Macro-AltAlt: Improving heuristic guided search using frequency based Action Macros

Murugeswari I & N.S.Narayanaswamy
Department of Computer Science & Engineering
Indian Institute of Technology Madras, India

Abstract

A knowledge base of macro actions is created by the learner from plans generated for sample problems in a domain. This knowledge is used in a heuristic guided state space search planner to improve the heuristic value of states according to the frequency of macro actions.

1. Introduction

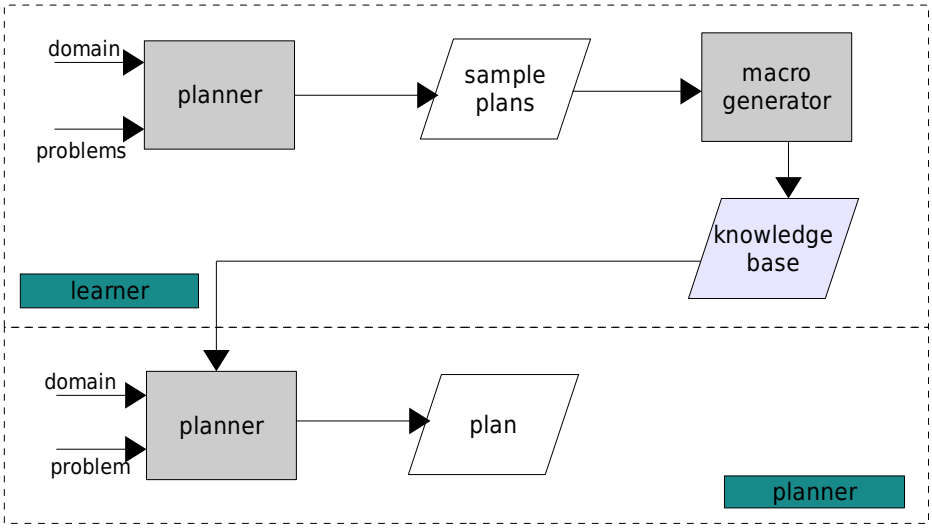
The learner is an offline learning mechanism consisting of a planner and a macro generating module. A macro-action is defined as an action sequence with adjacent actions acting on a common object. Sample problems in a domain are solved and macro actions are extracted from these plans. The macro actions are ranked according to frequency and compiled into a knowledge base.

The planner is a heuristic guided state space search planner. The knowledge base is used in the planner to change the order in which states are expanded in the search process by adjusting their heuristic values.

AltAlt is used as the planner in both the learning and the planning components.

2. Architecture

Macro-AltAlt is organized as shown below:



2.3 AltAlt

AltAlt is a heuristic guided backward state space search planner. It brings together the best of two well know planners, STAN and HSP-r. STAN is a graph plan based planner and HSP-r is a heuristic search planner. AltAlt constructs a plan graph and computes heuristic values from it for the states in the state space. It then performs a best first search on the state space based on the heuristic value of the states.

The following changes are made to AltAlt:

1. In the best first search, if there is a state which is as good as the best state chosen for expansion, then both are expanded and all their children are considered in the next level as open nodes.
2. The memory requirements are reduced by allocating only as much memory as is required for a state and by avoiding the usage of temporary linked lists.
3. An action that reverses the effect of the previous action is not considered for application.
4. The weightage given to the heuristic value of a state is changed from a constant to a function of the number of grounded facts in the problem.

2.2 Learner

The learner consists of a planner and a macro generator. It takes as input a set of sample problems in a domain and constructs a knowledge base of action macros.

The planner in the learner solves the input problems and writes the plans to a temporary file. These plans are formatted to remove action arguments introduced due to untyped pddl definitions of the domain and problems. Using untyped pddl definitions causes certain objects to be present in the argument list of all grounded instances of an action. This is necessary as these extra arguments are not to be considered as a common object between adjacent actions while extracting action macros.

The macro generator works on these plans to extract action macros. The length of the action macros extracted is currently restricted to two. The macro generator identifies macro actions by applying the classic 'apriori' algorithm in data mining adapted for plans and creates a knowledge file for the domain.

The knowledge file has a record for each action-macro with the following entries:

1. Action-Macro name
2. Rank among action macros with the same prefix
3. Length of the macro-action
4. Pointer to the prefix action-macro
5. Pointer to the first action-macro which has this action-macro as prefix
6. Pointer to the next action-macro with the same prefix

The first line of the knowledge file has a count of the number of action macros in it. Primitive actions are considered as action macros of length one and are also stored in the knowledge file. In the 'name' field, only the last action of a action-macro is stored. A prefix of a action-macro is defined as the sequence of actions in it except the last action. Rank is an integer in the range 1 to n, where n is the number of action macros with the same prefix. An action-macro with a rank of 'n' is the most frequently occuring one in the sample plans.

2.3 Planner

The planner performs a best first search of the state space based on heuristic values of states. A state with the lowest heuristic value is considered as the best one. At each step in the search, the best node seen so far is selected for expansion. After expansion, a new heuristic value is computed for each of the child nodes as follows:

new-h-value(s) = old-h-value(s) - rank(a, b)

where s denotes the child state; a denotes the action that created the child and b denotes the action that created the parent.

rank(a,b) is obtained from the knowledge file. It is zero if the action sequence (a,b) is not present in the knowledge file as an action macro.

This new heuristic value is used to choose the best node for the next iteration.

3. Conclusions

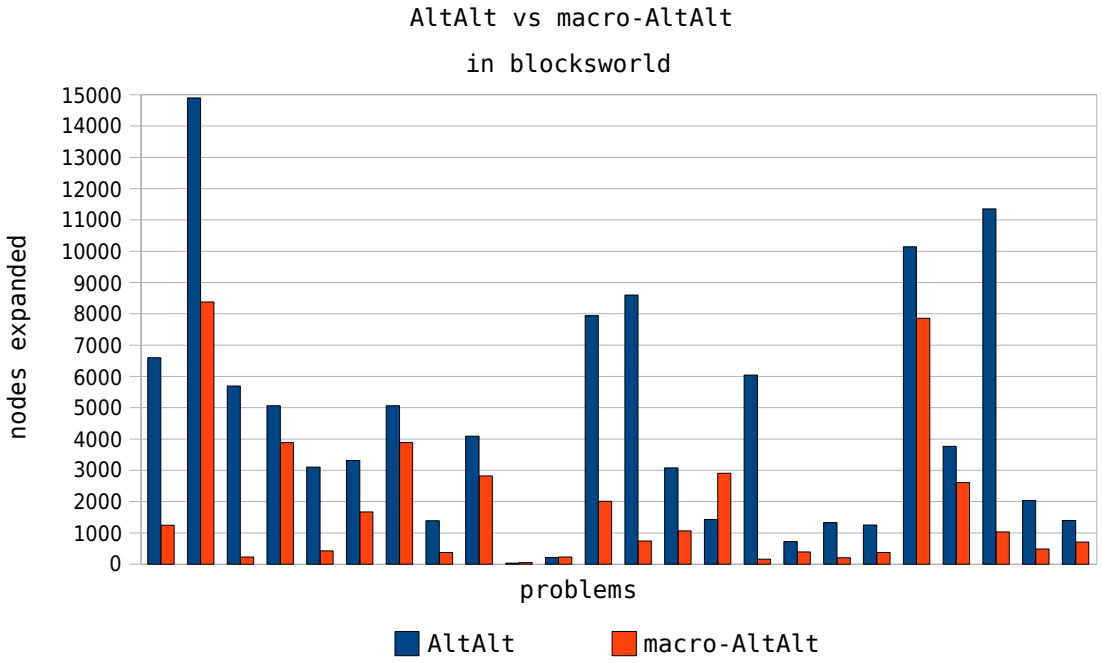
3.1 Experiments

This learning and planning approach is tested in the blocksworld domain. After learning, for 90% of the problems, plans are found in comparatively less time expanding lesser number of nodes in the search tree.

The following knowledge file of action macros is created by the learner :

```
7
unstack 3 1 0 5 2
stack 4 1 0 0 3
putdown 1 1 0 0 4
pickup 2 1 0 7 0
stack 2 2 1 0 6
putdown 1 2 1 0 0
stack 1 2 4 0 0
```

The number of nodes expanded to find a plan by AltAlt and macro-AltAlt are shown for a few problems in the following graph:



3.2 References

[1] AltAlt - Combining the advantages of GraphPlan and Heuristic State Search (*Romeo Sanchez Nigenda, XuanLong Nguyen & Subbarao Kambhampati)*
[2] Using Data Mining to Enhance Automated Planning and Scheduling (*Jeremy Frank)*