AI Planners and Tools

March 14, 2019
At the end of the lecture you will...

- ... know about some useful planners
- ... get a timeline picture that shows when new planners appeared over time
- ... learn about some useful AI tools to (re)use and build planning systems

- The contents covered in this lecture are not exhaustive!
Outline

- **Deterministic Planning**
  - Fast-Forward (FF)
  - Fast-Downward (FD)
  - Iterated Width (IW)
  - Others: SATPlan (using SAT), SIMBA (using BDDs), ...

- **Non-Deterministic Planning**
- **Probabilistic Planning**
- **Other types of planning**
Dinosaurs

1990's

Automatic heuristic generation:
2001: $h_{add}$, $h_{max}$, $h_{ff}$
2001: FF planning system

Early 2000's

1995: Graphplan

Late 2000's

2006: FD planning system

2006: Bandits and Monte-Carlo Planning

2010's

2012: Iterated Width Search
2012: PRP
Fast-Forward (FF)


- Planning with deterministic actions
- STRIPS model
- Hill-Climbing + Breadth-First Search
  - Hill climbing: choose most promising action according to $h_{FF}$ heuristic
    - $h_{FF}$ based on the delete relaxation
  - Breadth-First Search: an artifact to turn the planner into a complete algorithm
- Other planners are based on FF:
  - FF-replan, for probabilistic planning
  - Metric-FF, for numeric planning
  - Conformant-FF, Contingent-FF, …
- Its PDDL parser is still widely (re-)used because is fast (and somewhat reliable)
Planning with Heuristics

Fig. 2: Good-bye, Graphplan!
Fast-Downward (FD)


- Planning with deterministic actions
- SAS⁺ model
- Search technique:
  - A* equipped with good heuristics
  - FD rapidly became a framework to do research for better heuristics
- Other planners are based on FD. E.g.:
  - PRP, for non-deterministic planning
- Its PDDL parser is widely (re-)used
  - Converts PDDL into SAS⁺
  - Slower than the FF parser
http://www.fast-downward.org
http://www.fast-downward.org/PlannerUsage
Iterated Width (IW)


- Planning with deterministic actions
- STRIPS and SAS⁺ model
- Initially, it was a **blind** search algorithm
  - Breadth-First search with pruning
  - A state is pruned if its *novelty* measure is greater than a hyperparameter \( k \)
  - State novelty is the size of the smallest tuple of atoms that has been seen for the first time
- More recently, equipped with heuristics and reworked notions of novelty
- Good results at the **General Video Game** (GVG) Competition 2016
- Recent success in playing **Atari** games (no PDDL model is needed since it is blind search)
http://lapkt.org
https://github.com/LAPKT-dev/LAPKT-public
Analogy with Reinforcement Learning Frameworks

AI Planning
- Planning Problem Description (usually, PDDL)
- Parser
  - one-way
- Planning Algorithm

(Deep) Reinforcement Learning
- Environment
  - (e.g. Atari, Minecraft, sensory data, ...)
- OpenAI Gym, Malmo
  - two-way
- RL Algorithm
Analogy with Reinforcement Learning Frameworks

AI Planning

Planning Problem Description (usually, PDDL)

Parser

FF parser

FD parser

one-way

FF planner

FD planner

IW planner

(Deep) Reinforcement Learning

Environment
(e.g. Atari, Minecraft, sensory data, ...)

OpenAI Gym, Malmo

two-way

RL Algorithm
Analogy with Reinforcement Learning Frameworks

Planning Problem Description
(usually, PDDL)

Parser

Planning Algorithm

Analogy with Reinforcement Learning Frameworks

Environment
(e.g. Atari, Minecraft, sensory data, ...)

OpenAI Gym, Malmo

two-way

(Deep) Reinforcement Learning

RL Algorithm

AI Planning

FF Planning system

FF parser

FF parser

FD parser

FD parser

IW planner

FF planner

FF planner

FF planner

FD planner

FD planner

IW planner

planner

planner
Analogy with Reinforcement Learning Frameworks

AI Planning

Planning Problem Description (usually, PDDL)

FD Planning system

FF parser

FD parser

FF planner

FD planner

IW planner

(Deep) Reinforcement Learning

Environment
(e.g. Atari, Minecraft, sensory data, ...)

OpenAI Gym, Malmo

two-way

Tensorflow

PyTorch
Analogy with Reinforcement Learning Frameworks

AI Planning

Planning Problem Description (usually, PDDL)

LAPKT Planning system

FF parser
FD parser
IW planner

(Deep) Reinforcement Learning

Environment (e.g. Atari, Minecraft, sensory data, ...)

OpenAI Gym, Malmo
two-way

Tensorflow
PyTorch

FF planner
FD parser

Outline

- Deterministic Planning
- Non-Deterministic Planning
  - Reasoning backwards (e.g. Gamer)
  - Reasoning forwards
    - Based on (L)AO*: myND
    - Based on plan aggregation: PRP
- Probabilistic Planning
- Other types of planning
PRP

Christian J. Muise, Sheila A. McIlraith, J. Christopher Beck. Improved Non-Deterministic Planning by Exploiting State Relevance. ICAPS 2012

- Planning with non-deterministic actions
- PRP searches for strong cyclic policies using plan aggregation.
- PRP uses FD to compute plans in the determinization of the problem
- Its performance is based in two premises:
  1) Finding deterministic plans is cheap
  2) Plan regression is an effective method to compute families of policies

Plan regression reasons backwards to compute what state features were relevant to goal achievement.
Outline

- Deterministic Planning
- Non-Deterministic Planning
- **Probabilistic Planning**
  - Based on Monte-Carlo Tree Search (PROST)
  - Others: Find-Revise-Eliminate-Traps (FRET)
  - HIGHPROB: ProbPRP
- Other types of planning
Probabilistic Planning

Different classes of probabilistic planning problems exist:

- **HIGHPROB** (e.g. ProbPRP)
  compute policies that **attempt to** maximize prob. of goal achievement
- **MAXPROB**
  compute policies that **maximize** prob. of goal achievement
- **HIGHREWARD/MAXREWARD** (e.g. PROST)
  compute policies that (attempt to) maximize discounted cumulative reward
PROST

Thomas Keller, Patrick Eyerich. **PROST: Probabilistic Planning Based on UCT.** ICAPS 2012

- MDP Planning with **stochastic** actions
- PROST searches for strong cyclic policies using Monte-Carlo Tree Search
- PROST does not make much use of the (stochastic) MDP model, except that it uses **heuristics** to estimate the return value in the leafs of the Monte-Carlo rollouts
- PROST (and variations of PROST) have won the last probabilistic planning competitions
Analogy with Reinforcement Learning Frameworks

AI Probabilistic Planning

Planning Problem Description (usually, \textsc{PDDL}, \textsc{RDDL})

Parser

Planning Algorithm

one-way

(Deep) Reinforcement Learning

Environment (e.g. Atari, Minecraft, sensory data, ...)

OpenAI Gym, Malmo

two-way

RL Algorithm
https://bitbucket.org/tkeller/prost/wiki/Home
https://github.com/ssanner/rddlsim
https://sites.google.com/site/rddltutorial/
Analogy with Reinforcement Learning Frameworks

AI Probabilistic Planning

PPDDL

RDDL

PPDDL parser

RDDLsim

one-way
two-way

FRET planner

PROST planner

(Deep) Reinforcement Learning

Environment
(e.g. Atari, Minecraft, sensory data, ...)

OpenAI Gym, Malmo

two-way

RL Algorithm
Analogy with Reinforcement Learning Frameworks

What is the difference with Reinforcement Learning?

AI Probabilistic Planning

(Deep) Reinforcement Learning

one-way

two-way

FRET parser

RDDLsim

PROST planner

OpenAI Gym, Malmo

RL Algorithm

What is the difference with Reinforcement Learning?
Outline

- Deterministic Planning
- Non-Deterministic Planning
- Probabilistic Planning
- **Other types of planning:**
  - Planning with hybrid (discrete+continuous) variables
  - Planning with durative actions (with start and end times)
  - ROSPlan: a planner integrated in the Robot Operating System (ROS).
  - ...
Compilation-based Techniques

- Compiling Partial Observability Away
- K-fault tolerant planning to classical planning
- Temporally Extended Goals to Final-State Goals
- Infinite-length plan generation to finite-length plan generation
- A combination of the above:
  - E.g. Infinite-length temporally extended goals into finite-length final-state goals.
That’s all folks