

A METHOD FOR PONDERING PLANS IN AUTONOMOUS INTELLIGENT SYSTEMS

López, D., Merlino, H., Ierache, J., García Martínez, R.

*Intelligent Systems Laboratory. School of Engineering. University of Buenos Aires
Software & Knowledge Engineering Center. Buenos Aires Institute of Technology
PhD on Computer Science. School of Computer Science. National University of La Plata
School of Computer Science, Communicational Sciences & Special Technologies. University of Morón*

rgm@itba.edu.ar

Abstract

Autonomous intelligent systems (AIS) learn from their environment by forming theories (operators) about how the environment reacts to system's actions. Each AIS uses the theories it has acquired for building plans (operators sequences) that allow it to reach its self-proposed objectives. In order to improve the performance of the AIS in the process of reaching their objectives, it is necessary to have a method that allows estimating the reliability of the generated plans before their execution. Such a process is called ponderation of plans, and has been successfully applied in previous works. In this paper, a new algorithm for pondering plans is proposed with the goal of improving the performance (percentage of successful plans) and this way, solving problems more efficiently. The experimental results presented clearly show how the proposed method outperforms the classic one.

Keywords: *Autonomous Intelligent Systems, Planning in Robotics, Machine Learning.*

1 Introduction

LOPE is an agent architecture that integrates planning, learning, and execution in a closed loop, showing an autonomous intelligent behavior [García-Martínez and Borrajo, 1997] (LOPE stands for Learning by Observation in Planning Environments). Learning planning operators is achieved by observing the consequences of executing planned actions in the environment [Fritz *et al.*, 1989; Veloso, 1994; Borrajo and Veloso, 1997; Ierache *et al.*, 2008]. In order to speed up the convergence, heuristic generalizations of the observations have been used. Also, probability distribution estimators have been introduced to handle the contradictions among the generated planning operators. All these techniques have been integrated in a multiple agent architecture [Maceri and García-Martínez, 2001].

In this paper, a new algorithm for pondering the generated plans is proposed with the aim of increase the percentage of successful executed plans.

Section 2 describes the general architecture of the LOPE agents, defining its top-level algorithm. Section 3 defines the representation that will be used in the paper for situations, observations and planning operators. Section 4 introduces the concept of planning. Section 5 describes classical ponderation of plans. In Section 6 the proposed method for pondering plans is presented. Section 7 presents the experiments performed and their results. Finally, Section 8 draws some conclusions.

2 General descriptions

One of the main objectives of each LOPE agent is to autonomously learn operators (action models). Each agent receives perceptions from the environment, called situations, applies actions, and learns from its interaction with the outside world (environment and other agents).

The agent loops by executing an action, perceiving the resulting situation and utility, learning from observing the effect of applying the

action in the environment, and planning for further interactions with the environment when the previous plan has finished its execution, or the system observes a mismatch between the predicted situation by the agent's operators and the situation it perceived from the environment [Laird *et al.*, 1986; Minton, 1988; Sutton, 1990; Shen, 1993].

The planner does a backward chaining search from the initial situation (goal) of the operator with the highest utility in order to find a sequence of operators that will lead from the current state to that goal. If it succeeds, and the probability of its success is greater than a given bound, it executes the plan. If not, it selects the next highest utility operator and searches for a plan. This process loops until it finds a plan.

3 Representation

LOPE builds its operators based on the perceptions it receives from its sensors.

The planning operator model has the structure and meaning described in Table 1, where *p-list*, *action*, and *U* are domain-dependent and have the following meaning:

PLANNING OPERATOR: O_i		
Feature	Description	Values
<i>C</i>	Initial Situation (conditions)	<i>p-list</i>
<i>A</i>	Action	<i>action</i>
<i>F</i>	Final Situation	<i>p-list</i>
<i>P</i>	Time that the operator O_i was successfully applied (the expected final situation, <i>F</i> , was obtained)	<i>integer</i>
<i>K</i>	Times that the action <i>A</i> was applied to <i>C</i>	<i>integer</i>
<i>U</i>	Utility level reached applying the action to the initial situation, <i>C</i> , of the operator	<i>real</i> [0, 1]

Table 1. General description of an operator

- *p-list* is a list of propositions

- *action* can be any of the set of allowed actions that each agent can perform.
- *U* is a function that measures how useful the current situation is for the agent, and refers implicitly to the distance to the agent's goal. In the performed experiments, it has been measured as the distance of a robot (agent) to the closest energy point.

The parameters *P* and *K* allow the architecture to decrease the effect of noise in the sensors or in the environment, and hidden state problems [García-Martínez and Borrajo, 1997; 2000; García-Martínez *et al.*, 2006].

4 Planning

The planner builds a sequence of planning operators that allows to efficiently reach the top-level goal. This goal is implicit in the system and depends on the measure of utility. At each planning cycle, the system receives the highest utility (as, for instance, being on top of an energy point) if it reaches a situation yielding such utility.

Since each operator has its own utility, the planner will try to build plans that transform each current situation into the situation described by the condition of the operator with the highest utility; that is, the operator that achieves the highest utility situation (its final situation). The resulting plan will allow the execution module to perform a set of actions that interact with the environment in such a way that it arrives to a situation with highest level of utility.

More details and an example of a learning episode can be found in [García-Martínez and Borrajo, 2000].

5 Classical ponderation of Plans

In order to estimate the probability of success of plans, the planner is based on an extension of the theory of stochastic automata. The knowledge that the system has at a given time, the set of planning operators, can be viewed as a model of how its environment will react to the system's actions. The quotient P_{O_i} / K_{O_i} of a given operator O_i is the probability estimator [García-Martínez, 1997] of the fact that given the action A_{O_i} applied to a situation S_j that matches the operator

conditions ($C_{O_i} \subseteq S_j$) results in a situation S_k that verifies the predicted effects of the operator ($F_{O_i} \subseteq S_k$). Therefore, the knowledge that the system has about the effects of an action A_i at a given instant can be represented by the transition matrix M_{A_i} , that has, in the (j, k) position, the quotient P_{O_i} / K_{O_i} of operator O_i whose action is A_i , its conditions are S_j , and the predicted effects S_k [Calistri-Yeh, 1990].

The P s and K s of the plan operators can be used in the evaluation of the plans that are generated. This “a priori” estimation of the plan success probability allows to discard plans with a low probability of success ($P / K < \tau$), where τ is a predefined threshold. This property is critical when the system must act without supervision.

6 A new approach to ponderation of plans

In this section, an alternative to the preceding method is presented for the estimation of the probability of success of the plans (ponderation of plans).

In first place, for the proposed method it is taken as accepted from the previous algorithm (section 5) the fact that the quotient P_{O_i} / K_{O_i} of a given operator O_i , is the probability estimator of the fact that given the action A_{O_i} applied to a situation S_j that matches the operator conditions ($C_{O_i} \subseteq S_j$) results in a situation S_k that verifies the predicted effects of the operator ($F_{O_i} \subseteq S_k$) [García-Martínez, 1997]. This sounds reasonable, since the parameter P of an operator represents the quantity of times that this operator was used successfully, and the parameter K represents the number of times that the operator was used (section 3).

Therefore, the knowledge that the system has about the effects of applying an action A_k in a given moment could still be represented by a transition matrix M_{A_k} that has, in the position (i, j) , the quotient P / K of the operator whose action is A_k , its initial situation is S_i , and its final situation is S_j . However, the way in that the probability of success of the plan is calculated is different.

Suppose that in the planning process in order to reach the situation S_j from situation S_i , the plan P_{ij} can be built based on the following r operators that have the structure shown in section 3 (*Initial Situation, Action, Final Situation, P, K, U*):

$$P_{ij} = O_1 \circ O_2 \circ \dots \circ O_r \quad (1)$$

Where the symbol “o” represents the composition operation, and:

$$\begin{aligned} O_k &= (S_{i_k}, A_k, S_{f_k}, P_k, K_k, U_k) \text{ with } 1 < k < r \\ S_{i_1} &= S_i \\ S_{f_m} &= S_{i_{m+1}} \text{ with } 1 < m < r - 1 \\ S_{f_r} &= S_j. \end{aligned}$$

Then, the estimated probability that the plan P_{ij} applied to situation S_i , achieves the situation S_j (success of plan) is obtained as the result of the following product:

$$\hat{P}(\text{success}) = \frac{P_1}{K_1} \cdot \frac{P_2}{K_2} \cdot \dots \cdot \frac{P_r}{K_r}. \quad (2)$$

This result is based on the conditional probability concept, but the complete logical foundation of the method can be found in [Lopez, 2005].

As an example of how this algorithm proceeds, suppose that the system already built the following set of operators:

$$\begin{aligned} O_1 &= (S_1, A_1, S_2, 3, 4, 0.6), \\ O_2 &= (S_1, A_1, S_4, 1, 4, 0.4), \\ O_3 &= (S_2, A_2, S_3, 3, 4, 0.4), \\ O_4 &= (S_2, A_2, S_5, 1, 4, 1), \\ O_5 &= (S_3, A_1, S_1, 3, 3, 0.4), \\ O_6 &= (S_4, A_2, S_2, 1, 1, 0.8), \\ O_7 &= (S_5, A_1, S_4, 1, 1, 0.2). \end{aligned}$$

In that set there are two actions A_1 and A_2 , and five situations S_1 to S_5 . Suppose that the planner created the plan $O_1 \circ O_4$ (actions A_1 and A_2) in order to reach situation S_5 (self-generated goal) from S_1 (current situation).

The probability that the plan:

$$P_{15} = A_1 \circ A_2$$

or the same plan expressed in terms of operators:

$$P_{15} = O_1 \circ O_4$$

applied to the situation S_1 , achieves the situation S_5 is:

$$\hat{P}(\text{success}) = \frac{P_1}{K_1} \cdot \frac{P_2}{K_2} = \frac{3}{4} \cdot \frac{1}{4} = \frac{3}{16}.$$

In the figure 1 it is presented the algorithm associated to the process of ponderation of plans.

This algorithm is quite straightforward. Its input is the plan for pondering specified as a composition of the operators (S_i, A, S_f, P, K) in which is based.

The way that the pondering algorithm works is as follows.

In first place, it takes the first operator O of the plan (step 1) and calculates the value P as the quotient P_O / K_O (step 2). Then, begins a loop (step 3) in which the next operator O of the plan is obtained (step 3.1), and its quotient P_O / K_O is multiplied by the value P calculated previously (step 3.2). This loop ends when there are no more operators in the plan. After the loop is ended, the value P is returned (step 4). This value is the probability estimator for the plan.

INPUT: Plan specified as composition of the involved operators
OUTPUT: Estimator of the success probability of the plan
<ol style="list-style-type: none"> 1. O = first operator of the plan 2. $P = P_O / K_O$ 3. While there are more operators in the plan <ol style="list-style-type: none"> 3.1 O = next operator of the plan 3.2 $P = P \cdot P_O / K_O$ 4. Return the value of P

Fig. 1. Probability estimation algorithm.

From the point of view of the computational complexity, if the plan to ponder is represented by the expression (1), then from the expression (2) can be inferred that the amounts of arithmetical operations needed by the method in order to ponder a plan having r actions are the ones summarized in table 2.

r	divisions
$r-1$	multiplications

Table 2. Computational complexity of proposed method

The important point regarding these results is that the total amount of arithmetical operations

does not depend on the number of situation discovered by the system at the time of ponderation. This fact is in contrast with the computational complexity for the classical approach [Lopez, 2005], in which the amounts of arithmetical operations needed by the method in order to ponder a plan having r actions when the system has discovered n different situations are those summarized in table 3.

In this case, a represents the number of different actions that the system can execute. As can be seen, the computational complexity grows linearly with the number of steps of the plan (r), but it grows much more quickly (cubically) with the number of situations discovered by the system (n).

$a.n^2$	divisions
$(r-1).n^3$	multiplications
$(r-1).n^2.(n-1)$	additions

Table 3. Computational complexity for classical method

7 Experiments and results

We performed several experiments in previous papers to test the behavior of LOPE [García-Martínez and Borrajo, 1997]. In order to test the effect of ponderation of plans by means of proposed method, we performed new experiments. On each experiment, we averaged the results of running 300 tests. In each test, the initial setup (environment-grid and initial position of the agent) was randomly selected, and each LOPE agent performed 500 cycles of learning, planning and execution. Eight grids of 500 x 500 pixels were used consisting of energy points (goals) and obstacles. In the multiple agent setup, two agents were used that shared their knowledge acquired in the same grid. We compare here eight experiments:

- SC: a single LOPE agent learning in a single grid where a probability estimator is assigned to each operator. This estimator is the quotation P / K of each operator. It is used to assign a confidence to the generated plans by means of the classical ponderation method, so that plans with low confidence (< 0.6) are discarded.

- SP: same as SC except that the proposed method is used in order to assign a confidence factor to the generated plans.
- SGC: a single LOPE agent learning in a single grid, in which operators are generalized [García-Martínez and Borrajo, 2000], and a probability estimator is assigned to each operator to assign a confidence to the generated plans by means of the classical ponderation method, so that plans with low confidence are discarded.
- SGP: same as SGC except that the proposed method is used in order to assign a confidence factor to the generated plans.
- MC: a set of LOPE agents learning at the same time in the same grid configuration with the complete sharing strategy [Maceri and García Martínez, 2001], and where a probability estimator is assigned to each operator to assign a confidence to the generated plans by means of the classical ponderation method.
- MP: same as MC except that the proposed method is used to assign the confidence.
- MGC: a set of LOPE agents learning at the same time in the same grid with the complete sharing strategy. In this grid the operators are generalized, and a probability estimator is assigned to each operator to assign a confidence to the generated plans by means of the classical ponderation method.
- MGP: same as MGC except that the proposed method is used to assign confidence.

We used the percentage of successful plans when comparing these versions of the system, and the results of the experiments are shown in Figures 2 to 5.

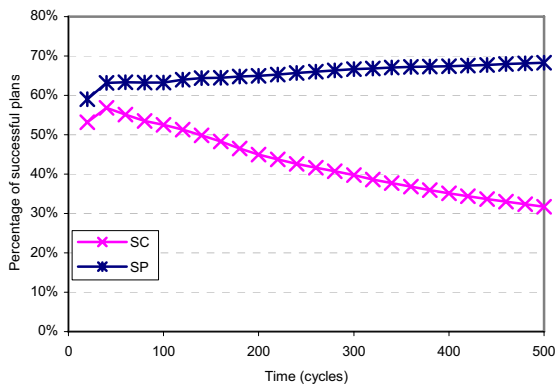


Fig. 2. Results using only ponderation

Figure 2 shows the experimental results of using only ponderation of plans. These results clearly show that the system using only probability estimation by means of the proposed ponderation method (SP) outperforms the system using the classical method (SC). This is so since the percentage of successful plans for the proposed method stays above the other one during the whole simulation interval.

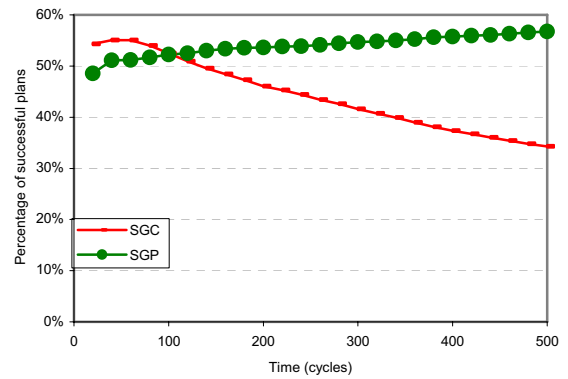


Fig. 2. Results using ponderation and generalization

Figure 3 shows the experimental results of using generalization and ponderation of plans. In this case, it can be seen that at the beginning of the simulation (when the system has little knowledge) it is better to use the classical ponderation method (SGC). However, after 100 cycles aprox. the use of the proposed method (SGP) outperforms the results of the system using the classical method.

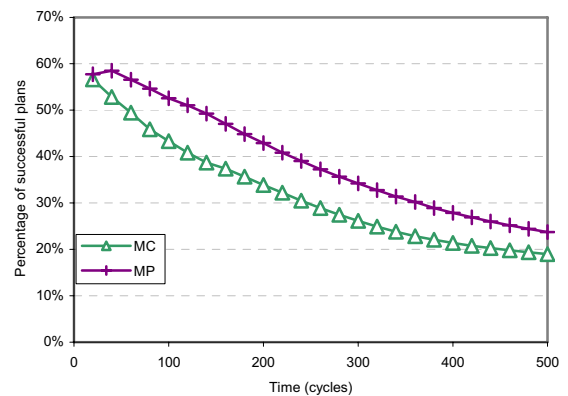


Fig. 4. Results using ponderation and knowledge sharing

Figure 4 depicts the experimental results of using ponderation of plans combined with knowledge sharing. It can be seen that use of the proposed method (MP) leads to a better performance than

the classical one (MC) during the whole simulation.

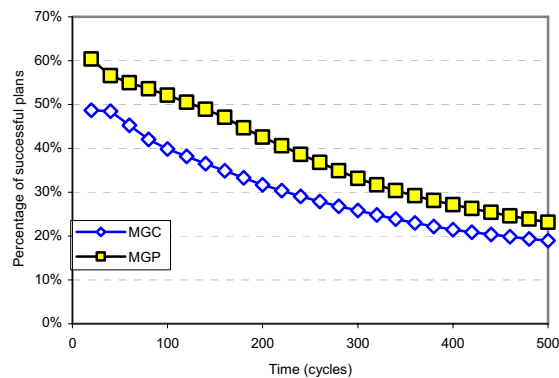


Fig. 5. Results using ponderation, generalization and knowledge sharing

Figure 5 shows the behavior of the system using ponderation of plans combined with generalization and knowledge sharing. From this figure can be seen that similarly to the showed by the figure 4, the use of the proposed method (MGP) leads to a better performance than the classical one (MGC) during the whole simulation.

8 Conclusions

In this paper, we have used an architecture that learns a model of its environment by observing the effects of performing actions on it. The LOPE agents autonomously interact with their environment and with other agents with the objective of learning operators that predict, with a given probability estimator, the resulting situation of applying an action to another situation. The operators are used to construct plans that allow the system to reach its top level goals.

A new method for pondering plans has been proposed with the goals of improving the performance (percentage of successful plans) of the system.

The experimental results clearly show that the proposed method causes a significant increase of the performance of the intelligent systems, measured as the percentage of plans executed successfully.

As a result of the above-mentioned, the proposed method accelerates the learning of the model of the environment, since a greater quantity of plans executed successfully, contributes to reinforce the involved theories (operators) more quickly,

accelerating the convergence of the estimators P/K to the probability distributions of the environment.

Finally, an additional advantage of the proposed method is that it presents a computational complexity much smaller than the one of the classic method. This is so since the matrix products of the classic method (whose complexity grows cubically with the amount of discovered situations) are transformed into products of scalar numbers, whose complexity does not depend on this variable. This is a very desirable feature in relation with the scalability of the approach in complex and noisy domains.

9 References

- Borrajo, D. and Veloso, M. (1997), *Lazy incremental learning of control knowledge for efficiently obtaining quality plans*. AI Review Journal. Special Issue on Lazy Learning, 11(1-5):371-405.
- Calistri-Yeh, R. (1990), *Classifying and detecting plan-based misconceptions for robust plan recognition*. PhD Thesis, Department of Computer Science, Brown University.
- Fritz, W., García-Martínez, R., Blanqué, J., Rama, A., Adobbati, R., and Samo, M. (1989). *The autonomous intelligent system*. Robotics and Autonomous Systems, 5(2):109-125.
- García-Martínez, R. and Borrajo, D. (1997). *Planning, learning, and executing in autonomous systems*. Lecture Notes on Artificial Intelligence 1348 : 208-220, Springer-Verlag.
- García-Martínez, R. and Borrajo, D. (2000). *An Integrated Approach of Learning, Planning and Executing*. Journal of Intelligent and Robotic Systems 29(1):47-78.
- García-Martínez, R., Borrajo, D., Britos, P. y Maceri, P. *Learning by Knowledge Sharing in Autonomous Intelligent Systems*. Lecture Notes in Artificial Intelligence, 4140: 128-137; 2006.
- García-Martínez, R.: (1997). *Un Modelo de aprendizaje por observación en planificación*. PhD Thesis. Facultad de Informática, Universidad Politécnica de Madrid.
- Ierache, J., García-Martínez, R., De Giusti, A. (2008). *Learning Life-Cycle in Autonomous*

- Intelligent Systems*. In *Artificial Intelligence in Theory and Practice II*, ed. M. Bramer, (Boston: Springer), in press.
- Laird, J. E., Rosenbloom, P. S., and Newll, A. (1986). *Chunking in soar: The anatomy of a general learning mechanism*. *Machine Learning*, 1(1):11-46.
- López, E. (2005). *Un Método de Ponderación de Planes en Sistemas Inteligentes Autónomos. Tesis de Grado en Ingeniería Informática*. Facultad de Ingeniería. Universidad de Buenos Aires.
- Minton, S. (1988). *Learning Effective Search Control Knowledge: An Explanation-Based Approach*. Kluwer Academic Publishers, Boston, MA.
- Shen, W. (1993). *Discovery as autonomous learning from environment*. *Machine Learning*, 12:143-165.
- Sutton, R. (1990). *Integrated architectures for learning, planning, and reacting based on approximating dynamic programming*. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216-224, Austin, TX. Morgan Kaufmann.
- Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Springer Verlag.