Automated Action Planning

Introduction

Carmel Domshlak

Automated Action Planning

— Introduction

Prerequisites

Course prerequisites:

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

See the complementary "background" set of slides.

Outline

The course is on *computational* aspects of physical autonomous systems, and in particular on AI techniques developed for

- Goal-oriented planning of action.

Focus on generic, domain-independent techniques.

Autonomous Systems

A sample of problems:

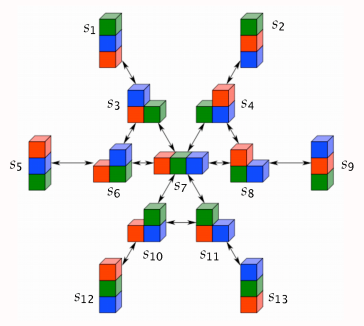
- Solving Rubik's cube (or 15-puzzle, or ...)
- Selecting and ordering movements of an elevator or a crane
- Scheduling of production lines
- Autonomous robots
- Crisis management
- ...

What is in common?

Autonomous Systems

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



Action Selection in AI

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

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

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

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!)

♠ 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

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

Three Key Ingredients of Planning

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

Planning is a form of general problem solving

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

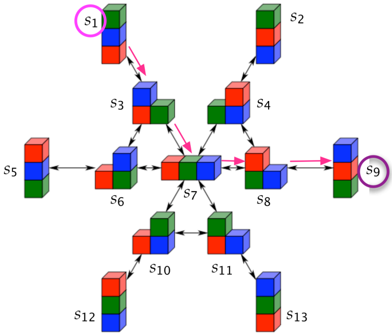
1. **models** for defining, classifying, and understanding problems
   - what is a *planning problem*
   - what is a *solution* (*plan*), and
   - what is an *optimal solution*
2. **languages** for representing problems
3. **algorithms** for solving them

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 $\Rightarrow$ $|V|$ of our interest is 10^{10}, 10^{20}, 10^{100}, ...

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



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.

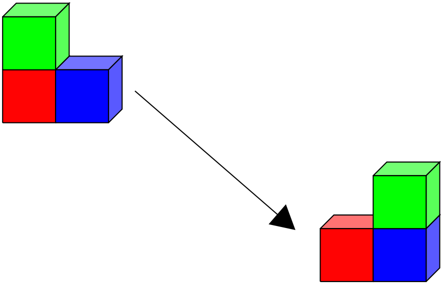
Probabilistic dynamics

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

Analogy: deterministic versus nondeterministic automata

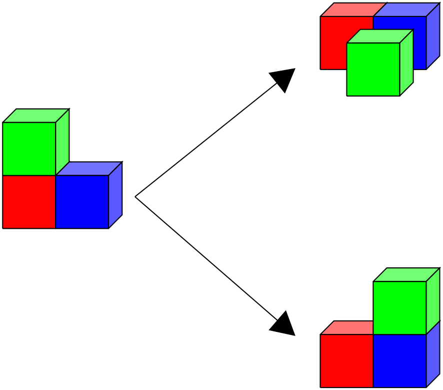
Deterministic dynamics example

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



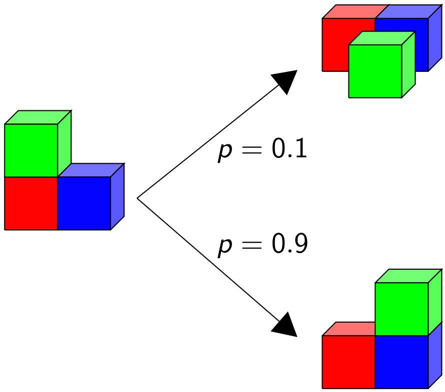
Nondetermistic dynamics example

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



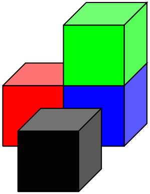
Probabilistic dynamics example

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

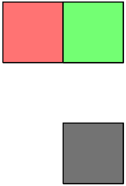


Properties of the world: observability

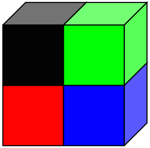
Camera A



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.

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

Formalization of the dynamics of the world/application

Definition (transition system)

A transition system is $\langle S, I, \{a_1, \dots, a_n\}, G \rangle$ 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'.

Transition systems

Deterministic transition systems

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)

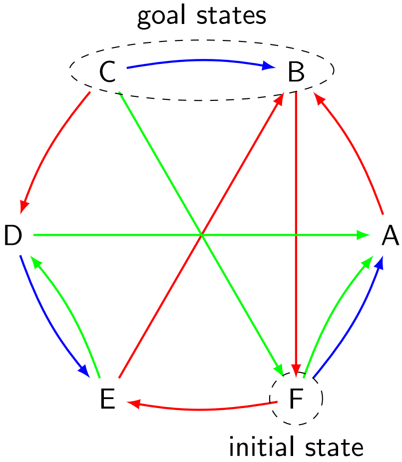
A deterministic transition system is $\langle S, I, A, G \rangle$ where

- S is a finite set of states (the state space),
- $I \in S$ is a state,
- actions $a \in A$ (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 $s' = app_a(s)$.

Transition systems



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

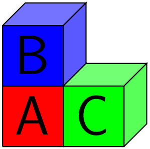
$vhour$: $\{0, \ldots, 23\} = 13$
$vminute$: $\{0, \ldots, 59\} = 55$
$vlocation$: $\{51, 52, 82, 101, 102\} = 101$
$vweather$: $\{dsunny, dcloudy, drainy\} = dcloudy$
$vholiday$: $\{dT, dF\} = dF$

Blocks world with state variables

State variables:
vlocation-of-A: $\{dB, dC, dtable\}$
vlocation-of-B: $\{dA, dC, dtable\}$
vlocation-of-C: $\{dA, dB, dtable\}$

Example

$$s(vlocation - of - A) = dtable$$
$$s(vlocation - of - B) = dA$$
$$s(vlocation - of - C) = dtable$$

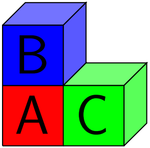


Not all valuations correspond to an intended blocks world state, e. g. s such that $s(vlocation - of - A) = dB$ and $s(vlocation - of - B) = dA$.

Blocks world with Boolean state variables

Example

$$
\begin{aligned}
s(vA - on - B) &= 0\\
s(vA - on - C) &= 0\\
s(vA - on - table) &= 1\\
s(vB - on - A) &= 1\\
s(vB - on - C) &= 0\\
s(vB - on - table) &= 0\\
s(vC - on - A) &= 0\\
s(vC - on - B) &= 0\\
s(vC - on - table) &= 1
\end{aligned}
$$



The FDR Language

Also known as SAS

A problem in FDR 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.

Three Key Ingredients of Planning

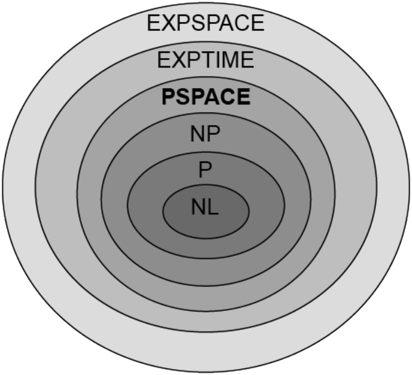
... and of AI approach to problems in general?

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
2. **languages** for representing problems
3. **algorithms** for solving them
   - NEXT: algorithms for **classical planning** where a significant progress has been recently achieved

Planning Tasks and Worst-Case Complexity



non-deterministic + NO $\leadsto$ EXPSPACE-complete

non-deterministic + FO $\leadsto$ EXPTIME-complete

deterministic $\leadsto$ PSPACE-complete

bounded deterministic $\leadsto$ NP-complete

No efficient algorithm $\mapsto$
Search techniques + Language-specific "know-hows"

Solving Problems Intelligently

Quote by a Famous Computer Scientist in a Famous Book

"How, then, are we to construct an intelligent problem-solver?

It appears that the clue to intelligent behavior, whether of men or machines, is highly selective search, the drastic pruning of the tree of possibilities explored.

For a computer program to behave intelligently, it must search problem mazes in a highly selective way, exploring paths relatively fertile with solutions and ignoring paths relatively sterile."

— Alan Turing, in: Computers and Thought (1963)

Planning as Heuristic Search

Planning as Heuristic Search

general search algorithm (e. g. A^*, greedy best-first search)
$+$ heuristic function ("heuristic")

- heuristic: estimates goal distance from the current situation
- challenge: precise domain-independent heuristics
- optimal planning: admissible (optimistic) heuristics

Where Do Heuristics Come From?

Of Heuristics and Academics

"Algorithms are conceived in analytic purity in the high citadels of academic research, heuristics are midwifed by expediency in the dark corners of the practitioner's lair."

— Fred Glover (1977)

It doesn't have to be this way!

In what comes next, we'll

- Develop a rigorous theory of heuristics for planning.
- In doing so, advance the practice of planning.

Where heuristics come from?

General idea

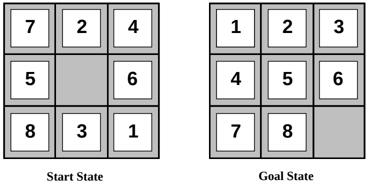
(Admissible) heuristic functions obtained as
(optimal) cost functions of relaxed problems

Examples

- Euclidian distance in Path Finding
- Manhattan distance in N-puzzle
- Spanning Tree in Traveling Salesman Problem
- Shortest Path in Job Shop Scheduling

Example

8-Puzzle



- A tile can move from square A to square B if A is adjacent to B and B is blank $\rightsquigarrow$ solution distance h^*
- A tile can move from square A to square B if A is adjacent to B $\rightsquigarrow$ manhattan distance heuristic h^{MD}
- A tile can move from square A to square B $\rightsquigarrow$ misplaced tiles heuristic h^{MT}

Here: $h^*(s_0) = ?$, $h^{MD}(s_0) = 14$, $h^{MT}(s_0) = 6$

In general, $h^* \geq h^{MD} \geq h^{MT}$. *(Why?)*

Are we solver?

General idea

(Admissible) heuristic functions obtained as (optimal) cost functions of relaxed problems

- OK, but heuristic is yet another input to our agent!
- Satisfactory for general solvers?
- Satisfactory in special purpose solvers?

Towards domain-independent agents

- How to get heuristics automatically?
- Can such automatically derived heuristics dominate the domain-specific heuristics crafted by hand?

A simple heuristic for deterministic planning

STRIPS (Fikes & Nilsson, 1971) used the number of state variables that differ in current state s and a STRIPS goal $G = \{g_1, \ldots, g_k\}$:

$$h(s) := |G \setminus s|.$$

Intuition: more true goal literals $\rightsquigarrow$ closer to the goal

$\rightsquigarrow$ STRIPS heuristic (properties?)

Criticism of the STRIPS heuristic

What is wrong with the STRIPS heuristic?

- quite uninformative:
  the range of heuristic values in a given task is small;
  typically, most successors have the same estimate
- very sensitive to reformulation:
  can easily transform any planning task into an equivalent one where
  $h(s) = 1$ for all non-goal states (how?)
- ignores almost all problem structure:
  heuristic value does not depend on the set of actions!

$\rightsquigarrow$ need a better, principled way of coming up with heuristics

Heuristics Toolbox

Just 15 years ago

Nothing, but “STRIPS heuristic” (missing goals counting).

- HSP is considered natural yet hopeless approach to planning *(cf. R&N, ed1)*.
- Surprising, given successes of HS in AI back then ...

In (just) 15 years

HSP is considered a leading approach to planning
(cf. R&N, ed3).

Heuristics for Planning

How do we come up with heuristics for general planning tasks?

$\rightsquigarrow$ four major approaches in the literature:

- abstraction
- delete relaxation
- critical paths
- landmarks

Example: FreeCell

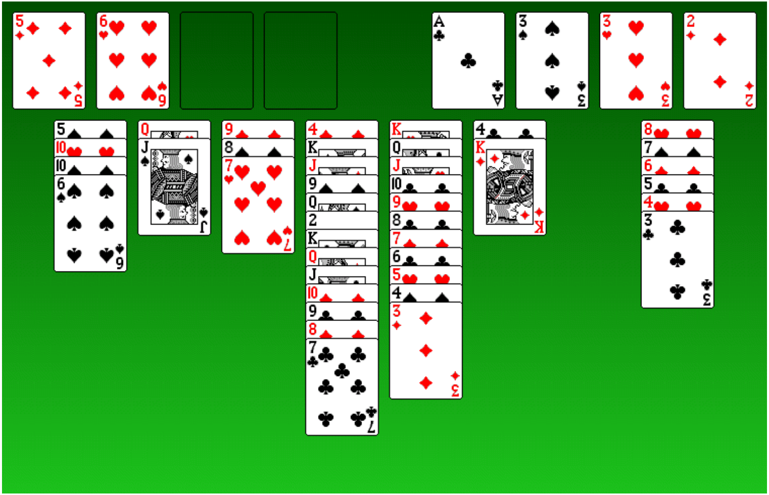


image credits: GNOME Project (GNU General Public License)

Planning Heuristics: Abstraction

Four classes of heuristics:

1. Abstraction

Estimate cost by projecting the state space to a smaller space (e.g., by applying a graph homomorphism).

Example: Abstraction in FreeCell

One possible abstraction for FreeCell:
project away all cards that are not 10s, Js, Qs or Ks.

Abstraction Heuristics in the Literature

Abstraction heuristics in the literature:

- pattern databases (PDBs) (Edelkamp, 2001; Haslum, Helmert, Bonet, Botea & Koenig, 2007)
- symbolic PDBs (Edelkamp, 2002)
- constrained PDBs (Haslum, Bonet & Geffner, 2005)
- merge-and-shrink (Helmert, Haslum & Hoffmann, 2007; Nissim, Hoffmann & Helmert, 2011)
- implicit abstractions (Katz & Domshlak, 2008)

Planning Heuristics: Delete Relaxation

Four classes of heuristics:

2. Delete Relaxation

Estimate cost to goal by considering simpler planning task without negative side effects of actions.

Example: Delete Relaxation in FreeCell

Problem constraints dropped by the delete relaxation in FreeCell:

- free cells and free tableau positions remain available after moving cards into them
- cards remain movable and remain valid targets for other cards after moving cards on top of them

Delete Relaxation Heuristics in the Literature

Delete relaxation heuristics in the literature:

- maximum heuristic h_{max} (Bonet & Geffner, 1999)
- additive heuristic h_{add}(Bonet & Geffner, 1999)
- FF heuristic, h^{+} heuristic (Hoffmann & Nebel, 2001)
- pairwise max heuristic (Mirkis & Domshlak, 2007)
- set-additive heuristic (Keyder & Geffner, 2008)
- Steiner tree heuristic (Keyder & Geffner, 2009)

Planning Heuristics: Critical Paths

Four classes of heuristics:

3. Critical Paths

Estimate cost as critical path length of a subgoal decomposition that ignores (or limits) dependencies between subgoals.

Example: Critical Paths in FreeCell

Possible critical path for single subgoals (h^1):

- Solving the FreeCell task requires four subgoals: have each of $\diamondsuit K$, $\heartsuit K$, $\spadesuit K$, $\clubsuit K$ at foundations
- Follow 3rd subgoal: getting $\spadesuit K$ to foundations requires first having $\spadesuit Q$ at foundations and having $\spadesuit K$ movable.
- Follow 2nd subsubgoal: having $\spadesuit K$ movable requires...

Critical Path Heuristics in the Literature

Critical path heuristics in the literature:

- $h^{(m)}$ heuristic family (Haslum & Geffner, 2000)
- additive $h^{(m)}$ (Haslum, Bonet & Geffner, 2005)
- additive-disjunctive heuristic graphs (Coles, Fox, Long & Smith, 2008)

Planning Heuristics: Landmarks

Four classes of heuristics:

4. Landmarks

An action set A is a landmark if all plans include an action from A.

Compute a set of landmarks and use its cardinality
(possibly modified by weights) as a cost estimate.

Example: Landmarks in FreeCell

Landmarks in FreeCell:

- The set of actions that move the $\heartsuit Q$ to foundations.
- The set of actions that move the $\clubsuit 7$ away from the $\diamondsuit 8$.
- ...

Landmark Heuristics in the Literature

Landmark heuristics in the literature:

- LAMA heuristic (Richter, Helmert & Westphal, 2008)
- cost-partitioned landmarks (Karpas & Domshlak, 2009)
- conjunctive landmarks (Keyder, Richter & Helmert, 2010)

Lets Dive into Details!