Heuristic Search Replanning with Regressed Goal Descriptions

Sandeep Kumar and **Deepak Khemani**
AIDB Lab, Department of CS&E,
Indian Institute of Technology Madras,
Chennai, India.
sandeep@cse.iitm.ac.in, khemani@iitm.ac.in

Abstract

Plans often fail during execution in the real world. This may happen if the world has changed in the intervening period and some actions are no longer applicable. Often it is desirable to salvage the failed plan as much as possible. This reason is not just computational efficiency, but it may also help in maintaining commitments to resources already made. This paper describes an approach using state-space search for reaching a state from which the failed plan can be taken up again. The algorithm employs the planning-graph based reachability analysis to compute the heuristic value of a state with respect to a set of goal descriptions and carry out the search effectively.

Introduction

An agent operating in real-world is often required to have replanning ability. While executing a plan external events may take place, and the agent may find itself in an unexpected state where the old plan is no longer applicable. For example, a robot agent may discover an object to be in a place different from the one expected. But the rest of the world may be unchanged. In such a scenario it is not only desirable to find a new plan quickly, but also to use the old plan as much as possible for maintaining commitments already made to resources.

Apart from just being able to handle unknown external events, the success of FF-Replan (Yoon, Fern, and Givan 2007) has shown replanning to be a very effective approach for dealing with probabilistic planning problems. FF-Replan first constructs a deterministic version of the planning problem and then calls FF (Hoffmann and Nebel 2001) to generate plan for the deterministic problem. If the plan fails it again calls FF to generate a new plan to the goal taking the unexpected state as the new initial state. We must point out here that FF-Replan does not attempt to reuse the old plan while replanning.

In the worst case replanning has been shown to be no more efficient than planning from scratch (Nebel and Koehler 1995). But in many practical situations we expect that the exogenous events modify the world only to a small degree. Replanning should aim to exploit this fact and try to use a part of the old plan and save on unnecessary computation. In general such systems require a base planning method which computes a sub plan for dealing with the failure.

Heuristic search has enjoyed considerable success in planning. Much of this success can be attributed to the development of strong domain independent heuristics using planning-graph based reachability analysis (Bryce and Kambhampati 2007). The replanner RHS (Replanner using Heuristic Search) described in this paper uses state-space heuristic search paradigm for replanning. It is based on the FF style planning (Hoffmann and Nebel 2001) and can solve replanning problems in STRIPS style domains.

RHS first constructs a set of goal descriptions from the old plan. The goal descriptions correspond to the states that the original plan would have gone through if it had not been disrupted. Once this set is ready, FF is used to search for a plan to reach one of the regressed goal descriptions. The key point of the search is that it is performed with respect to a set of goal descriptions instead of just one. Ideally, during search, when evaluating a state we would like to compute the heuristic distance with respect to each goal description and select the minimum as its heuristic value. But this is computationally very expensive. To overcome this difficulty we compute the heuristic estimate of a state only with respect to the goal description appearing first in the relaxed planning graph. This is based on the assumption that it will be close to the minimum estimate. Once an intermediate goal description is reached, and a plan to it found, the remaining actions are added from the old plan.

The main contribution of this paper is to show how state-space heuristic search using planning-graph based reachability analysis can be used effectively to solve replanning problems.

The next two sections discusses the related work and the basic definitions and notations. The following section describes how FF computes its heuristic using the relaxed planning graph. Then the process of constructing the goal descriptions set using the idea of plan annotation from execution monitoring systems is explained. Following this we discuss the process of replanning using the state-space search. Finally after presenting the experimental results we end with future work and conclusions.

Related Work

State-space heuristic search has been used in the replanning system SHERPA (Koenig, Furcy, and Bauer 2002). SHERPA uses Lifelong Planning A* to find new optimal

plans quickly. The algorithm was designed for robot path (re)planning problems and focuses on changes in the edge weights (action cost). It does not consider changes in the state while RHS can handle random changes in the current state while replanning.

SimPlanner (Onaindia et al. 2001) is another replanning system. It first computes a set of possible reachable states by a process similar to plan annotation (Fritz and McIlraith 2007). Then it selects one of these intermediate states as a goal. In the final step it constructs a plan to reach the intermediate state and appends to it the plan from that state to the goal state. The selection of the intermediate state is based on heuristic evaluation and may not always be optimal. This new goal selection in SimPlanner is done prior to replanning and is fixed henceforth. And replanning can be done by using any of the standard techniques to plan. While in RHS all the potential goals are evaluated at each point during the search and also it is strictly based on forward state-space search planing using relaxed planning-graph heuristic.

Other recent replanning systems have used partial-order planning (Krogt and Weerdt 2005) and planning-graph based techniques (Gerevini and Serina 2000). This makes them different from RHS in the fundamental paradigm of planning.

FF-Replan is the closest replanning system to RHS, using FF as the central planning system. But as pointed out earlier FF-Replan replans from scratch every time a plan fails. Hence RHS can make FF-Replan much more efficient, as it takes into account the old plan during replanning.

Background

The simple STRIPS planning problem is defined as follows:

Definition 1 $(State)$: A state is defined as a set of logical atoms.

Definition 2 $(Action)$: An action a is a triple

$$a = (pre(a), add(a), del(a))$$

where $pre(a)$ are the preconditions of a, $add(a)$ are the add effects and $del(a)$ are the delete effects, each being a set of atoms.

An action is applicable in a state S if $pre(a) \subseteq S$. The result of applying an action on a state S is defined as

$$S \cup add(a) \backslash del(a)$$

Definition 3 $(Planning\ Problem)$: A planning problem $\Pi = (A, I, g)$ is a triple where A is the set of actions, I is the initial state and g is a set of goal atoms.

Definition 4 $(Plan)$: Given a planning problem $\Pi = (A, I, g)$ a plan $\pi = [a_1, a_2, ...a_n]$ is a sequence of actions which changes I to a state S such that $g \subseteq S$.

RPG Heuristic

A common approach while deriving a heuristic for a problem is to relax it to a simpler form, and solve it efficiently. FF uses relaxed planning graph to compute the heuristic value for a state S in the following way.

First all the delete effects of the actions are ignored, this accounts for the relaxation of the problem. Then a planning graph is build until all the goal-atoms are reached. The graph consists of alternating fact and action layers. The first fact layer is same as the state S. The first action layer contains all actions applicable in S. The union of all add effects of actions in the action layer along with the facts of the first fact layer forms the second fact layer. The next action layer is set of all applicable actions in this fact layer. This process of constructing fact and action layers is continued, until a fact layer containing all the goal atoms is reached. The next step is to extract a relaxed plan. To do so, start at the last fact layer m, considering all goal-atoms. At each fact layer i if the goal is the layer $i-1$, then insert it into the goals to be achieved at $i-1$. For other goals select an action in action layer $i-1$ that adds that goal and insert the action's preconditions into the goals at $i-1$. Once all the goal-atoms at fact layer i are finished, continue the same process with goal-atoms at $i-1$ until the first fact layer is reached. The process results in a relaxed plan $< A_0, A_1, ...A_{m-1} >$ where, each A_i is the set of actions selected from the action layer i. The length of the solution is estimated by counting the actions in the plan.

$$h_{FF}(s) = \sum_{i=0}^{m-1} |A_i|$$

Generating Goal Descriptions

Execution monitoring and replanning systems together form a general strategy for dealing with a dynamic world (Russell and Norvig 2003). It is the function of execution monitoring module to decide when replanning is necessary. A common approach among execution monitoring systems is to annotate plans with conditions to be checked at the time of execution, for example PLANEX1 (Fikes 1971). The strategy of PLANEX1 is to find a goal description (kernel) from the annotated plan that is true in the current state of execution. Then the action corresponding to the matched goal description is executed. If none of the goal descriptions match the current execution state, STRIPS (Fikes and Nilsson 1971) is called to find a new plan.

We use the concept of plan annotation for replanning. If there is a plan failure the replanning algorithm tries to get back to a state in the old plan as soon as possible. This means that any of the state from the old plan can now serve as a goal state. As a goal description represents a set of states where the goal atoms are true, generating new goal descriptions increases the total number of goal states. The process of regressing the goal description removes all the unnecessary information (logical atoms) from the state sequence which produced the plan. The work of (Fritz and McIlraith 2007) formalizes plan annotation by regressing goal over actions of a plan in situation calculus. We follow the same approach while generating goal descriptions for problems in STRIPS style specification.

Given a sequential plan $\pi_{old} = [a_1, a_2, a_3.....a_n]$ and a goal description g_n, the corresponding set of goal descriptions is computed as follows:

$$g_{i-1} = [g_i - add(a_i)] \cup pre(a_i) \quad \forall a_i \in \pi_{old}$$

Using the above formula we get a set of goal descriptions $G = \{g_0, g_1, g_2.....g_n\}$.

Replanning

The replanning algorithm uses the heuristic forward state-space search algorithm as its basis. The key points of interest are the termination condition for search and heuristic evaluation of a state.

Termination of Search

The search terminates successfully at a state S when the following condition is found to be true:

$$\exists g_i \in G \; such \; that \; g_i \subseteq S$$

In cases where more than one g_i is found the one closer to the goal g_n is preferred. Once the search terminates, the next task is to construct the new plan. This step is carried out by adding the remaining actions from the old plan. Let g_i be the goal description achieved and π_{new} be the plan found for achieving it. The complete plan to g_n is found by updating π_{new} as follows:

$$\pi_{new} \leftarrow append(\pi_{new}, [a_{i+1}, a_{i+2}, a_{i+3}.....a_n])$$

Heuristic Computation

The heuristic function is a key deciding factor that determines performance in a search algorithm. It is meant to give an approximate measure of the distance of a state from a goal. But while replanning in RHS we wish to compute the approximate distance of a state with respect to the set of goal descriptions G.

Let the heuristic value of a state S with respect to a goal description be given by the function $h(s, g_i)$. A simple strategy to evaluate S against G is to take:

$$h^*(s) = min[h(s, g_i)] \quad \forall g_i \in G.$$

Computing each $h(s, g_i)$ for getting $h^*(s)$ can be expensive, as $h(s)$ is needed at each step in the search. Here a careful design of the heuristic function can significantly cut down the computing cost.

The replanning algorithm uses relaxed planning graph for computing the heuristic value of a state, in a way similar to the planner FF. FF uses the reachability analysis for estimating the distance to a goal. While replanning reachability analysis can be used for dual purpose one to judge which of the goal descriptions may be nearest to the current state and other to estimate the heuristic distance to that goal description.

Let us redefine the heuristic function of FF as $h_{FF}(s, g_i)$ i.e. $h_{FF}(s)$ with respect to the goal description g_i. To compute the heuristic $h(s)$ the replanning algorithm first builds up the relaxed planning graph to a fact layer P^*, until the following is found to be true:

$$\exists g_i^* \in G \; such \; that \; g_i^* \subseteq P^*$$

Once the relaxed planning graph is built and the goal description g_i^* is found. The heuristic value is taken as the length of the relaxed plan to g_i^*. We can define the heuristic as $h_{RHS}(s) = h_{FF}(s, g_i^*)$. This is based on the assumption that $h_{RHS}(s) \approx h^*(s)$.

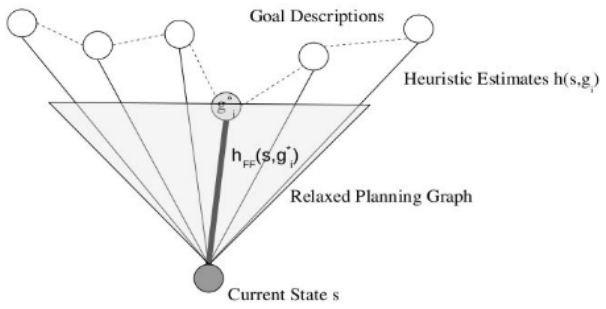

Figure 1: Illustration of $h_{RHS}(s) \approx h^*(s)$. The dotted lines represent the actions in the old plan π_{old} and the corresponding nodes are the generated goal descriptions G. Each line from current state to the goal description represents the heuristic estimate $h(s, g_i)$.

The assumption that the first goal description appearing in the relaxed planning graph is the one which will lead to the minimum heuristic value makes the computation much simpler. Though this may not always be the case, but as we see in figure 1 it is not a bad choice while computing the heuristic. Figure 1 shows the goal description g_i^* first appears in the relaxed planning graph built from the current state. At this point we estimate the heuristic $h_{FF}(s, g_i^*)$. For example in figure 1 if we calculate $h^*(s)$ by computing all $h(s, g_i)$ we will find it to be the same as $h_{FF}(s, g_i^*)$.

High Level Description of the Overall Algorithm

$Input$: A plan π_{old} (for a planning problem Π_{old}). A new planning problem Π_{new} (differing form Π_{old} only in initial state)

$Output$: A new plan π_{new} or fail.

1. Generate the set of goal descriptions G using.

$$g_{i-1} = [g_i - add(a_i)] \cup pre(a_i) \quad \forall a_i \in \pi_{old}$$

2. Perform,

$$\pi_{new} \leftarrow FFSearch(newInitialState)$$

using Terminal Condition:

$$\exists g_i \in G \; such \; that \; g_i \subseteq S$$

and Heuristic Function:

$$h_{RHS}(s)$$

3. If $\pi_{new} = null$ return $fail$.
4. Else Return

$$\pi_{new} \leftarrow append(\pi_{new}, [a_{i+1}, a_{i+2}, a_{i+3}.....a_n])$$

Experimental Results

We now present the empirical evaluation of our replanning approach. The focus of the experiments is on showing the effectiveness of RHS for solving replanning problems in the forward state space search planning paradigm. As the replanning algorithm builds on FF style planning, JavaFF (Coles et al. 2008) was used as the base planner to implement the replanning system. We used the GPG (Gerevini and Serina 2000) benchmark problem set. The same benchmark has also been used to evaluate the POPR system (Krogt and Weerdt 2005). The replanning problems are from the commonly used gripper, logistics and rocket domains. The problem set comprises of seven planning problems (two gripper, three logistics and two rocket). The replanning problems are modeled as a variation in the initial and final state of a planning problem. We used hundred replanning problems from the problem set to evaluate our work. The other problems were unsuitable as they made changes to the goal state, which our algorithm does not cater to.

The problems $grip10$ and $grip12$ are from the gripper domain and each has 10 replanning problems associated with it. Similarly $loga$, $logb$, $logc$ are from the logistics domain and have 20 replanning problems derived from each one. And the problems $roca$ and $rocb$ are from rocket domain also having 10 replanning problems each associated with them.

The replanning system RHS is evaluated against the computational effort of planning from scratch. This is commonly described in terms of the percentage savings. If x and y are the computational efforts required for replanning and planning from scratch respectively, then the percentage savings is defined as $100(y-x)/y$ (Hans and Weld 1995). The same measure is also used to evaluate SHERPA. In our case the replanning effort x and the planning effort y are measured in terms of the time taken to solve the problems. Since the replanning algorithm is implemented using JavaFF, for evaluation purpose we used JavaFF to plan from scratch as well.

The execution times for all the 100 replanning problems have been averaged over 5 trial runs. Figure 2 shows the percentage savings across the different planning problems. Each value in the figure is computed from the average execution times of the replanning problems associated with it. The average execution time of the seven planning problems during planning and replanning is shown in figure 3. The savings percentage in problem $grip10$ is not significant as the planning time is itself very low for replanning to make any notable improvement. In problem $logc$ we observe that the average planning time is quite high and hence we get a good savings percentage. From figure 3 we also observe that the variation in planning time across problems is much larger than variation in replanning time.

The experiments show that in general replanning using heuristic forward state-space search can be much faster than planning from scratch. The replanning algorithm on an average gives more than 75 percent savings, occasionally crossing 90. The overall average planning time of all 100 problems is 2.34 seconds and replanning time is 0.27 sec. The planning time is 8.67 times larger than the replanning time.

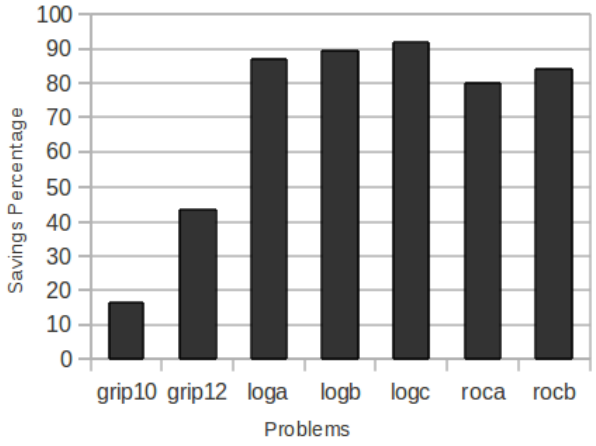

Figure 2: Average savings percentage across planning problems.

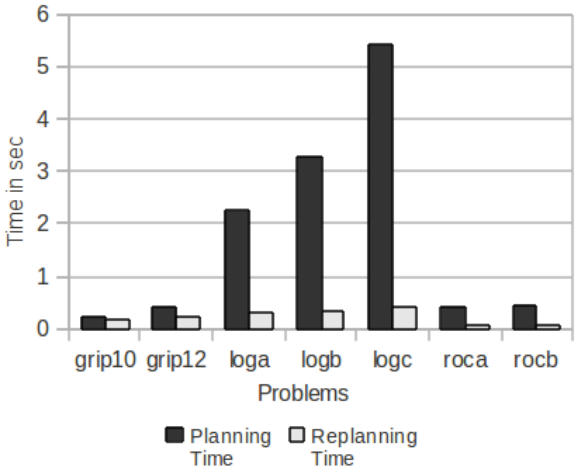

Figure 3: Average running time across planning problems.

Future Work

In this paper we have only considered domains with unit action cost. In future we plan to adapt the algorithm for actions with variable costs. We also plan to implement triangle tables (Fikes 1971) to improve efficiency. Triangle tables help to skip unnecessary comparisons while deciding if a goal description is met, during search and heuristic computation.

In this work we have used the relaxed planning graph for first selecting a goal description and then finding the heuristic value with respect to the selected goal, at each stage. As mentioned before this is based on the assumption that the first goal description appearing in the relaxed planning graph is likely to be closest to the current state. This assumption may not always be true. Hence it may be interesting to look for efficient heuristic functions independent of such assumptions.

The results of FF-Replan have shown that replanning is an effective approach for dealing with probabilistic planning

problems. In future we plan to adapt the replanning algorithm to the needs of a probabilistic problem so that it can induce a contingency plan.

Another interesting extension of this work is to explore how effectively replanning in partial satisfaction problems (Benton, Do, and Kambhampati 2009) can be handled by the algorithm. The replanning algorithm will also need adaptation while working with temporal planners using state space search like CRIKEY (Coles et al. 2009).

Conclusions

Plans often fail during execution in dynamic environments. The paper describes an approach using heuristic state-space search for replanning in the face of plan failure. The idea of plan annotation from execution monitoring can be used to generate goal descriptions which can be used for replanning. The paper also demonstrates how planning-graph based reachability analysis can be used as a heuristic which can evaluate a state against a set of goal descriptions efficiently. The empirical analysis of the approach shows that replanning using heuristic search can be far better than planning from scratch. Thus if FF-Replan uses RHS as the replanning module, instead of planning from scratch, it will have a gain in performance.

Acknowledgements

We thank I. Murugeswari and Bharat Ranjan Kavuluri for the discussions and constant support during this work.

References

Benton, J.; Do, M.; and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence* 173(5-6):562–592.

Bryce, D., and Kambhampati, S. 2007. A tutorial on planning graph based reachability heuristics. *AI Magazine* 28.

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*.

Coles, A. I.; Fox, M.; Halsey, K.; Long, D.; and Smith, A. J. 2009. Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence* 173(1):1–44.

Fikes, R., and Nilsson, N. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4).

Fikes, R. 1971. Monitored execution of robot plans produced by strips. *IFIP Congress 71*.

Fritz, C., and McIlraith, S. A. 2007. Monitoring plan optimality during execution. *In Proceedings of the International Conference on Automated Planning and Scheduling*.

Gerevini, A., and Serina, I. 2000. Fast plan adaptation through planning graphs: Local and systematic search techniques. *In Proceedings of the International Conference on AI Planning and Scheduling*.

Hans, S., and Weld, D. 1995. A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research* 319–360.

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

Koenig, S.; Furcy, D.; and Bauer, C. 2002. Heuristic search-based replanning. *In Proceedings of the International Conference on AI Planning and Scheduling* 294–301.

Krogt, R. V. D., and Weerdt, M. D. 2005. Plan repair as an extension of planning. *In Proceedings of the International Conference on Automated Planning and Scheduling* 161–170.

Nebel, B., and Koehler, J. 1995. Plan reuse versus plan generation: A theoretical and empirical analysis. *Artificial Intelligence* 76(1-2):427–454.

Onaindia, E.; Sapena, O.; Sebastia, L.; and Marzal, E. 2001. Simplanner: An execution-monitoring system for replanning in dynamic worlds. *In Proceedings of the EPIA* 393–400.

Russell, S. J., and Norvig, P. 2003. *Artificial intelligence: A Modern Approach*. Prentice-Hall, Inc.

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