Principles of AI Planning

Introduction

Malte Helmert Bernhard Nebel

Albert-Ludwigs-Universität Freiburg

October 27th, 2006

Course: Principles of AI Planning

Lecturer

Dr. Malte Helmert (helmert@informatik.uni-freiburg.de)
Prof. Dr. Bernhard Nebel (nebel@informatik.uni-freiburg.de)

Lecture

Wednesday 2-4pm, Friday 2-3pm in 51-00-034

`www.informatik.uni-freiburg.de/~ki/teaching/ws0607/aip/`

Text

Slides are partially based on similar course developed by Jussi Rintanen. They are available on the web page as the course proceeds.

Exercises and Examination

Exercises

Assistant: Robert Mattmüller
(mattmuel@informatik.uni-freiburg.de)
Friday 3pm after lecture
Assignments are given out on Wednesday, returned on Wednesday (before lecture).

Examination

Takes place in April (exact date to be determined).

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 simplest planning problems are paths from an initial state to a goal state in *the transition graph*. Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
  Why don't we solve all planning problems this way?
- State spaces may be huge: $10^9, 10^{12}, 10^{15}, \ldots$ states. Constructing the transition graph and using e.g. Dijkstra's algorithm is not feasible!!
- Planning algorithms try to avoid constructing the whole graph.
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

Why is planning difficult?

- Solutions to simplest planning problems are paths from an initial state to a goal state in *the transition graph*.
  Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
  Why don't we solve all planning problems this way?
- State spaces may be huge: $10^9, 10^{12}, 10^{15}, \ldots$ states.
  Constructing the transition graph and using e.g. Dijkstra's algorithm is not feasible!!
- Planning algorithms try to avoid constructing the whole graph.
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

Why is planning difficult?

- Solutions to simplest planning problems are paths from an initial state to a goal state in *the transition graph*.
  Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
  Why don't we solve all planning problems this way?
- State spaces may be huge: $10^9, 10^{12}, 10^{15}, \ldots$ states.
  Constructing the transition graph and using e.g. Dijkstra's algorithm is not feasible!!
- Planning algorithms try to avoid constructing the whole graph.
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

Why is planning difficult?

- Solutions to simplest planning problems are paths from an initial state to a goal state in *the transition graph*.
  Efficiently solvable e.g. by Dijkstra's algorithm in $O(n \log n)$ time.
  Why don't we solve all planning problems this way?
- State spaces may be huge: $10^9, 10^{12}, 10^{15}, \ldots$ states.
  Constructing the transition graph and using e.g. Dijkstra's algorithm is not feasible!!
- Planning algorithms try to avoid constructing the whole graph.
- Planning algorithms often are – but are not guaranteed to be – more efficient than the obvious solution method of constructing the transition graph + running e.g. Dijkstra's algorithm.

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Different classes of problems

actions	deterministic	nondeterministic	
probabilities	no	yes	
observability	full	partial	no
horizon	finite	infinite	
⋮			

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

Properties of the world: nondeterminism

Deterministic world/actions

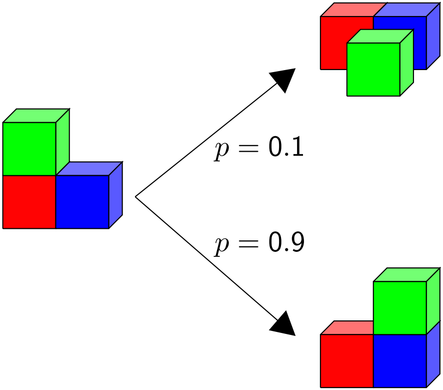
Action and current state uniquely determine the successor state.

Nondeterministic world/actions

For an action and a current state there may be several successor states.

Analogy: deterministic versus nondeterministic automata

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



Properties of the world: observability

Full observability

Observations/sensing allow to determine the current state of the world uniquely.

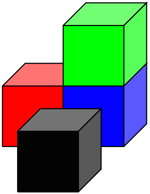
Partial observability

Observations/sensing allow to determine the current state of the world only partially: we only know that the current state is one of several of possible ones.

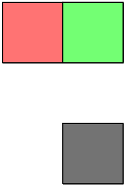
Consequence: It is necessary to represent the knowledge an agent has.

What difference does observability make?

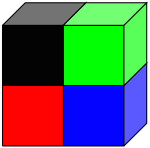
Camera A



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 2015 study hard and save money.
4. Collect the maximal *expected* rewards / minimal expected costs (infinite horizon).
   Example: Maximize your future income.
5. ...

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 2015 study hard and save money.
4. Collect the maximal *expected* rewards / minimal expected costs (infinite horizon).
   Example: Maximize your future income.
5. ...

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 2015 study hard and save money.
4. Collect the maximal *expected* rewards / minimal expected costs (infinite horizon).
   Example: Maximize your future income.
5. ...

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 2015 study hard and save money.
4. Collect the maximal *expected* rewards / minimal expected costs (infinite horizon).
   Example: Maximize your future income.
5. ...

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 2015 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 intelligent, 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.
- In certain special cases our techniques are applicable to multi-agent planning:
  - Finding a winning strategy of a game (example: 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.

Relation to games and game theory

- Game theory addresses decision making in multi-agent setting: "Assuming that the other agents are intelligent, 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.
- In certain special cases our techniques are applicable to multi-agent planning:
  - Finding a winning strategy of a game (example: 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.

Relation to games and game theory

- Game theory addresses decision making in multi-agent setting: "Assuming that the other agents are intelligent, 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.
- In certain special cases our techniques are applicable to multi-agent planning:
  - Finding a winning strategy of a game (example: 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.

Prerequisites of the course

1. basics of AI (you have attended an introductory course on AI)
2. basics of propositional logic
3. basics of structural complexity theory (reduction, NP-completeness, . . .)

What do you learn in this course?

1. Classification of different problems to different classes
   1. Classification according to observability, nondeterminism, goal objectives, ...
   2. computational complexity
2. Techniques for solving different problem classes
   1. search-based planning techniques
   2. algorithms based on heuristic search
   3. algorithms based on satisfiability testing (SAT)
   4. algorithms based on exhaustive search with logic-based data structures (BDDs)

Many of these techniques are applicable to problems outside AI as well.