Principles of AI Planning

1. Introduction

Malte Helmert

Albert-Ludwigs-Universität Freiburg

October 21st, 2008

Principles of AI Planning

October 21st, 2008 — 1. Introduction

People

Lecturer

Dr. Malte Helmert

- email: helmert@informatik.uni-freiburg.de
- office: room 052-00-044
- consultation: by appointment (email)

Assistants

Gabi Röger and Patrick Eyerich

- email: roeger@informatik.uni-freiburg.de,
  eyerich@informatik.uni-freiburg.de
- office: room 052-00-030
- consultation: by appointment (email)

Time & Place

Lectures

- time: Tuesday 11:15-13:00, Friday 11:15-12:00
- place: SR 051-00-031

Exercises

- time: Friday 12:15-13:00
- place: SR 051-00-031

Web site

http://www.informatik.uni-freiburg.de/~ki/teaching/ws0809/aip/

- main page: course description
- lecture page: slides
- exercise page: assignments, model solutions, software
- bibliography page: literature references and papers

Teaching materials

- no textbook, no script
- slides handed out during lectures and available on the web
- additional resources: bibliography page on web + ask us!

Acknowledgment:

- slides based on earlier courses by Jussi Rintanen, Bernhard Nebel and Malte Helmert

Target audience

Students of Computer Science:

- Diploma, advanced study period ($\sim$4th year)
- Master of Science
- Bachelor of Science, $\sim$3rd year

Students of Applied Computer Science:

- Master of Science, $\sim$2nd year

Other students:

- advanced study period ($\sim$4th year)

Prerequisites

Course prerequisites:

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

Exams & grades

- 6 ECTS points
- special lecture in concentration subject
  Artificial Intelligence and Robotics
- oral exam for degree courses:
  - BSc in Computer Science
  - MSc in Applied Computer Science
- written exam for degree courses:
  - MSc in Computer Science
  - diploma in Computer Science
- neither Computer Science nor Applied Computer Science
  $\rightsquigarrow$ contact us!
- if too few written exam candidates, oral exam for everyone

Exercises

Exercises (written assignments):

- handed out on Tuesdays
- due Tuesday following week, before the lecture
- discussed Friday that week
- may solve in groups of two students ($2 \neq 3$)
- copied/plagiarized solutions: 0 marks
- may earn bonus marks for oral/written exam

Projects

Projects (programming assignments):

- handed out every now and then
- more time to work on than for exercises
- may solve in groups of two students ($2 = 2$)
- copied/plagiarized solutions: 0 marks
- language: Java (maybe open for some projects)
- solutions that obviously do not work: 0 marks
  - may fix bugs uncovered by our testing if still within submission deadline
- may earn bonus marks for oral/written exam

Bonus marks

- may earn up to 10 bonus marks in exercises
- may earn up to 10 bonus marks in projects
- $\rightsquigarrow$ max. possible: 20 bonus marks
- 10 bonus marks $\approx \frac{1}{3}$ grade improvement (e. g., 1.7 $\rightarrow$ 1.3)

Bonus marks from exercises

- compute total score percentage for the semester
- $\leq$ 50%: no bonus marks
- 1 bonus mark for each 5% above 50%

Bonus marks from projects

- no minimum requirement: each project directly yields a certain number of bonus marks
- max. bonus marks from projects capped at 10

What is planning?

- intelligent decision making: What actions to take?
- general-purpose problem representation
- algorithms for solving any problem expressible in the representation
- application areas:
  - high-level planning for intelligent robots
  - autonomous systems: NASA Deep Space One, ...
  - problem solving (single-agent games like Rubik's cube)

Why is planning difficult?

- solutions to classical planning problems are paths from an initial state to a goal state in the transition graph
  - efficiently solvable by Dijkstra's algorithm in $O(|V| \log |V| + |E|)$ time
  - Why don't we solve all planning problems this way?
- state spaces may be huge: $10^9, 10^{12}, 10^{15}, \dots$ states
  - constructing the transition graph is infeasible!
  - planning algorithms try to avoid constructing whole graph
- planning algorithms often are – but not guaranteed to be – more efficient than obvious solution methods constructing the transition graph and using e. g. Dijkstra's algorithm

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)

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)

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)

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)

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)

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)

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.

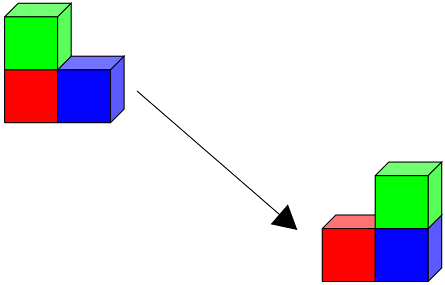
Probabilistic dynamics

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

Analogy: deterministic versus nondeterministic automata

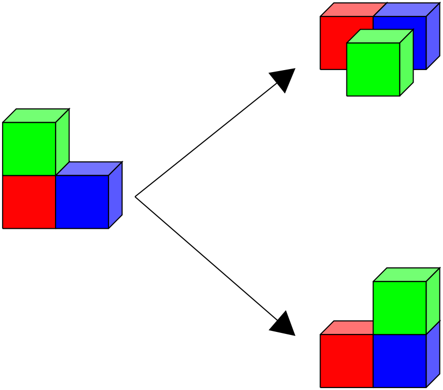
Determistic dynamics example

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



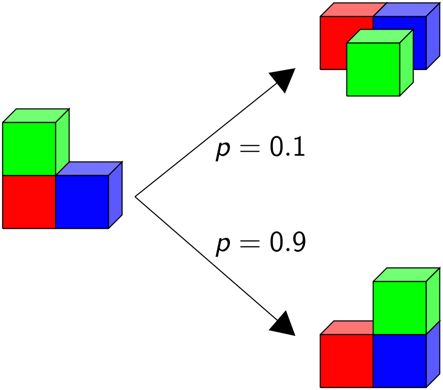
Nondetermistic dynamics example

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

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.

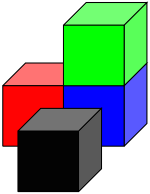
No observability

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.

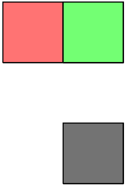
Consequence: If observability is not full, must represent the knowledge an agent has.

What difference does observability make?

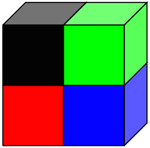
Camera A



Camera B



Goal



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.
- In this course we 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.

What do you learn in this course?

- "big picture" of different kinds of planning problems
  - classification according to dynamics, observability, objectives, ...
  - computational complexity for different problem classes
- algorithms for solving different problem classes,
  with an emphasis on the classical ("simplest") setting:
  - algorithms based on heuristic search
  - algorithms based on satisfiability testing (SAT)
  - algorithms based on exhaustive search with logic-based data structures (BDDs)

  Many of these techniques are applicable to problems outside AI as well.
- hands-on experience with a classical planner (optional)