The ROLLENT Planning and Learning System at the IPC-8 Learning Track

Raquel Fuentetaja and **Lukáš Chrpa** and **Tomás de la Rosa** and **Mauro Vallati**
{rfuentet,trosa}@inf.uc3m.es
Universidad Carlos III de Madrid, Spain
{l.chrpa,m.vallati}@hud.ac.uk
University of Huddersfield, United Kingdom

Abstract

This paper describes the ROLLENT system submitted to the Eight International Planning Competition, Learning Track. ROLLENT combines two machine learning techniques: generation of entanglements and decision tree learning by ROLLER. Entanglements capture causal relationships for a class of problems while ROLLER learns relational decision trees useful to sort the applicable operators at a given state. The key idea is to exploit both types of knowledge during planning. Specifically ROLLENT planning phase applies this knowledge to guide a "vanilla" search algorithm based on Depth First Search.

Introduction

ROLLENT has its roots in the ideas of combining general learning techniques for learning specific heuristics, which has a long history in several fields of AI (McCluskey 1987; García-Durán, Fernández, and Borrajo 2007; Balduccini 2011; Petrovic and Epstein 2008). Specifically, the objective of ROLLENT is to maintain the search algorithm simple and understandable and to adapt the behaviour of such algorithm by exploiting different forms of automatically extracted knowledge.

The application of machine learning techniques to planning has been explored mostly by exploiting a single type of knowledge. Usually, it is difficult to integrate several machine learning systems, particularly when they are planner-specific. On the other hand, in the area of machine learning, the combination of multiple learnt models has been well studied (Dietterich 2000). The base idea is to combine several learnt models that are precise and diverse enough to obtain a stronger model that compensates their individual deficiencies. Typically, this refers to predictive models with the same variable to be predicted and does not apply directly to learning systems for planning. However, the idea of obtaining stronger models by the combination of several learning techniques is also attractive in planning, but has been explored rarely.

This paper is structured as follows. First two sections provide an overview of the two learning tecniques integrated in ROLLENT: extraction of entanglements and decision tree learning. Then, it describes how they are combined. Finally, it provides some details of ROLLENT related to the Learning Track of IPC-8.

Entanglements

(Outer) entanglements are relations between planning operators and initial or goal predicates which are used to eliminate unpromissing instances of planning operators. In other words, (outer) entanglements say that to solve a given planning problem some operators can be restricted to be applicable only when some preconditions are in the initial state or some positive effects are target goals. In the well-known BlocksWorld domain, it can be observed that unstacking blocks only occurs from their initial positions. In this case an *entanglement by init* will capture that if an atom `on(a b)` is to be achieved for a corresponding instance of operator `unstack(?x ?y)` (`unstack(a b)`), then the atom is an initial atom. Similarly, it may be observed that stacking blocks only occurs to their goal positions. Then, an *entanglement by goal* will capture that atom `on(b a)` achieved by a corresponding instance of operator `stack(?x ?y)` (`stack(b a)`) is a goal atom. Encoding (outer) entanglements is done by introducing supplementary static predicates having the same arguments as predicates involved in the entanglement relations and instances of these static predicates corresponding to instances of original predicates in the initial or goal state are added to the initial state. For example, if `on(a b)` is in the initial state, then `static-on(a b)` is added to the initial state. These supplementary static predicates are added into preconditions of operators involved in the entanglement relations, so they allow only such operators' instances that follow conditions of entanglements.

Entanglements are extracted from training plans, solutions of simpler problems, in such a way that we check for each operator and related predicates whether the entanglement conditions are satisfied in all the training plans (some error rate might be allowed since training plans may consist of 'flaws' such as redundant actions). Although applying extracted entanglements might cause loss of solvability of some non-training problems, it has been empirically shown that it does not happen (or happens very rarely) if the structure of the problems is similar (only numbers of objects increase). For deeper insights, see (Chrpa and McCluskey

2012).

Roller: Decision Tree Learning for planning

Roller (de la Rosa et al. 2011) is an inductive learning system that learns relational decision trees for planning. These decision trees contain control knowledge useful to sort the successors of a given node for state space search planners. Roller receives as inputs a domain, expressed in PDDL (STRIPS), and a set of training problems. Then, it extracts training instances from the search trees generated to solve training problems. These training instances are used to train TILDE (Blockeel and De Raedt 1998), an off-the-shelf relational classification tool that generate decision trees.

Decision trees are binary trees in which each node represents a query about a feature expressed as a positive predicate logic literal. The main features considered by Roller are: (1) *helpful actions*, whether a particular action is helpful or not at the current state. The notion of helpful actions was first introduced in the FF planner (Hoffmann and Nebel 2001); (2) target pending and achieved goals, whether a target goal is pending or achieved in the current state; and (3) static facts, whether a fact is static, i.e. it is defined at the initial state and any action modifies it. The Roller approach is specific to planners that incorporate a notion of helpful actions, i.e. a heuristic able of dividing the successors of a given node in two sets representing bad and good successors. Roller provides an order of successors that can correct this heuristic by experience.

Roller generates two types of decision trees: operator trees and binding trees. The leafs of operator trees provide an order to sort applicable operators at a given state, depending on the features indicated by the path from the root node to the leaf. Only one operator tree is generated per domain. Binding trees allow to sort the instantiations of each operator. Leafs of binding trees recommend to select or reject an instantiation. For a domain, Roller generates one binding tree per operator. As an example, Figure 1 shows part of the operators tree learned in Depots. The branch in bold states that if there is a `load` helpful action and a `lift` helpful action and there is not an `unload` helpful action for the current state, the operators have to be applied in the order given by the reached leaf: first `lift` with a support of 3 over 6, then `drive`, `drop` and `load` with a support of 1 over 6, and finally unload. Figure 2 shows part of the bindings tree for the operator drive in Depots. Instantiations of drive matching with the branch in bold are recommended to be rejected.

Roller operates in two phases: the learning phase and the planning phase. In the learning phase, the set of decision trees for a domain is generated using a set of training problems. In the planning phase, a state space search algorithm is applied to solve a new problem in that domain. The base search algorithm is Depth-First Search (DFS) with chronological backtracking endowed with the helpful actions heuristic and an strategy to explore first the space generated by helpful actions. When this base algorithm considers Roller decision trees, applicable actions are sorted and re-classified as helpful/non-helpful following the advice they provide. Basically, decision trees are used to compute a priority for each applicable action. Then, these actions are

```
selected(-A,-B,-C)
helpful_load(A,B,-D,-E,-F,-G) ?
+--yes: helpful_lift(A,B,-H,-I,-J,G) ?
| +--yes: nothelpful_unload(A,B,-K,-L,-M,-N) ?
| | +--yes: [lift] 6.0
| |           [[drive:1.0,lift:3.0,drop:1.0,load:1.0,unload:0.0]]
| | +--no: [lift] 12.0
...
```

Figure 1: Partial view of the operators tree in Depots.

```
selected_drive(-A,-B,-C,-D,-E,-F)
helpful_drive(A,B,C,D,E) ?
+--yes: nothelpful_load(A,B,-G,-H,C,D) ?
| +--yes: helpful_unload(A,B,-I,-J,C,D) ?
| | +--yes: [rejected] 12.0 [[selected:0.0,rejected:12.0]]
| | +--no: nothelpful_lift(A,B,-K,-L,-M,E) ?
| |    +--yes: [selected] 20.0 [[selected:18.0,rejected:2.0]]
| |    +--no: helpful_unload(A,B,-N,-O,-P,-Q) ?
...
```

Figure 2: Partial view of the bindings tree for the operator drive in Depots.

sorted by decreasing priorities.[1] The only actions considered now as non-helpful are those with a priority of zero. Roller planning algorithm is the addition of both, the base algorithm and the mechanisms to sort and re-classify actions considering decision trees (originally this algorithm was called Depth-first Helpful Context Policy (DHCP) (de la Rosa et al. 2011)). This is a complete algorithm given that it considers all applicable actions at every node.

The Rollent System

For inductive learning tasks, one of the reasons that justify the use of a combination of several learning techniques, instead of a single one, is representational (Dietterich 2000).

[1]Ties are broken arbitrarily.

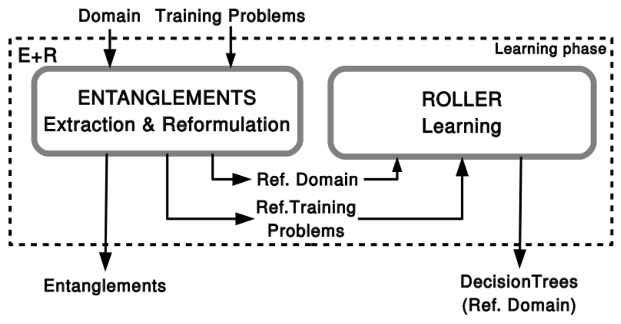

Figure 3: Rollent learning phase.

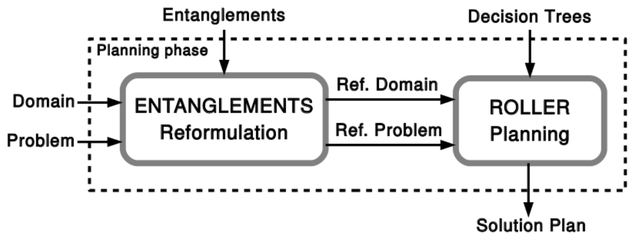

Figure 4: Rollent planning phase.

It could be the case that the true function cannot be represented. This is one of the ways in which a learning algorithm fails. There are also statistical reasons, related with the number of training problems; and computational reasons, related with the use of greedy algorithms to generate the model. However, we focus on the representational issue, since combining entanglements and relational decision trees learnt by Roller improves this aspect specially.

Given the different nature of the knowledge represented by entanglements and Roller, it seems clear that considering both types of knowledge can improve the representational power of the learnt model. This idea is also supported by the fact that previous results for both techniques considered separately showed they are valuable to guide search algorithms for solving planning tasks (Chrpa and McCluskey 2012; de la Rosa et al. 2011).

The ROLLENT system combines entanglements and decision trees in a modular way consisting on apply them in sequence. This is possible because entanglements can be expressed by reformulating the planning domain and problems. Otherwise, the only option would be to implement a new planner considering both knowledge. Figure 3 shows the learning phase. Inputs are a domain and a set of training problems. Entanglements are generated first, producing the reformulated domain and training problems. Then, decision trees are learnt considering the reformulated domain and problems. Thus, entanglements-related knowledge is incorporated naturally in decisions trees since this knowledge expressed by means of domain predicates. These new predicates are considered by the learning algorithm to be included as queries in decision trees. In this sense, the representational power of decision trees increases. Additionally, the search space for the reformulated problem is smaller than the original one which can facilitate the learning task when entanglements are successful.

In the planning phase, shown in Figure 4, inputs are the original domain and a problem. In a first step they are reformulated to include entanglements. Then, the Roller planning algorithm is applied using the decision trees generated in the learning phase.

The planning algorithm can be more aggressive than the individual Roller planning algorithm on the original domain. This algorithm uses decision trees to sort successors and is therefore complete. However, the incorporation of entanglements in the domain has the effect of pruning successors, which results in a reduction of the branching factor. For this reason, planning using the combination can be incomplete.

Competition Details

In this section we describe some specific details of the ROLLENT system submitted to the IPC:

The base planner The base search algorithm is Depth-First Search (DFS) with chronological backtracking endowed with the helpful actions heuristic and an strategy to explore first the space generated by helpful actions. This algorithm could be enough to solve problems in some domains, but we expect it to show very poor performance given the large size of the test distributions at IPC-8 Learning Track.

Training problems As in ROLLER, training problems for generating DCK should be small enough to be explored completely using a BFS + Branch and Bound algorithm. We have used the random problem generators provided by the organizers to generate appropriate training sets for ROLLENT. It has several parameter related to the learning algorithms, for instance, the type of entanglements to use (we used only outer entanglements), the different predicates that can appear as queries in decision trees or the number of minimal cases considered by TILDE to generate a leaf. The value of these parameters has been selected empirically. That is, we have trained ROLLENT (entanglements and decision trees) several times and evaluated its performance in a set of problems of the expected size for the test phase.

Reasoning about costs ROLLENT is a system designed to discover information about the domain and problem structure in terms of predicate logic (ROLLER part) or in terms of exclusivity relations between predicates and operators (entanglements part). The learning techniques used in ROLLENT do not learn knowledge related to action costs. Thus, the first solution found by ROLLENT may present poor quality in domains where the best policy depends on the cost structure of problems (e.g., a graph with ramdom arc costs for the navigation in a city). Since the quality of solutions is the main metric for the competition, we have adopted simple strategy to optimize the quality of the first solution found. Instead of stopping in the first solution, ROLLENT search algorithm continues searching until the time bound, trying to improve the best solution found so far. The search prunes any branch that exceeds the current best cost found.

References

Balduccini, M. 2011. Learning and using domain-specific heuristics in ASP solvers. *AI Communications* 24(2):147–164.

Blockeel, H., and De Raedt, L. 1998. Top-down induction of first-order logical decision trees. *Artificial Intelligence* 101(1-2):285–297.

Chrpa, L., and McCluskey, T. L. 2012. On exploiting structures of classical planning problems: Generalizing entanglements. In *Proceedings of ECAI*, 240–245.

de la Rosa, T.; Jiménez, S.; Fuentetaja, R.; and Borrajo, D. 2011. Scaling up heuristic planning with relational decision trees. *Journal of Artificial Intelligence Research (JAIR)* (40):767–813.

Dietterich, T. G. 2000. Ensemble methods in machine learning. In *Multiple classifier systems, LBCS-1857*, 1–15.

García-Durán, R.; Fernández, F.; and Borrajo, D. 2007. Inductive logic programming. chapter Combining Macro-operators with Control Knowledge, 229–243.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research (JAIR)* 14:253–302.

McCluskey, T. L. 1987. Combining weak learning heuristics in general problem solvers. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, 331–333.

Petrovic, S., and Epstein, S. L. 2008. Random subsets support learning a mixture of heuristics. *International Journal on Artificial Intelligence Tools* 17(03):501–520.