Integrating Task Planning and Interactive Learning for Robots to Work in Human Environments

Alejandro Agostini
Institut de Robòtica
i Informàtica Industrial
(CSIC-UPC)
Barcelona, Spain
agostini@iri.upc.edu

Carme Torras
Institut de Robòtica
i Informàtica Industrial
(CSIC-UPC)
Barcelona, Spain
torras@iri.upc.edu

Florentin Wörgötter
Bernstein Center
for Computational Neuroscience
Göttingen, Germany
worgott@bccn-goettingen.de

Abstract

Human environments are challenging for robots, which need to be trainable by lay people and learn new behaviours rapidly without disrupting much the ongoing activity. A system that integrates AI techniques for planning and learning is here proposed to satisfy these strong demands. The approach rapidly learns planning operators from few action experiences using a competitive strategy where many alternatives of cause-effect explanations are evaluated in parallel, and the most successful ones are used to generate the operators. The success of a cause-effect explanation is evaluated by a probabilistic estimate that compensates the lack of experience, producing more confident estimations and speeding up the learning in relation to other known estimates. The system operates without task interruption by integrating in the planning-learning loop a human teacher that supports the planner in making decisions. All the mechanisms are integrated and synchronized in the robot using a general decision-making framework. The feasibility and scalability of the architecture are evaluated in two different robot platforms: a Stäubli arm, and the humanoid ARMAR III.

1 Introduction

In the last years special emphasis has been placed on the development of robots capable of helping humans in carrying out human-like tasks. Human environments usually involve very large domains, where many unexpected situations can arise easily, and coding the behaviours to cope with every possible situation may result impractical. The alternative is to let the robot learn these behaviours autonomously. However, for this alternative to make sense, learning should occur rapidly to let the robot be operative in a reasonable amount of time, and without interrupting much the ongoing task every time a new behaviour should be learned.

This work presents a system that uses AI techniques for planning and learning to avoid the need of coding these behaviours in real robot platforms. The system integrates a logic-based planner and a learning approach that constantly enriches the capabilities of the planner for decision making, allowing the robot to fulfil a wide spectrum of tasks without a previous coding of planning operators. Learning and planning occur intertwined, without task interruption, and using experiences that arrive sequentially.

The strong requirements for learning behaviours pose insurmountable difficulties to the existing learning paradigms, where on-line learning is not considered [Oates and Cohen, 1996; Wang, 1996], a significant amount of prior knowledge need to be provided [Gil, 1994], or a large number of experiences are required [Walsh and Littman, 2008; Wang, 1996; Benson, 1995; Shen, 1989; Oates and Cohen, 1996]. To cope with the learning requirements, we devise a competitive approach that tries in parallel different explanations of the cause-effects that would be observed from action executions, and use the ones with higher probability of occurrence to code basic operators for planning. Trying different explanations of cause-effects in parallel increases the chances of having a successful explanation among the competing ones, which, in turn, increases the speed of learning. To determine the probability of occurrence we propose a specialization of the m-estimate formula [Cestnik, 1990] that compensates the lack of experience in the probability estimation, thus producing confident estimations with few examples.

Due to incomplete knowledge, the planner may fail in making a decision, interrupting the ongoing task. To prevent these interruptions, we include a human teacher in the planning-learning loop that provides the action to execute in the ongoing task when the planner fails. Finally, since the learning approach needs the actual experience of actions to evaluate the cause-effect explanations, the planning and learning mechanisms should be integrated in a more general framework that permits the grounding of the symbolic descriptions of actions, and the abstraction into attribute-values of the raw perceptions obtained from the sensors of the robot. To this end, we use a general decision-making framework that integrates and synchronizes all the involved mechanisms in the robot. The proposed system was successfully tested in two real robot platforms: a Stäubli robot arm, and the humanoid robot platform ARMAR III [Asfour *et al.*, 2008]. Next section presents the decision-making framework used for the integration. Section 3 briefly explains the planner and the role of the teacher. Then, in Section 4, the learning mechanisms are detailed and evaluated. The implementation in real robot platforms are described in Section 5. The paper ends with

some conclusions.

2 Decision-Making Framework

For the integration of planning and learning in the robot platform we use a conventional decision-making framework (figure 1). The PERCEIVE module, above the planner, generates a symbolic description of the initial situation by the abstraction of the values provided by the sensors into a set of discrete attribute-values. The initial situation, together with the GOAL specification, are used by the PLANNER to search for plans to achieve the goal from that situation. If a plan is found, the planner yields the first action to execute. If not, the TEACHER is asked for action instruction. The action, provided either by the planner or the teacher, is sent to the EXE module in charge of transforming the symbolic description of the action into low-level commands for the actual action execution. After the action execution, a symbolic description of the reached situation is provided by the other PERCEIVE module. The LEARNER takes the situations descriptions before and after action execution, together with the symbolic description of the action, and generates or refines cause-effect explanations and planning operators. After learning, the last situation perceived is supplied to the planner that searches for a new plan. This process is continued until the goal is reached, in which case the planner yields the end of plan signal (EOP).

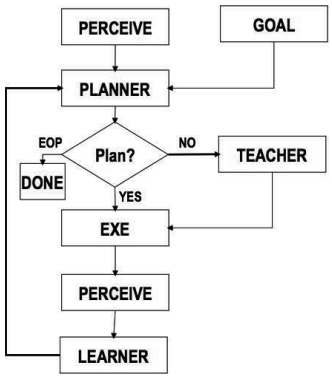

Figure 1: Schema of the decision-making framework.

3 The Planner and the Teacher

The logic-based planner implemented is the PKS planner [Petrick and Bacchus, 2002] which uses STRIPS-like planning operators [LaValle, 2006] for plan generation. STRIPS-like operators are widely used due to their simplicity and their capability of providing a compact representation of the domain, which is carried out in terms of relevant attribute-values that permit predicting the effects of executing an action in a given situation. This kind of representation is suitable for problems where the total number of attributes is large, but the attributes relevant to predict the effects in a particular situation is small. A STRIPS-like planning operator (PO) is composed of a *precondition* part, which is a logic clause with the attribute-values required for the given action to yield the desired changes, and the *effect* part which specifies additions and deletions of attribute-values with respect to the precondition part as a result of the action execution.

Since the PO database may be incomplete, which is the reason of why a learner is needed to complete it, the planner may fail to make a decision because of incomplete knowledge. In this case we may adopt two alternative strategies to support the planner in decision making. On the one hand, we may define an action selection strategy, e.g. select an action randomly or an action that would be taken in a similar situation, to provide the robot with an action to execute. On the other hand, we may simply use the help of a human teacher to instruct the action to execute. We choose to use a human teacher since this may diminish significantly the time spent for learning useful POs, and since its inclusion in the planning-learning loop is very simple and straightforward for the kind of applications we are dealing with: human-like tasks. Teacher instructions simply consist of a single action to be performed in the current situation according to the task in progress.

4 The Learner

The learner has the important role of providing the planner with POs. To this end, we propose a learning approach that evaluates in parallel cause-effect explanations of the form $CEC_i = \{C_i, E_i\}$, where C_i is the cause part, and E_i is the effect part. The cause part $C_i = \{H_i, a_i\}$ contains a symbolic reference of an action a_i, and a set of attribute-values H_i that, observed in a given situation, would permit to obtain the expected changes in the situation when a_i is executed. The effect part E_i codes these changes as the final values of the attributes that are expected to change with the action.

4.1 Planning Operator Generation

The execution of every action instructed by the teacher produces the generation of many alternative cause-effect explanations that compactly represent the observed transition, as well as the generation of a PO. First, a cause-effect explanation CEC_i is generated by instantiating a_i with the instructed action, H_i with the initial values of the attributes that have changed with the action, and E_i with final values of the changed attributes. From this initial CEC_i a planning operator is generated using a_i as the name of the operator, H_i as the precondition part, E_i as the additions in the effect part, while the values in H_i changed with the action (all of them in this case) are the deletions.

After the generation of the PO, many other cause-effect explanations are generated from the newly generated one. This is done to provide the learning method with more alternatives to try in parallel in case the newly generated PO fails (see next section). Every new additional explanation CEC_n is generated with the same action $a_n = a_i$ and effect $E_n = E_i$ of the CEC_i, but with a set H_n consisting of one among all the specializations in one attribute-value of H_i. This general to specific strategy is followed to keep a compact representation.

4.2 Planning Operator Refinement

A PO is refined every time its execution leads to an unexpected effect. First, all the cause-effect explanations that

share the same action a and effect E of the failed PO r are brought together,

$$\mathsf{CEC}_r = \{CEC_i | a_i = a, E_i = E\}. \quad (1)$$

Then, from the set CEC_r, the CEC with highest chance of occurrence,

$$CEC_w = \underset{CEC_i \in \mathsf{CEC}_r}{\operatorname{argmax}} P(E_i \mid H_i, a_i), \quad (2)$$

is selected for the refinement of the PO, using H_w to replace its precondition part.

Cause-Effect Evaluation

The problem of evaluating the $CEC_i \in \mathsf{CEC}_r$ can be handled as a classification problem, where each H_i represents a classification rule, and the classes are *positive*, when a situation covered by H_i permits to obtain E_i with a_i, and *negative*, otherwise. For example, the probability in (2) may be represented by a probability for a positive instance, $P_+ = P(E_i \mid H_i, a_i)$.

We require the system to rapidly generate and refine POs using as few experiences as possible. This implies that, if the lack of experience is not taken into account in the estimation, the approach may wrongly produce large premature estimations of these probabilities, degrading the performance of the system, mainly at early stages of the learning. To prevent premature estimations we use the m-estimate formula [Cestnik, 1990; Furnkranz and Flach, 2003],

$$P_+ = \frac{n_+ + m\,c}{n_+ + n_- + m}, \quad (3)$$

where n_+ is the number of experienced positive instances, n_- is the number of experienced negative instances, c is an *a priori* probability, and m is a parameter that regulates the influence of c. The m parameter plays the role of the number of instances covered by the classification rule. For a given c, the larger the value of m the lesser the influence of the experienced instances in the probability, and the closer the estimation to c. This permits to regulate the influence of the initial experiences with m, preventing large premature estimations. To illustrate how this regulation takes place, we use the extreme case of setting $m = 0$, which leads to the traditional frequency probability calculation,

$$P_+ = \frac{n_+}{n_+ + n_-} \quad (4)$$

where, if an estimation has to be done using only a couple of positive instances, it may produce a 100 % chances of being a positive, disregarding the uncertainty associated to the instances that are still pending to be tried. However, if we define a larger value of m, the influence of the observed instances decays and the estimation is closer to c. The setting of m is defined by the user according to the classification problem at hand. One known instantiation of the m-estimate is to set $m = 2$ and $c = 1/2$, in which case we have the Laplace estimate [Cestnik, 1990],

$$P_+ = \frac{n_+ + 1}{n_+ + n_- + 2}, \quad (5)$$

widely used in known classification methods such as CN2 [Clark and Boswell, 1991]. However, the original m-estimate does not provide a way of regulating the influence of m as more experiences are gathered since the value of m is assumed constant. This degrades the accuracy of the estimation as learning proceeds, it being worse for larger values of m, which makes the estimation to be biased towards c. To avoid this problem, we propose to use a variable m that consists in an estimation of the number of instances $n^\emptyset$ covered by the classification rule that are still pending to be tried,

$$P_+ = \frac{n_+ + \hat{n}^\emptyset\, c}{n_+ + n_- + \hat{n}^\emptyset}, \quad (6)$$

where $\hat{n}^\emptyset$ is an estimation of $n^\emptyset$. Equation (6) can be interpreted as the conventional frequency probability calculation (4), where each inexperienced instance contributes with a fraction c of a sample for each class. Note that, in (6), the value $\hat{n}^\emptyset$ is particular for each rule, regulating the influence of the lack of experience in each particular case. Since the kind of applications we are dealing with permits to calculate exactly the number of instances covered by a classification rule, n_T, we can calculate exactly the number of inexperienced instances as

$$n^\emptyset = n_T - n_+ - n_-. \quad (7)$$

Using (7) in (6), setting the prior probability as $c = 1/2$, and reformulating, we obtain,

$$P_+ = \frac{1}{2}\left(1 + \frac{n_+}{n_T} - \frac{n_-}{n_T}\right), \quad (8)$$

which we name *density*-estimate, and it is used hereafter for the probability estimation. Note that, with this equation, the probability of a class changes as a function of the density of samples for each class rather than as a function of the relative frequencies. Low densities of samples will produce low variations in the probability, preventing large premature estimations when few examples are collected. As learning proceeds, the influence of the densities will be larger and the probability estimation will tend to the actual probability. For instance, when all the instances are already experienced, we have $n_T = n_+ + n_-$, and equation (8) is equal to (4).

4.3 Performance Evaluation

The learning proposal has been thoroughly evaluated in different classification problems [Agostini *et al.*, 2011]. We present in this section an evaluation carried out in the binary classification problem of the Monk's problem number 2 [Thrun *et al.*, 1991]. We choose this problem since it is a complex classification problem that poses difficulties to many known classification methods, and since it permits a direct analogy with the binary classification problem of partitioning the set of attributes H_i into *positive* or *negative*. For the learning of the binary function, we use the competitive strategy (2), where each classification rule is equivalent to a set H_i of a $CEC_i \in \mathsf{CEC}_r$, a positive instance is equivalent to a situation in which E is obtained with a (see (1)), and a negative instance is equivalent to a situation in which E is not obtained when a is executed. We select, from all the classification rules covering a given instance, on the one hand, the rule

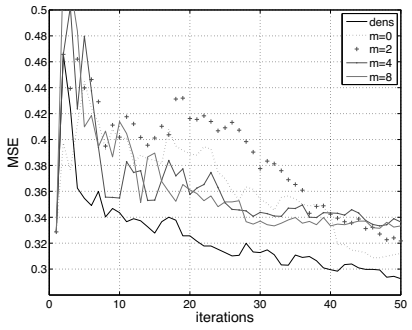

Figure 2: Comparing the performance of the *density*-estimate with that of the m-estimate at early stages of learning.

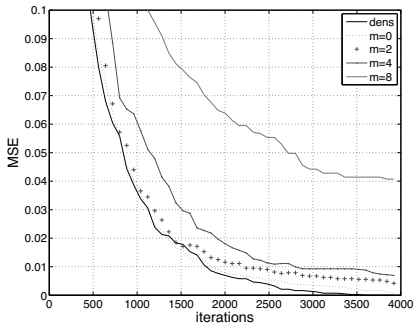

Figure 3: Comparing the performance of the *density*-estimate with that of the conventional m-estimate in the long run.

with highest P_+, and, on the other hand, the rule with highest P_-. Then, the classification for that instance is the class with highest probability. Two new rules are generated every time a misclassification occurs adding an attribute-value selected randomly.

We first compare the results obtained from using the original m-estimate, with $m = 0, 2, 4, 8$, and the *density*-estimate, to calculate the probability of a class. Note that, for $m = 0, 2$, we obtain (4) and (5), respectively. Training instances are selected randomly in the input space. After each training iteration, a test episode, consisting in calculating the classification error at every input in the input space, is run. The results present the average of the classification errors of 10 runs for each considered case. We set $c = 1/2$ for all cases. The results show that, when few instances are experienced (figure 2), the performance of the conventional m-estimate seems to improve as the value of m increases. However, this result is reversed as learning proceeds (figure 3) due to the inability of the original m-estimate to compensate the component introduced by the large m. Our proposal, instead, precisely compensates the effect of the lack of experience in the estimation of the probability, producing more confident estimations, and outperforming the original m-estimate at all the stages of the learning process.

To illustrate how the competitive strategy increases the speed of learning, we performed an experiment using only the density estimation and generating 10 rules, instead of 2 rules, every time a misclassification occurs. Figures 4 and 5 present the results for the average of 10 runs at early stages of the learning and in the long run, respectively. Note the improvement in the convergence speed for the case of 10 rules generation, which permits to achieve a classification without errors much faster than in the 2 rules generation case.

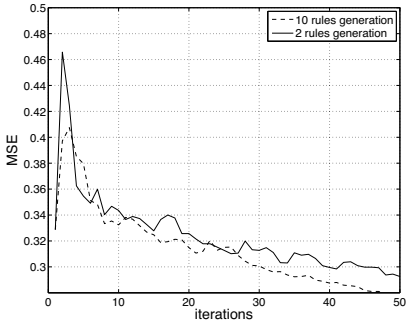

Figure 4: Comparing the performance of the *density*-estimate at early stages of learning for 2 and 10 rules generation.

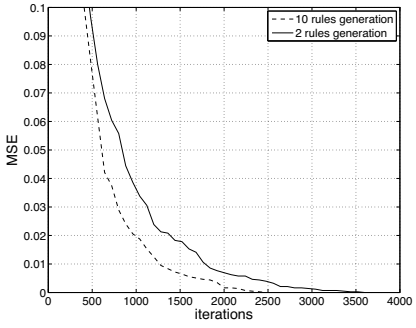

Figure 5: Comparing the performance of the *density*-estimate in the long run for 2 and 10 rules generation.

5 Implementation in Real Robot Platforms

The system has been implemented in two different robot platforms: the humanoid ARMAR III [Asfour *et al.*, 2008] and the Stäubli arm. To show the synergies between the integrated components, we use a task based on the test application of Sokoban [Botea *et al.*, 2003] since it permits a clear visualization of the interesting cases in which these synergies take place, and actions can be easily instructed by a lay person. Given a goal specification, consisting of a target object to be moved and its desired destination, the robot should learn to move the target object to the specified position using vertical or horizontal movements. To achieve the goal, the robot may be forced to move objects blocking the trajectory in an ordered way.

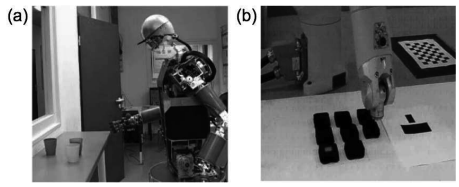

Figure 6: Scenarios. (a) ARMAR III. (b) Stäubli arm.

5.1 ARMAR III Robot

The task in the ARMAR III platform consists in moving the green cup (light grey in figures) on a sideboard where there are other blocking cups, and without colliding with others (figure 6a). The horizontal and vertical movements are performed through pick and place with grasping. Figure 7 presents a simple experiment that permits to illustrate all the cases for learning, where the green cup should be moved to the right but there is a blocking cup. At the time of this experiment, the robot has learned a single PO from a similar experiment, but without a blocking object. The PKS notation for this PO is [Petrick and Bacchus, 2002],

< action name = "TR2" >
  < preconds >
    K(*to*(0)) ^
    K(*e*(R2))
  < \ preconds >
  < effects >
    add(Kf, *to*(R2));
    add(Kf, *e*(0));
  < \ effects >
< \ action >

where "TR2" refers to the action of moving the target object two cells to the right, to is the position of the target object, and e indicates that the referenced cell is empty. "R2" refers to the cell two positions to the right of the initial position of the target object. Note that this PO does not indicate that the cell "R1" in the trajectory to the goal should be empty. In figure 7a, the planner provides PO "TR2" to be executed, but the action is bypassed to avoid a collision. Since no changes occur, the expected effects are not fulfilled and the PO refinement mechanism is triggered (Section 4.2). Then, from the set

$$\mathsf{CEC}_{\mathsf{TR2}} = \{CEC_i | a_i = \mathrm{TR2}, E_i = \{to(\mathrm{R2}), e(0))\}\},$$

the selected CEC_w (2) has $H_w = \{to(0), e(\mathrm{R2}), e(\mathrm{R1})\}$, and it is used to refine the precondition part of the PO, which now includes $e(\mathrm{R1})$. CEC_w has a probability $P_+ = 0.5001$, with number of situations experienced so far in the initial two experiments $n_+ = 1$ and $n_- = 0$, and with number of total situations covered by the cause part C_w, $n_T = 4096$. For the sake of illustration, the sets H of other competing $CECs$ in $\mathsf{CEC}_{\mathsf{TR2}}$ are: $H_j = \{to(0), e(\mathrm{R2})\}$, with $P_+ = 0.5$, $n_+ = 1$, $n_- = 1$, $n_T = 8192$, and $H_k = \{to(0), e(\mathrm{R2}), o(\mathrm{R1})\}$, with $P_+ = 0.4999$, $n_+ = 0$, $n_- = 1$, $n_T = 4096$, where $o(\mathrm{R1})$ indicates that an object is one cell to the right of the target object. After the PO refinement, the planner fails to find a plan since the refined PO is no longer applicable (figure 7b). Then, the teacher instructs to move the blocking cup up, and the learner generates a new PO using the generation mechanism presented in Section 4.1. Finally, in figure 7c, the freed path permits reaching the goal successfully using the refined PO. Figure 8 presents another experiment, performed at later learning stages, in which the green cup should be moved to the right, but there are more blocking cups than in the previous example. In this case, the cup to the right of the target cup cannot be moved up since there is another cup blocking this movement, and neither further to the right since there is not enough space for the hand of the robot to release the cup without knocking over the cup farthest to the right. With the POs learned so far, the robot is able to generate a three-step plan that permits to cope with all these restrictions, moving the cup blocking the target cup first one position to the right, where no cups block its upwards movement, and then up. This frees the path of the target cup which permits fulfilling the goal.

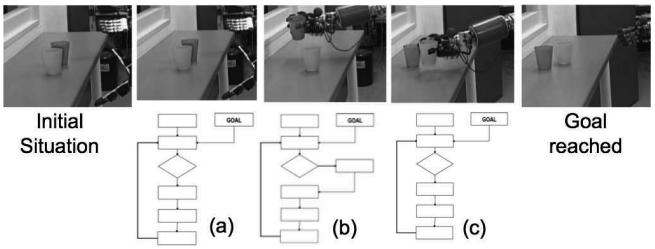

Figure 7: Experiment in which the three cases of learning take place. For further explanations see the text.

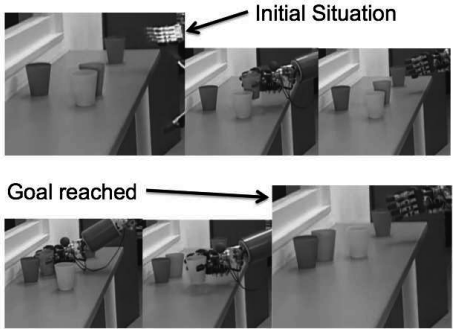

Figure 8: Example of the performance of the system in a more complex situation with many blocking cups.

5.2 Stäubli Arm Robot

The task implemented in the Stäubli arm uses counters instead of cups (figure 6b). The target counter is marked with a red label (light grey label in figures). In this case, we restrict the environment to be a 3 by 3 grid world, where the amount of counters ranges from 1 to 8. Collisions are now allowed. After the robot has learned a large enough set of POs, it is capable of solving difficult situations such as the one presented in figure 9, in which the target counter should be moved from the lower middle position, to the upper right

corner of the grid, starting from the difficult situation where all the cells are occupied except one.

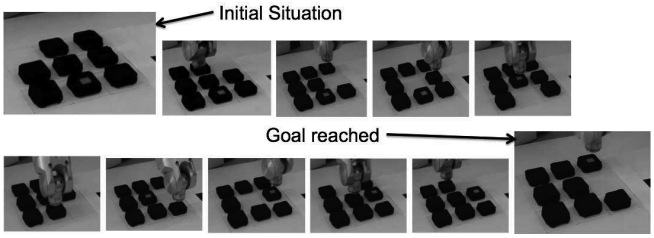

Figure 9: Snapshots that illustrate the sequence of actions executed to move the target counter to the upper right position.

6 Conclusions

In this work, we proposed a system that integrates AI techniques for planning and learning to enhance the capabilities of a real robot in the execution of human-like tasks. The learner enriches the capabilities of the planner by constantly generating and refining planning operators. In turn, the planner widens the capabilities of the robot, since it allows the robot to cope with different tasks in previously inexperienced situations using deliberation.

The system works reliably thanks to the rapid learning of planning operators using a competitive strategy that tries many alternatives of cause-effect explanations in parallel rather than sequentially. The inclusion of a human teacher in the planning-learning loop to support the planner in decision-making permits the robot to generate planning operators that are relevant for the ongoing task, increasing also the speed of learning. The teacher instruction, together with the capability of the learner of generating and refining planning operators at runtime, prevents undesired task interruptions. The AI techniques for planning and learning are integrated with the mechanisms of real robot platforms using a simple decision-making framework. Non-robotic applications can also be handled as long as a set of discrete actions and a set of perceptions, in the form of attribute-values, can be provided to the system. We believe that the proposed system for planning and learning can be used to enhance the performance of other real dynamic systems, such as industrial supply chains.

Acknowledgments

Thanks to Dr. Tamim Asfour, Prof. Rüdiger Dillmann, and their team from the Karlsruhe Institute of Technology for all the support provided in the implementation of the system on the ARMAR platform. This research is partially funded by the EU GARNICS project FP7-247947 and the Generalitat de Catalunya through the Robotics group.

References

[Agostini *et al.*, 2011] A. Agostini, C. Torras, and F. Wörgötter. A General Strategy for Interactive Decision-Making in Robotic Platforms. Technical Report IRI-TR-11-01, Institut de Robòtica i Informàtica Industrial, CSIC-UPC, 2011.

[Asfour *et al.*, 2008] T. Asfour, P. Azad, N. Vahrenkamp, K. Regenstein, A. Bierbaum, K. Welke, J. Schroeder, and R. Dillmann. Toward humanoid manipulation in human-centred environments. *Robotics and Autonomous Systems*, 56(1):54 – 65, 2008.

[Benson, 1995] S. Benson. Inductive learning of reactive action models. In *Proc. of the Twelfth Int. Conf. on Machine Learning*, pages 47–54. Morgan Kaufmann, 1995.

[Botea *et al.*, 2003] A. Botea, M. Müller, and J. Schaeffer. Using abstraction for planning in sokoban. *Computers and Games*, pages 360–375, 2003.

[Cestnik, 1990] B. Cestnik. Estimating probabilities: A crucial task in machine learning. In *Proc. of the Ninth European Conference on Artificial Intelligence*, volume 1990, pages 147–9, 1990.

[Clark and Boswell, 1991] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. of the Fifth European Working Session on Learning*, pages 151–163, 1991.

[Furnkranz and Flach, 2003] J. Furnkranz and P.A. Flach. An analysis of rule evaluation metrics. In *Proc. of the Twentieth Int. Conf. on Machine Learning.*, volume 20, pages 202–209, 2003.

[Gil, 1994] Y. Gil. Learning by experimentation: incremental refinement of incomplete planning domains. In *Proc. of the Eleventh Int. Conf. on Machine Learning*, 1994.

[LaValle, 2006] S.M. LaValle. *Planning algorithms*. Cambridge Univ Pr, 2006.

[Oates and Cohen, 1996] T. Oates and P. Cohen. Learning planning operators with conditional and probabilistic effects. In *Proc. of the AAAI Spring Symposium on Planning with Incomplete Information for Robot Problems*, pages 86–94, 1996.

[Petrick and Bacchus, 2002] R. Petrick and F. Bacchus. A knowledge-based approach to planning with incomplete information and sensing. In *AIPS*, pages 212–221, 2002.

[Shen, 1989] W. Shen. Rule creation and rule learning through environmental exploration. In *Proc. of the Eleventh Int. Joint Conf. on Artificial Intelligence*, pages 675–680. Morgan Kaufmann, 1989.

[Thrun *et al.*, 1991] S. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, D. Fisher, S. Fahlman, et al. The MONK's problems: A Performance Comparison of Different Learning Algorithms. (CMU-CS-91-197), 1991.

[Walsh and Littman, 2008] T.J. Walsh and M.L. Littman. Efficient learning of action schemas and web-service descriptions. In *Proc. of the Twenty-Third AAAI Conf. on Artificial Intelligence*, pages 714–719, 2008.

[Wang, 1996] X. Wang. Planning while learning operators. In *AIPS*, pages 229–236, 1996.