Integrating Macro-Operators and Control-Rules Learning

Rocío G. Durán
Departamento de Informática, Universidad Carlos III de Madrid
Avda. de la Universidad, 30 - 28911 Leganés (Madrid). Spain
rgduran@inf.uc3m.es

Abstract

Nowadays, planning is still a computationally unsolved task and many different learning techniques have been applied in order to improve its capabilities. In this paper we propose the integration of two learning methods sequentially: macro-operators and search control rules. Macro-operators provide us with a sequence of actions that are often executed in a given order. Thus, they avoid to plan that sequence each time it is required. However, the use of macro-operators increases the branching factor of the planning search tree, so the complexity of the planning process grows, and may produce a decrement of the planning performance. Our goal is to learn control rules that let us know when to use the macro-operators. Therefore, the search through the planning tree can be efficiently guided by the control rules. We show that this combination can be successfully applied in classical planning domains.

Introduction

Planning is a process that chooses and organizes a set of given actions by anticipating their expected outcomes. It is a task of Artificial Intelligence considered very complex and computationally hard, in which the search tree reaches a very big size and makes it difficult to find a solution. To reduce the difficulty of finding a solution plan, many solvers employ learning techniques, that acquire macro-operators, heuristics, search control rules, etc, whose results improve noticeably their original behaviour.

In this paper, we propose to use two of these learning techniques sequentially that acquire: macro-operators and search control rules. Firstly, we select the most common macro-operators, composed by two or three simple operators, obtained from the solution plans of a set of random problems. Secondly, we use the search tree of some problems solved with the macro-operators to learn control rules. These rules may include the macro-operators used in the plans previously generated. Finally, we compare the results of both learning techniques together with both techniques individually.

None of these two learning techniques are new, but the sequential use of both provides a novel way of applying the macros in the planning process. This method is much more selective that without the control rules, reducing the number of nodes of the search tree expanded and reducing the planning time.

In the next section we describe the planner and learning modules used in the experiments. The third section describes the method to decide the macro-operators and generate the control rules. The forth section shows some experiments with a version of the Logistics and the Miconic domains of the International Planning Competition. Finally, the last section introduces some conclusions and outlines future work.

The IPSS planner

Nowadays there are very different kinds of planners with different results for each domain. The planner used in this work is the IPSS planner, which provides the two learning modules we need in this work: macro-operators and control rules learning modules. IPSS is an integrated tool for planning and scheduling (Rodríguez-Moreno *et al.* 2004), which is based on PRODIGY (Veloso *et al.* 1995) as the planner component. PRODIGY is a nonlinear planning algorithm and it has been used for studying several machine learning techniques in the context of planning.

IPSS planner inputs are the domain and problem descriptions, generating as output a total-ordered plan, and the planning search tree. They can be used to learn macro-operators and seach control rules respectively, as explained next.

Macro-operators learning

A macro-operator is an operator composed by several simpler operators. It produces the same result than executing the simple operators sequentially. Their principal drawback is the utility problem (Minton 1988; McCluskey & Porteous 1997). The addition of macro-operators increases the branching factor and the processing cost per node, which can mean that they have worse search performance than not using them. Some other effects of using the macro-operators can be disadvantageous too: change of the order in which the search space is traversed (they change the order in which the primitive operators are used for obtaining a solution), change of the path costs, and increase of redundancy.

However, they can show significant improvement in different domains (Botea, Mueller, & Schaeffer 2005), by including into the macro-operators a partial ordering of its simple operators or combining the use of macro-operators

with techniques such as the relaxed graphplan computation implemented in FF. Therefore, a key issue consists on finding the good macro-operators, which can find faster a better plan.

In this work, we have selected the macro-operators using the frequency of appearance of several simple operators sequentially together in a set of obtained solution plans. IPSS provides a module to obtain a macro-operator from solving a problem in a given domain. It is also possible to select one operator subsequence from the solution plan to obtain a smaller macro-operator.

The HAMLET learning module

HAMLET is an incremental learning method based on EBL (Explanation Based Learning) and inductive refinement of control rules (Borrajo & Veloso 1997). The inputs of HAMLET are a domain, a set of training problems, and other learning-related parameters. HAMLET calls IPSS and receives as input the search tree expanded by the planner, in order to decide where and what to learn. HAMLET output is a set of control-rules that potentially guide the planner towards good quality solutions. In the context of this work, we use HAMLET to find a set of control rules that are able to learn when to use the acquired macro-operators.

Integration of macro-operators and control-rules

In this work, we have used both learning techniques together, with the aim of generating control rules that define when a specific macro-operator shall be used. To show the effectiveness of this approach, we show the results of using the two techniques separately and together: control rules in the original domain, macro-operators in the original domain and control rules after the macro-operators are acquired.

The first step is to select some macro-operators composed by two and three simple operators. We provide IPSS a set of random training problems to be solved. From the resulting total-ordered plans, all the different combinations of two and three operators have been obtained that appear one after the other and have, at least, one constant in common. The most common of them are selected for the second step.

The next step is, for each macro-operator, to insert them separately into the given domain and let the system learn control rules, using always the same training set of random problems. Learning control rules using the original domain (without macro-operators) is also executed, in order to compare the results.

Finally, the same test set is used for each resulting domain: (i) the original domain, (ii) the domains with each selected macro-operator, (iii) the original domain with its own learned control rules and (iv) the macro-operators and the control rules together. The main objective of this approach is to obtain good control-rules for each macro-operator and so, better results with this combination than using both techniques separately.

Experiments

This section describes the experiments performed in both the Logistics and Miconic domains. For each experiment, the learning parameters are the default ones. The only parameter modified is the time limit given to learn from the training problems and to solve the test problems. For both domains and in both cases, this value is always 30 seconds.

Logistics domain

We use the version of the Logistics domain, as it was first defined (Veloso 1994). The difference with the version created for the first IPC is that the predicates for describing where packages, trucks and airplanes are, have changed to at-object, at-truck and at-airplane.

We have used a random problem generator to create different problem sets for learning and test. These sets are the following:

- Macro-operator learning set (to obtain the most common macro-operator composed by 2 and 3 simple operators): 30 random problems with 3 cities, 3 objects and a maximum of 3 goals.
- Control-rules learning set: 30 random problems with 3 cities, 3 objects and a maximum of 3 goals.
- Test set 1: 30 random problems with 7 cities, 10 objects and from 1 to 10 goals.
- Test set 2: 40 random problems. 10 of them are of type (3, 5, 5), other 10 are (5, 10, 10), the next 10 problemas are (8, 15, 15) and the last 10 are (10, 20, 20), where (c, o, g) refers to number or cities (c), number of objects (o) and number of goals (g) respectively.

The characteristics of the sets are different because their different use. For instance, the problems generated for learning control-rules and macro-operators are “simple” problems (with small number of cities, goals and objects) to ensure that the planner is able to: (i) find solutions from which to generate macro-operators; and (ii) expand the whole search tree to obtain control rules. Test sets are also different. We have first created a test set with easy problems (from 1 to 10 goals), and a more complex set that contains problems with up to 10 cities, 20 objects and 20 goals.

We have learned several macro-operators of different types, following the approach introduced in the second section. The complete list is enumerated next, where we describe the operators that compose the macro-operator.

1. Macro m2-1: drive-truck unload-truck
2. Macro m2-2: fly-airplane unload-airplane
3. Macro m2-3: load-truck drive-truck
4. Macro m3-1: load-truck drive-truck unload-truck
5. Macro m3-2: drive-truck load-truck drive-truck
6. Macro m3-3: load-airplane fly-airplane unload-airplane

Tables 1 and 2 show the results of solving the problems of both test files respectively. They present percentages of solved problem (*Solved*) and number of used rules (*Rules*).

The IPSS column shows the results obtained with IPSS without control rules. The HAMLET column shows the results obtained by IPSS when using the learned control rules. The different rows describe the results obtained when different macro-operators are used. In the first row, no macro-operator is used. In the second one, the m2-1 macro is used, and so on.

Domain	IPSS	HAMLET	
	Solved	**Solved**	**Rules**
Logistics	20%	27%	9
Logistics + m2-1	13%	20%	4
Logistics + m2-2	43%	63%	8
Logistics + m2-3	8%	8%	6
Logistics + m3-1	13%	20%	4
Logistics + m3-2	13%	13%	9
Logistics + m3-3	23%	76%	9

Table 1: Percentage of solved problems of the test set 1.

The results obtained for the first test set are very satisfactory in two cases: m2-2 (fly-airplane+unload-airplane) and m3-3 (load-airplane+fly-airplane+unload-airplane). For both macros, the results are good with and without control rules. In the first case, the percentage of solved problems is 43%, more than double when compared with IPSS alone, that obtains a 20%. If we learn the control rules for that macro, the percentage increases up to 63%. When using control rules and the m3-3 macro-operator, this percentage increases up to 76% of solved problems. However, the table shows that the results depend on the macro-operator used and, for instance, when using the macros m2-1, m2-3, m3-1 and m3-2 without control rules, the performance is lower (13%) than when the macro-operator is not used (20%). In two of these cases (m2-1 and m3-1), their results using HAMLET improve the results of IPSS alone and, oddly, equal the results of IPSS in the original domain. In the other two cases (m2-3 and m3-2) their results using HAMLET are the same than the results of IPSS.

The results with the second test set provide a similar reading as described in Table 2. For macros m2-2 and m3-3 the performance raises from a 2% of problems solved up to 15% and 58% respectively. Thus, the macros that were useful in the previous test set are useful in this one too. IPSS default uses trucks before airplanes to load and unload objects in a location. With these airplane macro-operators it changes the preference and it seems to learn control rules that decide when to use the airplane macro-operators (m2-2 and m3-3). With the macros m2-1, m2-3 and m3-1, IPSS is not able to solve any problem, nor with HAMLET. Finally, the macro m3-2 keeps its results with IPSS equal than with HAMLET, only 3% solved problems.

Miconic domain

The version of this domain is the one used in the IPC-2000, as well as the 150 used problems. In this domain there are two types of objects: passengers and floors. The goal is to bring people using an elevator to different floors. We used the 10 most simple problems of the 150 problems of the competition to learn the control rules in HAMLET and the rest 140 to test. The first group are problems with two and four floors, while the second set has problems with from six up to sixty floors. The learned macro-operators in this domain are:

Domain	IPSS	HAMLET	
	Solved	**Solved**	**Rules**
Logistics	2%	0%	9
Logistics + m2-1	0%	0%	4
Logistics + m2-2	13%	15%	8
Logistics + m2-3	0%	0%	6
Logistics + m3-1	0%	0%	4
Logistics + m3-2	3%	3%	9
Logistics + m3-3	10%	58%	9

Table 2: Percentage of solved problems of the test set 2.

1. Macro m2-1: up board
2. Macro m2-2: board down
3. Macro m2-3: down depart
4. Macro m3-1: board down depart
5. Macro m3-2: up board down
6. Macro m3-3: board up depart

Table 3 shows the results of solving the test problems in the same format that in the previous tables.

Domain	IPSS	HAMLET	
	Solved	**Solved**	**Rules**
Miconic	3%	11%	3
Mic+m2-1	22%	17%	4
Mic+m2-2	12%	26%	3
Mic+m2-3	51%	52%	3
Mic+m3-1	17%	28%	3
Mic+m3-2	27%	34%	3
Mic+m3-3	51%	53%	3

Table 3: Percentage of solved problems of the test set.

Every macro-operator configuration has better results than the original domain, even with control-rules in HAMLET. So, except for the first macro-operator (up+board), the results with both techniques together improve over using only one of them or not using them. After analysing the solution plans from the Miconic domain using the macro-operators, these solutions are not semantically correct. In Figure 1, we can see the obtained solution plan using the macro-operator up+board for one simple problem. The first thing we can observe is the unnecessary use of operators. The second and third actions, for example, could be better replaced by down and board, instead of down and up+board. But the real problem is the fact of repeating the action **up-board f0 f2 p1** after boarding already the passenger p1 into the lift and, even **up-board f0 f1 p0** after serving the passenger p0 in the floor **f2**.

The reason of this behaviour is the definition of the board simple operator, which does not delete the predicate **origin**

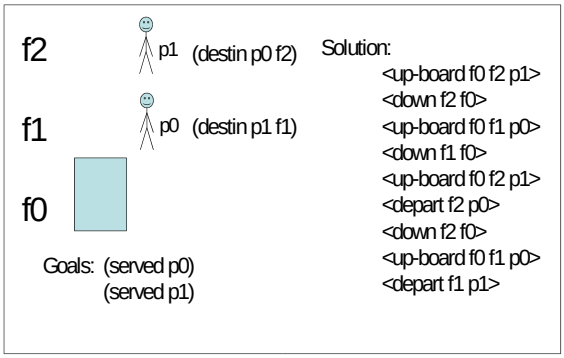

Figure 1: Example with macro-operator up+board.

p f. That means that the planner can board a passenger as many times as it needs, because no operator deletes the origin predicate. So, for example, if we have to go up to the second floor to leave passenger p0, the planner is going to select first the macro-operator up+board, which will try to move the lift up to the second floor, and board someone into the lift: passenger p1, who has there the origin. Nothing in this domain avoids this problem and the solution plan is incorrect.

In order to solve this problem, we added a new predicate: **(at-passenger p f)**, to know exactly where each passenger is and to avoid boarding them into the lift many times. This includes changing the definition of the Miconic domain and generating the correct definition of the macro-operators again. Finally, the new results for the Miconic domain are given in the Table 4.

Domain	IPSS	HAMLET	
	Solved	**Solved**	**Rules**
Miconic	3%	10%	2
Miconic + m2-1	16%	15%	4
Miconic + m2-2	1%	1%	3
Miconic + m2-3	11%	10%	3
Miconic + m3-1	4%	7%	3
Miconic + m3-2	1%	4%	3
Miconic + m3-3	15%	16%	3

Table 4: Percentage of solved problems of the test set.

Now, the results are not as good as before, but the plans are valid this time. Only with macro m3-3 we obtain better results with both learning techniques together than both techniques alone.

Conclusions and future work

In this paper, we have shown that the combination of macro-operators and control rules in the Logistics and Miconic domains can improve the results of the IPSS planner alone. We demonstrate that different macro-operators can be learned, and that their use does not always outperform the results of IPSS alone. However, when learning control rules to guide the search, the results over using the macro-operator alone improve.

We show, however, that there are some risks on the application of macro-operators: the learned macro-operators may solve no problem. Thus, to define which kind of macro-operators is good for this integration and which training problems are good to obtain the right rules, are two of the future research lines. That can include a new method to find good macro-operators.

Also, there are many domains in which this integration must be tested and we have to increase even more the number of simple operators that compose the used macro-operators.

A side effect of learning control rules on planning domains with macro-operators has also been finding extra knowledge about macro-operators: after acquiring macro-operators, we have seen a bug in the Miconic domain description that would be difficult to detect without using them, given when not using them IPSS would always generate valid plans.

Acknowledgements

This work has been partially supported by the Spanish MCyT project TIC2002-04146-C05-05, MEC project TIN2005-08945-C06-05 and regional CAM-UC3M project UC3M-INF-05-016.

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. Also in the book "Lazy Learning", David Aha (ed.), Kluwer Academic Publishers, May 1997, ISBN 0-7923-4584-3.

Botea, A.; Mueller, M.; and Schaeffer, J. 2005. Learning partial-order macros from solutions. In *Proceedings of ICAPS'05*.

McCluskey, T. L., and Porteous, J. M. 1997. Engineering and compiling planning domain models to promote validity and efficiency. *Artificial Intelligence* 95(1):1–65.

Minton, S. 1988. *Learning Effective Search Control Knowledge: An Explanation-Based Approach*. Boston, MA: Kluwer Academic Publishers.

Rodríguez-Moreno, M. D.; Oddi, A.; Borrajo, D.; Cesta, A.; and Meziat, D. 2004. IPSS: A hybrid reasoner for planning and scheduling. In de Mántaras, R. L., and Saitta, L., eds., *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, 1065–1066. Valencia (Spain): IOS Press.

Veloso, M.; Carbonell, J.; Pérez, A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical AI* 7:81–120.

Veloso, M. 1994. *Planning and Learning by Analogical Reasoning*. Springer Verlag.