Automated (AI) Planning

Autonomous Systems

Carmel Domshlak

Prerequisites

Course prerequisites:

- foundations of AI: search, heuristic search
- propositional logic: syntax and semantics
- computational complexity theory: decision problems, reductions, NP-completeness

Outline

The focus of the course is on Artificial Intelligence planning (= domain-independent planning) techniques

1. What planning problems are and why they are interesting?
2. The "Holy Triangle" of AI problem solving
3. Too hard or Too easy?, or Can this all be any practical?

What is AI?

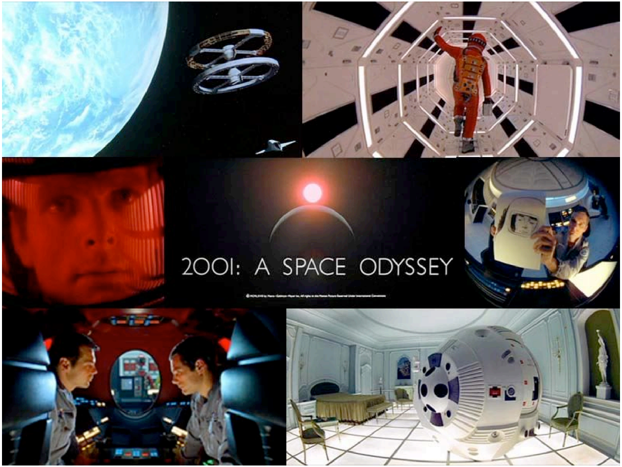
Two of somewhat more pragmatic attempts

The study of mental faculties through the use of computational models.
(E. Charniak & D. McDermott)

The science concerned with understanding intelligent behavior by attempting to create it in the artificial.
(T. Smithers)

- Intelligent behavior can be considered (postulated?) as ability to solve problems for which the machine has no knowledge of an suitable algorithm

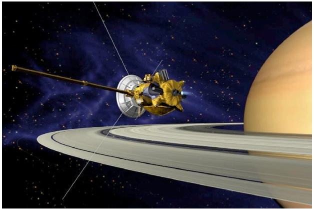
Why do we need such an AI?



NASA Experience

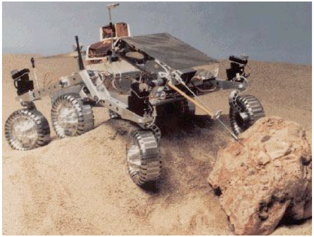
Galileo Jupiter or Cassini Saturn missions

- $1G budget
- Ground crew of 100-300 personnel



Mars micro-rover Sojourne

- $100M budget
- Small (and tired!) ground teams



Sojourne operated for two month, but future robots are expected to operate **much** longer!

NASA Vision

Space-exploring systems should be

- Low-cost and rapid development, low-cost control
- Autonomous operation for long periods of time
- Autonomous operation must guarantee success, given tight deadlines and resource constraints

Utopy?

NASA Vision

Space-explorating systems should be

- Low-cost and rapid development, low-cost control
- Autonomous operation for long periods of time
- Autonomous operation must guarantee success, given tight deadlines and resource constraints

Utopy? Not really. First progress in this direction has been accomplished in 1998 in the scope of the Deep Space One project!

Planning Problems

A sample of problems:

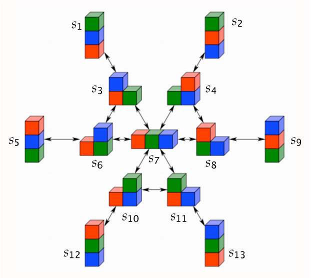
- Solving Rubik's cube (or 15-puzzle, or ...)
- Selecting and ordering movements of an elevator or a crane
- Scheduling of production lines
- Autonomous robots
- Crisis management
- ...

What is in common?

Planning Problems

What is in common?

- All these problems deal with action selection or control
- Some notion of problem state
- (Often) specification of initial state and/or goal state
- Legal moves or actions that transform states into other state



Planning Problems

For now focus on:

- Plans (aka solutions) are sequences of moves that transform the initial state into the goal state
- Intuitively, not all solutions are equally desirable

What is our task?

1. Find out whether there is a solution
2. Find any solution
3. Find an optimal (or near-optimal) solution
4. Fixed amount of time, find best solution possible
5. Find solution that satisfy property $\aleph$ (what is $\aleph$? you choose!)

Planning Problems

What is our task?

1. Find out whether there is a solution
2. Find any solution
3. Find an optimal (or near-optimal) solution
4. Fixed amount of time, find best solution possible
5. Find solution that satisfy property $\aleph$ (what is $\aleph$? you choose!)

- ♠ While all these tasks sound related, they are *very different*. The techniques best suited for each one are almost disjoint.
- In AI planning, (1) is usually assumed not to be an issue. (In contrast, in formal verification this is the central issue.)

Planning vs. Scheduling

Closely related but conceptually different problems

Scheduling

Deciding when to perform a given set of actions

- Time constraints
- Resource constraints
- Global constraints (e.g., regulatory issues)
- Objective functions

Planning

Deciding what actions to perform (and when) to achieve a given objective

- same issues

The difference comes in play in solution techniques, and actually even in worst-case time/space complexity

Planning and Action Selection in AI

Three approaches in AI (*in general?*) to the problems of action selection or control

- *Learning*: learn control from experience
- *Programming*: specify control by hand
- *Planning*: specify problem by hand, derive control automatically

All three have strengths and weaknesses; approaches not exclusive and often complementary.

Planning is a form of **general problem solving**

Three Key Ingredients of Planning
... and of AI approach to problems in general?

Planning is a form of general problem solving

$$\texttt{Problem} \Longrightarrow \texttt{Language} \Longrightarrow \texttt{Planner} \Longrightarrow \texttt{Solution}$$

1. **models** for defining, classifying, and understanding problems
   - what is a *planning problem*
   - what is a *solution* (*plan*), and
   - what is an *optimal solution*
2. **languages** for representing problems
3. **algorithms** for solving them

Three Key Ingredients of Planning

... and of AI approach to problems in general?

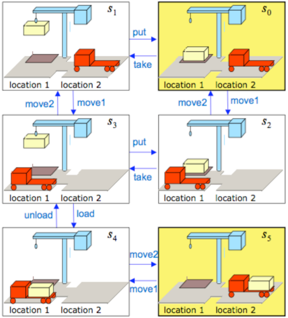
Planning is a form of general problem solving

`Problem` ⟹ `Language` ⟹ `Planner` ⟹ `Solution`

1. **models** for defining, classifying, and understanding problems
   - what is a *planning problem*
   - what is a *solution* (*plan*), and
   - what is an *optimal solution*
2. **languages** for representing problems
3. **algorithms** for solving them

State model for Classical AI Planning

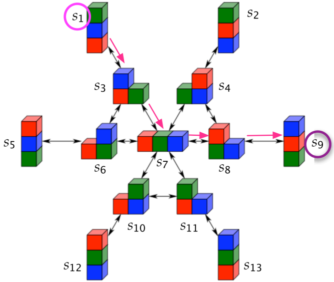
- finite state space S
- an initial state $s_0 \in S$
- a set $S_G \subseteq S$ of goal states
- applicable actions $A(s) \subseteq A$ for $s \in S$
- a transition function $s' = f(a, s)$ for $a \in A(s)$
- a cost function $c : A^* \rightarrow [0, \infty)$



A **solution** is a sequence of applicable actions that maps s_0 into S_G
An **optimal solution** minimizes c

Why planning is difficult?

- Solutions to planning problems are paths from an initial state to a goal state in the transition graph
- Dijkstra's algorithm solves this problem in $O(|V|\log(|V|) + |E|)$
- Can we go home??

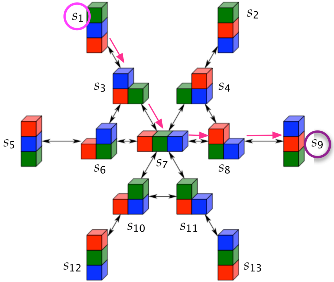


Why planning is difficult?

- Solutions to planning problems are paths from an initial state to a goal state in the transition graph
- Dijkstra's algorithm solves this problem in $O(|V| \log(|V|) + |E|)$
- Can we go home??

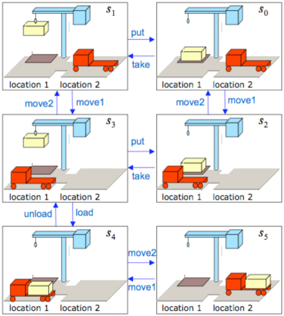
♠ Not exactly ⇒ $|V|$ of our interest is 10^{10}, 10^{20}, 10^{100}, ...

- *But do we need such values of $|V|$?!*



Why planning is difficult?

- Generalize the earlier example:
  - Five locations, three robot carts, 100 containers, three piles
  - $|V| \approx 10^{277}$
- The number of atoms in the universe is only about 10^{87}
  - The state space in our example is more than 10^{109} times as large (uppss ...)



And solving such a problem is not hopeless!

Beyond Classical Planning

Adding into the model

- Uncertainty about initial state and action outcomes
- Infinite state spaces (resources, time, ...)
- Continuous state spaces (resources, time, ...)
- Complex models of solution, and solution optimality
- Interleaving planning and execution
- ...

Side comment ...

- It is not that classical planning is easy
- It is not even clear that it is too far from modeling and/or solving real-world problems well!

Different classes of problems

- dynamics: deterministic, nondeterministic or probabilistic
- observability: full, partial, or none
- horizon: finite or infinite
- ...

1. classical planning
2. conditional planning with full observability
3. conditional planning with partial observability
4. conformant planning
5. Markov decision processes (MDP)
6. partially observable MDPs (POMDP)

Properties of the world: dynamics

Deterministic dynamics

Action + current state uniquely determine successor state.

Nondeterministic dynamics

For each action and current state there may be several possible successor states.

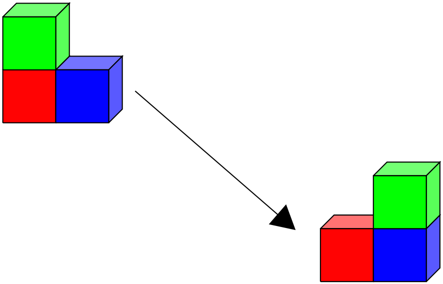
Probabilistic dynamics

For each action and current state there is a probability distribution over possible successor states.

Analogy: deterministic versus nondeterministic automata

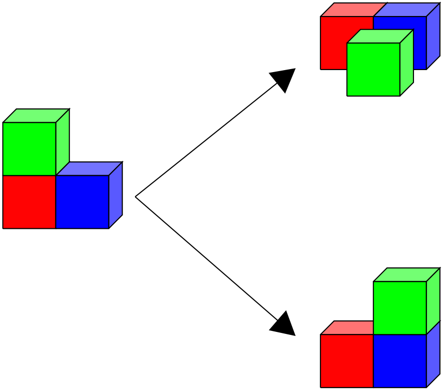
Determistic dynamics example

Moving objects with a robotic hand:
move the green block onto the blue block.



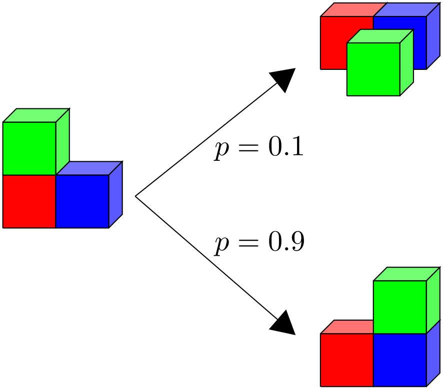
Nondetermistic dynamics example

Moving objects with an unreliable robotic hand: move the green block onto the blue block.



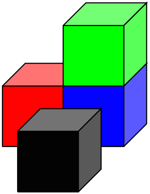
Probabilistic dynamics example

Moving objects with an unreliable robotic hand:
move the green block onto the blue block.

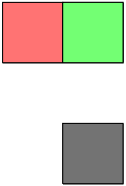


Properties of the world: observability

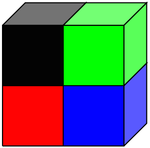
Camera A



Camera B



Goal



Properties of the world: observability

Full observability

Observations/sensing determine current world state uniquely.

Partial observability

Observations determine current world state only partially: we only know that current state is one of several possible ones.

No observability

There are no observations to narrow down possible current states. However, can use knowledge of action dynamics to deduce which states we might be in.

Consequence: If observability is not full, must represent the knowledge an agent has.

Different objectives

1. Reach a goal state.
   - Example: Earn 500 euro.
2. Stay in goal states indefinitely (infinite horizon).
   - Example: Never allow the bank account balance to be negative.
3. Maximize the probability of reaching a goal state.
   - Example: To be able to finance buying a house by 2018 study hard and save money.
4. Collect the maximal *expected* rewards/minimal expected costs (infinite horizon).
   - Example: Maximize your future income.
5. ...

Relation to games and game theory

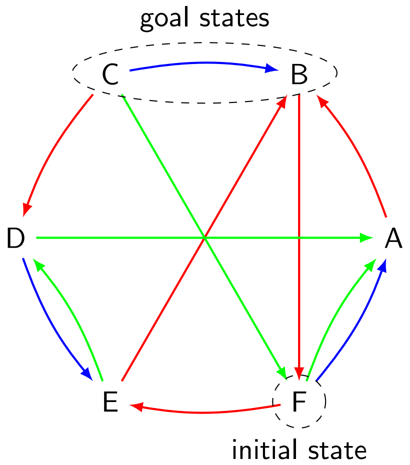
- Game theory addresses decision making in multi-agent setting: "Assuming that the other agents are rational, what do I have to do to achieve my goals?"
- Game theory is related to multi-agent planning.
- I will concentrate on single-agent planning.
- Some of the techniques are also applicable to special cases of multi-agent planning.
  - Example: Finding a winning strategy of a game like chess. In this case it is not necessary to distinguish between an intelligent opponent and a randomly behaving opponent.
- Game theory in general is about optimal strategies which do not necessarily guarantee winning. For example card games like poker do not have a winning strategy.

Where classical planning stands?

- dynamics: deterministic, nondeterministic or probabilistic
- observability: full, partial or none
- horizon: finite or infinite
- ...

1. classical planning
2. conditional planning with full observability
3. conditional planning with partial observability
4. conformant planning
5. Markov decision processes (MDP)
6. partially observable MDPs (POMDP)

Transition systems



Transition systems

Formalization of the dynamics of the world/application

Definition (transition system)

A transition system is $\langle S, I, \{a_1, \ldots, a_n\}, G\rangle$ where

- S is a finite set of states (the state space),
- $I \subseteq S$ is a finite set of initial states,
- every action $a_i \subseteq S \times S$ is a binary relation on S,
- $G \subseteq S$ is a finite set of goal states.

Definition (applicable action)

An action a is applicable in a state s if sas' for at least one state s'.

A transition system is deterministic if there is only one initial state and all actions are deterministic. Hence all future states of the world are completely predictable.

Definition (deterministic transition system)

A deterministic transition system is $\langle S, I, O, G \rangle$ where

- S is a finite set of states (the state space),
- $I \in S$ is a state,
- actions $a \in O$ (with $a \subseteq S \times S$) are partial functions,
- $G \subseteq S$ is a finite set of goal states.

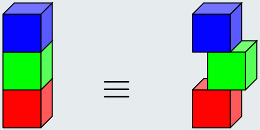
Successor state wrt. an action

Given a state s and an action a so that a is applicable in s, the successor state of s with respect to a is s' such that sas', denoted by $s' = app_a(s)$.

Blocks world
The rules of the game

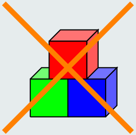
Location on the table does not matter.

Location on a block does not matter.

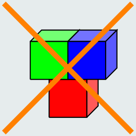


Blocks world
The rules of the game

At most one block may be below a block.

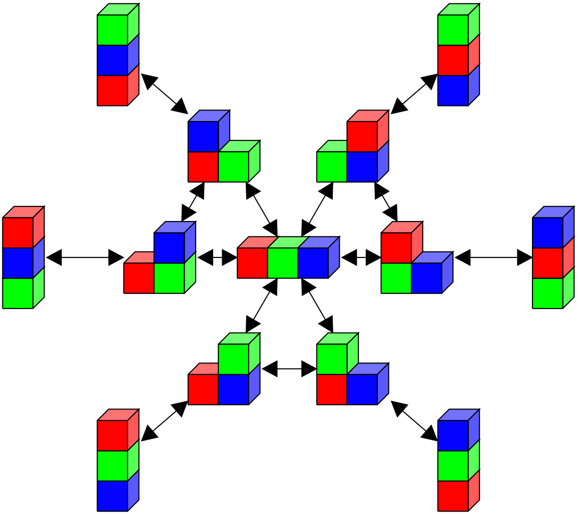


At most one block may be on top of a block.



Blocks world

The transition graph for three blocks



Blocks world

Properties

blocks	states
1	1
2	3
3	13
4	73
5	501
6	4051
7	37633
8	394353
9	4596553
...	
19	13564373693588558173

1. Finding a solution is polynomial time in the number of blocks (move everything onto the table and then construct the goal configuration).
2. Finding a shortest solution is NP-complete (for a compact description of the problem).

Deterministic planning: plans

Definition (plan)

A plan for $\langle S, I, A, G\rangle$ is a sequence $\pi = a_1, \dots, a_n$ of action instances such that $a_1, \dots, a_n \in A$ and $s_0, \dots, s_n$ is a sequence of states (the execution of π) so that

1. $s_0 = I$,
2. $s_i = app_{a_i}(s_{i-1})$ for every $i \in \{1, \dots, n\}$, and
3. $s_n \in G$.

This can be equivalently expressed as

$$app_{a_n}(app_{a_{n-1}}(\dots app_{a_1}(I) \dots)) \in G$$

Three Key Ingredients of Planning

... and of AI approach to problems in general?

Planning is a form of general problem solving

$$\texttt{Problem} \Longrightarrow \texttt{Language} \Longrightarrow \texttt{Planner} \Longrightarrow \texttt{Solution}$$

1. **models** for defining, classifying, and understanding problems
   - what is a *planning problem*
   - what is a *solution* (*plan*), and
   - what is an *optimal solution*
2. **languages** for representing problems
3. **algorithms** for solving them

Succinct representation of transition systems

- More compact representation of actions than as relations is often
  - possible because of symmetries and other regularities,
  - unavoidable because the relations are too big.
- Represent different aspects of the world in terms of different state variables. $\rightsquigarrow$ A state is a valuation of state variables.
- Represent actions in terms of changes to the state variables.

State variables

- The state of the world is described in terms of a finite set of finite-valued state variables.

Example

hour: $\{0, \ldots, 23\} = 13$
minute: $\{0, \ldots, 59\} = 55$
location: $\{51, 52, 82, 101, 102\} = 101$
weather: $\{\text{sunny}, \text{cloudy}, \text{rainy}\} = \text{cloudy}$
holiday: $\{\text{T}, \text{F}\} = \text{F}$

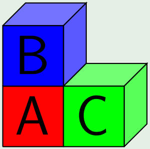
- Any n-valued state variable can be replaced by $\lceil \log_2 n \rceil$ Boolean (2-valued) state variables.
- Actions change the values of the state variables.

Blocks world with state variables

State variables:
location-of-A: {B, C, table}
location-of-B: {A, C, table}
location-of-C: {A, B, table}

Example

$$s(\textit{location-of-A}) = \text{table}$$
$$s(\textit{location-of-B}) = \text{A}$$
$$s(\textit{location-of-C}) = \text{table}$$

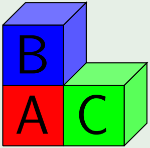


Not all valuations correspond to an intended blocks world state, e. g. s such that $s(\textit{location-of-A}) = \text{B}$ and $s(\textit{location-of-B}) = \text{A}$.

Blocks world with Boolean state variables

Example

$$s(\textit{A-on-B}) = 0$$
$$s(\textit{A-on-C}) = 0$$
$$s(\textit{A-on-table}) = 1$$
$$s(\textit{B-on-A}) = 1$$
$$s(\textit{B-on-C}) = 0$$
$$s(\textit{B-on-table}) = 0$$
$$s(\textit{C-on-A}) = 0$$
$$s(\textit{C-on-B}) = 0$$
$$s(\textit{C-on-table}) = 1$$



Deterministic planning tasks

Definition (deterministic planning task)

A deterministic planning task is a 4-tuple $\Pi = \langle V, I, A, G \rangle$ where

- V is a finite set of state variables,
- I is an initial state over V,
- A is a finite set of actions over V, and
- G is a constraint ($=$ formula) over V describing the goal states.

Notes:

- Unless stated otherwise, G will be a single partial assignment to V
- We will omit the word "deterministic" where it is clear from context.

Mapping planning tasks to transition systems

From every deterministic planning task $\Pi = \langle V, I, A, G \rangle$ we can produce a corresponding transition system $\mathcal{T}(\Pi) = \langle S, I, A', G' \rangle$:

1. S is the set of all valuations of V,
2. $A' = \{R(a) \mid a \in A\}$ where $R(a) = \{(s, s') \in S \times S \mid s' = \mathit{app}_a(s)\}$, and
3. $G' = \{s \in S \mid s \models G\}$.

Planning Languages

Key issue

Models represented **implicitly** in a **declarative language**

Play two roles

- **specification**: concise model description
- **computation**: reveal useful info about problem's *structure*

The SAS Language

A problem in SAS is a tuple $\langle V, A, I, G \rangle$

- V is a finite set of state variables with finite domains $dom(v_i)$
- I is an initial state over V
- G is a partial assignment to V
- A is a finite set of actions a specified via $\text{pre}(a)$ and $\text{eff}(a)$, both being partial assignments to V

- An action a is applicable in a state $s \in dom(V)$ iff $s[v] = \text{pre}(a)[v]$ whenever $\text{pre}(a)[v]$ is specified
- Applying an applicable action a changes the value of each variable v to $\text{eff}(a)[v]$ if $\text{eff}(a)[v]$ is specified.
- Example: 8-puzzle

The STRIPS language
Useful fragment of SAS

A problem in STRIPS is a tuple $\langle P, A, I, G \rangle$

- P stands for a finite set of **atoms** (boolean vars)
- $I \subseteq P$ stands for **initial situation**
- $G \subseteq P$ stands for **goal situation**
- A is a finite set of **actions** a specified via pre(a), add(a), and del(a), all subsets of P

- States are collections of atoms
- An action a is applicable in a state s iff pre$(a) \subseteq s$
- Applying an applicable action a at s results in $s' = (s \setminus \text{del}(a)) \cup \text{add}(a)$

Why STRIPS is interesting

- STRIPS operators are particularly simple, yet expressive enough to capture general planning problems.
- In particular, STRIPS planning is no easier than general planning problems.
- Many algorithms in the planning literature are easier to present in terns of STRIPS .

Three Key Ingredients of Planning

... and of AI approach to problems in general?

Planning is a form of general problem solving

Problem $\Longrightarrow$ Language $\Longrightarrow$ Planner $\Longrightarrow$ Solution

1. **models** for defining, classifying, and understanding problems
2. **languages** for representing problems
3. **algorithms** for solving them
   - NEXT: algorithms for **classical planning**
     where a significant progress has been recently achieved

More on the Motivation

Planning is a form of general problem solving

$$\texttt{Problem} \Longrightarrow \texttt{Language} \Longrightarrow \texttt{Planner} \Longrightarrow \texttt{Solution}$$

Modeling Time vs. Solution Time and Quality

- specialized methods are typically more efficient (though even that is not necessarily correct), but tend to require lots of programming
- goal in AI problem solving is to **facilitate modeling** and yet provide **efficient solutions**
- this involves **general languages** (*a la* SAS or STRIPS) and thus **language-specific algorithms**

What do you learn in this course?

- algorithms for solving different problem classes, with an emphasis on the classical ("simplest") setting:
  - algorithms based on heuristic search in state space
  - algorithms based on heuristic search in plan space
  - algorithms based on satisfiability testing (SAT) and general constraint satisfaction (CSP)

  Many of these techniques are applicable to problems outside AI as well.
- hands-on experience with problem modeling and (mostly classical) planners