Learning by Knowledge Sharing in Autonomous Intelligent Systems

Ramón García-Martínez, Daniel Borrajo, Pablo Maceri, and Paola Britos

Software and Knowledge Engineering Center, Graduate School,
Buenos Aires Institute of Technology, Argentina
Departamento de Informática, Universidad Carlos III de Madrid, Spain
Intelligent Systems Laboratory. School of Engineering,
University of Buenos Aires, Argentina
rgm@itba.edu.ar

Abstract. Very few learning systems applied to problem solving have focused on learning operator definitions from the interaction with a completely unknown environment. In order to achieve better learning convergence, several agents that learn separately are allowed to interchange each learned set of planning operators. Learning is achieved by establishing plans, executing those plans in the environment, analyzing the results of the execution, and combining new evidence with prior evidence. Operators are generated incrementally by combining rote learning, induction, and a variant of reinforcement learning. The results show how allowing the communication among individual learning (and planning) agents provides a much better percentage of successful plans, plus an improved convergence rate than the individual agents alone.

1 Introduction

Given unknown environments, real autonomous systems must generate theories of how their environment reacts to their actions, and how the actions affect the environment. Usually, these learned theories are partial, incomplete and incorrect, but they can be used to plan, to further modify those theories, or to create new ones. Previous work on machine learning applied to problem solving has mainly focused on learning knowledge whose goal was to improve the efficiency of the problem solving task [Borrajo and Veloso, 1997; Laird et al., 1986; Minton, 1988; Veloso, 1994]. There is also a current interest in learning state transition probabilities in the context of reinforcement learning [Sutton, 1990; Watkins and Dayan, 1992]. However, few researchers have approached the generalized operators acquisition problem [Carbonell and Gil, 1990; Wang, 1996], described as techniques for automatically acquiring generalized descriptions of a domain theory. This issue is crucial when dealing with systems that must *autonomously* adapt to an unknown and dynamic environment.

LOPE (Learning by Observation in Planning Environments) is an agent architecture that integrates planning, learning, and execution in a closed loop, showing an autonomous intelligent behavior [García-Martínez and Borrajo, 1997, 2000]. Learning planning operators (what we will call operators, is also referred to as action models within the reinforcement learning community) is achieved by observing

the consequences of executing planned actions in the environment. 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. In our previous work, we presented a single agent architecture. Here, we will concentrate on the multiple agents behavior. More concretely, we present the learning mechanism, generalizing the one presented in those papers, and extending it by demonstrating how knowledge may be shared among many agents. The results show how the learning mechanism, outperforms the behavior of the base planner with respect to the production of successful plans (plans that achieve self-proposed goals). But, more importantly, they also show how the interaction with other learning agents greatly improves learning convergence and successful behavior. Section 2 describes the general architecture of the LOPE agents. Section 3 defines the representation that will be used in the paper for situations, observations and planning operators. Section 4 presents the learning model and its components (high level learning algorithm and heuristic generalization of operators). Section 5 deals with how agents share learned knowledge. Section 6 present the experiments and their results. Section 7 compares our approach with related work. Finally, section 8 draws some conclusions.

2 General Description

One of the main objectives of each LOPE agent is to autonomously learn operators (action models) that predict the effects of actions in the environment by observing the consequences of those actions. In order to learn those descriptions, it is able to plan for achieving self-proposed goals, execute the plans, find out incorrect or correct behavior, and learn from the interaction with the environment and other agents. Each agent receives perceptions from the environment, called *situations*, applies actions, and learns from its interaction with the outside world (environment and other agents). At the beginning, the agent perceives the initial situation, and selects a random action to execute in the environment. Then, it loops by executing an action, perceiving the resulting situation and utility of the situation (a classical reward from the environment, further explained in section 3), 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. The planner will not be described in this paper. Basically, it 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 for any high utility operator. More details on how the planner works can be found in [García-Martínez and Borrajo, 1997, 2000]. Figure 1 shows an schematic view of the architecture, where there can be n LOPE agents. Each of the agents receives as input: perceptions from the environment (situation and utilities); set of actions that it

can perform; and operators learned by other agents. The output of each agent is a sequence of actions over time (for the environment), and, regularly the set of operators that it learned (for the other agents).

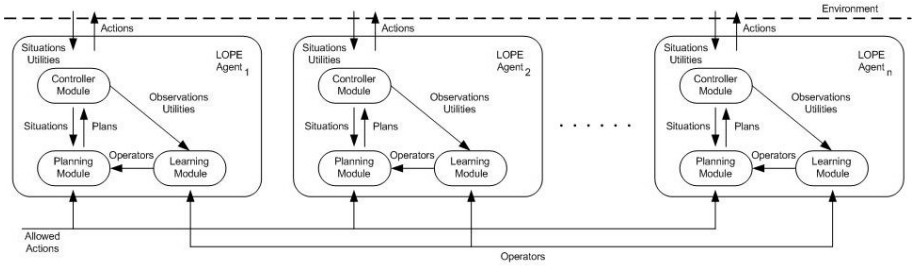

Fig. 1. Architecture of a group of LOPE agents

3 Representation

The autonomous agent type of world that we used for the experiments (robotic tasks) were two-dimensional grids, where each position within a grid can either contain obstacles, energy points, or be empty. For LOPE, as for many other systems, there is a difference between the world states, common to classical planning, and the observations it perceives. While classical planners are mainly interested in the high-level descriptions of the states (e.g. on(A,B) in the blocks world), LOPE builds its operators based on the perceptions it receives from its sensors (the word sensors refer to the generic idea of input, so this is applicable to non robotic domains); its "states" are the inputs it receives from its sensory system. Any post-processing of its inputs in order to translate them into high-level descriptions can be done without affecting the overall behavior. Because of the natural limitations of the sensory system, the agents map different states of the environment in to a single one, so our system manages noisy domains and hidden states. Previous work of one of the authors developed early versions of the learning mechanism. The representation was based on the model proposed in [Fritz *et al*., 1989], in which an observation (also called experience unit) had the following structure: [Initial Situation, Action, Final Situation]; where initial and final situations are lists of propositions that can be either true or false. In [Fritz *et al*., 1989], observations were directly used as planning operators. In [García-Martínez and Borrajo, 1997, 2000], while the concept of an observation does not change, the representation of operators is extended, by the addition of features that allow the system to determine their planning/execution acceptability. The proposed planning operator model has the structure and meaning described in Table 1, where *C*, *F*, *action*, and *U* are domain-dependent and have the following meaning: *C* and *F* describe Initial Situation and Final Situation through a list of propositions (p-list) that can be preceded by the ¬ symbol, denoting negation, if a proposition does not appear on that list, it is assumed that its value does not matter; *action*: can be any of the set of allowed actions that each agent can perform, for instance, in a robotic domain, it could be "go" and "turn" and *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

(similar concept to the reward in reinforcement learning [Watkins and Daya, 1992]). This function could be changed to allow different behaviors, each one depending on their specific goals. 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.

4 Learning Planning Operators

We will first define when operators and observations are similar, equal or one confirms another. Then we will present the high level learning algorithm and the heuristic generalization of operators.

Given two operators $O_1 = [C_1, A_1, F_1, P_1, K_1, U_1]$ and $O2 = [C_2, A_2, F_2, P_2, K_2, U_2]$, and an observation $o = [S_i, A, S_f]$, we say that: (i) two operators are **similar** if $C_1 = C_2$ and $A_1 = A_2$, (ii) two operators are **equal** if $C_1 = C_2$, $A_1 = A_2$, and $F_1 = F_2$, (iii) observation is **similar** to the operator O_1 if $S_i \subseteq C_1$ and $A = A_1$, (iv) observation **confirms** the operator O_1 if $S_i \subseteq C_1$, $A = A_1$, and $S_f \subseteq F_1$. Here, the predicate $\subseteq$ tests whether a list of propositions subsumes another list, and the predicate = tests whether a list of propositions is equal to another.

For presenting the **high level learning algorithm** suppose a situation S_i is perceived by the system, and there exists a set of operators, φ, such that each operator is of the form $O_i = [C,A,F,P,K,U]$. If the system applies the action A, arriving at a situation S_f, the learning method processes this new observation by the algorithm shown in Table 2. When a new observation arrives at the learning module, it checks if a similar operator exists. If it is *similar*, it checks to see if the observation confirms the operator. Then, it rewards all such operators and punishes *similar* ones. If a *similar* operator exists, but there is none that is *confirmed by the observation, it* creates a new operator, punishes *similar* operators to the new one, and generalizes those *similar* operators. The operators generated by the generalization procedure reward *equal* operators and punish *similar* ones. If it does not find a *similar* operator for the input observation, it creates a new one. *Punishing* operators means incrementing in a given quantity *r* the number of times that the pair (condition, action) of *similar* operators to O has been observed. As the algorithm shows, punishment not only occurs when observations are made, but also when new generated by the heuristic generalization. This is so because K really accounts for the number of times that similar operators have been generated or seen. Similarly, *rewarding* operators means incrementing in *r* the P and K of a successful operator. The parameter *r* usually is equal to one, but on Section 5 different values are used for integrating operators of several agents with different Ks. The terms *punish* and *reward* have been borrowed from the field of biological reinforcement rather than from reinforcement learning. The heuristic-generalization algorithm generates a set of new operators according to the generalization heuristics, which are incorporated into the set of planning operators. Since the number of operators that are created can potentially slow down the performance of the learning and planning modules, the system forgets operators with a very low quotient P/K, given enough observations have been made.

The **heuristic generalization of operators** is based on the heuristics defined in [Hayes-Roth, 1983] and [Salzberg, 1985]. For the following discussion, suppose that the new observation is described by (S_i, A, S_f), the domain operator is described by

[C, A, F, P, K, U], and the new generalized observation (m) is [C_m, A_m, F_m, U_m]. In order to apply a heuristic, there had to be a fault in using the corresponding operator in the observed initial and resulting situations.

Hayes-Roth [1983] proposed the following set of heuristics for revising a faulty (buggy) theory: *Retraction* generalizes an operator predicted situation so that it is consistent with the new observation, if $S_f \not\subset F$, then m = [C, A, F', U] where F' is a generalization of F and S_f; *Exclusion* restricts the conditions of the operator, so that it does not apply in the observed situation again, given that $S_i \subseteq C$ and $S_f \not\subset F$, then m = [C', A, F, U], where C' is built by selecting a proposition that does not belong to C (does not matter) and change it to the negation of what appears in the observation; *Inclusion*: generalizes the operator conditions, so that it will later apply in the observed situation. If $S_f \subseteq F$ and $S_i \not\subset C$, then m = [C', A, F, U], where C' is a generalization of C and S_i. It is the equivalent to retraction, but applied to the conditions of the operator.

The following Salzberg (1985) heuristics are used to correct prediction violations: *Inusuality* restricts the conditions of an operator, so that it will not longer apply to the observed initial situation, if $S_i \subseteq$ and $S_f \not\subset F$, then m = [C', A, F, U], where C' is a specialization of C, adding all propositions not appearing in C (does not matter) by the negation of their value in S_i, differs from Hayes-Roth exclusion, in that it adds all propositions to C; *Conservationism*: it is a meta-heuristic that selects the generalization heuristic (from the Salzberg ones), that proposes less modifications in the conditions of an operator; *Simplicity*: it is a generalization of the Hayes-Roth inclusion heuristics; *Adjustment*: when the P/K ratio of an operator falls below a given threshold, it is very unlikely that the operator will correctly predict any situation, if it is a generalization of a set of operators (for instance, by application of the simplicity heuristic), this heuristic generates other combinations of those operators that will increase the ratio.

5 Learning by Sharing

Previous work of the authors presented how the integration of this combined learning mechanism with planning and execution allowed the system to improve the ratio of successful plans (sequence of actions that lead to arrive at the locations of the energy points) [García-Martínez and Borrajo, 1997]. In order to improve the learning convergence, and to test the generality of the learned knowledge, we performed experiments in which the system remembers the operators learned in an example grid g1 when planning and learning in other configurations of the grid, g_2 and g_3 (obstacles and energy points in different places). In those experiments we showed how that prior knowledge provided a faster learning convergence than not using it, and the Section 6 shows the results obtained. We decided then to experiment with the inclusion of new agents of the same type, learning and sharing what they learned in the same grid configuration, and testing how that affected the learning and planning behavior. Agents cannot occupy the same position in the grid, and the sensors of one agent consider the other agents as obstacles. Under this framework, each agent continuously learns, plans and executes. However, when they were close to another agent, they were allowed to communicate in order to interchange what they learned, operator

descriptions. We devised two types of knowledge sharing strategies: Complete sharing and Most reliable operator sharing.

In **complete sharing strategy** every pair of agents integrate their respective theories (set of operators) using all operators in the sets. The algorithm is shown in Table 3. For each operator of another agent (a_2), an agent (a_1) looks for similar operators in its theory. If there is no such similar operator, then the a_2's operator is included in the set of operators of a_1. If a similar operator is found, then all such operators are punished with the P of a_2's operators. If there is no equal operator, then it is included in a_1's operators with the K of its similar operators in a1's theory.

Table 1. General description of an operator

Planning Operator: O_i		
Feature	*Description*	*Values*
C	Initial Situation (conditions)	p-list
A	Action	action
F	Final Situation	p-list
P	Times that the operator O_i was succesfully applied (the expected final situation, F, was obtained)	integer
K	Times that the action A was applied to C	integer
U	Utility level reached applying the action to the initial situation, C, of the operator	real 0..1

Table 2. Algorithm that integrates the operators of two agents

Function Complete-sharing (φ_1, φ_2) : φ_1

φ_1: Set of operator of agent 1
φ_2: Set of operator of agent 2

```
Forall O_i ∈ φ_2 do
    If exists O_j ∈ φ_1 such that CO_j = CO_i
  AND AO_j = AO_i
    Then φ_1 := punish-operators(O_j, φ_1, PO_i);
        If any of the similar operators to O_i,
      O_k, is such that FO_k = FO_i
        Then O_k := reward-operators(O_k,
      PO_i)
        Else KO_i := KO_k;
            φ_1 := φ_1 U {O_i}
    Else φ_1 := φ_1 U {O_i};
Return φ_1
```

Table 3. Algorithm that modifies operators descriptions after having seen a new observation

Function Learning (S_i, A, S_f, U, φ) : φ

S_i: Initial situation of the observation
A: Applied action of the observation
S_f: Observed final situation
U: Observed utility
φ: Set of operator descriptions

```
If exists O_i ∈ φ such that S_i ⊆ CO_i AND
A = AO_i Then
  If S_f ⊆ FO_i Then
    Forall O_i such that S_i ⊆ CO_i
    AND =AO_i AND S_f ⊆ FO_i do
      O_i := reward-operator(O_i, 1);
      UO_i := max(UO_i, U);
      φ := punish-operators(O_i, φ, 1)
    Else O_n := [S_i, A, S_f, 1, KO_i + 1, U];
      φ := φ U {O_n}
      φ := punish-operators(O_n, φ, 1)
      M := heuristic-generalization(S_i,A, S_f,φ);
      Forall M ∈ M do
        If exists O_j∈ φ such that CO_j=C_M
        AND AO_j=A_M Then
          If FO_j = F_M Then
            O_j := reward-operator(O_j, 1);
            φ := punish-operators(O_j, φ, 1)
          Else O_n := [C_m, A_m, F_m, 1, KO_j + 1,
        UO_j];
            φ := φ U [O_n];
            φ := punish-operators(O_n, φ, 1)
        Else φ := φ U {[C_m, A_m, F_m, 1, 1, U_m]};
Else φ := φ U {[S_i, A, S_f, 1, 1, U]}
Return φ
```

In **most reliable operator sharing strategy** every time two agents share their knowledge, only the most liable operators are share (the ones that maximize the

quotient P/K). The only difference with the prior algorithm is that instead of providing it as input with φ_2, the algorithm is called with the set of most liable operators. This set is computed by selecting from each set of similar operators of an agent, the one with maximum P/K.

6 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 sharing the knowledge among the agents, we performed new experiments which we then compared with a summary of the best results of the previous ones. On each experiment, we averaged the results of running 50 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. Grids of 700 x 100 pixels were randomly created consisting of 10-20 randomly situated energy points (goals) and obstacles (10% to 20% of the grid area was covered by them). In the multiple agents setup, the agents shared their knowledge when they were 20 pixels away from another. We compare here seven experiments: *SG*: a single LOPE agent learning in a single grid, in which operators are generalized; *SP*: 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 learned operator, also, it is used to assign a confidence to the generated plans, so that plans with low confidence are discarded. The decisions of the agent are based on sensory input only when there is no plan on execution. We have shown previously that the P/K of similar operators follows a multinomial distribution of probability and that is an unbiased estimator of the probability. Also, when an exact theory of the domain exists, the operators that have been built applying the learning mechanism based on observations convergence to the exact ones; *MC*: a set of LOPE agents (we used two for these experiments) learning at the same time in the same grid configuration with the complete sharing strategy; *SGP*: a single LOPE agent learning in a single grid, in which operators are generalized, and a probability estimator is assigned to each operator to assign a confidence to the generated plans, so that plans with low confidence are discarded; *MCG*: a set of LOPE agents learning at the same time in the same grid with the complete sharing strategy and in which operators are generalized; *MCP*: a set of LOPE agents learning at the same time in the same grid configuration with the complete sharing strategy, and where a probability estimator is assigned to each operator to assign a confidence to the generated plans, so that plans with low confidence are discarded; *MCGP*: 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, so that plans with low confidence are discarded.

We used the percentage of successful plans when comparing these versions of the system, and the results of the experiments are shown in Figure 2. First, these results clearly show that the combination of generalization and probability estimation (SGP) outperforms the system using only generalization (SG). Besides, using probability estimation in two agents with complete sharing (MCP) improves

in convergence rate with respect to the SP case, but it worse in the long run, since it converges to a lower percentage rate than the MCP case. It means that sharing knowledge among agents at the beginning is better than using only one agent. Other results show that the use of generalization in two agents with complete sharing (MCG) is worse without using generalization (MC) and even is worse when one agent use generalized operators (SG).

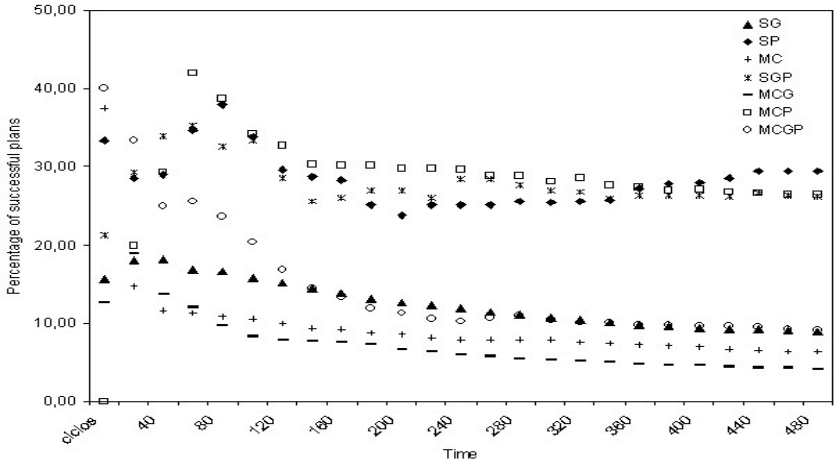

Fig. 2. Results of the experiments

7 Related Work

The closest related work is the one on reinforcement learning techniques within the Markov Decision Processes (MDP) paradigm (Mahavedan and Connell, 1992; Sutton, 1990; Watkins and Dayan, 1992). Also, current techniques that deal with Partially Observable Markov Decision Processes (POMDP) are very close to this approach [Kaebling *et al*., 1998]. Usually, they integrate reinforcement learning, planning and reacting based on approximated dynamic programming. It differs from our work in the fact that the reinforcement procedure is local to an operator, while, in our case, the reinforcement of an operator explicitly implies the punishment of similar ones (global reinforcement). The second difference refers to the fact that we use *symbolically generalized states*, instead of instantiated states (as most other work in reinforcement learning), or non-symbolically based generalized states (such as neural networks [Lin, 1995]. In fact, our approach can be viewed as a method for producing a generalized Q table using global reinforcement. Similar approaches, group sets of similar states and/or actions on big state/action spaces [Boutilier *et al*., 1995]. Most of this work uses different representation schemas, such as belief networks. A third difference lies on the type of planning scheme for which it is used. While reinforcement learning has been usually applied for more reactive planning (with some exceptions), our approach lies closer to the classical planning approach (plans are generated in a search-based fashion and later monitored for divergences between predicted and observed states) [García-Martínez and Borrajo, 1997; 2000]. A fourth difference with most work on

reinforcement learning lies in the fact that we do not deal with the temporal credit assignment problem. Each learning episode is handled independently of what happened before. Within this classical reinforcement learning framework, the work by Tan [1993] could be considered a predecessor of our work. He explores the cooperation among agents by sharing instantaneous information (perceptions, actions or rewards), sequences of perception-action-reward, and learned policies. The GINKO system [Barbehenn and Hutchinson, 1991], the LIVE system [Shen, 1993], and the work of Safra and Tennenholtz [1994] also integrate perception, action and learning. They differ from the proposed architecture in the fact that they do not take into account reinforcement nor heuristic-based refinement of operators. Chrisitansen [1992] also addresses the problem of learning operators (task theories) in a robotic domain. However, in his work there is no revision process as our heuristic-based refinement process. OBSERVER [Wang, 1996] uses an incremental approach for operators revision, where operators involve during the execution of the system. However, there is no memory of past versions of the operators as in LOPE. Also, OBSERVER uses predicate logic for representation, since its goal is to perform classical high-level planning. Our approach uses a representation that is close to the real inputs and outputs of system, with that intermediate type of planning between high-level and reactive planning. Other integrated planning and learning systems for robotic tasks are [Bennet and DeJong, 1996] and [Klingspor *et al*., 1996]. The first one deals with the concept of *permissiveness*, that defines qualitative behavior for the operators. The second one uses Inductive Logic Programming for learning the operators of the domain by doing a transformation from the sensor data into predicate logic. They both differ from our approach in that they need some type of prior background knowledge, either a predefined domain theory in the form of initial operators, or external instruction and knowledge on how to perform the transformation.

8 Conclusions

In this paper, we have presented 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. Two types of knowledge sharing strategies among the agents have been presented: sharing of all the acquired knowledge (operators), and sharing of only the best operator of different sets of operators. The results show that sharing the learned knowledge can greatly help an autonomous system to acquire a theory description that models the environment, thus achieving a high percentage of successful plans, and also improving the convergence rate for obtaining a successful theory. An important issue when allowing sharing of operators among agents, is related to the differences on their sensors, which causes different ways of perceiving the world, and, therefore, different biases towards the generation of operators. We have not yet studied this effect, although one possible way of solving it could be by learning other agents biases, in order to perform a more informed sharing of knowledge. With respect to the scalability of the approach, we are now performing experiments in a much more complex, noisy, with hidden states, and multi-agent domain, such as the Robosoccer. We believe that through the use of

the probabilities estimations, and the heuristic generalization of operators, we will be able to cope with the complexity of that domain.

References

Barbehenn, M. and Hutchinson, S. (1991). An integrated architecture for learning and planning in robotic domains. *Sigart Butlletin*, 2(4), 29-33.

Bennet, S. W. and DeJong, G. (1996). Real world robotics: Learning to plan for a robust execution. *Machine Learning*, 23, 121-162.

Borrajo, D. and Veloso, M. (1997). Lazy incremental learning of control knowledge for efficiently obtaining quality plans. *AI Review Journal*, 11, 371-405.

Boutilier, C., Dearden, R., and Goldszmidt, M. (1995). Exploiting structure in policy construction. *Proc. 14th Int. Joint Conf. on AI:* 1104-1111, Morgan Kaufmann.

Carbonell, J. G. and Gil, Y. (1990). Learning by experimentation: The operator refinement method. In (Michalski and Kodratoff, eds) *Machine Learning: An AI Approach, Vol. III* (pp. 191-213). San Francisco: Morgan Kaufmann.

Christiansen, A. (1992). *Automatic Acquisition of Task Theories for Robotic Manipulation*. PhD thesis, School of Computes Science, Carnegie Mellon University.

Fritz, W., García-Martínez, R., Blanqué, J., Rama, A., Adobbati, R., and Sarno, M. (1989). The autonomous intelligent system. *Robotics and Autonomous Systems*, 5, 109-125.

García-Martínez, R. and Borrajo, D. (1997). Planning, learning, and executing in autonomous systems. *Lecture Notes in Artificial Intelligence*, 1348, 208-220.

García Martínez, R. y Borrajo, D. (2000). An Integrated Approach of Learning, Planning and Executing. *Journal of Intelligent and Robotic Systems*, 29, 47-78.

Hayes-Roth, F. (1983). Using proofs and refutations to learn from experience. In (Michalski, Carbonell, and Mitchell eds) *Machine Learning, An AI Approach* (pp. 221-240). Palo Alto: Tioga Press.

Kaebling, L., Littman, M. and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 10, 99-134.

Klingspor, V., Morik, K., and Rieger, A. (1996). Learning concepts from sensor data of a mobile robot. *Machine Learning*, 23, 305-332.

Lin, L. (1995). Reinforcement Learning of Non-Markov Decision Processes. *Artificial Intelligence*, 73, 271-306.

Mahavedan, S. and Connell, J. (1992). Automatic programming of behavior-based robots using reinforce-ment learning. *Artificial Intelligence*, 55, 311-365.

Safra, S. and Tennenholtz, M. (1994). On planning while learning. *JAIR*, 2, 111-129.

Salzberg, S. (1985). Heuristics for inductive learning. *Proc. 9th International Joint Conference on AI*, pages 603-609, Los Angeles, CA.

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. *Proc. 7th Int. Conf. on ML*: 216-224. Kaufmann.

Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. *Proc. 10th International Conference on ML* (pp. 330-337), Amherst: Morgan Kaufman.

Wang, X. (1996). *Planning while learning operators*. PhD thesis, School of Computes Science, Carnegie Mellon University.