

The PELA architecture: integrating planning and learning to improve execution

Sergio Jiménez Celorrio Fernando Fernández Rebollo
Daniel Borrajo Millán

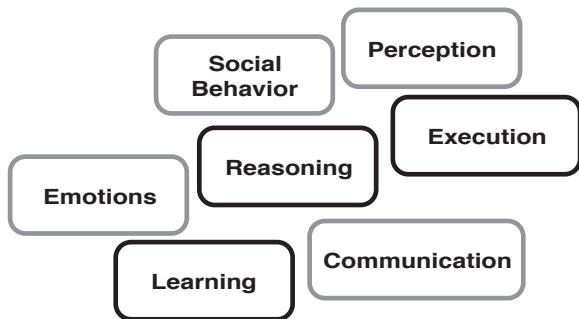
Planning and Learning Group, Universidad Carlos III de Madrid

July, 2008

Integrated Intelligence

Cognitive Architectures integrate diverse AI components to implement intelligent behavior

- ▶ This work focuses on the integration of **planning** and **learning** for the improvement of plan **execution**



Integrated Intelligence

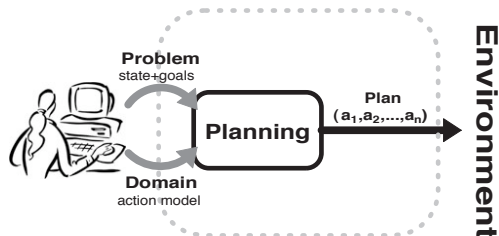
Open issues in planning and learning integration

- ▶ **Learning for declarative planning in stochastic environments**
[Pasula et al., 2007, Fern et al., 2006, Garcia-Martinez and Borrajo, 2000]
- ▶ **Integration based on off-the-shelf AI components**
- ▶ **Online learning** [Coles and Smith, 2007]

Planning

Finding a set of actions (**plan**) to achieve the **goals** starting from a given **initial state**

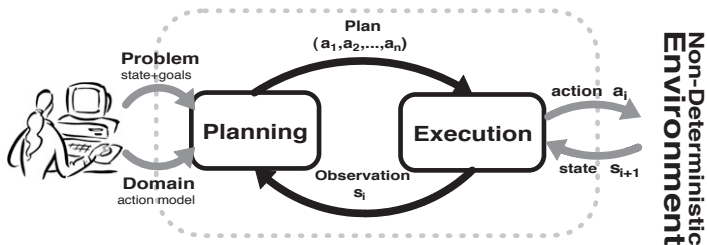
- ▶ Planning models
 - ▶ **Classical** reasons about **causality**, language **PDDL**
 - ▶ **Cost-Based** reasons about **causality + costs**, language **PDDL**
 - ▶ **Probabilistic** reasons about **causality + probabilities**, language **PPDDL**



Planning + Execution

Plan execution is monitored and plans are repaired when needed

- ✓ **Easy to implement:** off-the-shelf planners work
- ✗ **Fragile plans:** off-the-shelf planners do not reason about probabilities



Planning + Execution + Learning = PELA

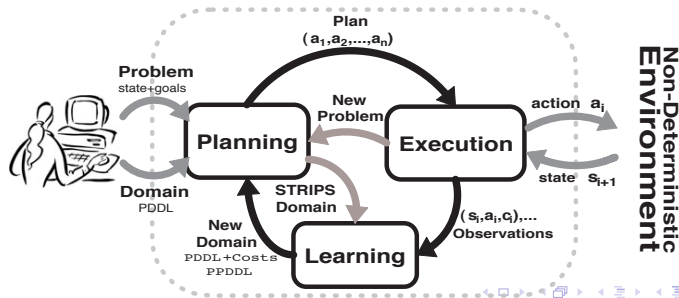
Plan execution is monitored, plans repaired and models updated

✓ Easy to implement:

- ▶ off-the-shelf planners work
- ▶ off-the-shelf relational classifiers work
- ▶ standard representation language, knowledge expressed in **PDDL**

✓ Robust plans: PELA uses learning for updating the action models

- ▶ capturing probability of success of actions
- ▶ predicting execution dead ends



Outline

Introduction and Motivation

The PELA architecture

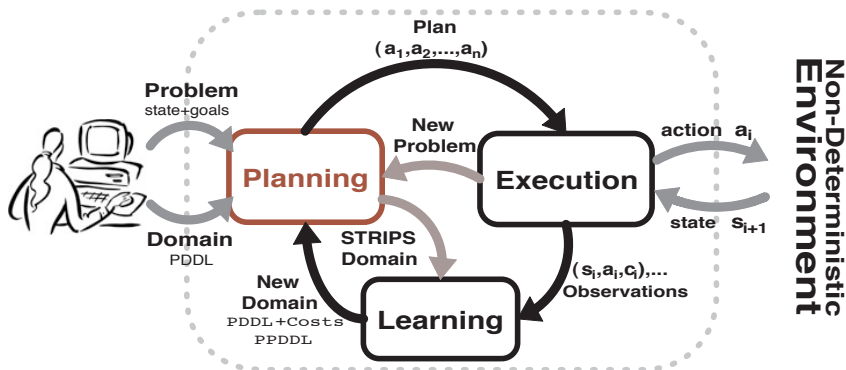
Experiments

Conclusions

Planning

A standard planner proposes a plan to solve a given planning task

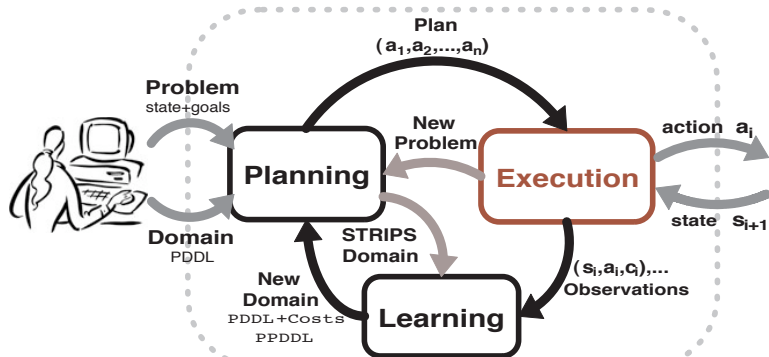
- ▶ **Input:** Strips action model + training problem
- ▶ **Output:** [plan|noplan]



Execution

Plans are executed step by step and executions are tagged as:

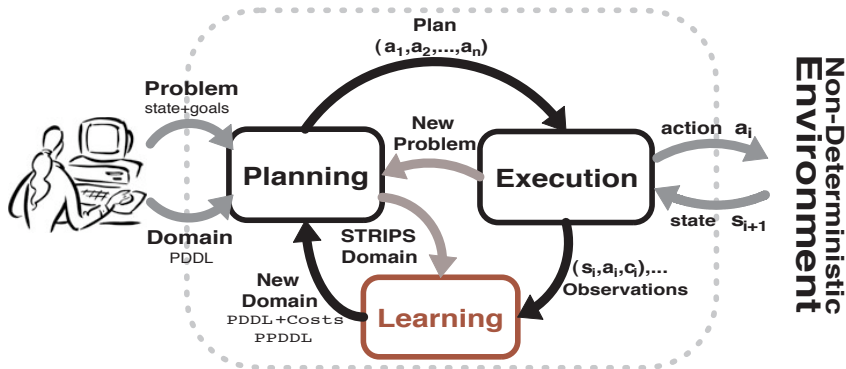
- ▶ **Success:** expected outcome according to the STRIPS model
- ▶ **Failure:** replanning is needed for reaching the goals
- ▶ **Dead-end:** there is no way to reach the goals



Learning

The planning model is updated following a two steps process

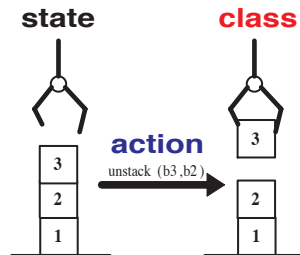
1. **Classification:** ([Success|Failure|Dead-End])
2. **Compilation:** Update of the action model



Learning - 1.Classification

The examples

- ▶ **state**: Context of the execution
- ▶ **action**: Executed action
- ▶ **class**: [Success|Failure|Dead-End]



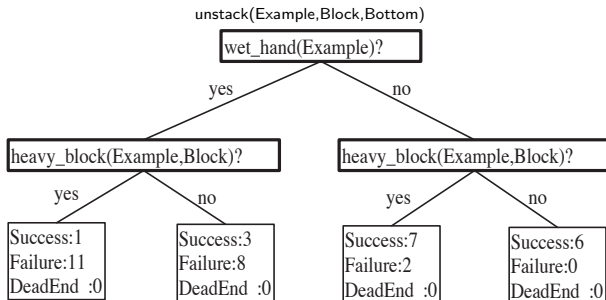
```

ontable(examplei,b1). emptyhand(examplei).
on(examplei,b2,b1). on(examplei,b3,b2).
clear(examplei,b3).
unstack (examplei,b3,b2,success).
  
```

Learning - 1. Classification

The algorithm

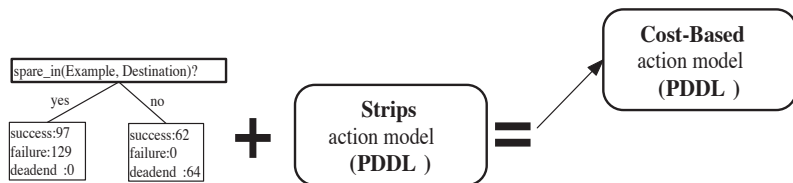
- ▶ A standard relational classification algorithm
- ▶ Learning relational decision trees [Blokceel and Raedt, 1998]



Learning - 2.Compilation

Two possible compilations:

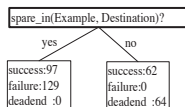
1. For cost-based planners



Learning - 2.Compilation for cost-based planners

1. Parameters: remain the same
2. Preconditions: remain the same
3. Effects: Action Effects + Conditional Cost (fragility)

movecar(E,Origin,Destination)



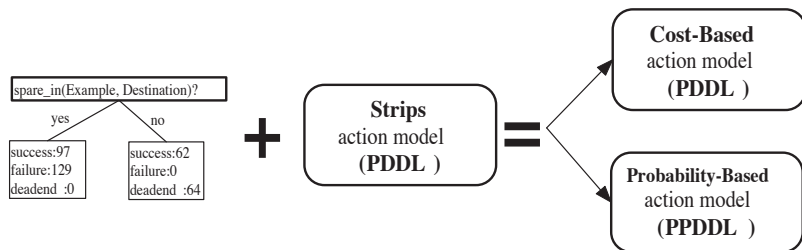
$$\text{cost} = -\log(S/(S + F + D))$$

```
(:action move-car
:parameters (?v1 - location ?v2 - location)
:precondition (and (vehicle-at ?v1)
                  (road ?v1 ?v2)
                  (not-flattire))
:effect (and (when (and (spare-in ?v2))
                 (and (vehicle-at ?v2)
                     (not (vehicle-at ?v1))
                     (increase (fragility) 0.845))))
           (when (and (not (spare-in ?v2)))
                 (and (vehicle-at ?v2)
                     (not (vehicle-at ?v1))
                     (increase (fragility) 999999999))))))
```

Learning - 2.Compilation

Two possible compilations:

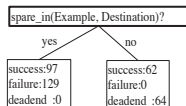
1. For cost-based planners
2. For probability-based planners



Learning - 2.Compilation for probabilistic planners

1. Parameters: remain the same
2. Preconditions: remain the same
3. Effects: Probabilistic Conditional Effects

movecar(E,Origin,Destination)



$$prob = S / (S + F + D)$$

```
(:action move-car
:parameters (?v1 - location ?v2 - location)
:precondition (and (vehicle-at ?v1)
                  (road ?v1 ?v2)
                  (not-flattire))
:effect (and (when (and (spare-in ?v2))
                  (probabilistic 0.43
                    (and (vehicle-at ?v2)
                         (not (vehicle-at ?v1))))))
            (when (and (not (spare-in ?v2))
                  (probabilistic 0.001
                    (and (vehicle-at ?v2)
                         (not (vehicle-at ?v1)))))))))
```

Outline

Introduction and Motivation

The PELA architecture

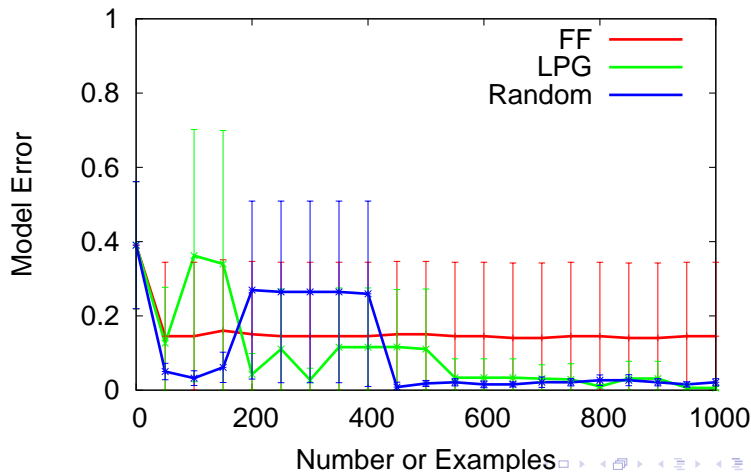
Experiments

Conclusions

Experimental Evaluation

1. Model Learning - Triangle Tireworld

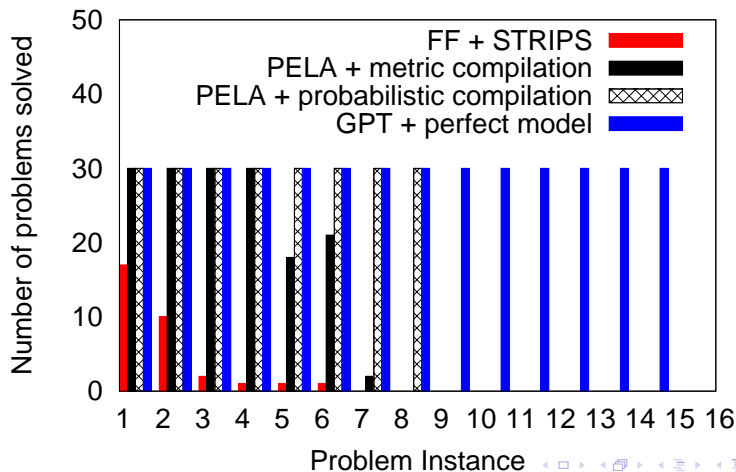
movecar(location,location)



Experimental Evaluation

2. Offline Integration

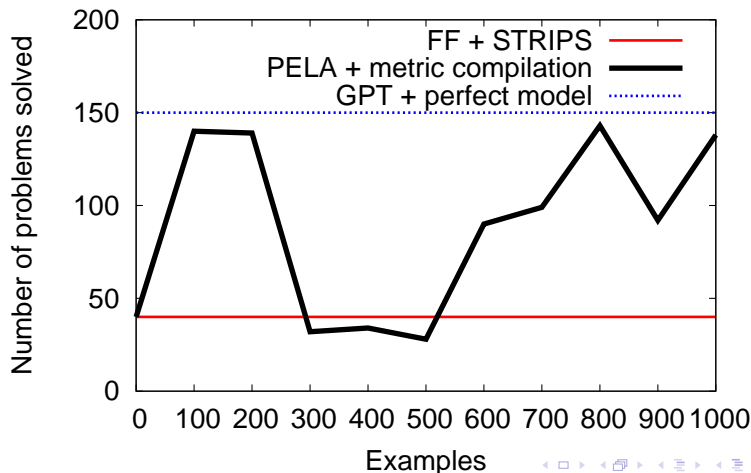
Triangle Tireworld



Experimental Evaluation

3. Online Integration

Triangle Tireworld



Conclusions

- ▶ **Learning for planning in stochastic environments**
PELA generates more robust plans than the *re-planning* approach
- ▶ **Integration based on off-the-shelf AI components**
In PELA, the planner and the learner are standard AI tools
- ▶ **Online Integration**
The model update of PELA does not affect to the actions causality

- Thanks -

The PELA architecture: integrating planning and learning to improve execution

Sergio Jiménez Celorrio Fernando Fernández Rebollo
Daniel Borrajo Millán

Planning and Learning Group, Universidad Carlos III de Madrid

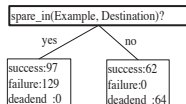
July, 2008

Learning

Compilation for cost-based planners

1. Parameters: remain the same
2. Preconditions: remain the same
3. Effects: Action Effects + Conditional Cost (fragility)

movecar(E,Origin,Destination)



$cost = -\log(S/(S + F + D))$

```
(:action move-car
:parameters (?v1 - location ?v2 - location)
:precondition (and (vehicle-at ?v1)
                   (road ?v1 ?v2)
                   (not-flattire))
:effect (and (when (and (spare-in ?v2))
                  (and (vehicle-at ?v2)
                       (not (vehicle-at ?v1))
                       (increase (fragility) 0.845))))
           (when (and (not (spare-in ?v2)))
                  (and (vehicle-at ?v2)
                       (not (vehicle-at ?v1))
                       (increase (fragility) 99999999))))))
```

Learning

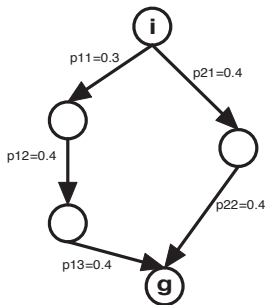
The fragility cost, $frag(p_i) = -\log(p_i)$

Maximize: $p_1 * p_2 * .. * p_n$

Maximize: $\ln(p_1 * p_2 * .. * p_n)$

Maximize: $\ln(p_1) + \ln(p_2) + .. + \ln(p_n)$

Minimize: $-\ln(p_1) - \ln(p_2) - .. - \ln(p_n)$

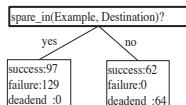


Learning

Compilation for probabilistic planners

1. Parameters: remain the same
2. Preconditions: remain the same
3. Effects: Probabilistic Conditional Effects

movecar(E,Origin,Destination)



$$prob = S / (S + F + D)$$

```
(:action move-car
:parameters (?v1 - location ?v2 - location)
:precondition (and (vehicle-at ?v1)
(road ?v1 ?v2)
(not-flattire))
:effect (and (when (and (spare-in ?v2))
(probabilistic 0.43
(and (vehicle-at ?v2)
(not (vehicle-at ?v1))))))
(when (and (not (spare-in ?v2)))
(probabilistic 0.001
(and (vehicle-at ?v2)
(not (vehicle-at ?v1))))))
```

Experimental Evaluation

1. Learning

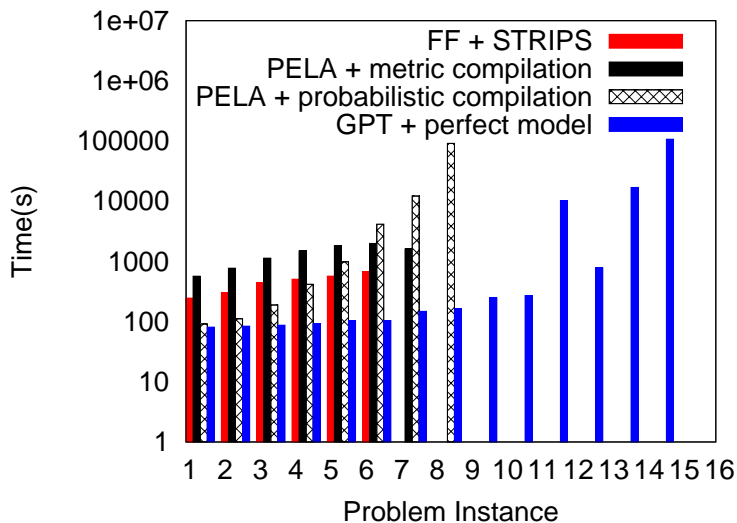
- ▶ Exploration Strategies: **FF**, **LPG**, **Random**
- ▶ Exploration Framework: 25 random problems, 50 actions max.
- ▶ Model Error: Average computed over 1000 random test examples

Experimental Evaluation

2. Offline Integration

- ▶ Planning Strategies: **Strips**, PELA-Compiled1, PELA-Compiled2, **Perfect Model**
- ▶ Models compiled after 500 action executions
- ▶ Each test problem attempted 30 times, average values shown

Offline Integration

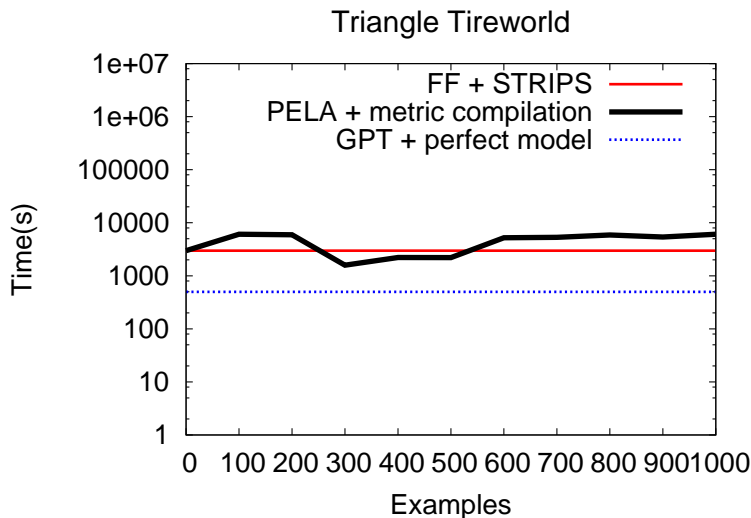





Experimental Evaluation

3. Online Integration

- ▶ Planning Strategies: **Strips**, PELA-Compiled1, **Perfect Model**
- ▶ Each test problem attempted 30 times, average values shown

Online Integration



-  Blockeel, H. and Raedt, L. D. (1998).
Top-down induction of first-order logical decision trees.
Artificial Intelligence, 101(1-2):285–297.
-  Coles, A. and Smith, A. (2007).
Marvin: A heuristic search planner with online macro-action learning.
JAIR, 28:119–156.
-  Dzeroski, S., Raedt, L. D., and Driessens, K. (2001).
Relational reinforcement learning.
Machine Learning, 43(1/2):7–52.
-  Fern, A., Yoon, S. W., and Givan, R. (2006).
Approximate policy iteration with a policy language bias: Solving relational markov decision processes.
JAIR, 25:85–118.
-  Garcia-Martinez, R. and Borrajo, D. (2000).
An integrated approach of learning, planning, and execution.
Journal of Intelligent and Robotics Systems, 29:47–78.
-  Pasula, H. M., Zettlemoyer, L. S., and Kaelbling, L. P. (2007).
Learning symbolic models of stochastic domains.
JAIR, 29:309–352.