CONTINGENT PLANNING AS BELIEF SPACE SEARCH

Incheol Kim and Hyunsik Kim
Department of Computer Science, Kyonggi University, San 94-6, Yiui-Dong, Youngtong-Gu, 443-760, Kyonggi, Korea

Keywords: Contingent planning, Partial initial condition, Nondeterministic actions, Belief states, Heuristic search.

Abstract: In this paper, we present a new heuristic search algorithm for solving contingent planning problems with the partial initial condition and nondeterministic actions. The algorithm efficiently searches through a cyclic AND-OR graph with dynamic updates of heuristic values, and generates a contingent plan that is guaranteed to achieve the goal despite of the uncertainty in the initial state and the uncertain effects of actions. Through several experiments, we demonstrate the efficiency of this algorithm.

1 INTRODUCTION

Classical AI planning has the assumption that the entire initial state is known at planning time, and each action should deterministically produce an exact one outcome. A more realistic assumption is that the planner only knows part of this information at planning time, the rest must be acquired at plan-execution time through observations, and each action may nondeterministically produce any of several outcomes. Several planning algorithms have been formulated to address these partially observable, nondeterministic planning problems (Ghallab, et al., 2004). Contingent planning algorithms construct a plan that includes both sensing actions and conditional execution of the actions in the plan. In general, contingent planning can be considered as search in a belief space to find a solution plan of which all possible different execution paths beginning from the initial belief state should end at one of goal belief states. In this paper, we present a new heuristic search algorithm for solving contingent planning problems with the partial initial condition and nondeterministic actions. The algorithm efficiently searches through a possibly cyclic AND-OR graph with dynamic updates of heuristic values, and generates a contingent plan that is guaranteed to achieve the goal despite of the uncertainty in the initial state and the uncertain effects of actions. Through several experiments, we demonstrate the efficiency of this algorithm.

```
(:action sense_door_open
 :parameters (?R – robot  ?L1 ?L2 - location)
 :precondition (and (robot_in ?R ?L1)
                    (unknown_door_open_between ?L1 ?L2))
 :effect (or
       (and (not (unknown_door_open_between ?L1 ?L2))
            (door_open_between ?L1 ?L2))
       (and (not (unknown_door_open_between ?L1 ?L2))
            (door_closed_between ?L1 ?L2))))
(:action carry
 :parameters (?R - robot ?L1 ?L2 - location ?O - object)
 :precondition (and (robot_in ?R ?L1)
                    (door_open_between ?L1 ?L2)
                    (object_in ?O ?L1) )
 :effect (or  (and (not (robot_in ?R ?L1)) (robot_in ?R ?L2))
                                 (grap_object   ?R   ?O) (not
 (object_in ?O ?L1)))
              (and (robot_in ?R ?L1))))
(:action move
  :parameters (?R - robot ?L1 ?L2 - location)
  :precondition (and (robot_in ?R ?L1)
                     (door_open_between ?L1 ?L2))
  :effect (and (not (robot_in ?R ?L1)) (robot_in ?R ?L2)))
(:action put_down
  :parameters (?R - robot ?L1 - location  ?O - Object)
  :precondition (and (robot_in ?R ?L1) (grap_object ?R ?O))
   :effect (and (not (grap_object ?R ?O)) (object_in ?O ?L1)))
```

Figure 1: Domain actions.

2 CONTINGENT PROBLEMS

Planning with incomplete information can be formulated as a problem of search in belief space, where belief states can be sets of states.

In this paper, we use the corresponding meta-predicate formula, (*unknown_predicate term*$_1$, .., *term*$_n$) to represent that we do not know whether the

Kim I. and Kim H..
CONTINGENT PLANNING AS BELIEF SPACE SEARCH.
DOI: 10.5220/0003288906940697
In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence* (ICAART-2011), pages 694-697
ISBN: 978-989-8425-40-9

fact (*predicate term$_1$, ..., term$_n$*) is true or not. A belief state b is represented as a set of literals, including some *unknown* meta-predicate literals. A deterministic action a_d is described by a precondition and an effect. However, a nondeterministic action a_{nd} may have more than one effect. A sensing action $a_{s\text{-}l}$ has at least one unknown literal $l_{unknown}$ in the precondition and two different effects including l_{true} and l_{false} respectively. Figure 1 shows some action descriptions of the Robot Navigation domain.

The contingent planning problem with both partial initial condition and nondeterministic actions is $P_{pond} = (D, b_I, b_G)$ and the domain is D = (L, B, A), where L is the set of all known and unknown literals, B is the set of all belief states, and $A=A_d \cup A_{nd} \cup A_s$ is the set of actions, that includes deterministic, nondeterministic, and sensing actions. b_I, b_G are the respective initial and goal belief states. Figure 2 shows an example of contingent planning problem with partial initial condition.

```
(:init
   (robot_in robot l_corridor)
   (object_in cup office1)
  (unknown_door_open_between conf1 r_corridor)
  (unknown_door_open_between lounge r_corridor)
   (door_open_between l_corridor office1)
   (door_open_between office1 conf1)
   . . .
   (door_open_between reception r_corridor))
(:goal
   (and (robot_in robot reception) (object_in cup reception) ))
```

Figure 2: A planning problem.

A solution of the given contingent planning problem P_{pond} can be represented as a policy π, the set of (belief state, actions) pairs, as shown in Table 1.

Table 1: A solution policy.

Belief State	Action
(robot_in robot l_corridor) (object_in cup office1) (unknown_door_open_between lounge r_corridor) **...**	(move robot l_corridor office1)
(robot_in robot office1) (object_in cup office1) (unknown_door_open_between lounge r_corridor) **...**	(carry robot office1 l_corridor cup)
. . .	. . .
(robot_in robot lounge) (grap_object robot cup) (unknown_door_open_between lounge r_corridor) **...**	(sense_door_open lounge r_corridor)
. . .	. . .

3 HEURISTIC SEARCH

```
AND-OR-Search-Trial(b0)
Begin
  stateStack.CLEAR();
  policyTable.CLEAR();
  localHistory.CLEAR();
  stateStack.PUSH(b0);
  While ¬stateStack.EMPTY() do
  /* if there remain any unexplored branches */
    b = stateStack.POP();
    If b.GOAL() then
      /* b is a goal state on the end of one branch */
      solved = true;
      b.VALUE = 0;
      valueTable.UPDATE(b, b.VALUE);
      localHistory.CLEAR();
      continue;
    If localHistory.CONTAINS(b) then
      /* b is a cycle state, not be included in a solution */
      solved = false;
      break;
    else
      localHistory.PUT(b);
    If policyTable.CONTAINS(b) then
      /* b is already in a partial solution */
      localHistory.CLEAR();
      continue;
    a = b.GREEDY_ACTION();
    If a = NULL then
      /* b is a dead-end state */
      solved = false;
      b.VALUE = MAX_VALUE
      valueTable.UPDATE(b, b.VALUE);
      break;
    policyTable.PUT(b, a);
    If a is one of sensing actions then
      b' = b.P_NEXTSTATE(a);  /*a positive state*/
      b'' = b.N_NEXTSTATE(a); /*a negative state*/
      localHistory.CLEAR();
      stateStack.PUSH(b');
      stateStack.PUSH(b'');
      b.VALUE = cost(b,a) + 0.5*valueTable.GET(b')
                + 0.5*valueTable.GET(b'')
      valueTable.UPDATE(b, b.VALUE);
    else if a is one of non-deterministic actions then
      next_states = b.ND_NEXTSTATE(a);
      localHistory.CLEAR();
      For each b' in next_states do
        stateStack.PUSH(b');
      b.VALUE = cost(b,a) +
        $\sum_{b' \in next\_states}$ valueTable.GET(b') / |next_states|
      valueTable.UPDATE(b, b.VALUE);
    else if a is one of deterministic actions then
      b' = b.NEXTSTATE(a);
      stateStack.PUSH(b');
      b.VALUE = cost(b,a) + valueTable.GET(b')
      valueTable.UPDATE(b, b.VALUE);
End
```

Figure 3: AND-OR-Search-Trial.

We propose a heuristic search algorithm for solving contingent planning problems, called HSCP (Heuristic Search for Contingent Planning). It repeats multiple AND-OR-search trials beginning from the initial belief state until it obtains a complete solution (i.e., until *solved* becomes true), as shown in Figure 4.

```
HSCP(bs$_0$ : initial belief state)
Begin
  valueTable.CLEAR();
  solved = false;
  While solved = false do
    AND-OR-Search-Trial(bs$_0$);
  return (policyTable.GET_POLICY());
End
```

Figure 4: The HSCP algorithm.

Figure 3 summarizes the AND-OR search trial of the HSCP algorithm. During AND-OR search trial, just one candidate of solution subgraphs keeps to be expanded until its every tip node successfully arrives at one of goal belief states. Therefore, the AND-OR search trial is like the heuristic depth-first search that traverses along every AND branch of the solution subgraph. The initial value of a belief state b is assumed to be its heuristic value, *h*(b). Whenever a node is selected to be expanded, the value of the belief state in the node is updated with the values of its successors. However, HSCP does not backpropagate value updates over the entire subgraph. So the AND-OR search trial proceeds fast. When it meets a dead end state that has no successors, it sets the value of the state to the infinite number ∞. If the candidate subgraph is no longer expanded to be a complete solution, the search trial fails. And then a new search trial begins again with the updated value table. With the help of updated values of the states, the new search trial is able to expand another part of the AND-OR graph. As the number of search trials increases, the possibility to obtain a complete solution also increases. In this way, HSCP can find a suboptimal solution of the given contingent planning problem very efficiently.

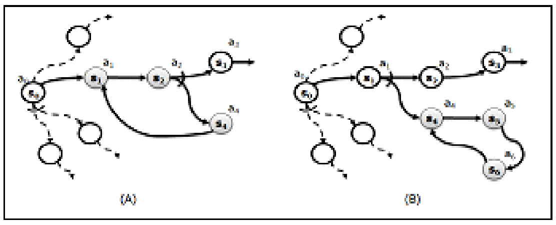

Figure 5: Open and closed cycles.

Our HSCP algorithm can distinguish open cycles from closed ones. The former has the possibility to arrive one of goal belief states, but the latter does not. (A) and (B) in Figure 5 show an open cycle and a closed cycle, respectively.

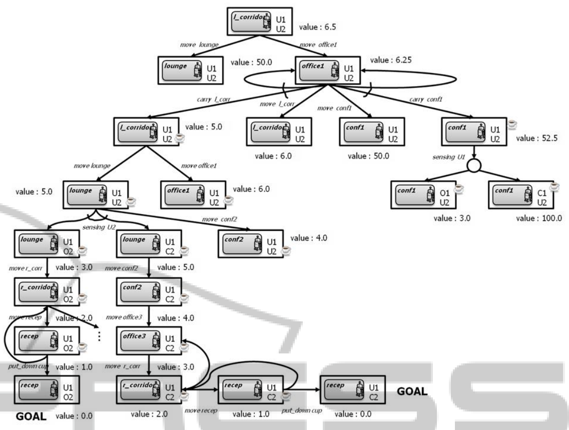

Figure 6: Search space expanded by HSCP.

Figure 6 shows the part of search space expanded by HSCP for solving the example problem of Figure 2.

4 RELATED WORK

There are several different algorithms for contingent planning problems: LAO*, RTDP, CondFCP, and ND-FCP. LAO*(Hansen and Zliberstein, 2001) is an extended version of AO*, the heuristic AND-OR search algorithm, so that it can find a solution subgraph containing open cycles. This algorithm obtains an optimal solution with just one search trial. However, during the search trial, it evaluates and expands multiple candidates of solution subgraphs simultaneously, and backpropagates value updates over the entire solution subgraph whenever it gets a new heuristic value of a tip node. Moreover, whenever it meets an open cycle, it continues tracing and value updating along the cycle until every state nodes in the cycle gets a converged value.

RTDP (Bonet and Geffner, 2003) is a well-known dynamic programming algorithm for solving nondeterministic contingent planning problems. It has two key advantages comparing with other DP algorithms: first, it obtains an optimal policy without computing the whole space, second, it has a good anytime behavior. However, RTDP usually requires a lot of search trials and value updates, so its convergence is slow. CondFCP(Kuter, et al, 2007) is an extended version of the classical forward-chaining algorithm for solving planning problems with partial initial condition. On the other hand, ND-FCP(Kuter and Nau, 2004) is an extended one for solving nondeterministic planning problems. Both algorithms do just one search trial and expand only one candidate of solution subgraph during the search trial. They do not try to update heuristic values with new information. Therefore, they do not guarantee to find a solution and cannot be used for solving contingent planning problems with both partial observations and nondeterministic actions.

5 EXPERIMENTS

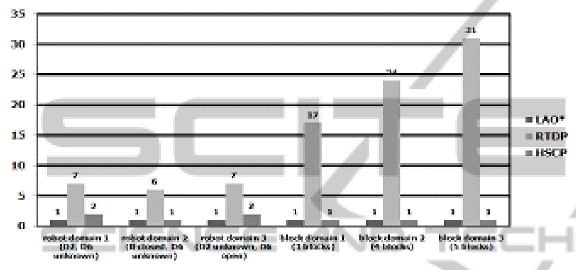

Figure 7: The number of search trials.

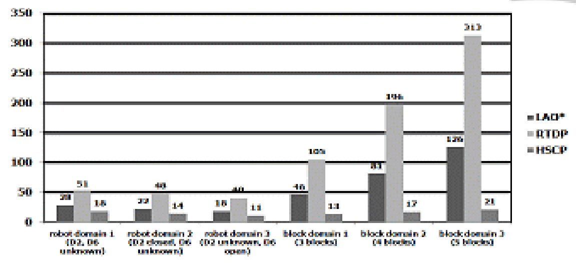

Figure 8: The number of value updates.

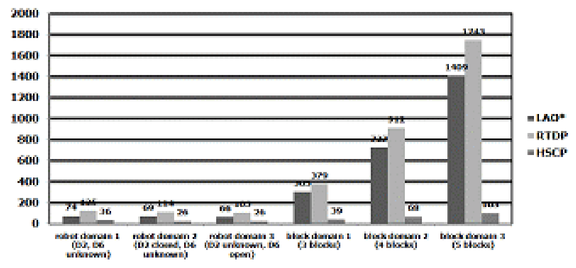

Figure 9: The number of generated states.

We implemented the HSCP algorithm, and compared it with LAO* and RTDP on two partially-observable, nondeterministic planning domains that are well-known from previous experimental studies: Robot Navigation and Blocks World. Three random problems were generated from each domain for experiments. We compared three different search algorithms in terms of the number of search trials, the number of value updates, and the number of generated states. Figure 7 ~ Figure 9 show the experimental results. While RTDP has performed lots of search trials to get the optimal values of states, our HSCP and LAO* have done just one or two trials for each problem. Out of three search algorithms, RTDP has tried value updates the most, but HSCP has done the least. Considering the number of generated states, we find out HSCP has explored much smaller search space than the other two algorithms.

6 CONCLUSIONS

We have presented a new heuristic search algorithm for solving contingent planning problems with the partial initial condition and nondeterministic actions. Through several experiments, we have evaluated the efficiency of this algorithm.

ACKNOWLEDGEMENTS

Industrial Strategic Technology Development Program (10032108), funded by the Ministry of Knowledge Economy(MKE, Korea).

REFERENCES

Bonet, B., Geffner, H., 2003. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *ICAPS'03, 13th International Conference on Automated Planning and Scheduling*. AAAI Press.

Ghallab, M., Nau, D., Traverso, P., 2004. *Automated planning: theory and practice*, Morgan Kaufmann.

Hansen, E., Zilberstein, S., 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, vol. 129, No. 1-2, pp. 35-62.

Hoffmann, J., Brafman, R., 2005. Contingent planning via heuristic forward search with implicit belief states. In *ICAPS'05*, *15th International Conference on Automated Planning and Scheduling*. AAAI Press.

Kuter, U., Nau, D., Reisner, E., Goldman, R., 2007. Conditionalization: Adapting forward-chaining planners to partially observable environments. In *ICAPS'07, 17th International Conference on Automated Planning and Scheduling*. AAAI Press.

Kuter, U., Nau, D., 2004. Forward-chaining planning in nondeterministic domains. In *AAAI'04*, *19th National Conference on Artificial Intelligence*. AAAI Press.