Principles of AI Planning

1. Introduction

Bernhard Nebel and Robert Mattmüller

Albert-Ludwigs-Universität Freiburg

October 25th, 2011

About the course

People

Lecturers

Prof. Dr. Bernhard Nebel

- email: `nebel@informatik.uni-freiburg.de`
- office: room 052-00-029
- consultation: by appointment (email)

Robert Mattmüller

- email: `mattmuel@informatik.uni-freiburg.de`
- office: room 052-00-045
- consultation: by appointment (email) or just drop by in the office

People

Assistant

Thomas Keller

- email: `tkeller@informatik.uni-freiburg.de`
- office: room 052-00-030
- consultation: by appointment (email) or just drop by in the office

Tutor

Yusra Alkhazraji

- email: `yusra.alkhazraji@uranus.uni-freiburg.de`

Time & place

Lectures

- time: Tuesday 16:15-18:00, Friday 14:15-15:00
- place: SR 101-01-018

Exercises

- time: Friday 15:15-16:00
- place: SR 101-01-018

Web site

Course web site

`http://gki.informatik.uni-freiburg.de/teaching/ws1112/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!

Acknowledgments:

- slides based on earlier courses by Jussi Rintanen, Bernhard Nebel and Malte Helmert
- many figures by Gabi Röger

Target audience

Students of Computer Science:

- Master of Science, any year
- 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

Credit points & exam

- 6 ECTS points
- special lecture in concentration subject
  Artificial Intelligence and Robotics
- oral exam of about 30 minutes B.Sc. students
- written or oral exam for M.Sc. students
  (depending on their number)

Exercises

Exercises (written assignments):

- handed out on Tuesdays (exception: sheet 1 handed out this Friday instead of Tuesday next week because of All Saints' Day)
- due Tuesday following week, before the lecture
- discussed Friday that week
- may be solved in groups of two students ($2 \neq 3$)
- successful participation prerequisite for exam admission

Projects

Projects (programming assignments):

- handed out every now and then
  (probably three times over the course of the semester)
- more time to work on than for exercises
- may be solved in groups of two students ($2 = 2$)
- language: Python
- codebase: `https://bitbucket.org/malte/pyperplan`
- solutions that obviously do not work: 0 marks
  - may fix bugs uncovered by our testing
    if still within submission deadline
- successful participation prerequisite for exam admission

Admission to exam

- points can be earned for “reasonable” solutions to exercises and projects (one project counts like two exercise sheets).
- at least 50% of points prerequisite for admission to final exam.

Plagiarism

What is plagiarism?

- passing off solutions as your own that are not based on your ideas (work of other students, Internet, books, ...)
- `http://en.wikipedia.org/wiki/Plagiarism` is a good intro

Consequence: no admission to the final exam.

- We may (!) be generous on first offense.
- Don't tell us "We did the work together."
- Don't tell us "I did not know this was not allowed."

Introduction

What is planning?

Planning

"Planning is the art and practice of thinking before acting."
— Patrik Haslum

- 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: $10^{10}, 10^{100}, 10^{1000}, \ldots$ states
  - constructing the transition graph is infeasible!
  - planning algorithms try to avoid constructing whole graph
- planning algorithms are often much 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.

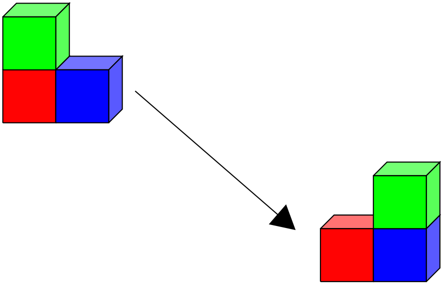
Probabilistic dynamics

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

Analogy: deterministic versus nondeterministic automata

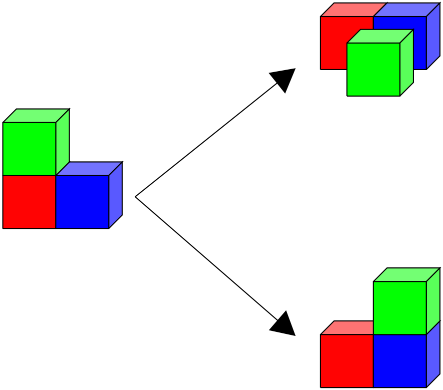
Deterministic dynamics example

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



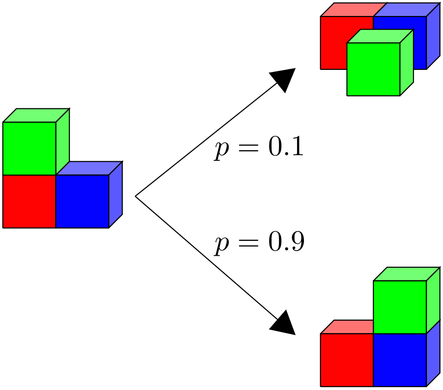
Nondeterministic dynamics example

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.

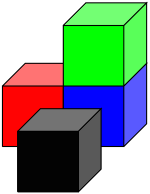
No observability

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.

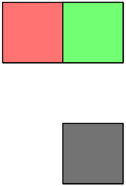
Consequence: If observability is not full, must represent the knowledge an agent has.

What difference does observability make?

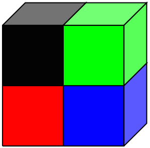
Camera A



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 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 2022 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?

- emphasis on classical planning ("simplest" case)
- theoretical background for planning
  - formal problem definition
  - basic theoretical notions
    (e. g., normal forms, progression, regression)
  - computational complexity of planning
- algorithms for planning:
  - based on heuristic search
  - based on satisfiability testing (SAT)
    (time permitting)

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