Planning and Learning under Uncertainty

Sergio Jiménez Celorrio

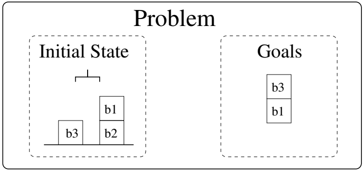
Directores: Daniel Borrajo Millán y Fernando Fernández Rebollo

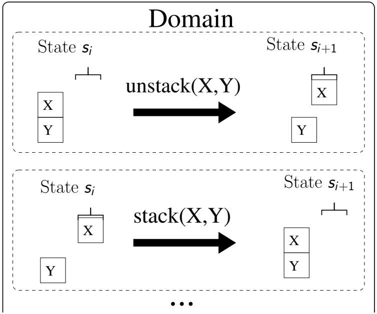
May, 2011

Automated Planning

Selection of **actions** for achieving **goals**

- Declarative representation
  - **Problem**, initial state and goals
  - **Domain**, set of actions
- General algorithms
  - **Search** strategies
  - **Heuristics**, automatically extracted

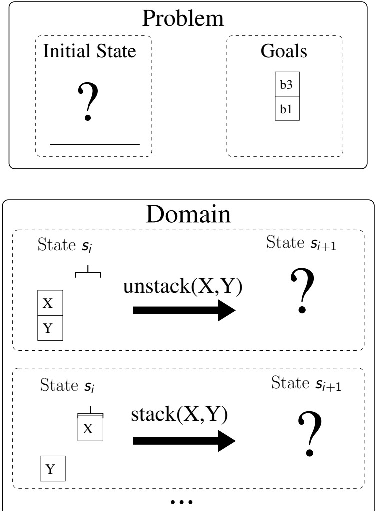




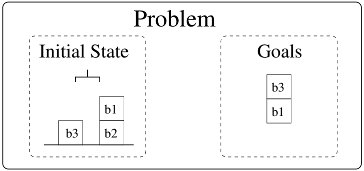
Automated Planning under Uncertainty

Selection of **actions** for achieving **goals**

- Declarative representation
  - **Problem**, initial state and goals
  - **Domain**, set of actions
- General algorithms
  - **Search** strategies
  - **Heuristics**, automatically extracted

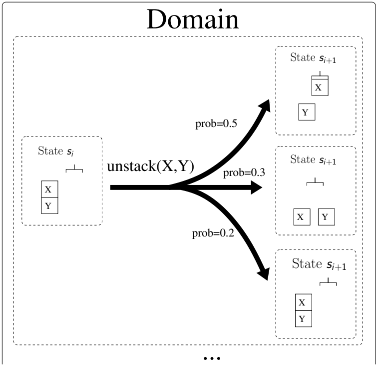


Planning under Uncertainty, Probabilistic Planning



Selection of **actions** for achieving **goals**

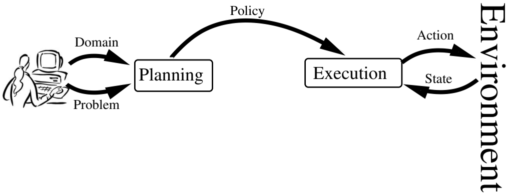
- Declarative representation
  - **Problem**, initial state and goals
  - **Domain**, set of actions
- General algorithms
  - **Search** strategies
  - **Heuristics**, automatically extracted



Planning under Uncertainty, Probabilistic Planning

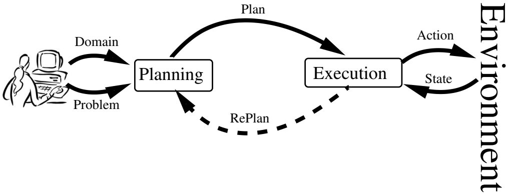
A) Perfect Model [Bonet and Geffner, 2006]

- Probabilistic Effects
- Probabilistic Planner



B) Deterministic Model [Fikes et al., 1972]

- Deterministic Effects
- Deterministic Planner

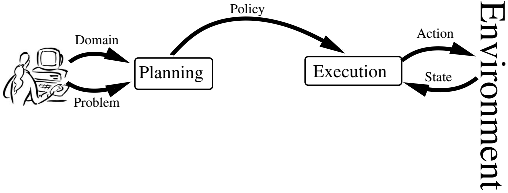


Outline

Probabilistic Planning

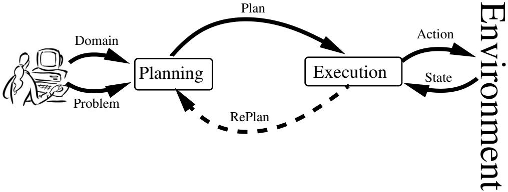
A) Perfect Model [Bonet and Geffner, 2006]

- × Hard to define
- × Hard to solve



B) Deterministic Model [Fikes et al., 1972]

- × Fragile at *probabilistically interesting*[a] domains

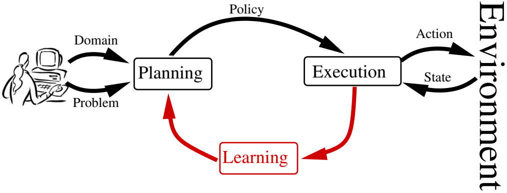


[a] [Little and Thiébaux, 2007]

Probabilistic Planning

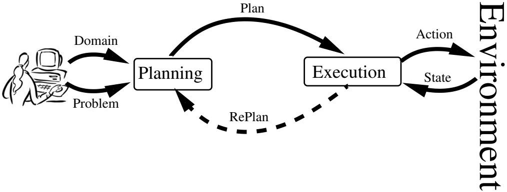
A) Perfect Model [Bonet and Geffner, 2006]

- Hard to define [Pasula et al., 2007]
- Hard to solve [Fern et al., 2006]



B) Deterministic Model [Fikes et al., 1972]

× Fragile at *probabilistically interesting*[a] domains

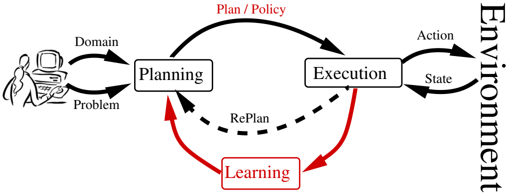


[a] [Little and Thiébaux, 2007]

Probabilistic Planning

A) Perfect Model [Bonet and Geffner, 2006]

- Probabilistic Effects
- Probabilistic Planner

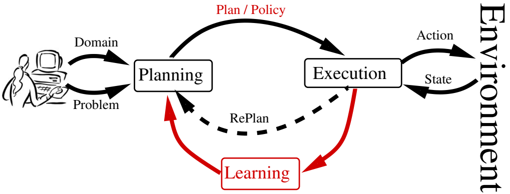


B) Deterministic Model [Fikes et al., 1972]

- Learning probability of success
- Learning execution dead-ends

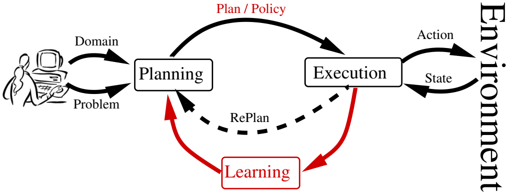
Our approach

```
(:action unstack
  :parameters (?x - block ?y - block)

  :precondition(and (on ?x ?y) (clear ?x) (handempty))

  :effect (and (holding ?x)(clear ?y)
               (not (clear ?x)) (not (handempty))
               (not (on ?x ?y))))
```



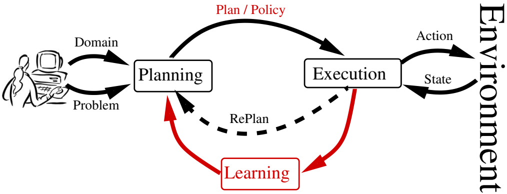
Our approach

```
(:action unstack
  :parameters (?x - block ?y - block)

  :precondition (and (on ?x ?y) (clear ?x) (handempty))

  :effect
     (and (holding ?x)(clear ?y)
          (not (clear ?x)) (not (handempty))
          (not (on ?x ?y))

     (when (and (not(blocked-hand))(not(wet-block ?x)))
          (increase (fragility) 10))))
```



Our approach

```
(:action unstack
  :parameters (?x - block ?y - block)

  :precondition (and (on ?x ?y) (clear ?x) (handempty))

  :effect
    (and
      (when (and (not(blocked-hand))(not(wet-block ?x)))
        (probabilistic 0.8
              (and (holding ?x)(clear ?y)
                   (not (clear ?x))(not (handempty))
                   (not (on ?x ?y))))))
```



Outline

Objectives

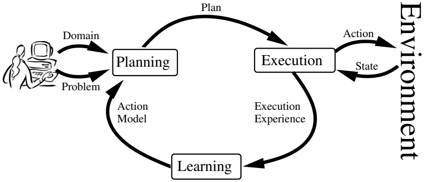
Integration of **planning**, **execution** and **learning** to synthesize robust plans in probabilistic planning

1. Learning
   - Collecting experience
   - Generalizing from experience
   - Exploiting the learned generalizations
2. Offline and Online learning
3. Adaptable to other kinds of planning knowledge
4. Use standard AI components

Outline

Planning, Execution and Learning Architecture

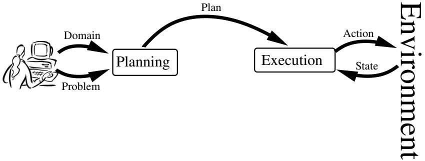
1. Collecting experience about the model
2. Generalizing from experience
3. Exploiting the learned generalizations



Planning, Execution and Learning Architecture

❶ Collecting experience about the model

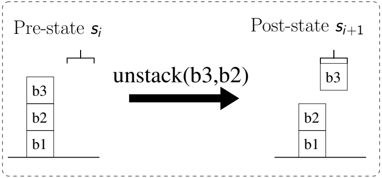
- Initial definition of a **deterministic model**
- A standard planner synthesizes a **plan** to solve problems
- Plans are **executed** step by step



;;; Pre-state s_i
ontable(b1). emptyhand().
on(b2,b1). on(b3,b2).
clear(b3).

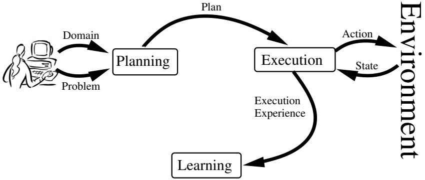
;;; Action executed
unstack(b3,b2).

;;; Post-state s_{i+1}
ontable(b1). on(b2,b1).
clear(b2). holding(b3).



Planning, Execution and Learning Architecture

1. Collecting experience about the model
2. Generalizing from experience
   - **Labeling examples**
   - **Finding patterns** in labeled examples



;;; Pre-state s_i
ontable(b1). emptyhand().
on(b2,b1). on(b3,b2).
clear(b3).

;;; Action executed
unstack(b3,b2,✓).

;;; Post-state s_{i+1}
ontable(b1). on(b2,b1).
clear(b2). holding(b3).

- ✓ **Success**: post-state matches model
  $s_{i+1} = \{s_i / Del(a_i)\} \cup Add(a_i)$
- × **Failure**: mismatch and goals reachable by re-planning
- † **Dead-end**: mismatch and goals are unreachable

- Organizing examples by actions
- Building patterns for each action

Examples

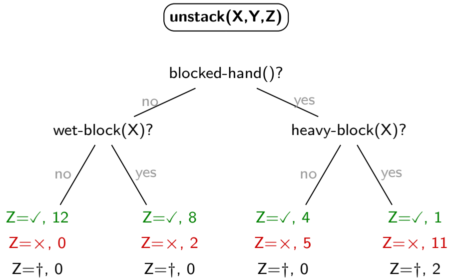
unstack(b3,b2,✓),pickup(b7,✓),stack(b7,b2,†),stack(b1,b3,✓),
unstack(b1,b4,✓),putdown(b6,×),stack(b4,b7,×),pickup(b1,✓),
putdown(b6,✓),unstack(b1,b2,×),unstack(b2,b6,†),
pickup(b2,×),pickup(b4,×),unstack(b4,b6,✓),unstack(b7,b1,×),
stack(b4,b6,×),stack(b2,b1,✓),stack(b1,b8,✓),pickup(b5,✓),
...

- Organizing examples by actions
- Building patterns for each action

unstack(X,Y,Z)

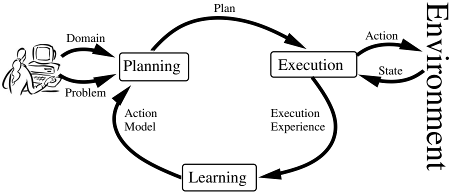
Action, pre-state
unstack(b3,b2,✓), emptyhand(),on(b3,b2),clear(b3), ...
unstack(b1,b4,✓), emptyhand(),on(b1,b4),clear(b1), ...
unstack(b1,b2,×), blocked-hand(),emptyhand(),on(...
unstack(b2,b6,†), blocked-hand(),emptyhand(),hea ...
unstack(b7,b1,×), blocked-hand(),emptyhand(),on(...
unstack(b4,b6,✓), emptyhand(),on(b4,b6),clear(b4), ...
unstack(b8,b9,✓), emptyhand(),on(b8,b9),clear(b8), ...
...

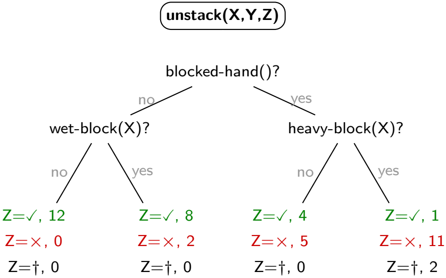
- Organizing examples by actions
- Building patterns for each action
  - Relational decision trees
    [Blockeel and Raedt, 1998]



Planning, Execution and Learning Architecture

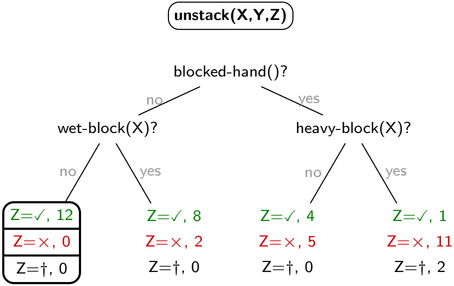
1. Collecting experience about the model
2. Generalizing from experience
3. Exploiting the learned generalizations
   - **Compiling patterns** into useful structures for further planning
     - Probabilistic planners
     - Deterministic planners






```
(:action unstack
   :parameters (?x - block ?y - block)

   :precondition(and (on ?x ?y) (clear ?x) (handempty))

   :effect (and (holding ?x)(clear ?y)
                (not (clear ?x)) (not (handempty))
                (not (on ?x ?y))))
```

(I) Compiling patterns for probabilistic planning

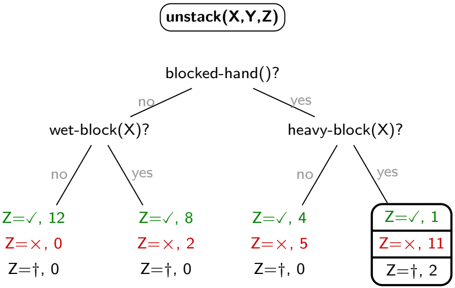


(:action unstack
:parameters (?x - block ?y - block)

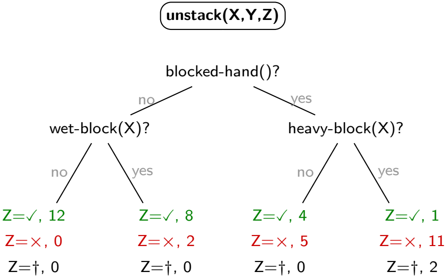
:precondition (and (on ?x ?y) (clear ?x) (handempty))

:effect
(and
(when (and (not(blocked-hand))(not(wet-block ?x)))
(probabilistic $\frac{12}{12+0+0}$
(and (holding ?x)(clear ?y)
(not (clear ?x))(not (handempty))
(not (on ?x ?y)))))

(I) Compiling patterns for probabilistic planning

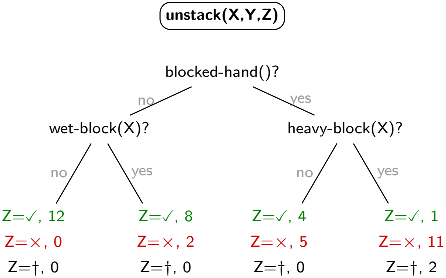



```
(:action unstack
  :parameters (?x - block ?y - block)
  :precondition (and (on ?x ?y) (clear ?x) (handempty))
  :effect
    (and
      (when (and (not(blocked-hand))(not(wet-block ?x)))
        (probabilistic 12/(12+0+0)
              (and (holding ?x)(clear ?y)
                   (not (clear ?x))(not (handempty))
                   (not (on ?x ?y)))))

      (when (and (not(blocked-hand))(wet-block ?x))
        (probabilistic 8/(8+2+0)
              (and (holding ?x)(clear ?y)
                   (not (clear ?x))(not (handempty))
                   (not (on ?x ?y)))))

      (when (and (not(blocked-hand))(wet-block ?x))
        (probabilistic 4/(4+5+0)
              (and (holding ?x)(clear ?y)
                   (not (clear ?x))(not (handempty))
                   (not (on ?x ?y)))))

      (when (and (blocked-hand)(heavy-block ?x))
        (probabilistic 0.001
              (and (holding ?x)(clear ?y)
                   (not (clear ?x))(not (handempty))
                   (not (on ?x ?y)))))
```




```
(:action unstack
   :parameters (?x - block ?y - block)

   :precondition(and (on ?x ?y) (clear ?x) (handempty))

   :effect (and (holding ?x)(clear ?y)
                (not (clear ?x)) (not (handempty))
                (not (on ?x ?y))))
```

(II) Compiling patterns for deterministic planning




```
(:action unstack
  :parameters (?x - block ?y - block)

  :precondition (and (on ?x ?y) (clear ?x) (handempty))

  :effect
     (and (holding ?x)(clear ?y)
          (not (clear ?x)) (not (handempty))
          (not (on ?x ?y))

     (when (and (not(blocked-hand))(not(wet-block ?x)))
           (increase (fragility) f( 12/(12+0+0) ))))

     (when (and (not(blocked-hand))(wet-block ?x))
           (increase (fragility) f( 8/(8+2+0) ))))

     (when (and (blocked-hand)(not(heavy-block ?x)))
           (increase (fragility) f( 4/(4+5+0) ))))

     (when (and (blocked-hand)(heavy-block ?x))
           (increase (fragility) ∞)))
```

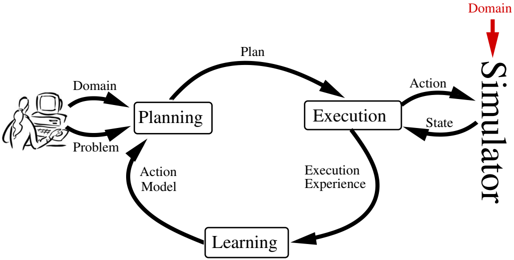
---

f(psuccess)=-log(psuccess)

Outline

Schema

1. Offline learning
2. Online learning

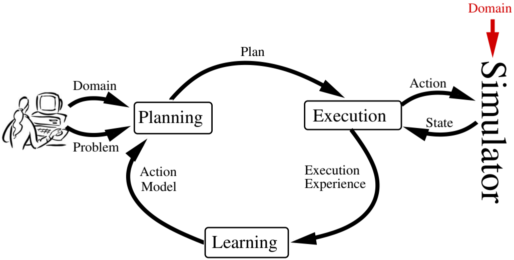


Domains

Probabilistically interesting domains [Little and Thiébaux, 2007]

Probabilities	**State-Dependent + Probabilities**
Blocksworld	*Slippery-Gripper*, *Rovers*
(†) *OpenStacks*	(†) *Triangle-tireworld*, (†) *Satellite*

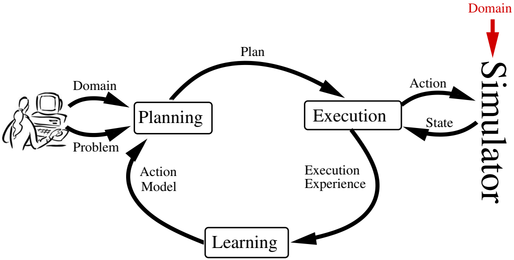
1. Offline learning
2. Online learning



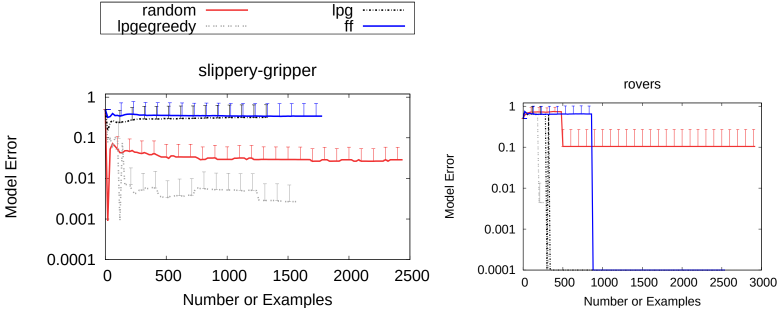
Evaluation of offline learning

Strategies for collecting examples

1. FF [Hoffmann and Nebel, 2001]
2. LPG [Gerevini et al., 2003]
3. LPG-$\mathcal{E}$greedy
4. Random



Evaluation of offline learning

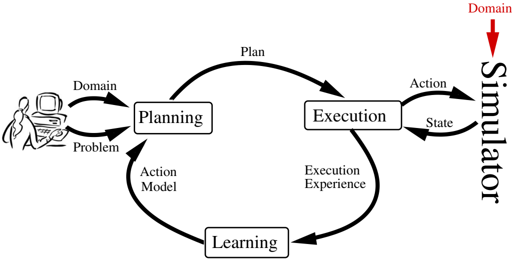


- Learning curves may present discontinuities
- Stochastic explorations introduce diversity in the examples ($FF < LPG < LPG\,\varepsilon greedy$)
- Pure random explorations miss portions of the state space

Evaluation of offline learning

Planning strategies

1. FF + learned model (cost)
2. GPT + learned model (prob)



Evaluation of offline learning

15 problems $\times$ 30 runs $=$ 450 instances, 900 secs per instance

	Problems Solved			
	FF STRIPS model	**(1) FF learned model (cost)**	**(2) GPT learned model (prob)**	GPT perfect model
(†)**OpenStacks**	0	90	300	**450**
(†)**Triangle-tireworld**	5	50	**373**	304
(†)**Satellite**	0	300	300	**420**

Planning at domains **with** execution dead-ends

- Learned models solve more problems than classic replanning
- Some learned models even more useful than the perfect model

Evaluation of offline learning

15 problems $\times$ 30 runs = 450 instances, 900 secs per instance

	Problems Solved			
	FF STRIPS model	**(1) FF learned model (cost)**	**(2) GPT learned model (prob)**	GPT perfect model
Blocksworld	443	**450**	390	390
Slippery-Gripper	369	**450**	**450**	450
Rovers	**450**	421	270	270

Planning at domains **free from** execution dead-ends

- Learned models solve more problems than classic replanning
  - Reduces the number of replanning episodes

Evaluation of offline learning

15 problems $\times$ 30 runs $=$ 450 instances, 900 secs per instance

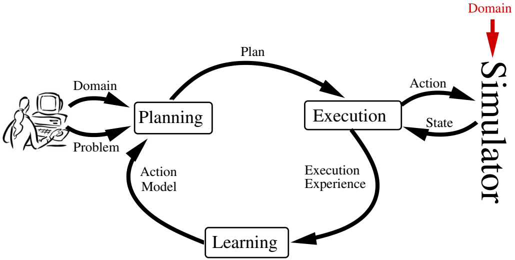
	Planning Time of Problems Solved (seconds)			
	FF STRIPS model	**(1) FF learned model (cost)**	**(2) GPT learned model (prob)**	GPT perfect model
Blocksworld	78454	35267	**26389**	38416
Slippery-Gripper	36771	4302	**1238**	2167
Rovers	28220	349670	18635	**18308**

Planning at domains **free from** execution dead-ends

- Learned models solve more problems than classic replanning
  - Reduces the number of replanning episodes
  - More effective when replanning is expensive

Evaluation

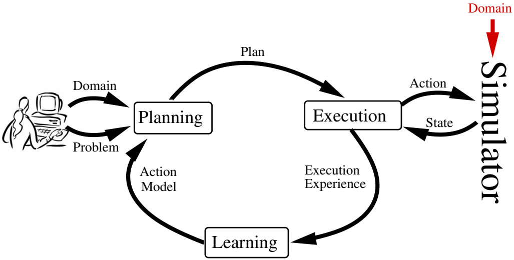
1. Offline learning
2. Online learning



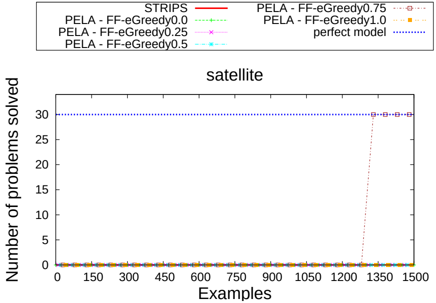
Evaluation

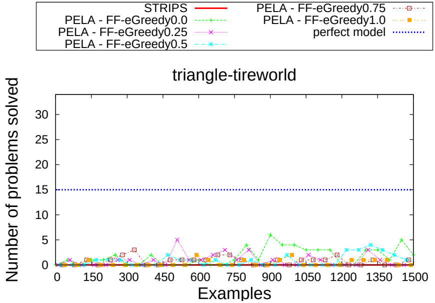
Strategies for collecting examples

1. FF-$\mathcal{E}$greedy0.0 (Random)
2. FF-$\mathcal{E}$greedy0.25
3. FF-$\mathcal{E}$greedy0.5
4. FF-$\mathcal{E}$greedy0.75
5. FF-$\mathcal{E}$greedy1.0 (FF)



Evaluation of online integration



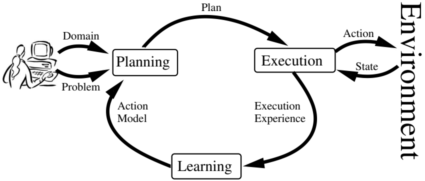


- Learned models solve more problems than the classic replanning
- It is more complex to collect good examples
- Effective learned knowledge may degenerate

Outline

Planning, Execution and Learning Architecture

1. Collecting experience about the model
2. Generalizing from experience
3. Exploiting the learned generalizations



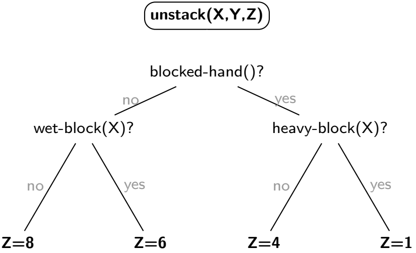
Learning durations of actions

Learning numeric values

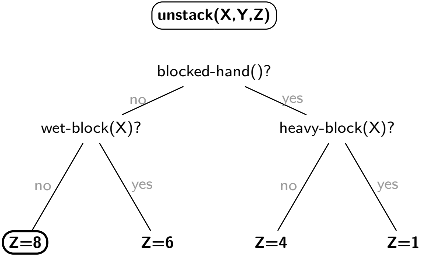
unstack(X,Y,Z)

Action,duration, pre-state

unstack(b3,b2,**8**) , emptyhand(),on(b3,b2),clear(b. . .
unstack(b1,b4,**8**), emptyhand(),on(b1,b4),clear(b. . .
unstack(b1,b2,**1**), blocked-hand(),emptyhand(),. . .
unstack(b2,b6,**1**), blocked-hand(),emptyhand(),. . .
unstack(b7,b1,**4**), blocked-hand(),emptyhand(),. . .
unstack(b4,b6,**6**), emptyhand(),on(b4,b6),clear(b. . .
unstack(b8,b9,**8**), emptyhand(),on(b8,b9),clear(b. . .
. . .

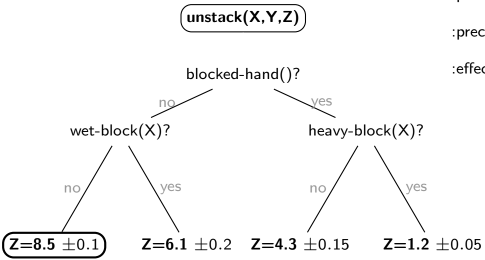


Learning durations of actions



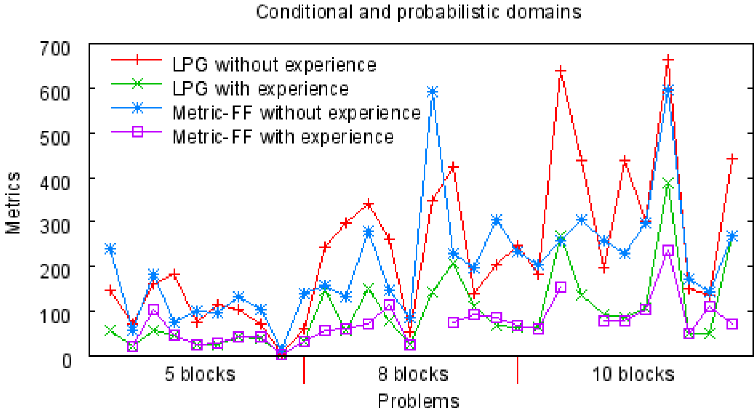

```
(:action unstack
  :parameters (?x - block ?y - block)

  :precondition (and (on ?x ?y) (clear ?x) (handempty))

  :effect (and (holding ?x)(clear ?y)
               (not (clear ?x)) (not (handempty))
               (not (on ?x ?y))

               (when (and (not(blocked-hand))(not(wet-block ?x)))
                     (increase (duration) 8))

               (when (and (not(blocked-hand))(wet-block ?x))
                     (increase (duration) 6))

               (when (and (blocked-hand)(not(heavy-block ?x)))
                     (increase (duration) 4))

               (when (and (blocked-hand)(heavy-block ?x))
                     (increase (duration) 1))))
```

Learning durations of actions




```
(:action unstack
   :parameters (?x - block ?y - block)

   :precondition (and (on ?x ?y) (clear ?x) (handempty))

   :effect (and (holding ?x)(clear ?y)
                (not (clear ?x)) (not (handempty))
                (not (on ?x ?y))

                (when (and (not(blocked-hand))(not(wet-block ?x)))
                      (increase (duration) 8.5))

                (when (and (not(blocked-hand))(wet-block ?x))
                      (increase (duration) 6.1))

                (when (and (blocked-hand)(not(heavy-block ?x)))
                      (increase (duration) 4.3))

                (when (and (blocked-hand)(heavy-block ?x))
                      (increase (duration) 1.2))))
```

Learning durations of actions



Outline

Summary of conclusions

1. Learning
   - Stochastic exploration is necessary but guided by planners
   - Learned models are more useful when replanning is expensive
2. Offline and Online learning
   - It is more complex to collect good examples
   - Learned models may degenerate
3. Adaptable to other kinds of planning knowledge
   - Same schema with small modifications for learning durations
4. Use standard AI components
   - Standard planners and learners

Future Work

1. **Planning**
   - Model-lite planning [Kambhampati, 2007]
   - Diversity exploration [Felner et al., 2003, Bjarnason et al., 2009]
2. **Execution**
   - Sophisticated execution monitoring [Bresina et al., 2005, Fox et al., 2006]
3. **Learning**
   - Learning full models [Pasula et al., 2007, Yang et al., 2007, Amir, 2006]

Planning and Learning under Uncertainty

Sergio Jiménez Celorrio

Directores: Daniel Borrajo Millán y Fernando Fernández Rebollo

May, 2011

Amir, E. (2006).
Learning partially observable action schemas.
In *National Conference on Artificial Intelligence (AAAI'06)*.

Bjarnason, R., Fern, A., and Tadepalli, P. (2009).
Lower bounding klondike solitaire with monte-carlo planning.
In *International Conference on Automated Planning and Scheduling (ICAPS 2009)*.

Blockeel, H. and Raedt, L. D. (1998).
Top-down induction of first-order logical decision trees.
Artificial Intelligence, 101(1-2):285–297.

Bonet, B. and Geffner, H. (2004).
mgpt: A probabilistic planner based on heuristic search.
Journal of Artificial Intelligence Research, 24:933–944.

Bonet, B. and Geffner, H. (2006).
Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs.
In *International Conference on Automated Planning and Scheduling, ICAPS06*.

Bresina, J. L., Jnsson, A. K., Morris, P. H., and Rajan, K. (2005).
Mixed-initiative activity planning for mars rovers.
In *IJCAI*, pages 1709–1710.

Felner, A., Kraus, S., and Korf, R. E. (2003).

Kbfs: K-best-first search.
Annals of Mathematics and Artificial Intelligence, 39.

Fern, A., Yoon, S. W., and Givan, R. (2006).
Approximate policy iteration with a policy language bias: Solving relational markov decision processes.
Journal of Artificial Intelligence Research, 25:75–118.

Fikes, R., Hart, P., and Nilsson, N. J. (1972).
Learning and executing generalized robot plans.
Artificial Intelligence, 3:251–288.

Fox, M., Gerevini, A., Long, D., and Serina, I. (2006).
Plan stability: Replanning versus plan repair.
International Conference on Automated Planning and Scheduling (ICAPS'06).

Gerevini, A., Saetti, A., and Serina, I. (2003).
Planning through stochastic local search and temporal action graphs in LPG.
Journal of Artificial Intelligence Research, 20:239–290.

Hoffmann, J. and Nebel, B. (2001).
The FF planning system: Fast plan generation through heuristic search.
Journal of Artificial Intelligence Research, 14:253–302.

Kambhampati, S. (2007).
Model-lite planning for the web age masses: The challenges of planning with incomplete and evolving domain models.

In *Senior Member track of the AAAI*, Seattle, Washington, USA. AAAI Press/MIT Press.

Little, I. and Thiébaux, S. (2007).
Probabilistic planning vs replanning.
In *Workshop on International Planning Competition: Past, Present and Future. ICAPS 2007, Providence, Rhode Island, USA.*

Pasula, H. M., Zettlemoyer, L. S., and Kaelbling, L. P. (2007).
Learning symbolic models of stochastic domains.
JAIR, 29:309–352.

Yang, Q., Wu, K., and Jiang, Y. (2007).
Learning action models from plan traces using weighted max-sat.
Artificial Intelligence Journal, 171:107–143.