CSPs are specialized for identification problems

Main representational dimensions (lecture 2)

Domains can be classified by the following dimensions:

- Uncertainty
  Deterministic vs. stochastic domains

- How many actions does the agent need to perform?
  Static vs. sequential domains

- An important design choice is: Representation scheme
  Explicit states vs. features (vs. relations)

Explicit state vs. Features (lecture 2)

How do we model the environment?

- You can enumerate the possible states of the world
- A state can be described in terms of features
  - Assignment to (one or more) variables
  - Often the more natural description
  - 30 binary features can represent $2^{30} = 1{,}073{,}741{,}824$ states

Variables, domains, possible worlds

We formulate CSPs using

- **Variables/features**: Feature of possible world
- **Domains**: The set of values the variable can take
- **Possible worlds**: A possible way the would could be

Variables, domains, possible worlds

- **Variable**: a synonym for **feature**
  We denote variables using capital letters. (E.g., A, V)

- **Domains**: Each variable V has a domain dom(V) of possible values. (E.g., {1,2}).
  Boolean: |dom(V)| = 2
  Finite: |dom(V)| is finite
  Infinite but discrete: the domain is countably infinite
  Continuous: e.g., real numbers between 0 and 1

- **Possible worlds**: Complete assignment of values to each variable (includes every last detail in the environment). In contrast, states also include partial assignments.

Example: Mars explorer

Variables and domains

Weather: {Sunny, Cloudy}

Temperature: [-40, +40]

Longitude: [0, 359]

Latitude: [1, 180]

A possible world:
{Sunny, -28, 320, 100}

How many total number of mutually exclusive worlds?

$$2 \times 81 \times 360 \times 180$$

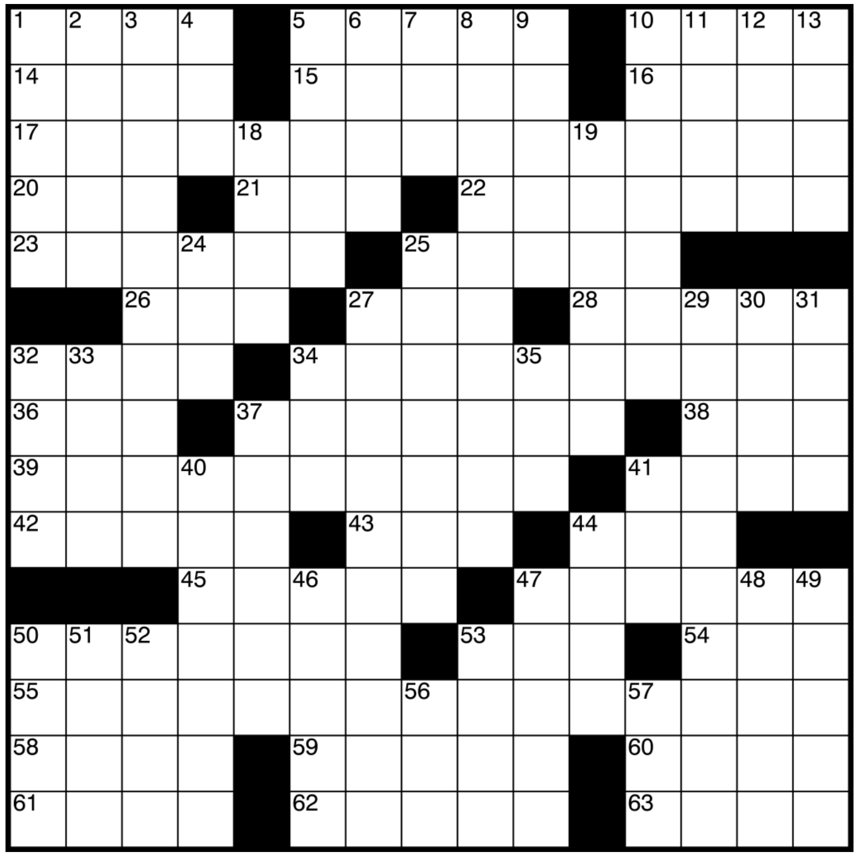
Product of cardinality of each domain

Always exponential in the number of variables

Example: Crossword puzzle

Representation 1

- Variables are words that have to be filled in
- Domains are English words of correct length
- Possible worlds: all possible ways of assigning words



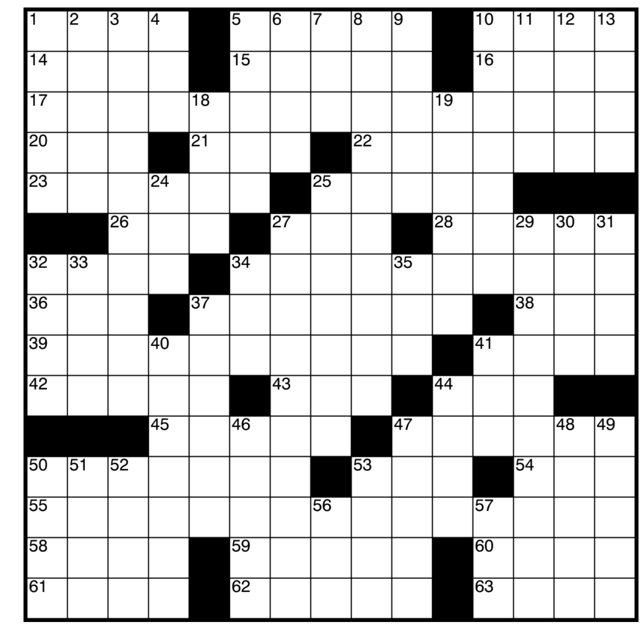
Example: Crossword puzzle

Representation 1

- Variables are words that have to be filled in
  63 variables in this case

- Domains are English words of correct length
  Number of words in English? (Say 150,000)
  Number of words of different lengths?
  (Say 15,000)

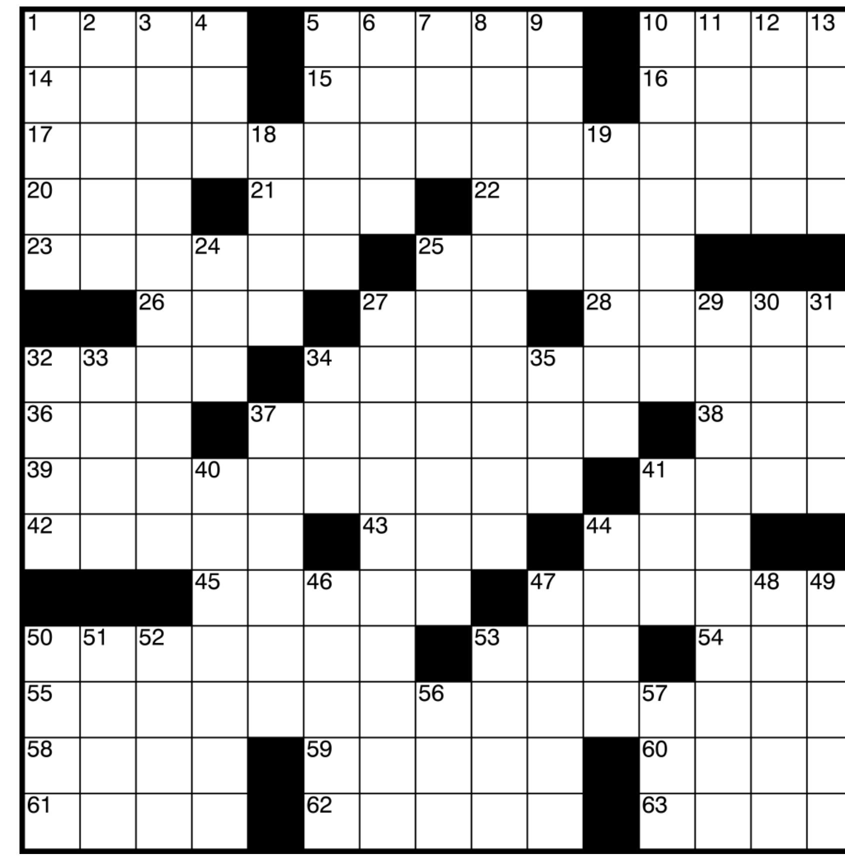
- Possible worlds: all possible ways of assigning words: 15000^{63}



Example: Crossword puzzle

Representation 2

- Variables are cells
- Domains are letters of alphabet
- Possible worlds: All possible ways of assigning letters to cells



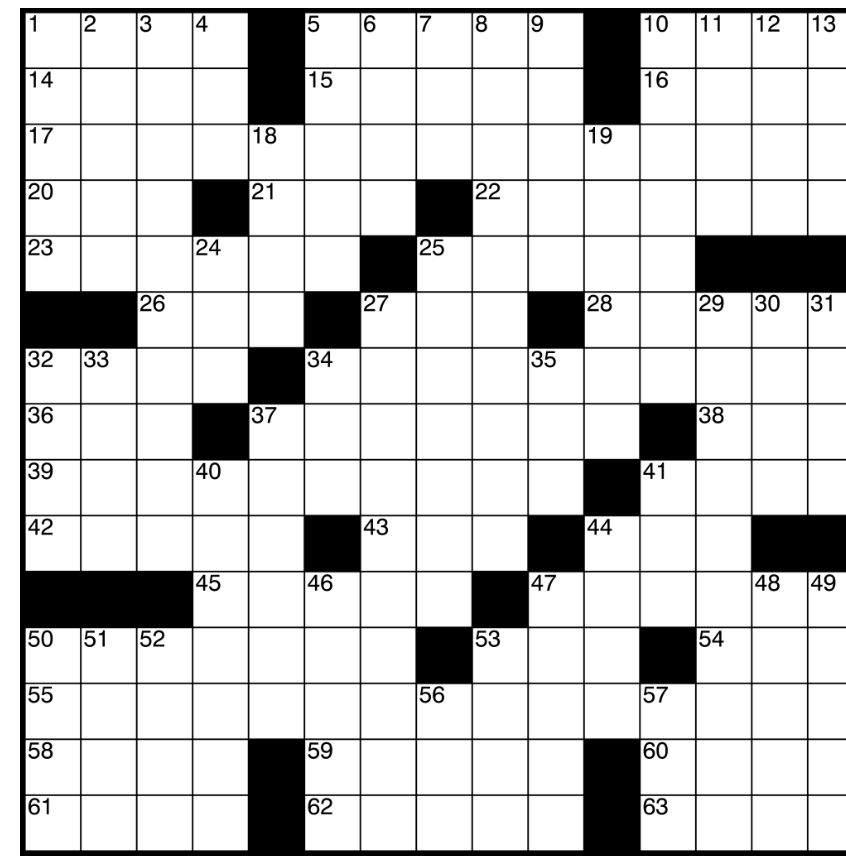
Example: Crossword puzzle

Representation 2

- Variables are cells
  Number of empty cells:
  $15 \times 15 - 32 = 193$

- Domains are letters of alphabet: 26

- Possible worlds: All possible ways of assigning letters to cells: 26^{193}



In general the number of possible worlds is:
$(\textbf{domain size})^{\textbf{number of variables}}$

Example: Sudoku

- Variables: ?
- Domains: ?
- Possible worlds: ?

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

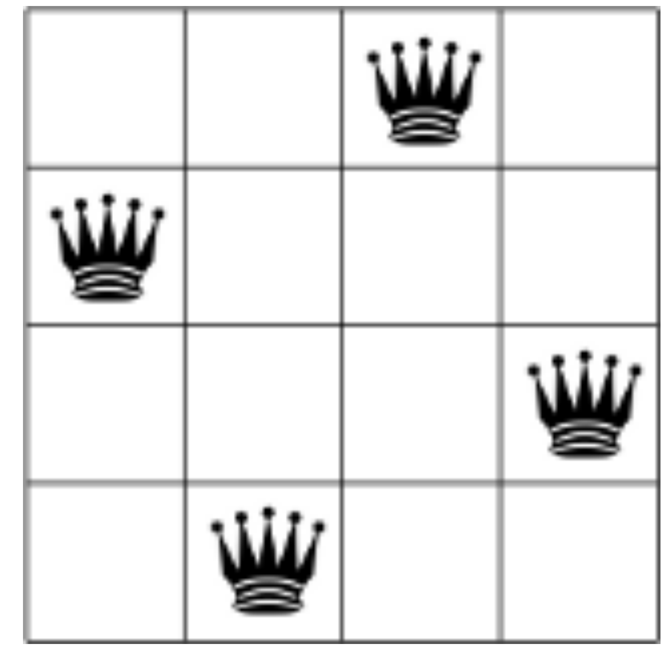
Example: Sudoku

- Variables are empty cells: 51
- Domains are integers between 1 to 9
- Possible worlds: All possible ways of assigning numbers to cells: 9^{51}

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

Example: N-Queens problem

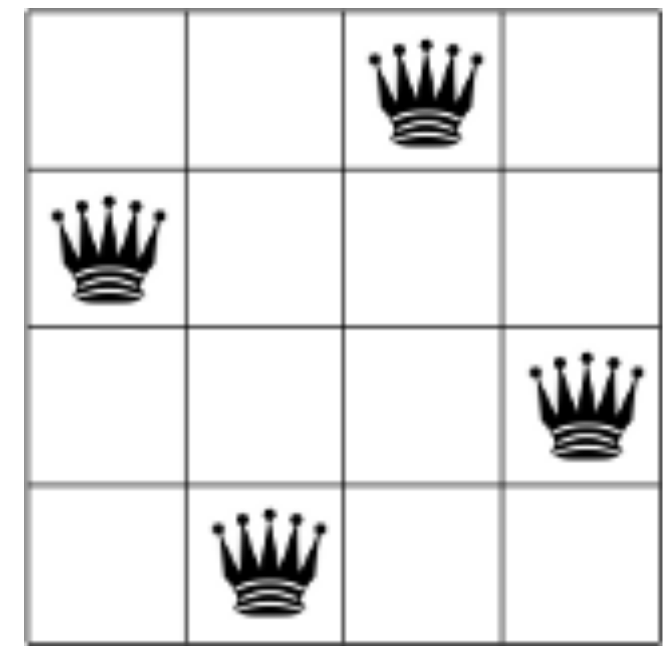
How can N queens be placed on an $N \times N$ chessboard so that no two of them attack each other?



- Variables: Location of a queen on a chess board
- Domains: grid co-ordinates
- Possible worlds: locations of all queens

Example: N-Queens problem

How can N queens be placed on an $N \times N$ chessboard so that no two of them attack each other?



- Variables: Location of a queen on a chess board N
- Domains: grid co-ordinates N^2
- Possible worlds: locations of all queens: Possible ways to choose N locations out of N^2

$$\binom{N^2}{N} = \frac{(N^2)!}{(N^2 - N)!N!}$$

$$\frac{(4^2)!}{(4^2 - 4)!(4!)}$$

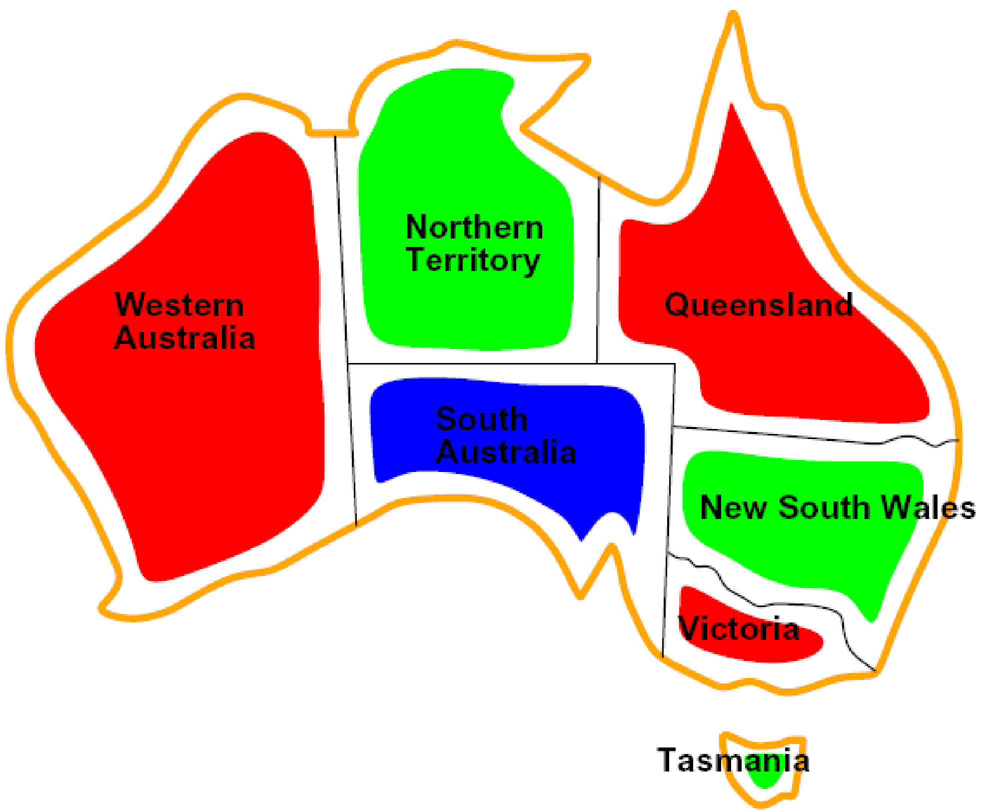
Why not $(N^2)^N$?

Example: Graph colouring

- Variables: regions on the map
  WA, NT, Q, SA, NSW, V, T

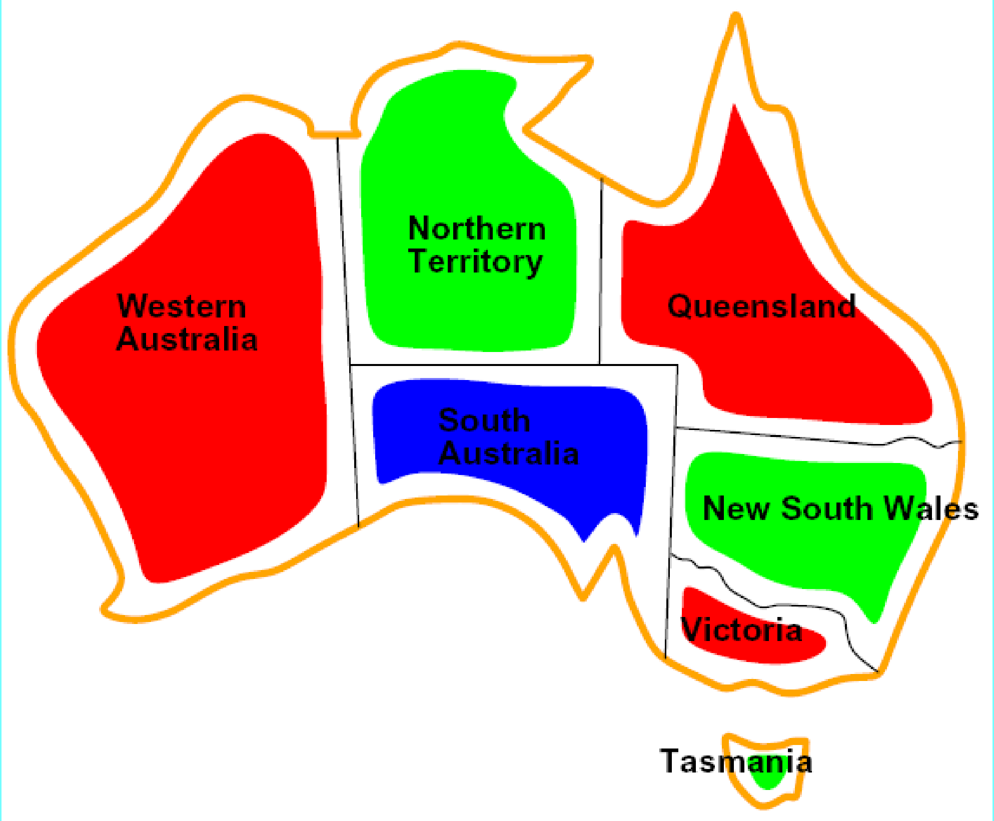
- Domains: possible colours
  {red, green, blue}

- Possible worlds: colour assignments for each region



Example: Graph colouring

How many possible worlds?



A. $2^{num\_regions}$

B. $2^{num\_colours}$

C. $num\_colours \times num\_regions$

D. $num\_regions^{num\_colours}$

E. $num\_colours^{num\_regions}$ ✅

Constraints

- Constraints are restrictions on the values that one or more variables can take
- Unary constraint: restriction involving a single variable
  Example: $A \neq 10$
- k-ary constraint: restriction involving the domains of k different variables: it turns out that k-ary constraints can always be represented as binary constraints, so we'll mainly only talk about this case
  Example: $A < B$ (binary), $A + B + C < 8$ (tertiary)

Scope of a constraint

The **scope** of a constraint is the set of variables that are involved in the constraint.

Examples:

- $V2 \neq 2$ has scope {V2}
- $V1 > V2$ has scope {V1,V2}
- $V1 + V2 + V4 < 5$ has scope {V1,V2,V4} •
- How many variables are in the scope of a k-ary constraint? k variables

Specifying constraints

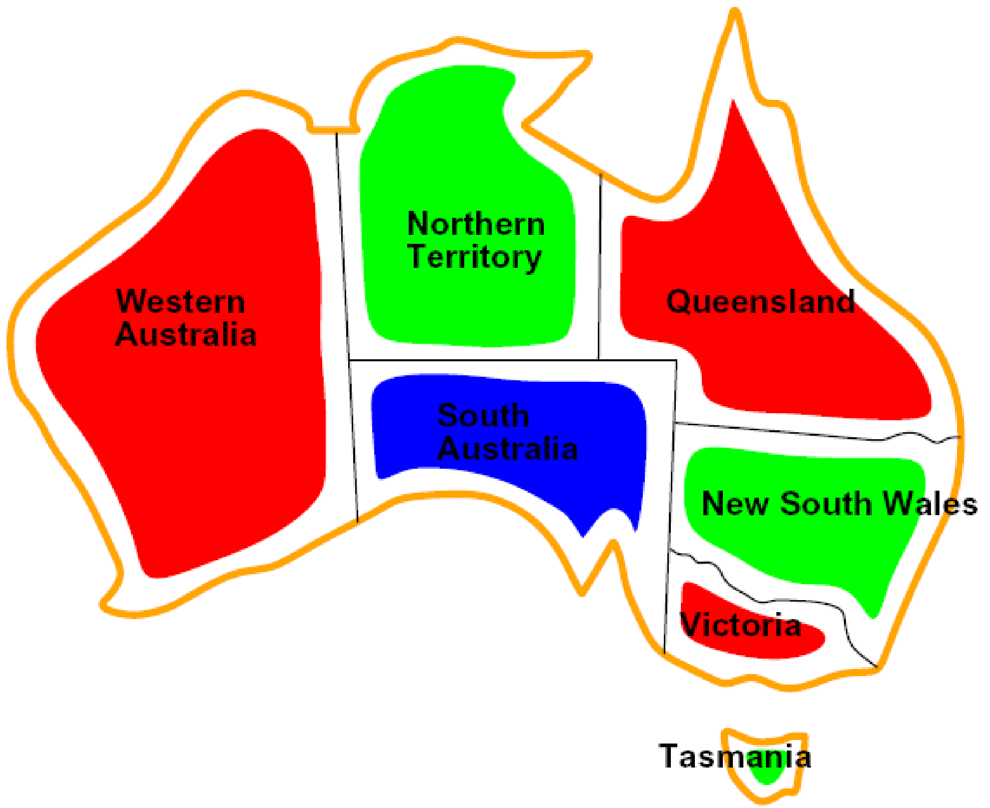
- Explicitly listing all combinations of valid domain values for the variables participating in the constraint
  Example: $(A, B) \in \{(1,1), (1,2), (2,2)\}$

- Giving a function that returns true when given values for each variable which satisfy the constraint: $A \leq B$

Variables

A	B
1	1
1	2
2	1
2	2

Example: Map colouring

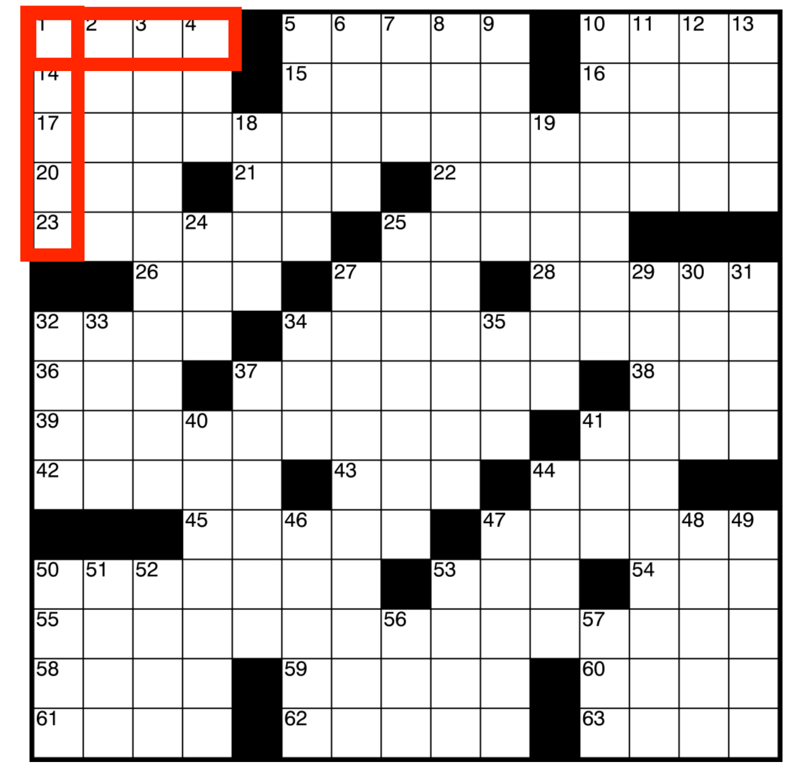


- Variables: regions on the map
  WA, NT, Q, SA, NSW, V, T
- Domains: possible colours
  {red, green, blue}
- Constraints: adjacent regions must have different colours. For example,
  Implicit: WA $\neq$ NT
  Explicit: (WA, NT) in {(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)}

Example: Crossword puzzle

Representation 1

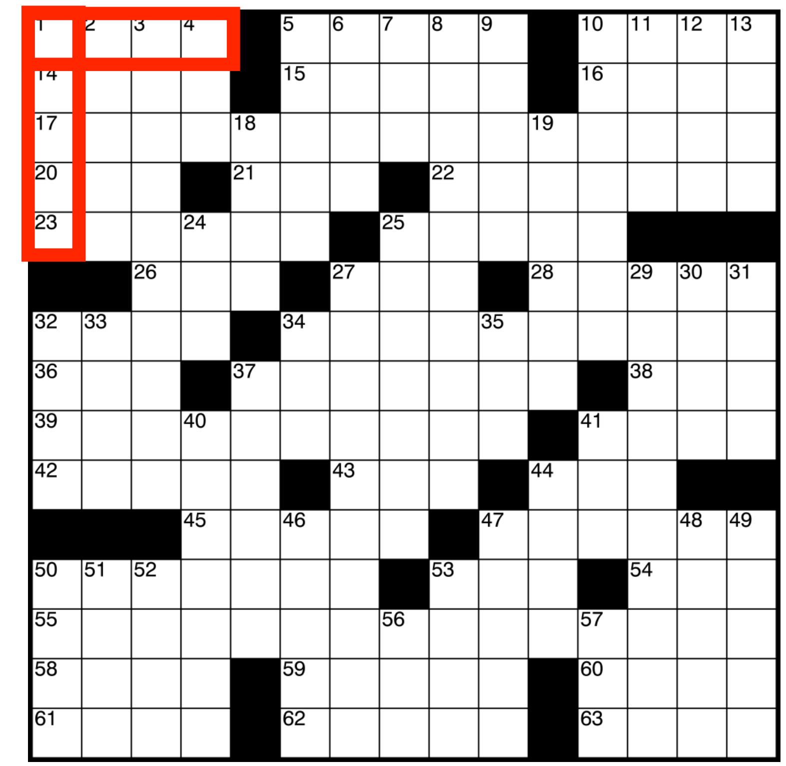
- Variables: words that have to be filled in
- Domains: English words of correct length
- Constraints: Words have the same letters at points where they intersect. For example,
  - verticle_w1[0] = horizontal_w1[0]
  - How many constraints?



Example: Crossword puzzle

Representation 1

- Variables: words that have to be filled in
- Domains: English words of correct length
- Constraints: Words have the same letters at points where they intersect. For example,

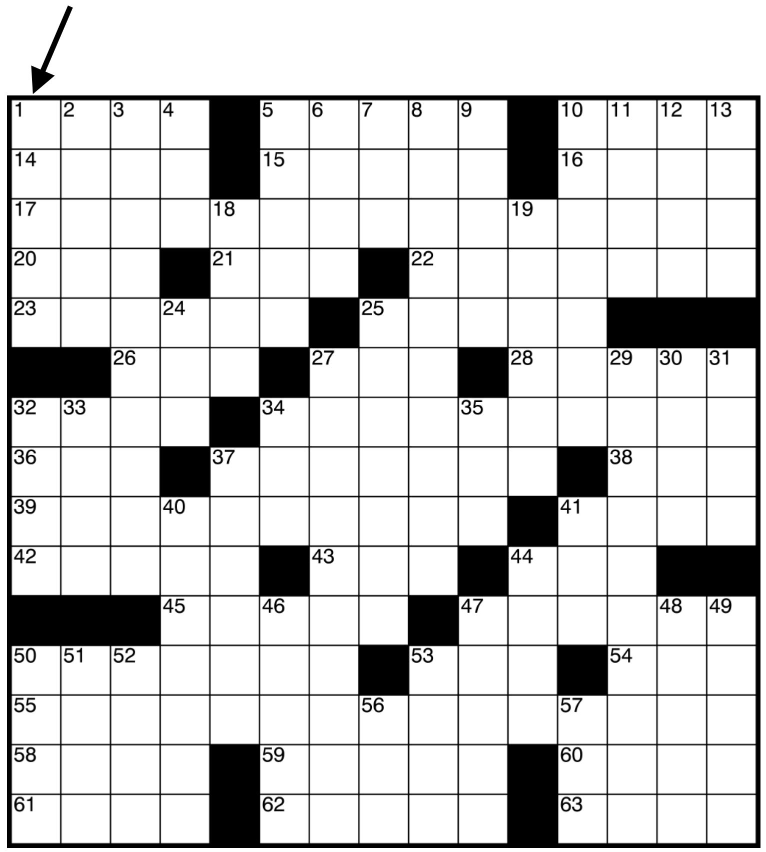


  - verticle_w1[0] = horizontal_w1[0]
  - How many constraints?
    225 - # black cells

Example: Crossword puzzle

Representation 2

cell[0,0]



- Variables: cells
- Domains: letters of alphabet
- Constraints: sequences of letters form valid English words. For example,
  - concat(cell[0,0], cell[0,1], cell(0,2), cell(0,3)) is a valid English word
  - How many constraints?

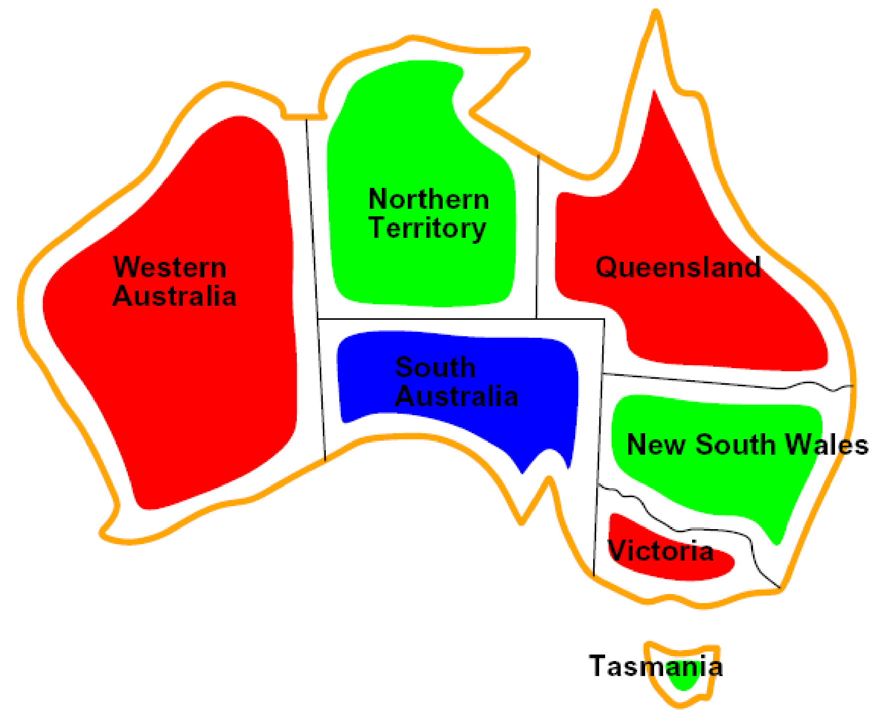
Example: Sudoku

- Variables: empty cells
- Domains: Integers between 1 to 9
- Constraints: rows, columns, boxes contain all different numbers.

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

Satisfying a set of constraints

- A possible world **satisfies** a set of constraints
  - If the values for the variables involved in each constraint are **consistent** with that constraint.



Possible world W {WA = red, NT = green, SA = blue, Q = red, NSW = green, V = red, T = green} satisfies the map-colouring constraint .

Satisfying constraints

Variables: {A, B, C}, Domains: [1,…,10]
Possible world W = {A = 1, B =2, C = 10}
Constraint set1 = {A = B, C > B}
Constraint set2 = {A $\neq$ B, C > B, (A,C) in {(10,1), (1,10)}}

A. W satisfies both set1 and set2

B. W satisfies set1 but not set2

C. W satisfies set2 but not set1 ✅

D. W does not satisfy any constraint

E. I would rather have a quick nap

Constraint satisfaction problems

A **constraint satisfaction problem** (CSP) consists of

- a set of **variables**
- a **domain** for each variable
- a set of **constraints**

A **model/solution** of a CSP is an assignment of values to all of its variables that **satisfies** all of its constraints.

A Simple CSP example

Example:

$V = \{V1, V2\}$

$dom(V1) = \{1,2,3\}$
$dom(V2) = \{1,2\}$

$C = \{C1, C2, C3\}$
$C1: V2 \neq 2$
$C2: V1 + V2 < 5$
$C3: V1 > V2$

Which ones are **models** for this CSP?

1. $\{V1=2, V2=1\}$
2. $\{V1=1, V2=1\}$
3. $\{V1=3, V2=1\}$

A Simple CSP example

Example:

$V = \{V1, V2\}$

$dom(V1) = \{1,2,3\}$
$dom(V2) = \{1,2\}$

$C = \{C1, C2, C3\}$
$C1: V2 \neq 2$
$C2: V1 + V2 < 5$
$C3: V1 > V2$

Which ones are **models** for this CSP?

1. $\{V1=2, V2=1\}$ ✅
2. $\{V1=1, V2=1\}$
3. $\{V1=3, V2=1\}$ ✅

CSP: Variants

We may want to solve the following problems with a CSP:

- determine whether or not a model exists
- find a model
- find all of the models
- count or enumerate all of the models
- find the best model, given some measure of model quality
- determine whether some statement holds in all models

CSP: Game plan

- Even the simplest problem of determining whether or not a model exists in a general CSP with finite domains is NP-hard
  - There is no known algorithm with worst case polynomial runtime
  - We can't hope to find an algorithm that is efficient for all CSPs

CSP: Game plan

However, we can try to:

- find consistency algorithms that reduce the size of the search space

- identify special cases for which algorithms are efficient (polynomial)

- work on approximation algorithms that can find good solutions quickly, even though they may offer no theoretical guarantees

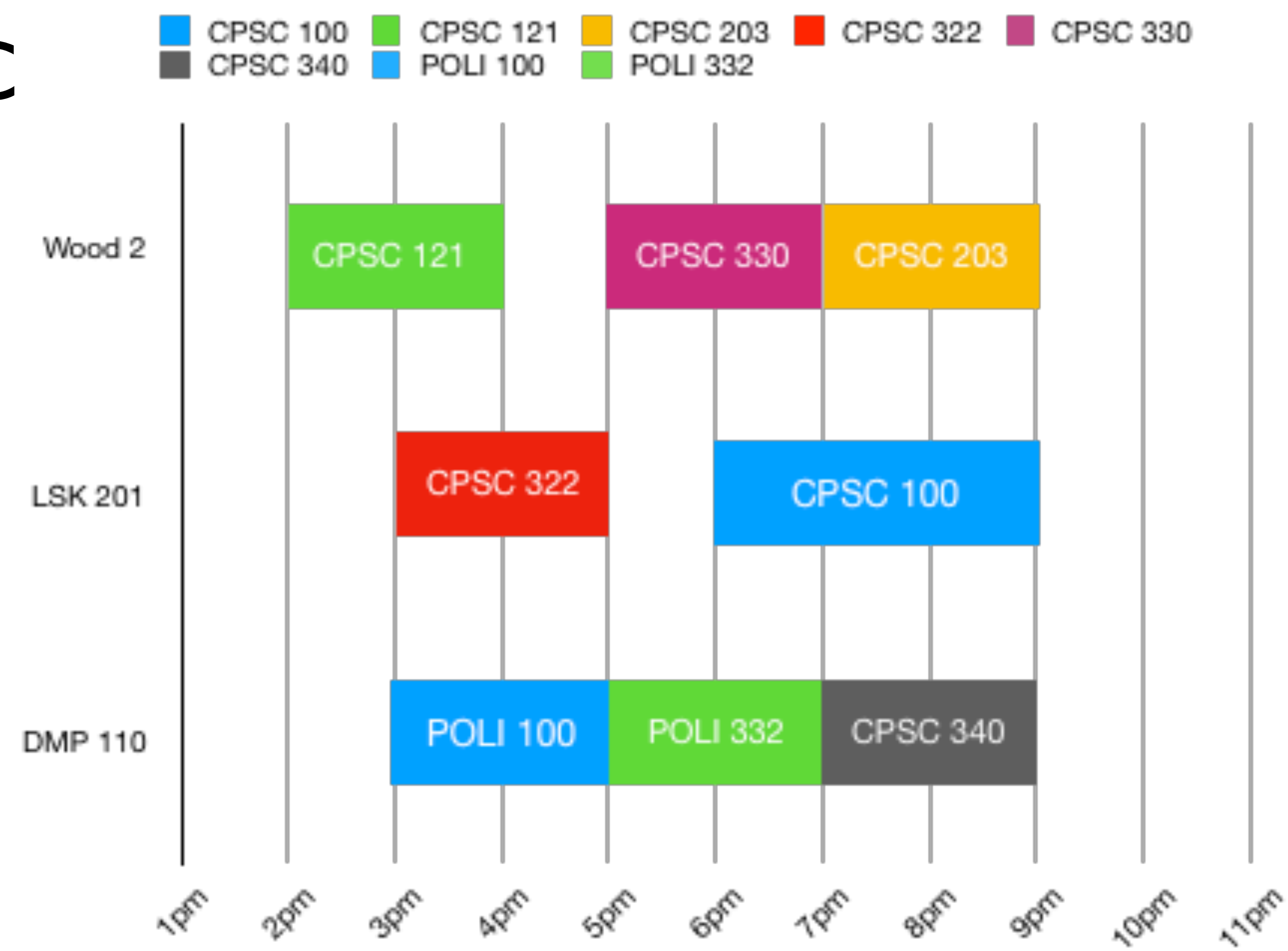
- find algorithms that are fast on typical cases

Class activity: Scheduling

Courses: CPSC 100, CPSC 121, CPSC 203, CPSC 322, CPSC 330, CPSC 340, POLI 100, POLI 332

Classrooms: Wood 2, LSK 201, DMP 110

Start times: 1pm, 2pm, 3pm, 4pm, 5pm, 6pm, 7pm, 8pm, 9pm, 10pm



Summary

- Need to think of search beyond simple goal driven planning agent.

- We started exploring the first AI Representation and Reasoning framework: CSPs

Coming up

Readings for next class

- 4.2 Generate-and-Test Algorithms
- 4.3 Solving CSPs Using Search
- 4.4 Consistency Algorithms