Variable Resolution Planning through abstraction

Moisés Martínez
Universidad Carlos III de Madrid
Avenida de la Universidad, 30
28911 Leganés, Spain
moises.martinez@uc3m.es

Abstract

Use Automated planning in real time environments is the main aim of the doctoral research behind this dissertation abstract. Planning a sequence of actions is particularly difficult in stochastic and dynamic environments, where actions execution might have unexpected results or new information can be discover. Also, in many domains, agents cannot wait a long time for plan generation. In these cases, computational cost makes automated planning not appropriate. My first work is based on the fact that in stochastic or dynamic domain, it is usually not worth computing a valid (sound) long plan, since a part of it may not be used. A plan can be generated where the first k actions are applicable if the environment does not change, while the rest of the plan might not be necessarily applicable, since it has been generated using an abstraction by removing some predicates.

Introduction

Real world environments are challenging for Automated Planning, where a planner has to generate a plan of actions, and the plan execution may yield unexpected states. For instance, a robot may not know the real traversability of a terrain or the precision of the actuators, a logistic system may not be able to predict future traffic congestions or even be sure about present traffic conditions when defining a route for different trucks. In these situations a planner tries to define a solution using the information about the environment in a given state. Thus, generated plans cannot always be executed completely. For these reasons, Automated Planning can be prohibitively expensive for most real world scenarios.

There are different ways to approach planning and execution in these kinds of environments. Different techniques can be applied depending of the information known about the environment. If we have information about the dynamics of the environment (failure in the actuators of the robots, the structure of the terrain, accuracy of sensors, etc) can build a non-linear conditional plan (Peot and Smith 1992) where the plan takes into account all possible situations that may occur. Another alternative is to generate a set of policies by a Markov Decision Process (MDP) as shown by the LAO* (Hansen and Zilberstein 2001), LRTDP (Bonet and Geffner 2004) and BLAO* (Hansen and Zilberstein 2001) algorithms. If the dynamics of the environment are not known, we could first learn it. However, usually the learning effort is impractical in small problems (Zettlemoyer, Pasula, and Kaelbling 2005).

A common solution is using a repair or re-planning system (Fox et al. 2006; Yoon, Fern, and Givan 2007). In re-planning, the planner generates an initial applicable plan and executes it, one action at a time. If a failure is detected, the system generates a new plan. This process is repeated until the system reaches the problem goals. Therefore, at each planning (re-planning) step, including the initial one, the system is devoting a huge computational effort on computing a valid plan (applicable plan that achieves the goals), when most of it will not be applied. On the other extreme, reactive systems greedily select the next action to be applied according to some configured - or learned - knowledge. They are "mostly" blind with respect to the future; they usually ignore the impact of the selected action on the next actions and states. Thus, they often get trapped in local minima, or dead-ends.

We advocate here for a middle-ground approach where planners try to decrease planning time and avoid dead-ends. On one hand we propose a technique where a planner devotes time to compute a valid first portion of the plan, and checks that there is a potential continuation of the plan, trying to avoid dead-ends, while the rest of the plan has been generated using an abstraction by removing some predicates.

This dissertation abstract is organized as follows: first in Section 2, we introduce the first technique design during my doctoral research. In Section 3 we define our abstraction mechanism. In Section 4 we describe how to implement Variable Resolution Planning (VPR). In Section 5 we show an experimental evaluation of our techniques over different domains. Finally, in Section 6 we conclude and introduce future work

Relaxing information about the environment

My first work is based on the idea that spending a big effort to compute a valid plan in real world environment may

not be useful. This is because actions execution might fail or new information can be discovered during planning or execution. This work is inspired by the work of Zickler and Veloso (Zickler and Veloso 2010), in which a motion planning technique is presented. It generates a collision-free trajectory from an initial state to a goal state in dynamic environments. They introduced the concept of Variable Level-Of-Detail (VLOD), which focuses search on obtaining accurate short-term motion planning, while considering the far future with a different level of detail by selectively ignoring the physical interactions with dynamic objects. This technique decreases the computational cost of the motion planning process, so that information about different elements of the environment is not used to search a path to reach all goals. I introduce Variable Resolution Planning (VRP) through predicate relaxation, a new approach to reduce the computational overhead of Automated Planning in dynamic and stochastic environments.

Abstractions

An abstraction can be defined as a function that transforms a planning problem into another (simpler) one. Usually, the goal is to reduce the problem complexity. There have been already many approaches that generate abstractions in Automated Planned. The rst work in abstractions was presented by Sacerdoti (Sacerdoti 1972) in 1972. He extended the work of Newell and Simon on GPS (Newell and Simon 1969) and used it to develop Abstrips, which combined abstraction with Strips. In recent years, abstractions have been used to generate heuristic techniques. In recent years, abstractions have been used to generate heuristic techniques. Fast-Forward (Hoffmann and Nebel 2001) used abstractions to compute the heuristic value of nodes by building a relaxed plan to guide the search process, where the delete effects of actions are ignored (Hoffmann and Nebel 2001). Another use of abstractions to build heuristics are pattern databases (PDBs), which have been used in Automated Planning (Edelkamp 2001). More recently, a new form of abstraction, in which DTGs are generated together (to merge them) and then shrunk by abstracting nodes that share the same relaxed distance to the goal (Haslum et al. 2007).

But in this work, we generate abstractions by removing some predicates from the lifted representation of the domain. Thus, since planners work at the propositional (instantiated) level, it is equivalent to remove literals from the preconditions and effects of actions in the grounded representation.

First it defines a mapping between a predicate in the PDDL domain definition and its corresponding groundings for problem P.

Definition 1 *$g(p, P)$ is the set of propositions (facts) of predicate p in problem P.*

For instance, in the Rover domain, if p is `at(?w,?r)` and a PDDL problem P_1 defines one rover R three waypoints (A, B, and C), among which can move, then:

$$g(p, P_1) = \{\texttt{at(R,A)},\ \texttt{at(R,B)},\ \texttt{at(R,C)}\}$$

For this approach, an abstraction of an action a over a predicate p removes all propositions that are groundings of predicate p – $g(p, P)$ from the preconditions and effects of action a. In order to select which predicates to remove, we split the set of facts F into three subsets: $F_d \subseteq F$ is the set of dynamic facts, or facts that appear in the effects of at least one action ($F_d = \{f \in F \mid \exists a_i \in A, f \in \mathit{Eff}(a_i)\}$). $F_s \subseteq F$ is the set of static facts, or facts that do not appear in the effects of any action; and $F_f \subseteq F$ is the functions set, or non-boolean facts that are groundings of PDDL functions instead of PDDL predicates. We are not going to remove all domain predicates (or their corresponding facts), so we define the candidate subset of facts to be potentially removed, $C^{abs} \subseteq F$. We will not use all predicates to generate abstractions. First, static predicates are not added or deleted during the planning process. Therefore, applying an abstraction over a static predicate does not reduce the number of actions to achieve the goals. Deleting a static predicate only increases the number of actions that may be selected during the search process to reach the goal state. Second, predicates which are part of the goal subset cannot be removed, because the planner would not reach a solution. For instance, in Rover domain, if p is `communicated_rock_data(?p)` and G=`communicated_rock_data(W1)`, `communicated_rock_data(W5)`} if predicate p is removed, it will not be to reach a state where goals are true and the problem cannot be solved.

C^{abs} is composed of all facts except for the static facts (F_s), the functions facts (F_f) and facts that are part of the problem goals (G).

$$C^{abs} = F \setminus (F_s \cup G \cup F_f)$$

Next, we include some definitions of abstraction by one predicate:

Definition 2 *An abstraction of an instantiated action $a \in A$ over a predicate p in problem P is defined by function $f(a, p) = (Pre_p^{abs}(a), Add_p^{abs}(a), Del_p^{abs}(a))$, where:*

- $Pre_p^{abs}(a) = Pre(a) \setminus g(p, P)$
- $Add_p^{abs}(a) = Add(a) \setminus g(p, P)$
- $Del_p^{abs}(a) = Del(a) \setminus g(p, P)$

Definition 3 *An abstraction of a PDDL problem P over a predicate p, called P_p^{abs} is the result of abstracting all components of P over p:*

$$P_p^{abs} = (F_p^{abs}, A_p^{abs}, I_p^{abs}, G_p^{abs})$$

where:

- $F_p^{abs} = F \setminus \cup_p g(p, P)$
- $A_p^{abs} = \{a^{abs} \mid a \in A, a^{abs} = f(a, p)$
- $I_p^{abs} = I \setminus \cup_p g(p, P)$
- $G_p^{abs} = G$, *given that the predicates in the goals are not candidates to abstract*

Variable Resolution Planning

The goal of VRP in stochastic domains is to quickly come up with a plan whose first k actions are sound and whose rest of actions could potentially solve the problem by adding the removed details. Thus, we will generate a search tree where nodes of depth less or equal to k use the original definition of the problem, and nodes below that depth will use an abstracted version of the problem. In order to use current planners under this approach, modifications in the planners code are necessary. In this work, Metric-FF (Hoffmann 2003) has been used as a base to implement our approach. We call the new planner Abstract-MFF. Apart from the standard inputs of Metric-FF, Abstract-MFF also receives k and L as input. Once the initial instantiation is performed (computing the instantiated problem and all its components, $P = (F, A, I, G)$), Abstract-MFF computes the abstracted problem P^{abs} with its corresponding data structures.

During search, when the depth of a node is greater than the Temporal Horizon, k, the abstract space, P^{abs}, is activated for all the nodes in the subtree of that node while the original space, P is deactivated. The implementation basically changes the data structures corresponding to the original problem P for the ones corresponding to the abstracted problem P^{abs}. So, the nodes above depth k use P and the ones below that threshold use P^{abs}.

For instance, Figure 1 shows an example of the generation of a regular plan in the original planning space versus an abstract plan in the abstract space. Suppose that we have a planning problem P_3 defined as:

$$P_3 = (\langle a(x), b(x), c(x), d(x), e(x), f(x)\rangle, \langle t_1, t_2, t_3, t_4\rangle, \langle a(1)\rangle, \langle e(1), g(1)\rangle)$$

and we apply an abstraction over predicate d with function $f(P_3, d)$. It would generate the abstracted problem:

$$P_{abs} = (\langle a(x), b(x), c(x), e(x), f(x)\rangle, \langle t'_1, t'_2, t'_4\rangle, \langle a(1)\rangle, \langle e(1), g(1)\rangle)$$

where $t'_1 = t_1$, $t'_2 = t_2$, $t'_3 = f(t_3, d)$ and $t'_4 = f(t_4, d)$. Actions t_3 and t_4 include predicate d in their structure and function $f(t_i, p)$ can be applied to generate a new abstract action. If function f is applied over action t_3, its effects become empty. Thus, t_3 can be removed from the abstracted problem.

Preliminary Results

To evaluate the approach presented in this dissertation abstract, we have performed a planning-execution-replanning loop on some domains and problems.

We report the total planning time, as well as the mean of re-planning steps. We have selected some benchmark domains: Depots, DriverLog, Satellite and Rovers domains from IPC-3. For each benchmark two problems are solved, using abstractions over two different predicates. Table 1 presents the results. These results indicate that this approach has good result in easy and difficult problems. For instance, in the Depots domain it has good performance in easy problems (21 and 22); it needs less planning time if it abstracts predicates *in* and *available*. On the other hand, in the Satellite domain results present a similar performance to the Rovers domain, where we obtain good results in hard problems. But these abstractions do not offer good results in all scenarios. For instance, removing the predicate *Link* in DriverLog domain, increases the cost to solve the problem, because this predicate defines the route among different locations. When is deleted, it is possible to move to any location from every location increasing the cost to reach the goals. Finally, Rovers domain confirms our initial hypothesis: in problems where planning time is high, abstractions significantly reduce planning time. But when the predicate *at* is removed, results are worse. Our initial assumption was that the predicates like at in domains where a robot or a vehicle has to move through different positions would offer better performance than the rest of predicates. But, in the Rovers domain this is not the case. Thus, the decision on which predicates to remove is an important one, in order to obtain a good performance.

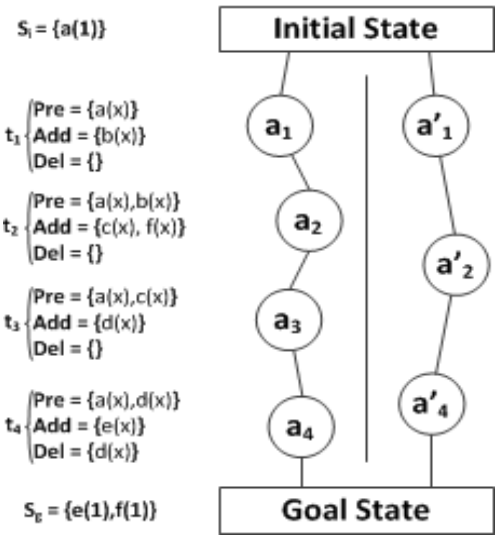

Figure 1: Valid regular plan in problem space P versus abstract plan in abstracted space P_{abs}.

Conclusions

In this dissertation abstract, Abstract-MFF has been presented as first work of my doctoral research, a planner based on Metric-FF that uses an abstraction mechanism that dynamically removes some predicates during the planning process based on a temporal horizon, in order to improve planning performance in stochastic environments. The main contribution of this work consists on understanding the effects of this idea over the planning process, and how it can be used to improve the application of planning techniques in real environments. As further work, I would like to explore which combinations of predicates are useful to be removed, automatic ways of selecting those, and changing the temporal horizon dynamically. Also I am interested on testing the performance in dynamic environments, where new information can be discovered changing the complexity of the planning problem or even in some cases needed to solve the prob-

Domain	Problem	Predicate	Metric-FF	AMFF (k=3)	AMFF (k=5)	AMFF (k=10)	AMFF (k=20)
			Time	Time	Time	Time	Time
Depots	21	In	**0.90**	1.20	1.85	2.15	1.60
Depots	21	Available	**0.90**	1.31	1.78	2.21	1.92
Depots	22	In	220.10	45.30	**42.10**	108.21	131.43
Depots	22	Available	220.10	**85.25**	86.25	134.92	149.12
Satellite	27	Calibrate	12854.32	4828.85	4320.78	3745.83	**3397.15**
Satellite	27	Power On	12854.32	3951.60	2769.55	**3289.32**	3653.10
Satellite	29	Calibrate	16542.91	**4467.32**	4582.43	4832.12	5104.34
Satellite	29	Power On	16542.91	3934.42	**3582.43**	4232.12	4604.34
Driver Log	14	Empty	10.98	**4.67**	5.32	6.32	5.12
Driver Log	14	Link	10.98	8.73	9.21	**8.19**	9.42
Driver Log	19	Empty	1828.52	**460.80**	878.34	793.29	620.31
Driver Log	19	Link	**1828.52**	8720.43	8892.82	9621.64	9053.53
Rover	33	Take rock	528.91	260.42	**153.42**	177.95	170.24
Rover	33	At	528.91	578.24	502.94	**382.24**	381.27
Rover	35	Take rock	9657.32	1880.21	**1564.64**	1564.34	1796.75
Rover	35	At	9657.32	8865.54	7954.21	**5456.24**	6024.25

Table 1: Planning time for different domains with a 30% probability of failure. The rst column corresponds to domain. Second column with one problem of domain, the third column corresponds to predicate removed. Four column corresponds to Metric-FF and the rest corresponds to Abstract-MFF with different values of k. For each planner, each column shows the average of the sum of planning times of each run i seconds. In bold, we highlight the best results per row.

lem. Finally, it would be interesting to select a set of problems that cannot be solved by current planners like Metric-FF (Hoffmann 2003) or LAMA (Richter and Westphal 2008) and analyze if our technique can solve these problems applying different levels of abstraction. Also, we would like to explore the effects of using parallelization in Abstract-MFF. Planning time could be even less by using this technique (Martinez 2011).

Acknowledgements

I extend my thanks to supervisors Daniel Borrajo and Fernando Fernández for their helpful recommendations. This research is partially supported by the Spanish MICINN projects TIN2008-06701-C03-03, TRA-2009-008 and Comunidad de Madrid - UC3M(CCG10-UC3M/TIC-5597).

References

Bonet, B., and Geffner, H. 2004. A probabilistic planner based on heuristic search. In *In Proceedings of Fourth International Planning Competition*.

Edelkamp, S. 2001. Planning with pattern databases. *In Proceeding 6th European Conference on Planning* 13–24.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling*, 212–221.

Hansen, E. A., and Zilberstein, S. 2001. Lao* : A heuristic search algorithm that nds solutions with loops. *Articial Intelligence* 129(1-2):35–62.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. *In Proceeding of the 22nd Conference on Artificial Intelligence* 1007–1012.

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

Hoffmann, J. 2003. The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research* 20:291–341.

Martinez, M. 2011. Parallel heuristic search fast-forward. In *In Proceedings of the Seventh International Planning Competition*.

Newell, A., and Simon, H. A. 1969. Gps: A case study in generality and problem solving. *Artificial Intelligence* 2(5):109–124.

Peot, M. A., and Smith, D. E. 1992. Conditional nonlinear planning. In Kaufmann, M., ed., *In Proceedings of First International Conference on Artificial Intelligence*, 189197.

Richter, S., and Westphal, M. 2008. The lama planner using landmark counting in heuristic search. In *Proceedings of the International Planning Competition*.

Sacerdoti, E. D. 1972. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence* 2(5):115–135.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. Ff-replan: A baseline for probabilistic planning. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling*.

Zettlemoyer, L. S.; Pasula, H.; and Kaelbling, L. P. 2005. Learning planning rules in noisy stochastic worlds. In *AAAI*, 911–918.

Zickler, S., and Veloso, M. 2010. Variable level-of-detail motion planning in environments with poorly predictable

bodies. *In Proceeding of the 19th European Conference on Artificial Intelligence*.