FAP: Foward Anticipating Planner

Guy Camilleri and **Joseph Zalaket**
IRIT CCI-CSC,
Université Paul Sabatier,
118 route de Narbonne,
31062 Toulouse Cedex 4 FRANCE
camiller@irit.fr zalaket@irit.fr

Abstract

In this paper we introduce a new planning system FAP based on the heuristic search. For the heuristic calculation, FAP combines the techniques used in abstraction and heuristic planning. FAP calculates his heuristic by projecting the planning problem in a relaxed problem where the delete lists of the actions are ignored and the actions are grouped in sequences according to their order of application. FAP uses the calculated heuristic to guide its search on a N-Best-Search Hill-Climbing algorithm which is a combination of the N-Best-Search and Hill-Climbing algorithms.

Introduction

The heuristic search has enhanced the performance of planning algorithms. Planners like HSP (0) HSPr (0) and FF (0) has shown the ability of solving large planning problems according to the classical previous planners. The heuristic used buy the most of the current planners is based on the idea of McDermott (0) as well as Bonet et al. (0), which propose the relaxation of the problem in a simpler problem by ignoring the delete lists of the actions. Also the heuristic idea was early used in the hierarchical planning in a king of relaxing the problem buy projecting it in an abstract problem where the solution can be found faster (see planners like NOAH (0), NONLIN (0)). The abstraction used in hierarchical planning was often based on the actions or states grouping. In this paper we introduce a new planning system FAP based on the heuristic search. For the heuristic calculation, FAP combines the techniques used in abstraction and heuristic planning. FAP calculates the heuristic by projecting the planning problem in a relaxed problem where the delete lists of the actions are ignored and the actions are grouped in sequences according to their order of application. FAP uses the calculated heuristic to guide its search on a N-Best Hill-Climbing heuristic Search algorithm which is a combination of the N-Best heuristic Search and Hill-Climbing algorithms. In the rest of this paper we present an overview of our work. We explain the sequences meta-actions calculation. We present the generation of the sequences to finish with the main search algorithm.

Overview

FAP is a forward planner in a state space which combines heuristic search planning techniques with a ”state grouping” approach. As HSP (0), FF (0), etc. state's heuristic[1] is computed from a solution of a relaxed problem. The relaxed problem[2] ignores action's delete list and is solved through a planning graph similar to the GraphPlan's planning graph (0). The state grouping approach constitutes the main originality of this work. It aims at reducing state search space by grouping states, and is done through the generation of meta-actions ”sequences” rather than building states shapes as in ShaPer (0) or states abstractions as in some hierarchical planning systems like ALPINE (0).

During the search, FAP generates new actions (or meta-actions) corresponding to the actions ”sequence” called anticipations. These actions ”sequence” are used like the other ones in the planning graph, in the states search space and can belong to other actions ”sequence”. In this way, all states are not considered in the search space.

All candidate actions to the sequence generation are pulled out from the planning graph. The actions selection during the extraction of the relaxed solution is essential because they do not only participate to the heuristic calculation but also to the state grouping. Currently, FAP extracts the relaxed solution in regression (from the last level) and uses some local criteria to select actions in the planning graph.

The main search algorithm used in FAP is an extension of the N-Best heuristic Search algorithm NBS (proposed in (0)) called N-Best heuristic Hill-Climbing Search algorithm NBHCS. This algorithm is complete and can be viewed as a kind of Hill-Climbing algorithm with a backtracking. Therefore FAP considers all applicable actions (not only the anticipations) to be complete.

For each state, FAP applies the following steps:

1. Relaxed planning graph building (similar to FF),

[1] The heuristic corresponds to an estimation of the distance in number of actions between the initial state and the goal.

[2] In STRIPS, a planning problem $P = (O, I, G)$ is define by a set of operators O which change the world state, an initial state I and a goal G to satisfy. The operators of the considered relaxed problem $P' = (O', I, G)$ correspond to the operators of the problem P without the delete list.

2. Relaxed solution extraction which defines the candidate actions and the heuristic,
3. Sequence actions generation (in progression and then in regression).

In the first part of this paper, the meta-action "sequence" is briefly presented . Then, we expose the selection of relevant actions corresponding to the relaxed solution, the sequence actions generation and the state search algorithm NBHCS

Meta-action "Sequence"

A ground action α in STRIPS is described by the following lists: param(α) is the list of action's parameters, pre(α) is the list of preconditions which must hold for action's application, add(α) and del(α) lists are respectively the list of addition and the list of deletion of the action.

Definition 1 *The meta-action "sequence" $\rhd(\alpha_1,\alpha_2)$ is defined by:*

- *param($\rhd(\alpha_1,\alpha_2)$)=(α_1,α_2)*
- *pre($\rhd(\alpha_1, \alpha_2)$)= pre(α_1) $\cup$ (pre(α_2) $\setminus$ add(α_1))*
- *add($\rhd(\alpha_1,\alpha_2)$)= [add(α_2)$\cup$ (add(α_1) $\setminus$ del(α_2))] $\setminus$ pre($\rhd(\alpha_1, \alpha_2)$)*
- *del($\rhd(\alpha_1,\alpha_2)$)= [del(α_2) $\cup$ (del(α_1) $\setminus$ add(α_2))] $\cap$ pre($\rhd(\alpha_1, \alpha_2)$)*

Moreover, Fap used the following properties on the meta-action "sequence":

Definition 2 *Two ground actions α_1 and α_2 are* S-independent *iff pre($\rhd(\alpha_1, \alpha_2)$)= pre($\rhd(\alpha_2, \alpha_1)$), add($\rhd(\alpha_1, \alpha_2)$)= add($\rhd(\alpha_2, \alpha_1)$) and del($\rhd(\alpha_1, \alpha_2)$)= del($\rhd(\alpha_2, \alpha_1)$).*

Definition 3 *A sequence $\rhd$ is* correct *iff it exists a state s reachable from the initial state such as $\rhd$ is applicable in s.*

Relevant actions extraction and sequences generation

For each state FAP builds a relaxed planning to calculate the heuristic of that state. Actions are extracted from this planning graph in regression. The extraction process starts from the goals in the last level and go back to the first level 0. For each goal in the current level, only one action is selected from the previous level for sequence building according to some local criteria. The preconditions of the chosen actions are then added to the goal set and then the process goes back to the previous level until the first level is reached.

The local criteria use the following relation of authorization:

Definition 4 *An action α_1* authorizes *α_2 iff $del(\alpha_1) \cap pre(\alpha_2) = \varnothing$*

Definition 5 *A sequence $\rhd(\alpha_1, \alpha_2)$ where level(α_1)=i and level(α_2)=$i+1$ is considered* useful *at a level i iff it exists an atom p such as level(p)=i and $p \in add(\alpha_1) \cap pre(\alpha_2)$.*

The local criteria describe some selection rules between actions belonging to two successive action's levels. For each goals g at a level i, an action α is chosen at the level $i-1$ if 1) $g \in add(\alpha)$ and 2) for all actions β in the level i such as $g \in pre(\beta)$, α $authorize$ β and α minimize the ***difficulty*** of $\rhd(\alpha, \beta)$ whith $difficulty(a) = \sum_{p \in pre(a)} level(p)$. From this selection, only actions which maximize the number of goals of the level i are chosen so as all level goals belong to an add list of these actions.

The meta-actions "sequences" are generated from a partial planning graph containing only the extracted actions. A first generation is done in forward from the actions in the level 0 in the following way: if all actions α_i in level 0 are many to many *S-independent* then generate the sequence $\underset{i}{\rhd}\alpha_i$. Then for all generated sequences $\underset{k}{\rhd}$ in a level i and all actions β in the level $i+1$, only the *useful* sequences $\rhd(\underset{k}{\rhd}, \beta)$ are computed. The process stops when the last level is reached or if any sequences can be generated at the current level c.

In the second generation, only the *useful* sequences are computed by pairs of successive levels in backward from the last level to the level c.

NBHCS algorithm

The search algorithm used in FAP is an instantiation of the N-Best heuristic Search Algorithm (NBS). The NBS algorithm is at a time a functional extension and a simpler implementation of the First Best Search algorithm. In many planning problems, a state has a big number of successors, which decreases the planning performance if all of them are visited. The idea of the NBS algorithm is to generate a limited number N of successors at a time instead of generating all of them, then to expand the graph for the next N successors if no solution found and so on. Moreover, because the graph can be expanded every time the solution is missed up to containing all the successors, this algorithm is complete.

In the figure 1, the NBS algorithm is presented. The search process could be defined as a quadruplet (S_c,G,Γ_c,S_i) where S_c is the current best state, G is the goal, Γ_c is the set of operators applicable to S_c and S_i is the initial state. Any state S_n is completely expanded when its successors states are memorized and this state is kept in a list of all completely expanded states named Closed. A state S_k is partially expanded whenever it does not have any memorized successor or a part of its successor states are kept, all of these states are included in a list of states called Open.

Our N-Best heuristic Hill-Climbing Search algorithm is an NBS algorithm with a specification of the generate_best_successors function (see figure 2). As in Hill-Climbing search algorithm, the process of generating successors stops when a best successor is found. Let remark that in FAP the order in which the actions are memorized in the set Γ_c is very important because it defines the expansion strategy. The memorized order is: the generated sequences, the helpful actions (like the ones used in FF) and then the others, of course all these actions are applicable in the current state. Therefore, in a first stage the sequences are applied, in a second the helpful and then the other actions. By this way, FAP is complete.

Algorithm 1 The N-Best heuristic Search Algorithm

```
Open ← {(S_i,Γ_0)};
Closed ← ∅;
while Open ≠ ∅ do
  (S_c,Γ_c)=get_state_with_min_f(Open);
  generate_best_successors(N,S_c,Γ_c);
  if best_successors(S_c) ∩ G ≠ ∅ then
    return S_c;
  end if
  Open ← Open ∪ best_successors(S_c);
  if Γ_c = ∅ then
    Open ← Open ∖ {(S_c,Γ_c)};
    Closed ← Closed ∪ S_c;
  else
    update_Γ(S_c,Γ_c);
  end if
end while
return Failure;
```

Algorithm 2 generate_best_successors strategy

```
Successors ← ∅;
repeat
  γ ← element(Γ_c);
  S ← apply(S_c,γ);
  Successors ← Successors ∪ {S};
until f(S) < f(S_c)
return Successors;
```

Conclusion

This paper shows a new heuristic search planner based on the problem relaxation by action grouping. In the palnnigngraph, the generation of sequences "actions group" and their application can be more informative as a heuristic guide than the separated actions application. The main search algorithm can recuperate the time that the computation of sequences takes. Therefore, the main search algorithm can use a shorter path to achieve the goal with sequences than with direct heuristics. The main stake is to build the best sequences by choosing the actions as possible in the order of their applications to access the result as fast as possible. This will be our future work where we aim to refine the local criteria in a way to obtain optimal sequences, and by consequence to reduce the search time and the search space. Another extension to FAP will be the introduction of actions with conditional effects, where we thought the local criteria refinement would be harder to generate relevant sequences instead of generation a sequence for each possibility.
The main search algorithm of FAP is the N-Best-Hill-Climbing which is complete and in which we can go back to revisit previous actions when needed. But our experiments have showed that the result is often achieved in the first pass.

References

B. Bonet, G. L., and Geffner, H. 1997. A robust and fast action selection mechanism for planning. *to appear in the AAAI-97 Proceedings*.

Blum, A. L., and Furst, M. L. 1995. Fast planning through planning graph analysis. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI95)* 1636–1642.

Bonet, B., and Geffner, H. 2000. HSP: Heuristic search planner. *Entry at AIPS-98 Planning Competition, AI Magazine* 21(2).

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129:5–33.

Guéré, E., and Alami, R. 2001. One action is enough to plan. *IJCAI* 17:439–444.

Hoffman, J. 2001. FF: The fast-forward planning system. *AI Magazine* 22:57 – 62.

Knoblock, C. 1994. Automatically generation abstractions for planning. *Artificial intelligence* 68(2):243–302.

McDermott, D. 1996. A heuristic estimator for means ends analysis in planning. *Proceedings of the 3rd International Conference on Artificial Intelligence Planning Systems. AIPS.*

Pais, J., and Pinto-Ferreira, C. 1999. he n-best heuristic search algorithm. *In proceedings of the 18th Workshop of the UK Planning and Scheduling Special Insterest Group PLANSIG99, England*.

Sacerdoti, E. D. 1975. The nonlinear nature of plans. *In Proceedings of the Fourth International Joint Conference on Artificial Intelligence (IJCAI-75)* 206–214.

Tate, A. 1977. Interacting goals and their use. *In proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77)* 888–893.