International Journal of Business, Human and Social Sciences
ISSN: 2517-9411
Vol:1, No:4, 2007

Creating or Destroying Objects Plan in the Graphplan Framework

Wen-xiang Gu, Zeng-yu Cai, Xin-mei Zhang, Gui-dong Jiang

***Abstract*—** At present, intelligent planning in the Graphplan framework is a focus of artificial intelligence. While the Creating or Destroying Objects Planning (CDOP) is one unsolved problem of this field, one of the difficulties, too. In this paper, we study this planning problem and bring forward the idea of transforming objects to propositions, based on which we offer an algorithm, Creating or Destroying Objects in the Graphplan framework (CDOGP). Compared to Graphplan, the new algorithm can solve not only the entire problems that Graphplan do, but also a part of CDOP. It is for the first time that we introduce the idea of object-proposition, and we emphasize the discussion on the representations of creating or destroying objects operator and an algorithm in the Graphplan framework. In addition, we analyze the complexity of this algorithm.

***Keywords*—**Graphplan, object_proposition, Creating or destroying objects, CDOGP.

I. Introduction

The intelligent planning is an important method of artificial intelligence research and has been applied to many fields. Specifically, the Graphplan through planning graph analysis [1],[2], which made revolutionary progress in the intelligent planning, arouse extensive concerns and study. Furthermore, people have obtained many achievements from it, which include conditional effects in Graphplan [3], probabilistic planning in the Graphplan framework[4], uncertain planning[5] and goal recognition through graph analysis [6],and so on.

However, so far CDOP brought forward by Blum and Furst hasn't been solved. There are a great many of such planning problems in the real world. With the development of intelligence planning in the Graphplan framework, these problems get more serious, and have became a bottleneck of the progress of the AI planning. In this paper, we thoroughly study this problem and introduce a method to solve it in the Graphplan framework, which can solve the CDOP problem through making a few changes to the Graphplan. Hence, we can apply the results of Graphplan to the CDOP easily.

II. Definitions and Notations

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

Manuscript received December 10, 2004.This work described in this paper was fully supported by the National Nature Science Foundation of China under grant 60473042.

Wen-xiang Gu is with Department of Computer Science of Northeast Normal University, Changchun, Jilin, China(corresponding author, e-mail: gwx@nenu.edu.cn).

Zeng-yu Cai is with Department of Computer Science of Northeast Normal University, Changchun, Jilin, China(e-mail: caizy767@nenu.edu.cn).

McDermott and James Hemdeler think a *plan* is devising the sequence of actions for an agent [9] .We generally think a *plan* is a set of actions that will achieve the goals of a problem.

A planning *problem* **consists** of:

(1) a STRIPS-like domain(a set of operators),
(2) a set of objects,
(3) a set of propositions(literals) called the initial conditions,
(4) a set of problem goals which are propositions that are required to be true at the end of a plan.

No-op: a special kind of action that does nothing to a proposition at time step *i*, whose add effects is the same proposition as its preconditions.

Planning graph: a planning graph is a directed, leveled graph with two kinds of nodes and three kinds of edges. The two kinds of node are *proposition nodes* and *action nodes*. The action nodes express common action appearing at some time step, and proposition nodes express common proposition appearing at some time step. The levels alternate between proposition-level containing proposition nodes and action-level containing action nodes.

III. The Classifications of CDOP

Here, we divide the CDOP into two types:

(1). The attributes of new objects created in planning are unpredictable.

Before planning starting, we can't know about what kind of objects will be created, or the attributes of new objects in some planning problems. For example, an operator is to mix two kinds of chemical liquids together. In the planning, a new substance will be produced, but we know nothing about it before planning starting.

(2). The attributes of new objects created in a planning are predictable

That is to say, before planning starting, we know about what kind of objects will be created, and the attributes of new objects in some planning problems. In fact, in the operator set there exists this kind of operators, the type of whose parameters will not exist in the initial object set. For example, in a transportation planning, given a train head and a train tail, we can know that it is possible to create a train by linking them together. Moreover, we can predict the train can be loaded with cargo and move before planning starting.

Because the majority of planning problems in the reality are domain-specific, we can know what kind of objects will be created in a planning, which belongs to the second type stated above. We primarily discuss it in this paper.

IV. The Representation of the Operator in the CDOGP

A. The type and representation of the operator

We divide the operators into two types: common operator and creating or destroying objects (CDO) operator.

Definition 1: An operator creating or destroying objects is called *creating or destroying objects (CDO) operato*r. The action instantiated from it is called CDO action.

Definition 2: The operator other than CDO operators is called *common operator*. The action instantiated from them is called common action.

Representations of operators are demonstrated as follows.

Common operator:

Operator {
: The set of parameter {…}
: The set of precondition {…}
: The set of effect {the effect propositions}
}.

CDO action:

Operator {
: The set of parameter {{ the set of common parameter},{ the set of decisive parameter } }
: The set of precondition { …}
: The set of effect {{the set of object added},{ {{ the set of object destroyed },{ the effect propositions }}
}.

B. The restrictions to creating objects operator

(1). In general, a CDO operator destroys some objects when creating new objects, and a new object can be known what kind of it is according to some parameters of creating it. Therefore, we assume that the new objects can be determined by some parameters in all the creating objects actions. And we call these parameters *decisive parameters*. For instance, in a planning a creating objects operator is instantiated twice, and the decisive parameters are all the same, so we can make sure that the objects created by the two actions are exactly the same.

(2). The restrictions to the initial operator set

Assume that all the creating objects operators are in the set A (A={op1, op2, op3, …op n}). We restrict the set A as follows:

The type of any parameter of an operator mustn't be the same as that of the object it creates, we call it avoiding the recursion call of creating objects operator.

In set A, there doesn't exist an operator sequence, op i_1, op i_2, op i_3, …op i_m. The type of a new object created by op i_j is a decisive parameter of op i_{j+1}, and the type of a new object created by op i_m is a decisive parameter of op i_1. We call it avoiding the recursion of creating objects operator through loop call.

V. The Concept of Transforming Object to Proposition

In the CDOP, some actions can create new objects or destroy existing objects. Therefore, the number of objects will change during the planning. That is, the object is dynamic. In addition, in the process of a planning, a certain operator possibly destroys some objects when creating new objects. Hence, the interaction between objects is also one feature of the objects.

The propositions in Graphplan have the same features as the objects in the CDOP analyzed above, and the sophisticated methods to handle propositions are available. Therefore, we can apply them to objects. This is just the source of the idea of transforming object to proposition. The method is: we use a proposition to indicate that an object exists at some time step, and the usual form is *alive+ "object name"*. For example, *alive C* indicates that an object C exists at some time step. So we can divide the proposition into two types: *common proposition* and *object proposition*.

Definition 3: the *object-proposition* is the proposition to express whether an object exists at some time step.

Definition 4: the rest of the proposition is the *common proposition, which* usually expresses the attribute or the state of the object.

Simple as the idea is, we will find out its great advantages later. Because of the introduction of it, we can treat object as proposition and handle their dynamic change and mutual interactions conveniently. Furthermore, we can apply the results of Graphplan to CDOGP easily without many changes.

VI. CDOGP: Creating or Destroying Objects in the Graphplan Framework

In this paper, we try to solve the planning problems involving finite objects and new objects predictable, which belong to the second type of CDOP problem in section 3. The main thought is making the Graphplan handle CDOP problems through appropriate changes.

A. The description of CDOP problem

The CDOP problem, similar to the Graphplan, consists of:

(1) A set of operators, including the common operator and CDO operator;
(2) A set of objects;
(3) A set of proposition called the initial condition;
(4) A set of proposition called goals;

B. The graph of CDOGP

The planning graph of CDOGP is similar with the Graphplan's, also a directed, leveled graph with two kinds of nodes which are *proposition nodes* and *action nodes* and three kinds of edges. The action nodes represent common actions or CDO actions, and the proposition nodes represent common propositions or object propositions. The levels alternate between proposition levels containing proposition nodes and action levels containing action nodes. The three edges represent relations between actions and propositions. Similar with the Graphplan, The precondition edges are to connect the actions to their preconditions in the previous proposition-level, and the add edges are to connect the actions to their add effects in the next proposition level including common propositions and object propositions. Moreover, the delete edges are to connect the actions to their delete effects in the next proposition level including common propositions and

object-propositions.

C. Exclusion relations among planning graph nodes of CDOGP

1) Exclusion relations among objects

Definition 5: At proposition level t, two objects of A and B are *mutually exclusive* in a planning graph, if no valid plan could possibly contain both A and B at that proposition-level.

The method of judging exclusion relations among objects:

If the actions creating A and all the actions creating B that are marked as mutually exclusive of each other in the previous action-level, A and B will be mutually exclusive.

2) Exclusion relations among action nodes

Definition 6: two actions A and B at action-level t are *mutually exclusive* if no valid plan could possibly contain both actions A and B at that action-level.

The method of judging exclusion relations among actions:

(1) If one precondition of action A and one precondition of action B that are marked as mutually exclusive in the previous proposition-level;
(2) If an action deletes a precondition or an add effect of another;
(3) If an action destroys an object, whose name exists at the preconditions or add effects of another.

Under any circumstance above, two actions will be marked mutually exclusive at the current level.

3) Exclusion relations among propositions

Definition 7: At proposition-level t, two propositions of P and Q are *mutually exclusive* in a planning graph if no valid plan could possibly contains both P and Q at that proposition-level.

The method of judging exclusion relation among proposition nodes:

(1) If any action achieving P and any action achieving Q are marked as mutually exclusive of each other in the previous action-level;
(2) If two object-propositions are mutually exclusive at the current level, and the two objects' name appears at P and Q respectively;

Under either of circumstances above, two propositions will be marked mutually exclusive.

D. The algorithm of expanding the planning graph

(1). Transforming the objects in the initial state into object-propositions. The algorithm starts with a planning graph that only has a single proposition level containing the initial conditions and object-propositions of initial objects. Guarantee that no pair of propositions is mutually exclusive.

(2). We suppose that the proposition-level n has been obtained completely. If it contains all the goals and no pair of the propositions in the goal set is exclusive, the algorithm turns to searching valid plan, else if the proposition level n is exactly the same as the proposition level n-1 and the same exclusion relations, returns no valid plan, else turns to the next step.

(3). For each operator and each way of instantiating preconditions of that operator to propositions in the previous level, it inserts an action node if no pair of its preconditions are marked as mutually exclusive. Insert all the no-op actions and the precondition edges. Create a generic proposition level , add all the effects of the actions in the previous level, and transform the new objects created by the action in previous level to object-propositions. Add the object-propositions that are not in current proposition-level to the current proposition-level. Connect them to their preconditions with add edges and delete edges. Last, the algorithm creates an "acions_that_I_am_exclusive_of" list for each action in level n and a "propositions_that_I_am_exclusive_of" list for each proposition in level n+1.

E. The algorithm of searching for a valid plan

(1). For each subgoal at time step t, select some actions at time step t-1 achieving these goals that are not exclusive of any action that has been selected, and continue recursively with the next goals at time step t-1. If the recursive call returns failure, the algorithm tries a different action achieving the current goal and so forth. It returns failure once all the actions have been tried. Upon finishing with all the goals at time step t, the preconditions of the selected actions make up the new goal set at time step t-1, and then it continues this procedure at time step t-1.

(2). If the goals at the current level are a subset of the initial conditions, we get the valid plan as the actions been selected. If it reruns failure at the previous level, it backs up right way. If it still returns failure at the last level *t*, there is no valid plan in *t* steps, then it turns to then expanding planning graph algorithm.

It is similar with the Graphplan[2].

F. Terminating on unsolvable problem

According to the algorithm described in section 7, it is obvious that the number of propositions is finite in any proposition-level. If the propositions in *proposition-level n* are exactly same as the propositions *in proposition-level n-1*, and they have the same exclusion relations. We call the planning graph "level off ". If there is no valid plan at time step n, the problem is unsolvable. Please read the reference [2] for the detailed proof.

VII. The analysis of CDOGP

A. Space needed

Theorem 1: *Consider a planning problem with n objects, p propositions in the initial conditions, and m operators each having a constant number of formal parameters. Let L be the sum of the number of the objects and the propositions added by any operators. The new objects created in planning are no more than N. Then, the size of a t-level planning graph created by CDOGP, is polynomial in n, p, m, L, t, and N.*

Proof. Let k be the largest number of formal parameters in any operator. Since there are n objects in the initial state, and the new objects created in a planning are no more than N. The maximum number of different common propositions and object propositions that can be created by instantiating an operator is O ($L(n+N)^k$). Therefore, the maximum number of nodes in any proposition-level of the planning graph is O (p + L $(n+N)^k$). Since any operator can be instantiated in at most O ($(n+N)^k$) distinct ways. The maximum number of nodes in any

action level of the panning graph is O (m (n+N) ^{k}). Thus, the total size of the planning graph is polynomial in n, p, m, L, t, and N, since k is constant. □

B. Time needed

Intuitively, with the introduction of object proposition, our algorithm is similar with the Graphplan's. Therefore, the time needed by our algorithm is finite, too.

Theorem 2: *The time needed to create the graph is polynomial in the number of nodes of the planning graph.*

Proof. The times needed to create a new action and proposition level of planning graph can be broken down into (a) The time to instantiate the operator s in all possible ways to preconditions in the previous proposition level, (b) the time to determine mutual exclusion relations between actions, and (c) the time to determine the mutual exclusion relations in the next level of propositions. It is obvious that the time is polynomial in the number of nodes in the current level of the graph. So the time needed to create the graph is polynomial in the number of nodes of the planning graph.. □

C. Limitations

The purpose of this paper is to introduce a new method of solving the CDOP problem. We omit some details and don't utilize some optimizing techniques such as goal-driven planning. Therefore, our algorithm is not very efficient. Besides, we make some restrictions to the operators of our algorithm; as a result, some planning problems can't be solved. However, it is not serious. Because we aim at extending and improving Graphplan, and the less we make changes to the Graphplan, the easier we apply the existing results of Graphplan to the CDOP problems.

VIII. Conclusion

In this paper, we do much study on the CDOP problems and thorough analysis. We divide this planning problem into two types and give the concept of transforming object to proposition, through which we transform the reclusion relations among objects to propositions' successfully. We also offer an algorithm, CDOGP, in the Graphplan framework. Compared to the Graphplan, our algorithm can solve not only all the problems that Graphplan can do, but also part of the CDOP problems. As our work is still in the Graphplan framework, we can apply the results of Graphplan such as conditional effects, probabilistic planning, to CDOGP conveniently.

References

[1] Blum and M. Furst. Fast planning through planning graph analysis. In *Proc. 14th Int. Joint Conf. AI*, pages 1636--1642, 1995.

[2] Blum and M. Furst. Fast planning through planning graph analysis. *J. Artificial Intelligence*, 90(1--2):281--300, 1997.

[3] Anderson, C. R.Shimith,D.E., and Weld,D.S.(1998).Conditional Effects in Graphplan. In Simmons, R., Velose, M .and Shimith,S.(eds.),*Proceedings of the 4th International Conference on Artificial Intelligence Planning Systems (aips-98)*,pp.44-53.AAAIPress,melo Park..

[4] Avrim L. Blum and John C. Probabilistic Planning in the Graphplan Framework. *In the 5th European Conference on Planning (ECP'99)*.URL: http://www-2.cs.cmu.edu/~jcl/papers/planning/ecp.ps.

[5] D.S.Weld, C.R Anderson, and D.E.Smith. Extending Graphplan to handle uncertainty and sensing actions. *In AAAI98,1998.* URL: http://www.cs.washington.edu/ai/sgp.html

[6] Hong, J ..Graph Construction and Anlysis as a Paradigm for Plan Recognition, S*eventeenth National Conference on Artificial Intelligence, Austin, Texas, USA, Publisher: AAAI Press.*pp774-779,2000.

[7] R.E. Fikes and N.J. Nilsson, STRIPS: a new approach to the application of theoren proving to problem solving, *Artif. Intell.* 2(1971) 189-208.

[8] McDermott D,et al .Planning:What is, What it could be An introduction to the Special Issue on Planning and Scheduling. *Artificial Intelligence,*1995,76:1-16.

[9] Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for modeling time and metric resources. Technical report, University of Durham, UK, 2001. 61-124.

[10] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. 13th Nat. Conf. AI*, pages 1194--1201, 1996.

Wen-xiang Gu is with Department of Computer Science of Northeast Normal University, Changchun, Jilin, China(corresponding author, e-mail: gwx@nenu.edu.cn). Professor of computer science, school of Computer Science. 1970-1972 Department of Mathematics of NENU, student; 1972-1993 Department of Mathematics of NENU, teacher; 1993-date Department of Computer Science of NENU, professor.

Research Interests: Artificial intelligent planning method and intelligent user interface.

Projects undertaken: He projects from the Province Committee of Science and Technology, National Natural Science Foundation and Science and Technology Research from Ministry of Education.

Zeng-yu Cai received his bachelor degree in computer science Northeast Normal University in 2003, China. He is currently in the school of computer, Northeast Normal University, as a master. *Research Interests:* His search interests include artificial intelligent planning method, automata theory, and intelligent user interface. (e-mail: caizy767@nenu.edu.cn)