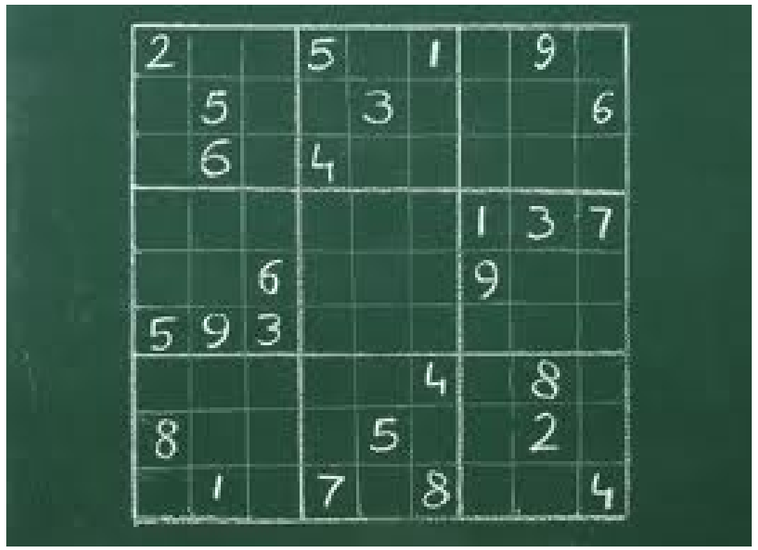
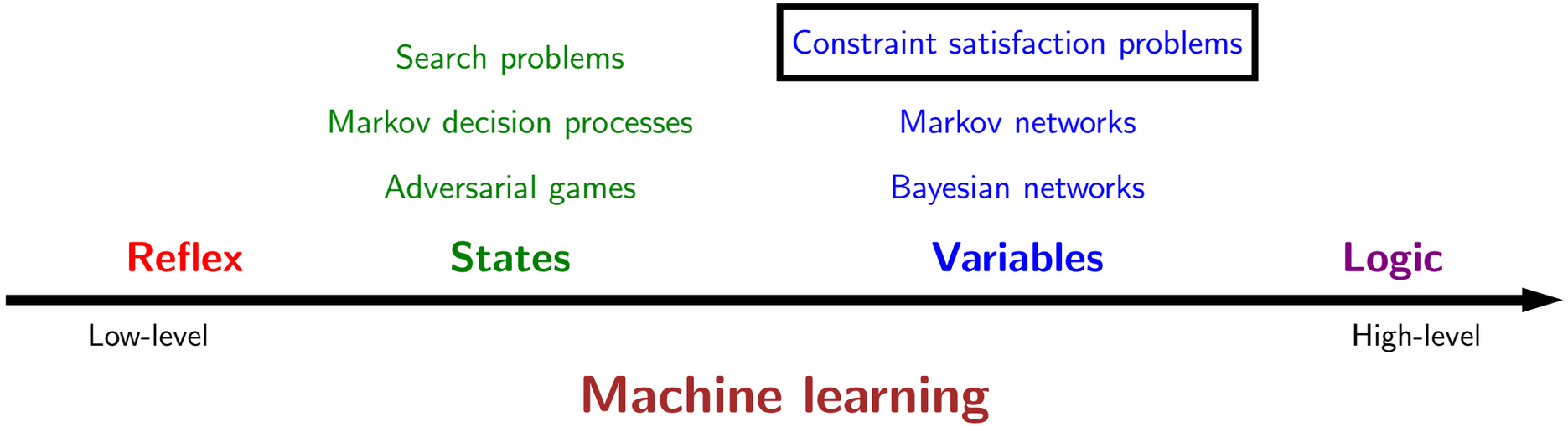
CSPs: overview



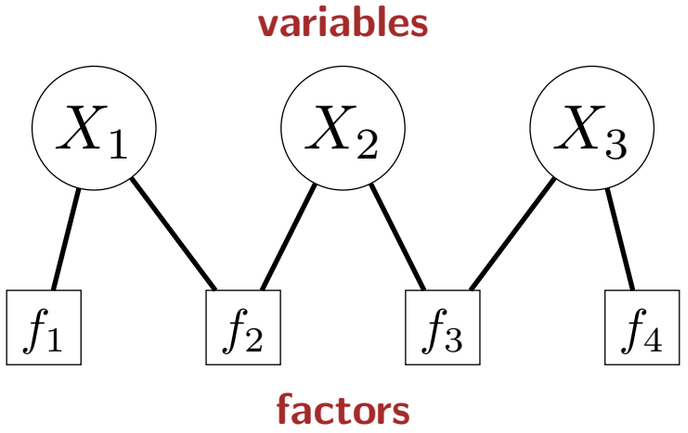
- In this module, I will introduce constraint satisfaction problems (CSPs).

Course plan


Search problems
Markov decision processes
Adversarial games
Constraint satisfaction problems
Markov networks
Bayesian networks
Reflex
States
Variables
Logic
Low-level
High-level
Machine learning

- We started with machine learning and reflex-based models, which simply produce a single output or action (classification or regression).
- Then we looked at state-based models, where we thought in terms of states, actions, and costs/rewards.
- Now we embark on our journey through **variable-based models**, a different modeling language, in which we will think in terms of variables, factors, and weights.

Factor graphs



Objective: find the best assignment of values to the variables

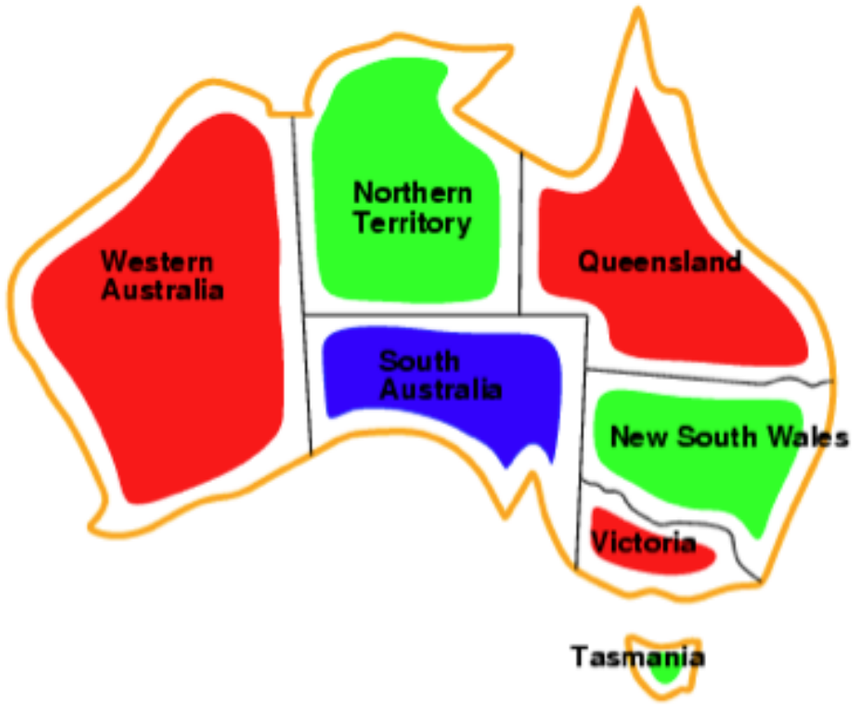
- All variable-based models have an underlying **factor graph**. Before formally defining what a factor graph is, let me first provide some intuition.
- A factor graph contains a set of **variables** (circle nodes), which represent unknown values that we seek to ascertain, and a set of **factors** (square nodes), which determine how the variables are related to one another.
- The objective of a constraint satisfaction problem is to 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?

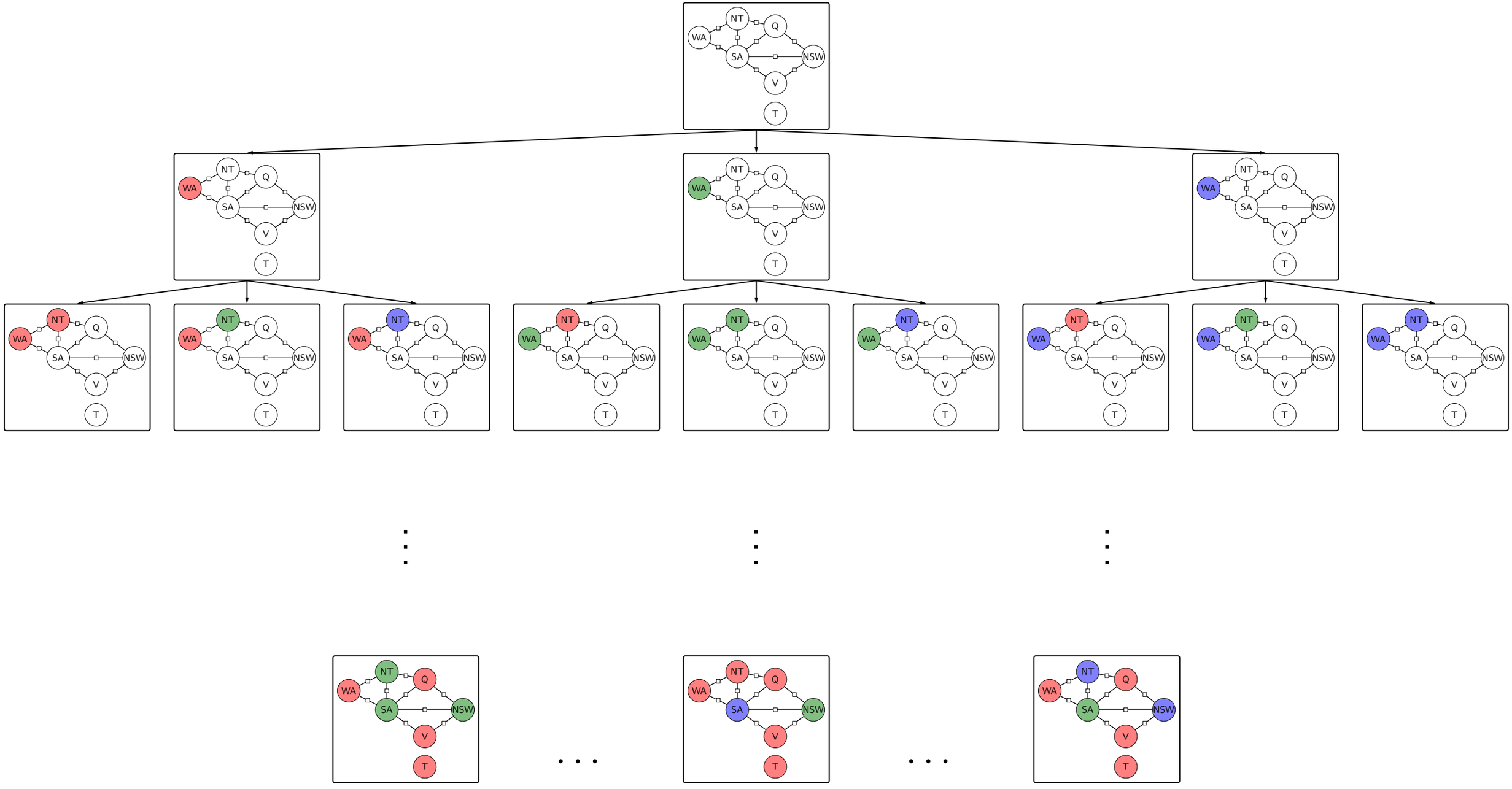
- Let us consider an example problem: map coloring.
- Here's Australia. It has 7 provinces, which might be hard to see, so let's color the provinces. How can we color the provinces with three colors so that no two neighboring provinces have the same color?

Map coloring



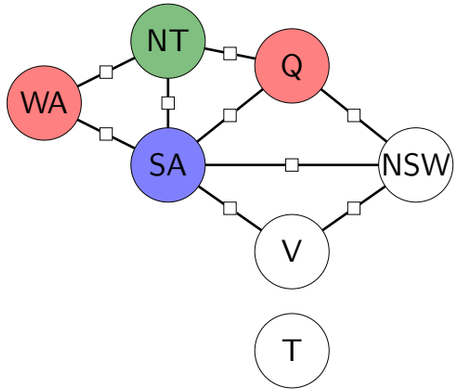
(one possible solution)

- Here is one solution.


NT
Q
WA
SA
NSW
V
T
0
0
1

- How do we solve this problem algorithmically? Let's use the hammer that we know: casting it as a search problem.
- We start with the state in which no colors are assigned. The possible actions from this state are to color one of the variables (WA) some color.
- In general, each state contains an assignment of colors to a subset of the provinces (a **partial assignment**), and each action corresponds to choosing a color for the next unassigned province.
- The leaves of the search tree are complete assignments, where every province has a color.
- Each leaf is either consistent — i.e., all neighboring provinces have different colors (1), or not (0).
- We then simply return any leaf that is consistent.

As a search problem

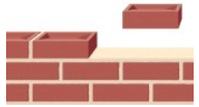


- State: partial assignment of colors to provinces
- Action: assign next uncolored province a compatible color

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

- This is a fine way to solve this problem, and in general, it shows how powerful search problems are: we don't actually need any new machinery to color Australia. But the question is: can we do better?
- First, **the order in which we assign variables doesn't matter for correctness**. This gives us the flexibility to dynamically choose a better ordering of the variables. That, with a bit of lookahead will allow us to dramatically improve the efficiency over naive tree search.
- Second, it's clear that Tasmania's color can be any of the three colors regardless of the colors on the mainland. This is an instance of **independence**, and later we'll see how to exploit this observation.



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.

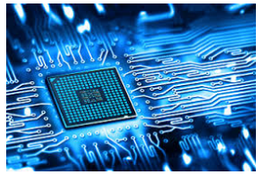
Higher-level modeling language than state-based models

- Varibale-based models allow us to capture this additional structure. Variable-based models is an umbrella term that includes constraint satisfaction problems (CSPs), Markov networks, and Bayesian networks.
- Aside: The term graphical models can be used interchangeably with variable-based models, and the term probabilistic graphical models (PGMs) generally encompasses both Markov networks (also called undirected graphical models) and Bayesian networks (directed graphical models).
- The unifying theme is the idea of thinking about solutions to problems as assignments of values to variables (this is the modeling part). All the details about how to find the assignment (in particular, which variables to try first) are delegated to the inference algorithm. So the advantage of using variable-based models over state-based models is that it's making the algorithms do more of the work, freeing up more time for modeling.
- An (imperfect) analogy is programming languages. Solving a problem directly by implementing an ad-hoc program is like using assembly language. Solving a problem using state-based models is like using C. Solving a problem using variable-based models is like using Python. By moving to a higher language, you might forgo some amount of ability to optimize manually, but the advantage is that (i) you can think at a higher level and (ii) there are more opportunities for optimizing automatically.
- Once a new modeling framework become second nature, it is almost as if it was invisible. It's like when you master a language, you can "think" in it without thinking about the language.

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

- Constraint satisfaction problems appear in many applications, most of which involve large-scale logistics, scheduling, and supply-chain management.
- Companies such as Amazon have to figure out how to put packages on vehicles to **deliver** them to customers to minimize cost and meet delivery times promised to the customer. Here, the variables include the assignment of packages to vehicles, and the factors encode travel times and costs. Ride-sharing serivces such as Uber and Lyft also have to figure out how to best assign drivers to riders. There are all extensions of the classic vehicle routing problem (VRP).
- Each year, the NFL has to make a **schedule** of which teams play what other teams and when. The schedule should minimize travel, fit into TV broadcast slots, be fair across teams, etc. Other scheduling problems involve assigning the courses that are offered one quarter to various classrooms at various time slots.
- A final application is **formal verification** of circuits and programs. Here, the variables are the unknown inputs to a program, and the factors encode the program/circuit execution. Then you can ask the question of whether there exists any program inputs that produce an error or incorrect result.

Roadmap

Modeling

Definitions

Examples

Backtracking (exact) search

Dynamic ordering

Arc consistency

Approximate search

Beam search

Local search

- Here's the roadmap for the rest of the modules on CSPs. First we will define constraint satisfaction problems and factor graphs formally, and give a few examples of CSPs.
- We then talk about backtracking search, which solves the problem exactly, though it takes exponential time in the worst case. To speed up search, we can take advantage of the fact that we can assign variables in any order to do dynamic ordering, where we heuristically figure out which variables to assign first. Arc consistency provides an lookahead algorithm called AC-3 to eagerly prune the search space, so that dynamic ordering can be more effective.
- Sometimes, you might not want to wait an exponential amount of time. If a crude solution suffices, one can apply approximate search algorithms. **Beam search** heuristically explores a small fraction of the exponentially-sized search tree, while **local search** takes an initial assignment and iteratively tries to improve it by changing one variable at a time.