Machine learning of plan robustness knowledge about instances

WHY?

Our future goal is to build a system:

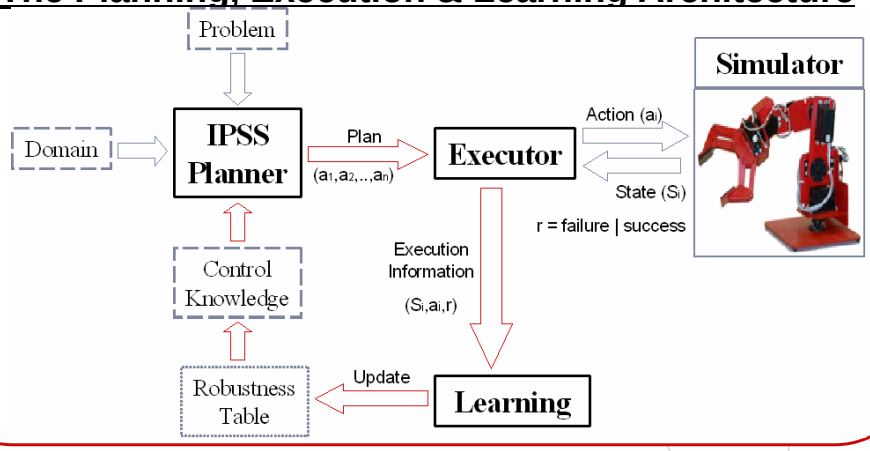
- **Able to act in uncertain domains**, as Reinforcement Learning does
- **With a representation of the action model richer** than Reinforcement Learning, as **deliberative planning**
- **Able to deal with large state spaces domains**

WHAT?

We have developed an architecture that:

- Gradually and automatically **acquires knowledge about the objects behavior in the real world** repeating **cycles of planning, execution and learning**, as it is commonly done in most real world planning situations by humans.
- **Learns knowledge** about **which actions are the most Robust** in the real world
- The system uses this knowledge as **control knowledge** to **prefer the more robust actions** in future planning.

The Planning, Execution & Learning Architecture



HOW?

PLANNING: We use the nonlinear backward chaining **planner IPSS** that integrates P&S .
To **use the learnt knowledge** (the actions robustness), the system defines **control rules that decide the instantiation of the actions**. These control rules choose the best bindings **according to the acquired knowledge about he robustness of the actions**.

EXECUTION: The system **executes step by step the** sequence of **actions** proposed by the planner to solve a problem **and observe the results**, to estimate the robustness of the actions.

LEARNING: **The Robustness Table stores the robustness of the actions** as tuples: **(op-name, op-params, robustness-value)**. Example `(prepare-visit (PRADO MONDAY) 1.1487)`.
This table is updated with the result of the actions executions following **the reinforcement algorithm**:

```
If the action is executed successfully
        new_robustness_value  = old_robustness_value + 1
 else
        new_robustness_value  = √old_robustness_value
```

EXPERIMENTS

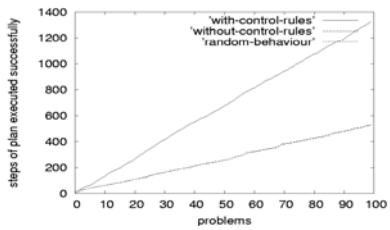
- Domain:

  Planning Tourist Visits Domain

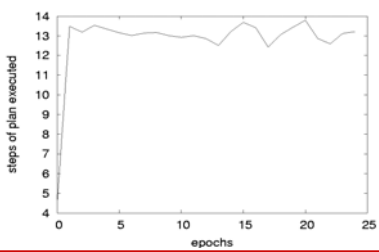
- Problems State
  - Tourist Available Money
  - Tourist Available Time
  - Places Time-Tables
  - Visit Duration
  - Visit Cost
- Problems Goals
  - Places to visit

RESULTS

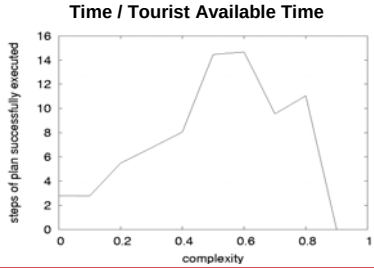
1.- Comparison of the number of steps successfully executed with and without the control knowledge



2.- Convergence of the number of plan steps successfully executed



3.- How problems complexity affects to the number of step plan successfully executed. Complexity = Goals Time / Tourist Available Time



ICAPS 2005 June 5-10. Monterrey, California, USA.
The International Conference on Automated Planning and Scheduling
Doctoral Consortium Poster Session

Sergio Jiménez Celorrio
Planning & Learning Group - Universidad Carlos III de Madrid