On the Inference of Intermediate Goals in Automated Planning

Vidal Alcázar
Departamento de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad, 30. Leganés (Madrid). Spain
valcazar@inf.uc3m.es

Abstract

Using intermediate goals to guide the search or decompose a problem into smaller instances has proved to be a successful approach in Automated Planning. Goal subsets and more recently landmarks have been used for this purpose, potentially reducing the search in an exponential way. However, these approaches have limitations on their own. In this paper, we propose alternative ways of obtaining possible intermediate goals. In particular, we analyze how using information extracted from the last actions in the relaxed plan can be used to generate intermediate goals backwards. Besides, we propose other lines of research that aim to accomplish this, overcoming some of the limitations that goal subsets and landmarks have.

Introduction

The use of intermediate goals is one of the most promising techniques in Automated Planning. Both providing the search with additional information and reducing the depth of the search space are benefits of using intermediate goals. Many state-of-the-art planners are based on this premise (Hoffmann, Porteous, and Sebastia 2004; Burfoot, Pineau, and Dudek 2006; Richter, Helmert, and Westphal 2008). Those planners depend on goal subsets and landmarks used as intermediate goals to partition the problem or to develop new heuristics. However, the difficulty of guessing the right orderings and the assumption of independence between goals/landmarks make their usage non-trivial. To overcome this, we will consider different methods to obtain intermediate goals. This paper describes the first steps of one of the approaches we are developing and proposes in the last section an additional line of research to be followed during the elaboration of the thesis.

Most state-of-the-art planners are forward state heuristic planners that use a reachability analysis to compute their heuristic function. The most common one is the Relaxed Plan (RP) heuristic used by FF (Hoffmann 2001). Apart from the heuristic numeric value obtained, the actions that compose the RP offer additional information that can be exploited in the search process. Examples are helpful actions (Hoffmann 2001), look-ahead states (Vidal 2003) and macro-actions (Botea, Müller, and Schaeffer 2007) that can be inferred online from the RP. Here we propose taking advantage of information from the last actions of the RP to generate intermediate goals.

The motivation behind this approach is that the last actions in the RP are often similar across different calls to the heuristic function. By doing regression using those actions, intermediate goals can be generated online. In subsequent computations of the heuristic, the closest intermediate goal to the current state in the relaxed problem can be detected and the search biased towards it. We have called this technique the Backwards Generated Goals (BGG) heuristic.

Those goals provide additional information that can be used to reduce the number of levels used in the reachability heuristic or to get heuristic estimations closer to the real value than other heuristics. Besides, this technique allows to finish the search earlier when satisfying an intermediate goal, as the path to the original goal can be built by tracing back the generation of the intermediate goal. This relies on a careful generation of extra information together with the intermediate goal.

To ensure the validity of the backwards generated goals, actions must legally support the reached goal. This is done by using concepts from regression heuristic planners like HSPr (Haslum and Geffner 2000): considering delete effects when supporting goals and taking into account static mutexes between the preconditions of the actions and the goal propositions not satisfied by the supporting actions. Experimental evaluation of the techniques presented in this paper shows improvements in coverage and number of expanded nodes over the regular RP heuristic.

Intermediate Goals and Reachability Heuristics

Reachability heuristics are computed using a breadth-first search in a relaxed version of the problem. The relaxation consists on ignoring the delete effectes of the operators. Once the goals are reached, a non-optimal relaxed plan is extracted. Due to the breadth-first search nature of the process, it not only gives an estimation of the distance to a goal, but also estimates that the first reached goal is the closest to the state in the relaxed problem in case multiple goal states are present. In most domains the set of goal propositions

does not describe a complete state (an exception being the N-Puzzle domain, for instance), so the reachability analysis deals with multiple goal states implicitly. The goal (set of goal propositions in G) does not describe a complete state when the value of some proposition in the instantiated problem is not defined. For example, if S=$\{a,b\}$ and G=$\{a\}$, both s_1=$\{a,b\}$ and $s_2 = \{a, \neg b\}$ are goal states. However, this fact does not change the behavior of the heuristic, meaning that regular reachability heuristics effectively take into account multiple potential goals. This interesting property can be extended to the case in which there are different sets of goal propositions as well.

Taking advantage of this fact, the key observation of this work is to generate multiple intermediate goals that lead to the original goal. Actions known to lead to the problem goals are used to generate them. At each call to the heuristic function, a new intermediate goal set G' is obtained so it can be added to the set of goals $\mathcal{G}$. Initially, $\mathcal{G} = \{G\}$. In the first call to the heuristic function, there will only be a set of goals G. Then, G' is generated by choosing an action a from the RP that supports one of the goal propositions $g \in G$. We remove the propositions in G supported by a and add the preconditions of the action (technique known as goal regression, which determines the qualification of a *backwards generated goal*). Then, this new set G' may be added to $\mathcal{G}$, a list of goals relevant to subsequent computations of the heuristic. Whether a backwards generated goal is added or not to the list depends on its potential usefulness. Usually, the intermediate goal obtained from a node with a bad heuristic value is probably not useful and hence discarded, although the criteria may vary depending on the forward search algorithm used. Lastly, the chosen action a and the originally reached goal proposition g are also stored together with G' with two purposes: allowing reconstructing a path by tracing back the chosen actions; and obtaining the distance from the intermediate goal to the original goal, that is the cost of the intermediate goal. Optimality on this distance cannot be ensured, as there may be shorter paths from the intermediate goal to the original goal.

In the following iterations, there will be several goal sets in $\mathcal{G}$. So, instead of finishing the expansion of the relaxed planning graph when all goal propositions $g \in G$ appear in a given layer P_i, the expansion finishes in a propositional layer P_i when $\exists G_j \in \mathcal{G}$ such that $G_j \subseteq P_i$. In that case, the steps in the previous paragraph are executed to generate a new goal set G' to be added to $\mathcal{G}$.

The implementation of this technique is closely related to how actions are known to be applicable at a given level, as the preconditions of any action form a subgoal themselves. Each proposition maintains a list of indexes of the sets of goals $G_i \in \mathcal{G}$ they appear in. Also, we define a counter that keeps the number of unsatisfied propositions in each intermediate goal set G_i. Whenever a goal proposition p is satisfied by a new action in the relaxed planning graph, the counter of the goal sets G_i where it appears ($p \in G_i$) is decreased. When the counter of any G_i reaches zero, there is a RP that can reach G_i.

A key difference with respect to the standard computation of the RP heuristic is that we require at least one action that reached the intermediate goal set in the last step to be a *legal support*. The concept of legal support is based on the application of the action in the search space: it must be able to appear as the last action in a valid solution plan. There are three conditions that must be fulfilled to ensure that a supporting action a is legal for supporting G_i (thus, being able to generate a new G'):

- its delete effects must not include a proposition appearing in the reached goal ($del(a) \cap G_i = \emptyset$);
- the preconditions of the action must not be mutually exclusive with any proposition of the goal not supported by the action; and
- the new set of goal propositions must not have been previously generated ($G' \notin \mathcal{G}$)

The first constraint is straightforward: an action that deletes a goal proposition can never be the last action of a plan, as it necessarily leads to a non-goal state. In fact, this constraint ensures that there is a legal path from a given intermediate goal to the original goal, although alone it does not suffice as unreachable sets of goal propositions may be generated.

The second constraint is related to the concept of static mutual exclusivity between propositions as it was originally described in (Haslum and Geffner 2000). Static mutexes may not suffice to detect all the unreachable sets of propositions in the search space, but in many domains they are able to prune most unreachable states, as shown by regression planners. In this work, only mutexes between pairs of propositions will be computed, as such sets can be computed in a reasonable time.

The basic idea of the third constraint is that when a duplicated goal is generated, that goal was necessarily already supported in the relaxed graphplan. Then, if the heuristic computation did not stop at an earlier level when supporting that goal, this means that it may be an unreachable goal as no legal action was found for it so far.

Preliminary Experimentation

Both the RP heuristic and the technique presented in this work (which we called BGG, Backwards Generated Goals) have been implemented on top of JavaFF (Coles et al. 2008) using a dual queue. One of the queues uses the BGG heuristic, while the other uses the regular RP heuristic. The reason behind this is that when computing the BGG heuristic, extracting a RP from the original goal takes little additional effort and may be helpful for the cases in which the BGG heuristic is strongly guided towards an unreachable intermediate goal.

Greedy best-first search has been used as the search algorithm. Greedy best-first search expands in every step the most promising node determined by the function *f(n) = h(n)* in which *h(n)* is the heuristic function of the node. Intermediate goals are associated with the state along with its heuristic value in the open list instead of being added to the goal list. When a state is expanded, its intermediate goal is added to the list. This way, all the goals are generated from the best state at every iteration.

Helpful actions for the BGG heuristic are the union of the sets of helpful actions obtained using the BGG RP and the regular RP from the original goal. Helpful action pruning has been enabled, but no restarts with the full set of successors have been enabled. This is done in order to analyze the impact of the extra helpful actions provided by the more informed RPs from intermediate goals.

The focus of the experimentation is the coverage (percentage of solved problems out of the total number of problems). In the experimentation, domains generaly regarded as difficult for FF were preferred. The selected domains were Gold-Miner, Matching-BW, N-Puzzle and Sokoban from the learning track of IPC-6; Peg-Solitaire and Scanalyzer from the deterministic track of the same competition; and Storage from IPC-5. For the sake of completeness, three more traditional domains were also included: Transportation from IPC-6; Driverlog from IPC-3; and Pipesworld No-Tankage from IPC-4.

Table 1 shows the results for these domains in terms of percentage of solved problems. Geometric mean of the ratio between states evaluated by BGG and RP for problems solved by both is also displayed, with values above one meaning that BGG evaluates fewer nodes. Overall, the coverage improves for BGG, mostly in the domains in which the RP alone does not behave well. Not taking into account Gold-Miner, which has a very hard constraint close to the goal and hence is the ideal case for BGG, domains in which BGG performs better have dead-ends and strong order interactions (Matching-BW, Sokoban) whereas in those without them there is no improvement. Regarding the number of evaluated nodes, BGG shows a similar behavior. Also, as the size of the problem increases, BGG tends to expand fewer nodes compared to the regular RP heuristic in spite of having a higher branching factor because of the additional helpful actions.

Domain	RP	BGG	Mean-S	Ratio-BGG
Driverlog	70	**80**	0.89	0.17
Gold-Miner	50	**100**	5.59	0.27
Matching-BW	3	**33**	9.72	0.24
N-Puzzle	10	**13**	4.13	0.01
Peg-Solitaire	**97**	80	1.67	0.32
Pipesworld	**28**	22	0.94	0.14
Scanalyzer	33	**67**	1.47	0.25
Sokoban	13	**17**	5.52	0.19
Storage	53	**60**	1.91	0.21
Transportation	63	**70**	0.7	0.04
Average	42	**57.5**	3.25	0.18

Table 1: Comparison between the RP heuristic and the BGG heuristic in terms of coverage and number of evaluated nodes (geometric mean of the ratio between evaluated states). Ratio-BGG is the percentage of the length of the solution part that belongs to the sequence of actions traced back from the reached intermediate goal.

Also, theoretically, the higher the average length of the part of the solution plan traced back from the reached intermediate goals is, the more relevant the backwards generation of goals is. Therefore, this can be considered a measure of how appropriate BGG is for some domains. Table 1 shows that this holds for most domains with the exceptions of N-Puzzle and Transportation. On average, around a fifth of the plan is retrieved from the reached intermediate goal. This also explains why BGG expands fewer nodes: the search space that must be explored to find a solution is potentially much smaller due to the reduced depth.

Related Work

A field under intense study, landmarks for automated planning (Hoffmann, Porteous, and Sebastia 2004; Richter, Helmert, and Westphal 2008) is our main frame of reference. Being computed in a previous step to the search, landmarks are intermediate goals themselves and thus provide some valuable information to the search. In particular, they capture some constraints in the problem as well as orders among propositions, in a similar fashion to what backwards generated goals do in areas close to the original goal. However, they have important shortcomings derived from the way they are generated. Landmarks are generated from a relaxed instance of the problem in the initial state. They are either actions or propositions that are necessary for some of the goals to be achieved in a reachability analysis or propositions which are common preconditions of the achievers of other landmarks (including goals, which are landmarks by definition). Because of this, unachieved landmarks from any state during the search will always be achieved in the RP. Therefore, if the RP is used as a heuristic, using landmarks as a guide conceptually represents little advantage over the RP heuristic alone. A similar case occurs when landmarks are used to partition a problem (as the RP heuristic already guides the search towards landmarks), although the greedier nature of this approach can be useful.

From a search point of view, there are many links between backwards goal generation and bidirectional/perimeter search (Dillenburg and Nelson 1994). In fact, this approach can be seen as a sort of imbalanced bidirectional algorithm in which the backwards search uses the last actions of the RPs as a guide and the forward search estimates the distance to the frontier of the backwards search instead of to the original goal. Nevertheless, the difficulty of formally defining the technique as such (no generation of successors when doing regression, no heuristic evaluation per se,...) leaves the study of this relationship for future work.

A conceptually close idea, RRT-Plan (Burfoot, Pineau, and Dudek 2006), relies on generating intermediate states by incrementally choosing subsets of the goal and satisfying them. The intermediate goals generation is completely different, being a random selection of subsets of the original goal set in each iteration in their case. The main critique to this work though is that it benefits from guesses over the order of the goals and not so much from actual intermediate states. Interestingly, the authors proposed implementing a bidirectional search algorithm as future work, but this idea has not been further developed.

From a procedural point of view, a recent work on low-conflict RPs (Baier and Botea 2009) shares important as-

pects with this approach. In their work, they take into account delete effects and pairs of mutually exclusive propositions in the backwards search done to retrieve a RP from the reachability analysis to provide more accurate RPs. Furthermore, they expand additional levels in the computation of the heuristic, being the main difference that the number of extra levels are expanded based on a preliminary estimation whereas the expansion of levels in this work is systematic.

Future Work

Future work for the presented approach can be: on a further analysis of the impact of the techniques developed in this work; and implementing other known techniques to improve the performance. It is still unclear how much every technique helps to improve the traditional RP heuristic. An insightful example of this is the aforementioned work on low-conflict plans (Baier and Botea 2009), which uses a similar approach by taking into account delete effects and mutexes, but in a less ambitious way. This way, ideas like forcing the reachability analysis to legally support every proposition instead of only one might greatly affect the performance, which requires a deeper understanding.

On the other hand, some known techniques intuitively seem to be well suited to the backwards generation of goals. Look-ahead techniques are a prominent example, which are known to work well in many domains with forward state planners. The twist in this case is to combine actions from the last part of the RP to generate intermediate goals more than one step farther from the reached goal. Another interesting potential improvement is the use of reasonable orders. So far, they have been used to create a goal agenda (Hoffmann 2001) and to add precedence constraints in heuristic computations (Cai, Hoffmann, and Helmert 2009). The most notable advantage is that reasonable orders are as sound as necessary orders when working with goals or in regression. This way, they can be used to prune actions that support goals reasonably ordered before other goals, greatly reducing the search space when doing regression.

As a last remark, we propose an alternative way of generating intermediate goals. One of the main difficulties of Automated Planning is the strong interaction between actions and propositions. The assumption on the independence of goals is one of the greatest shortcomings of many of the techniques used in the field. In order to overcome this, we propose estimating the probability of some propositions and actions to be true at some point in a solution plan (in opposition to landmarks, which ensure that an action or proposition is true with a probability of 1). For this, we plan on building a planning graph enriched with landmarks and apply a message passing algorithm like survey propagation (Braunstein, Mézard, and Zecchina 2005) to estimate the likelihood of propositions and actions to be true at the different levels of the graph. This way, since probabilities are affected by all the goals and landmarks of the problem, they are more likely to capture the possible interactions among them. Once the probabilities are computed, a sampling method can be used to generate intermediate goals to be used in a Rapidly-Exploring Random Tree (Lavalle 1998) search algorithm. Of course, estimation of probabilities can be exploited in different ways, like for example as a heuristic in local search algorithms, although that possibility is currently out of our scope.

Acknowledgments

This work has been partially supported by a FPI grant from the Spanish government associated to the MICINN project TIN2008-06701-C03-03.

References

Baier, J. A., and Botea, A. 2009. Improving planning performance using low-conflict relaxed plans. In *19th International Conference on Automated Planning and Scheduling (ICAPS)*, 10–17.

Botea, A.; Müller, M.; and Schaeffer, J. 2007. Fast planning with iterative macros. In *IJCAI'07: Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 1828–1833. San Francisco, CA

Braunstein, A.; Mézard, M.; and Zecchina, R. 2005. Survey propagation: An algorithm for satisfiability. *Random Struct. Algorithms* 27(2):201–226.

Burfoot, D.; Pineau, J.; and Dudek, D. 2006. RRT-Plan: a randomized algorithm for STRIPS planning. In *16th International Conference on Automated Planning and Scheduling (ICAPS)*, 362–365.

Cai, D.; Hoffmann, J.; and Helmert, M. 2009. Enhancing the context-enhanced additive heuristic with precedence constraints. In *ICAPS*.

Coles, A. I.; Fox, M.; Long, D.; and Smith, A. J. 2008. Teaching forward-chaining planning with JavaFF. In *Colloquium on AI Education, Twenty-Third AAAI Conference on Artificial Intelligence*.

Dillenburg, J. F., and Nelson, P. C. 1994. Perimeter search. *Artificial Intelligence* 65(1):165–178.

Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *5th International Conference on AI Planning and Scheduling (AIPS)*, 140–149.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–287.

Hoffmann, J. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research* 14:253–302.

Lavalle, S. M. 1998. Rapidly-exploring Random Trees: A new tool for path planning. Technical report, Computer Science Dept, Iowa State University.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, 975–982.

Vidal, V. 2003. A lookahead strategy for solving large planning problems. In *IJCAI'03: Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 1524–1525.