Exploiting Label Field in Intelligent Planning

Li Xu¹, Wen-Xiang Gu¹, Xin-Mei Zhang¹, and Long-Mei Jie¹

¹School of Computer, Northeast Normal University, Postal Code 13 01 17, Changchun, Jilin, China
{xul094, gwx, zhangxm372, jie1m}@nenu.edu.cn

Abstract. The paper introduces a new approach to planning based on the generation of label member, which provides a new perspective on the planning problem and speeds up the process of the problem solving in the artificial intelligence system plan. On the basis, we propose a novel planning algorithm on label field by thoroughly exploring the structure of planning graph and improve Graphplan in several ways. The algorithm is provably sound, complete, polynomial-time and polynomial-space of label member generation. The paper demonstrates it has excellent performance in terms of time and space. Else, because of wide application of intelligent planning, our research is very helpful to the development of robotology, natural language understanding, intelligent agent etc.

1 Introduction

Intelligent planning is an intersectional subject which deals with knowledge representation, data mining, human-machine interaction, biology, mathematics, cognitive science and so on. Not only is its development of importance in artificial intelligence, but it also will fundamentally change the traditional way the human operates a computer. Its research started in the 1950’s. The problem solving system QA3¹ designed by Green in 1969 is considered as the first intelligent planning system. The STRIPS² planner, designed by Fike and Nilsson in 1971, have historical significance in intelligent planning, in which knowledge representation and reasoning methods deeply affect later planning systems. But confined to objective conditions, the field was under the conservative state at one time. It was not until the end of 1980’s that the planning field was improved greatly. Over the last few years we have seen a significant increase of the efficiency of planning systems. This is mainly due to several new approaches in plan generation. The most remarkable approach was Graphplan proposed by Blum and Furst. In their seminal paper³,⁴ on the Graphplan system, they described a new plan generation algorithm, based on planning graph, which was much faster than any other algorithms known at that time. It possesses some important properties, including: optimality in the number of actions and length of a synthesized plan; soundness and completeness; and polynomial time and space complexity of the creation of planning graph structure. Graphplan caused revolutionary progress in intelligent planning⁵, and it started a whole series of research efforts that refined this approach by making it even more efficient and by

©GESTS-Oct.2005
extending it to cope with more expressive planning languages. Such as Blackbox\(^6\), FF\(^7\), LGP\(^8\), SGPlan\(^9\), the most successful automatic planner at recent AIPS planning competition from 1998 to 2004, all adopted Graphplan technique more or less. The paper introduces a novel planning algorithm based on label field by thoroughly exploring the structure of Graphplan, which extends and improves the original algorithm in several ways. The memory of the approach is much less than Graphplan’s and the speed are faster. Therefore, the algorithm has excellent performance in terms of time and space.

2 Definitions and Notations

In this section, we give some notations relevant to our algorithm.

McDermott and James Hemdeler think a plan is devising the sequence of actions for an agent.\(^10\) We generally think a plan is a set of actions that will achieve the goals of a problem.

A planning problem consists of:
(a) a STRIPS-like domain (a set of operators),
(b) a set of objects,
(c) a set of propositions (literals) called the initial conditions,
(d) a set of problem goals which are propositions that are required to be true at the end of a plan.

Interfere: two actions interfere with each other if one deletes a precondition or an effect of the other.

Valid plan: a valid plan for a planning problem consists of a set of actions and specified time steps in which each is to be carried out. There will be actions at time step 1, actions at time step 2, and so forth. Several actions may be specified to occur at the same time step so long as they do not interfere. A valid plan must make all the problem goals true at the final time step.

Nop: a special kind of action that does nothing to a proposition at time step \(i\).

Mutex: we define this relation recursively as follows (see Fig.1):

Two actions at level \(i\) are mutex if either
(a) inconsistent effects: the effect of one action is the negation of another action’s effect, or
(b) interference: one action deletes the precondition of another, or
(c) competing needs: the actions have preconditions that are mutually exclusive at level \(i-1\).

Two propositions at level \(i\) are mutex if one is the negation of the other, or if all ways of achieving one proposition (i.e. actions at level \(i-1\)) are exclusive of all ways of achieving another (inconsistent support).

Level Off: At the stage of Graph Expansion, there exists a minimum \(n\), for any positive integer \(m \geq n\), the proposition level, the action level and mutex at time step \(m\) are the same as those at time step \(n\), we call that from time step \(n\) on the Planning Graph has leveled off. \([3,4]\) have proved that any Planning Graph can reach Level Off.
Exploiting Label Field in Intelligent Planning

Label field: Label field is a totally ordered set, it is composed of a finite number of label member. \{0,1,2,3...\}

Label member: label member is a positive integer, an element of label field. For each proposition or action, the label member is the number of the first layer at which it appears in the graph. For each mutex, the label member is the number of the last layer at which it appears in the graph.

This paper discusses the problems in STRIPS-like domains.

Fig.1. graphical depiction of mutex definition (devised by David Smith), circles denote propositions, squares represent actions, and thin, curved lines denote mutex relations. The first three parts illustrate deduction of an action-action mutex (between the dark boxes), and the last part depicts the discovery of mutex between propositions (the dark circles).

3 Graphplan

In order to shed light on the sources of strength of Graphplan, we introduce Graphplan algorithm at first.

Graphplan algorithm is composed of two phases: Construction of Planning Graph and Solution Extraction. The algorithm begins by explicitly constructing planning graph. Planning graph offers a means of organizing and maintaining search information. Solution extraction involves searching for a sequence of actions and solves the planning problem. This can be done by a backward search over the planning graph structure. The two parts will be carried out alternately.

3.1 Construction of Planning Graph

All the initial conditions are placed in the first proposition level of the graph.

To create a generic action level, we do the following. For each operator and each way of instantiating preconditions of that operator to propositions in the previous level, insert an action node if no two of its preconditions are labeled as mutex. Also insert all the nops and insert the precondition edges. Then check the action nodes for exclusivity and memorize these mutexes.

To create a generic proposition level, simply look at all the add effects of the actions in the previous level (including nops) and place them in the next level,
connecting them via the appropriate add and delete edges. Mark two propositions as
mutex if all ways of generating the first are mutex of all ways of generating the
second.

3.2 Solution Extraction

Given a planning graph, Graphplan searches a valid plan using a backward-chaining
strategy. In order to best make use of mutex, it uses a level-by-level approach. In
particular, given a set of goals at time $t$, it attempts to find a set of actions (nops
included) at time $t-1$ having these goals as add effects. The preconditions to these
actions form a set of subgoals at time $t-1$ having the property that if these goals can be
achieved in $t-1$ steps, then the original goals can be achieved in $t$ steps. If the goal set
at time $t-1$ turns out not to be solvable, Graphplan tries to find a different set of
actions, continuing until it either succeeds or has proven that the original set of goals
is not solvable at time $t$.

4 Analyzing the Structure of Graphplan

From the introduction above, we can see Graphplan algorithm is based on
constructing and analyzing a planning graph.

By exploiting the following observations concerning monotonicity in the planning
graph, we can draw a conclusion one can avoid duplicated work during expanding a
planning graph.

Propositions are monotonically increasing: if proposition $P$ is present at level $i$, it
will appear at level $i+1$ and at all subsequent proposition levels.

Actions are monotonically increasing: if action $A$ is present at level $i$, it will appear
at level $i+1$ and at all subsequent action levels.

Mutexes are monotonically decreasing: if mutex $M$ between actions $A$ and $B$ is
present at level $i$, then $M$ is present at all previous action levels in which both $A$ and $B$
appear. The same is true of mutexes between propositions.

Nogoods are monotonically decreasing: if subgoals $P$, $Q$ and $R$ are unachievable at
level $i$, then they are unachievable at all previous proposition levels.

These observations suggest one can discard a multi-level planning graph. Instead,
all one needs is memorizing a set of label members. Action, proposition, mutex, and
nogood structures are all annotated with an integer label field: for proposition and
action nodes, this integer denotes the first planning graph level at which the
proposition or action appears. For mutex and nogood nodes, the label member marks
the last level at which the relation holds. One may interleave forward expansion and
backward extraction to search a solution plan by adding an additional label field.
Using this scheme, the costs of time and space during expansion phase are vastly
decreased. The idea is shown in the following algorithm.
5 Description of the Planning Algorithm Based on Label Field

Our algorithm is composed of two parts: Generation of Label Member and Plan Extraction. The two parts will be carried out alternately.

5.1 Generation of Label Member for Each Proposition, Action and Mutex

(1) The label members of all propositions and operators are initialized to $\infty$.

(2) The construction of the proposition set with label member “1”

All the initial conditions are set to label member “1”.

(3) The construction of the action set with label member “1”

For each operator, there is a counter, which is initialized to 0. If a precondition of the operator appears in the proposition set with label member “1”, get its counter incremented. As soon as the counter for an operator reaches the total number of its preconditions, the operator can be instantiated. All the operators initialized come into being the action set with label member “1” (not including nops).

(4) The construction of the action-action mutex set with label member “1”

Examine action-action mutexes in the nops that maintain the propositions with label member “1” and the actions with label member “1” by using the definition of mutex stated above. If exist, set the label member of these mutexes to “1”. The mutex set varies with the expansion of planning graph.

(5) The construction of the proposition set with label member “$i$”

The effects of actions (not including nops), if not already present in the proposition set with label member lower than or equal to “$i-1$”, just have the label member “$i$”.

(6) The construction of the proposition-proposition mutex with label member “$i$”

Examine proposition-proposition mutexes in the propositions with label member “$i$” by using the definition of mutex stated above. If the proposition-proposition mutexes with label member “$i-1$” still exist, change their label members to “$i$”, and set the label members of new proposition-proposition mutexes to “$i$”. The mutex set varies with the expansion of planning graph.

(7) Exit

If each goal has a label member lower than $\infty$, the largest label member of the goals is $t$, and no pairwise propositions of the goal set is in the proposition-proposition mutex set with label member “$t$”, then the algorithm is over and turn to search a valid plan.

Else, if no propositions are labeled “$i$” in step (5), and the proposition-proposition mutex set with label member “$i-1$” is empty after step (6), we say the Planning Graph has leveled off from time step $i-1$ on, which shows no valid plan exists. The algorithm is over.

(8) The construction of the action set with label member “$i$”

For each operator, there is a counter, which is initialized to 0. If a precondition of the operator appears in the proposition set with label member “$i$”, get its counter incremented. As soon as the counter for a operator reaches the total number of its preconditions and the label members of the proposition-proposition mutexes concerning any two of these preconditions lower than “$i$”. The operator can be
instantiated. All the operators initialized come into being the action set with label member “i”.

(9) The construction of the action-action mutex with label member “i”

Examine action-action mutexes in the nops that maintain the propositions with label member “i” and the actions with label member “i” by using the definition of mutex stated above. If the action-action mutexes with label member “i-1” still exist, change their label member to “i”, and set the label members of new action-action mutexes to i. The mutex set varied with the expansion of planning graph.

5.2 Solution Extraction

A main task is solution extraction in the planning problem. The paper adopts the approach of backward-chaining strategy. From step (5) of Generation of Label Member, after the construction of the proposition level at each step, we examine whether at this time all the propositions in the goal set appear and none of them are mutex, if not, continue to generate label members. If it is true, we will begin to search a valid plan.

The basic idea of solution extraction is: above all, judge whether there exists the valid plan, if not, the algorithm over, otherwise, the valid plan extraction. We will follow up the way of searching the plan backward-chaining on the basis of the label members of propositions, actions and mutexes. Firstly the goals are viewed as a proposition set. Then we begin searching the valid plan, until finding it. ★ Choose a proposition in the set. Look for an action, action1, which satisfies the following requirements: firstly, the action is a nop or in the action set with label member lower than or equal to t-1; secondly, the effects of the action include the proposition. Then, choose another proposition in the set. Look for an action, action2, which satisfies the following requirements: firstly, the action is a nop or in the action set with label member lower than or equal to t-1; secondly, the effects of the action include the proposition; last, action1 and action2 must be guaranteed not in the action-action mutex set with label member “i-1”. If such action does not exist, the algorithm backtracks at once. Go on with this job in this way, until we have found such actions for each proposition in the set and any pair of these actions is not in action-action mutex set with label member “t-1”. Let the effects of these actions comprise a proposition set. Carry out ★ until the proposition set is the superset of initial conditions.

6 An Example

Consider the problem of preparing a surprise date for one’s sleeping sweetheart. The goal is to take out the garbage, arrange dinner, and wrap up a present. There are four possible actions: cook, wrap, carry, and dolly. Cook requires cleanHands and achieves dinner. Wrap has precondition quiet (since the gift is a surprise, one mustn’t wake up the recipient) and produces a present. Carry eliminates the garbage, but the touch with a smelly container negates cleanHands. The final action, dolly, also
eliminates the garbage, but it negates quiet because of the noisy truck. Initially, you have cleanHands, there is garbage in the house and the house is quiet; all other propositions are false.

Initial Conditions: (and(garbage)(cleanHands)(quiet))
Goal: (and(dinner)(present (not(garbage))))

Actions:
cook : precondition(cleanHands)
    : effect(dinner)
wrap : precondition(quiet)
    : effect(present)
carry : precondition[ ] garbage[]
    : effect(and(not(garbage))(not(cleanHands)))
dolly: precondition[ ] garbage[]
    : effect(and(not(garbage))(not(quiet)))

Mutex[ ] cook is interfered by carry[]
wrap is interfered by dolly[]

~quiet and present are inconsistent supports.

Carrying out the Graphplan algorithm above, we will gain the Planning Graph of dinner-date problem as follows (see Fig. 3). In contrast to it, carrying out our algorithm above, we will gain the output of dinner-date problem as follows.

The proposition set with label member “1” is {garb, cleanH, quiet}; The action set with label member “1” is {carry, dolly, cook, wrap}; The proposition-proposition mutex set with label member “1” is {(¬cleanH, dinner), (¬quiet, present)}; The proposition set with label member “2” is {(¬garb, ¬cleanH, ¬quiet, dinner, present)}; The proposition-proposition mutex set with label member “2” is {(garb, ¬garb), (cleanH, ¬cleanH), (quiet, ¬quiet)}; The action-action mutex set with label member “2” is {(nop-garb, carry), (nop-garb, dolly), (carry, nop-cleanH), (carry, cook), (dolly, nop-quiet), (dolly, wrap)}.

Then, Carrying out any of Solution Exaction of Graphplan and our algorithm above, we will achieve the goal and gain one of valid plan as follows.

The valid plan[ ]
cook [] carry[] wrap[]
cook [] wrap[] carry[]
wrap [] cook[] carry[]
wrap [] dolly[] cook[]
cook [] wrap[] dolly[]
wrap [] cook[] dolly.
Fig. 3. The Planning Graph of dinner-date problem. Names of the actions are surrounded by boxes, and horizontal lines between proposition levels represent nops.

Now, leaving the two algorithms alone, we can transform between Fig. 3 and the output of the Generation of Label Member algorithm. In fact, it shows our algorithm has equivalent effect with Graphplan, the process of constructing the planning graph corresponds to Generation of Label Member. At the same time, we can see the memory space of our algorithm is much smaller than Graphplan’s.

7 Conclusion and Future Work

Nowadays, intelligent planning is a very hot branch in AI, and many highly efficient planning methods have been developed. The most remarkable approach in them was Graphplan. In this paper, we intensively analyze the structure of planning graph at first and introduce a novel planning algorithm based on label field in the following. The algorithm maintains the strength of Graphplan, possesses some preeminent properties, and has excellent performance in terms of time and space. However, to apply the algorithm to real world fields better, we need work on it further. Recently, representative planners such as Blackbox\(^6\) F\(F\)^7 are the combination of several planning methods. We are considering combining our algorithm with other planning methods (such as heuristic approach), exploring the new algorithm and developing the planner with higher performance. In addition, since the problems handled by our algorithm are limited in STRIPS domain, it is one of our meaningful future jobs to extend action representations to ADL\(^{11}\) and PDDL\(^{12}\) domain.
Acknowledgements

The work described in this paper is sponsored in part by the National Nature Science Foundation of China grant 60473042. The authors are grateful to the anonymous reviewers for providing their detailed, thoughtful and helpful comments on improving the work presented here.

References