A Novel Heuristic Usage of Helpful Actions for Conformant-FF System

Wei Wei, Dantong Ouyang, Tingting Zou and Shuai Lu
College of Computer Science and Technology, Jilin University, Changchun, China
Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, China
Email: wei_wei10@mails.jlu.edu.cn, ouyd@jlu.edu.cn, zoutingt@163.com, lus@jlu.edu.cn

***Abstract*—Conformant planning is usually transformed into a search problem in the space of belief states, where the combinatorial explosion of search space has been one of the most intractable problems. In this paper, we present a novel usage of the helpful action pruning technique in the Conformant-FF planner. The key idea is to change the way it deals with helpful actions and first consider actions from the so-called implication path which was used by Conformant-FF for concluding which subgoal would be considered known to be true in the relaxed planning graph. We first point out the semantics of solving by cases, indicated by the implication paths of the relaxed planning process. In line with the semantics, we then propose our heuristic idea of using these implication paths further by attempting to collect certain groups of helpful actions such that executing all actions within a group can achieve some subgoal while executing an individual action in the group cannot due to incomplete information. This technique usually leads to the goal faster and cuts down the search space dramatically. We evaluate the idea experimentally. In a number of conformant benchmarks, our heuristic pruning technique outperforms helpful actions pruning in both planning efficiency and the size of search space.**

***Index Terms*—Helpful actions, Conformant-FF, Heuristic Pruning, Belief state**

I. INTRODUCTION

Planning is an area of Artificial Intelligence that studies choosing and organizing actions to achieve some objectives. Over the last few years we have seen a significant increase of the efficiency of planning systems [1, 2]. There are several promising approaches in plan generation including planning graph analysis [3, 4], planning as satisfiability [5] and heuristic search planning [6, 7, 8]. Classical planning refers to planning under a restricted model which is deterministic, static, finite and fully observable with restricted goals and implicit time. However, these assumptions are often unrealistic when modeling real-world tasks. For instance, the initial state may be incompletely specified, actions may have non-deterministic effects or the environment may be only partially observable. Several interesting models are obtained by relaxing some of the restrictive assumptions. Conformant planning is the problem of finding a plan that guarantees goal achievement given nondeterministic initial state or action effects, and no information can be observed at run time. A conformant plan is a sequence of actions that should be successful to achieve the goal regardless of uncertainty about the initial state and action effects. Therefore, conformant planning turns out to be considerably harder than classical planning [9]. Effective approaches for conformant planning include heuristic guidance [10, 11], translation in to classical ones [12], approximation-based planning and so on [13].

Heuristic search is one of the strongest trends in the planning community. We focus on the formulation that transfers a conformant planning task into a search problem in the belief state space. In this way, uncertainty about the true current world state is modeled via a belief state, i.e., the set of world states that we consider possible at this time. Then a heuristic function is derived from the specification of the planning instance and used for guiding the search through the search space. FF's heuristic function based on a relaxation of the planning task turns out to be the most successful idea in classical planning. Hoffmann and Brafman extended this technique to conformant planning and implemented the well known planner Conformant-FF [14]. In a number of benchmarks, Conformant-FF can provide very informative heuristic values and shows fine scalability.

Conformant-FF uses the same overall search arrangement as FF, with reasonable modifications for conformant setting. In Conformant-FF, belief states are represented by an implicit approach where the truth values of all the propositions in a belief state are determined using a CNF reasoning. FF's relaxed planning process is also extended to handle conformant problems. For each action effect the relaxation is to ignore the effect's delete list as well as all but one proposition of the effect's condition. After building the relaxed planning graph successfully, a relaxed plan is extracted and the length of the relaxed plan is used to provide the heuristic value of the belief state. When solving the relaxed planning task for a belief state, implication relation is maintained simultaneously to

Manuscript received July 1, 2012; revised October 6, 2012; accepted October 11, 2012.
Corresponding author: Dantong Ouyang

doi:10.4304/jcp.8.6.1385-1393

capture constraints between propositions at adjacent time steps. Implication paths in this machinery can be used to check the truth value of uncertain propositions when building the relaxed planning graph and to determine actions that must be inserted to the solution during relaxed plan extraction. Besides the estimate of goal distance, Conformant-FF uses a pruning technique of helpful actions selecting a set of promising successors to each search node.

In conformant setting, we observe that some branches of the search tree are usually non-independent with each other when expanding a belief state. All the actions on these branches will be involved into the final plan in different search iterations. In this paper, we recognize such branches and consider them as a group. With that we propose a powerful pruning technique suggested by implication paths when solving the relaxed planning task. Briefly, those actions which are inserted into the relaxed plan by implication paths at the first time step concern the unknown propositions of the current belief state and thus permit to remove the uncertainty. We execute these actions in sequence with higher priority than other regular helpful action provided by Conformant-FF.

Our novel usage of helpful actions is useful in reducing the degree of uncertainty about the current belief state and get closer to the goal quickly. We run experiments to evaluate our idea. In a number of conformant benchmarks, the experimental results show that our pruning technique can get a much smaller search space than before and improves the original Conformant-FF system in both planning efficiency and the size of search space.

The paper is organized as follows. In Section 2 we briefly describe the conformant planning framework we consider and give an overview of Conformant-FF's architecture. In Section 3 we characterize the semantics of the implication paths in Conformant-FF firstly. Then we propose our idea of pruning based on the usage of helpful actions, and illustrate the enforced hill-climbing procedure adopting this pruning technique. Section 4 gives the experimental results and our analysis. We conclude the paper in Section 5.

II. BACKGROUND

A. Conformant planning problem

The conformant planning problem considered in this paper extends a subset of the ADL language with uncertainty about the initial state. The extensions to handle uncertainty about effects are conceptually straightforward by taking account of the non-deterministic effects when computing state transitions.

Definition 1 (*Conformant planning problem*)

A conformant planning problem P is a triple (A, I, G) where A corresponds the action set, I is a propositional CNF formula denoting the possible initial world states and G is an non-empty set of propositions defining the goal conditions.

The initial situation is a belief state represented by a propositional CNF formula I. Any world state that satisfies this formula is a possible initial state. We use S_I to denote the initial belief state. An action a is a pair of $(pre(a), E(a))$ where $pre(a)$ is a set of propositions representing the preconditions and $E(a)$ is a set of conditional effects. A conditional effect e is a triple $(con(e), add(e), del(e))$ that correspond to e's condition, add list and delete list respectively. An action a is applicable in a world state w if $pre(a) \subseteq w$, i.e., all of a's preconditions are satisfied in w. If action a is applicable in w, then all conditional effects $e \in E(a)$ that satisfies $con(e) \subseteq w$ get executed. Executing a conditional effect e results in the world state w- $del(e) \cup add(e)$.

Definition 2 (*Conformant plan*)

An action sequence $act \in A^*$ is a conformant plan for problem P if, no matter what initial world state one starts from, all actions in act are applicable at their point of execution and the associate run results in a goal state.

B. Conformant-FF system

Conformant-FF system transforms a conformant planning problem into a search problem in belief state space. A belief state S is represented by the initial belief state formula together with the action sequence that leads to S. For each belief state encountered during search, the sets of known, negatively known and unknown proposition are computed. Given a conformant planning problem, a belief state S reached by an action sequence act and a proposition p, p is known in S if p is contained in the intersection of the worlds in S, i.e., p is always true after executing act and equally p is negatively known in S if p is always false after executing act. A proposition p is unknown in S if it is neither known nor negatively known. The truth value of a proposition in a belief state can be computed by a SAT solver checking the CNF formula that captures the semantics of the respective action sequence.

Conformant-FF's overall architecture is identical to FF system, illustrated in Fig. 1. The basic search algorithm is enforced hill-climbing which combines local and systematic search. Starting from current search state S, enforced hill-climbing algorithm performs a breadth-first search for a better state S' such that $h(S') <$

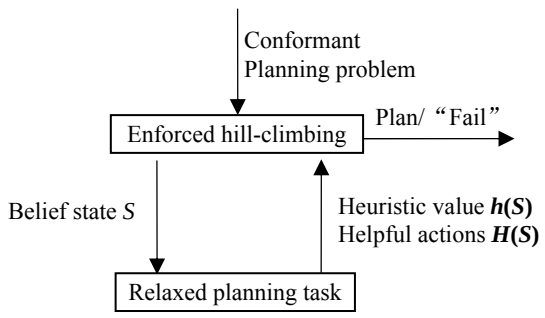

Figure 1. Conformant-FF's architecture

$h(S)$. Here, $h(S)$ denotes the heuristic value, and $H(S)$ denotes the helpful actions, which are the considered actions when expanding S. If a state with lower heuristic value was reached, then $S= S'$, else the hill-climbing fails, a complete best-first search is invoked to solve the problem from scratch.

III. The Heuristic Technique of Helpful Actions

During search, the considered successors of a search state are generated by the helpful actions in $H(S)$. In classical planning, $H(S)$ is the set of actions at the first level of the relaxed planning graph that add a subgoal of this level. In the conformant setting, $H(S)$ is the set of such actions at the first level of the relaxed planning graph, plus those actions that are selected for an implication path during relaxed plan extraction.

In this section, we propose our heuristic idea of helpful actions pruning, which can be used to reduce the search space further in the enforced hill-climbing procedure of Conformant-FF. The idea is derived according to the semantics of the relaxed planning graph and the implication paths. For the purpose of exposition, we first point out the relation between the implication paths of the relaxed plan and the uncertain aspect of the estimated belief state.

A. Semantics of implication paths

Consider a simplified example from the Blockworld domain.

Example 1 There are four blocks, *b1*, *b2*, *b3* and *b4*. Initially *b2*, *b3* and *b4* are on the table and *b1* is on *b2* or *b3*. The initial situation of *b1* is not known, modeled as *oneof*((*on b1 b2*), (*on b1 b3*)). The goal is (*on b1 table*) and (*on b4 b1*). We have a simple (*move ?b ?from ?to*) action that can change the location of the blocks. The graphical sketch of this example is given in Fig. 2.

To get the heuristic value, the relaxed task starting from the initial belief state is solved. In Fig. 3 we give the conformant relaxed planning graph built for Example 1. Proposition layers $P(t)$ and action layers $A(t)$ are built alternatively. Propositions on the dashed area are unknown at their respective layers. Dashed lines denote implication edges between two adjacent proposition layers and empty actions are represented by dots.

We find that $A(0)$ only includes actions given by implication edges plus some empty actions NOOP. The implication edges yielded by $A(0)$ are:

(*move b1 b2 table*): (*on b1 b2*)(0)→(*on b1 table*)(1)
(*move b1 b3 table*): (*on b1 b3*)(0)→(*on b1 table*)(1)
(*move b1 b2 b4*): (*on b1 b2*)(0)→(*on b1 b4*)(1)
(*move b1 b3 b4*): (*on b1 b3*)(0)→(*on b1 b4*)(1)
NOOP: (*on b1 b2*)(0)→(*on b1 b2*)(1)
NOOP: (*on b1 b3*)(0)→(*on b1 b3*)(1).

Here, timed implication edges represent that the truth values of some propositions at layer t are uncertain and usually depend on their truth values at layer t-1.

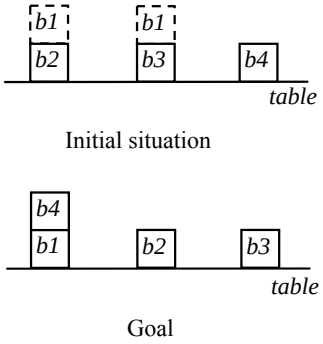

Figure 2. An example of Blockworld

Relaxed plan extraction starts with $G_2(S_I)$={(*on b1 table*), (*on b4 b1*)}. The subgoal (*on b1 table*) is inserted at layer 1 of its first appearance. (*on b1 b4*) is not a subgoal since it does not contribute to achieve the goal. However, there is no action in $A(0)$ that can guarantee achievement of (*on b1 table*). According to implication edges, to make (*on b1 table*) true at layer 1, we have to check the truth value of (*on b1 b2*) or (*on b1 b3*) at layer 0. In Conformant-FF, the implication edges of subgoal $g(t)$ form an *Imp* tree in the relaxed planning graph between the lowest layer and layer t. For the subgoal $g(t)$ that is not proved to be true at t, it is checked if the current state formula implies the disjunction of the leafs in the *Imp* tree, where the leafs are all the propositions in the current belief state whose truth values determine the truth value of g at layer t. *Impleafs*($g(t)$) is the set of such proposition leafs that are reachable from $g(t)$ and then *min_Impleafs*($g(t)$) is computed to be the minimal subset of *Impleafs*($g(t)$). For this example, we get *min_Impleafs* ((*on b1 table*)(1))={(*on b1 b2*)(0), (*on b1 b3*)(0)} whose disjunction is obviously implied by the initial state formula. Then we know that (*on b1 table*) can be achieved by (*move b1 b2 table*) or (*move b1 b3 table*). These two actions form the implication paths from *min_Impleafs* ((*on b1 table*)(1)) to (*on b1 table*)(1). To guarantee (*on b1 table*) become true at layer 1, both of the two actions are selected into the relaxed plan. Also these are collected as helpful actions. Since (*on b1 b4*) is not a necessary subgoal at layer 1, other actions of $A(0)$ are not considered as helpful.

For a subgoal g at layer t, sometimes there does not exists any supporting action that guarantees to always achieve g, i.e., any action with an effect that adds g and whose condition is known to hold at layer t-1. This complicated situation should trace back to the uncertain initial state. Intuitively, g has been shown that at least one possible world state could make it true. To achieve g, implication paths for g are determined from layer t going downwards to layer 0. All the actions that are responsible for the implication paths are selected at the respective times. *min_Impleafs*($g(t)$) is computed by a back chaining loop over the implication edges ending in $g(t)$ and it must be a subset of unknown propositions at layer 0, which corresponds to the uncertain aspect of the estimated belief state. Checking out the disjunction of the

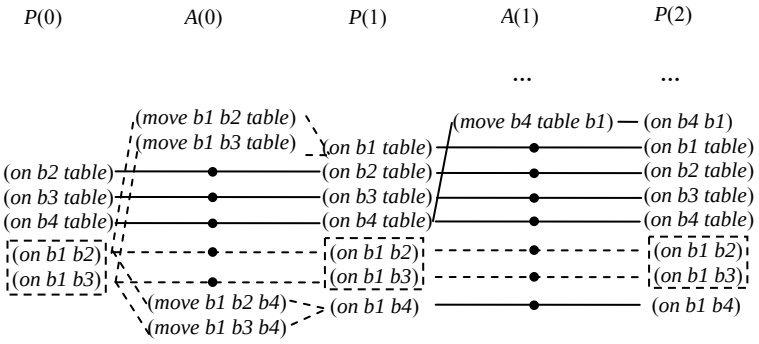

Figure 3. The relaxed planning graph for the initial belief state

min_Impleafs($g(t)$) against the current state formula, g can be proved to be always true at layer t in the relaxed planning procedure. Each implication path from $g(t)$ to one of its leaf corresponds to an action sequence that is executable in a particular current world state. Trying out all these actions in the current belief state is the only way to make g to be true in despite of the uncertainty arisen from initial state. This way, we remark that implication paths capture a form of solving by cases.

In Example 1, as the situation of *b1* is unknown, implication paths in the relaxed plan indicate that action (*move b1 b2 table*) achieves (*on b1 table*) if (*on b1 b2*) is true in the initial world state, otherwise action (*move b1 b3 table*) achieves (*on b1 table*). Since *b1* is only at one of the two locations, executing these two actions one by one could guarantee to make *(on b1 table)* true. Namely, given uncertainty about the initial state, to solve the task, a plan must be able to solve each possible initial world state.

B. Pruning of helpful implication paths

In line with our analysis of implication paths, we give our changing on the usage of helpful actions during search. To illustrate the algorithm, let us reconsider Example 1. After solving the relaxed planning task, we get the heuristic value supplied by the relaxed plan and collect the helpful actions. This information helps to lead the search procedure. The helpful actions here are (*move b1 b2 table*) and (*move b1 b3 table*), given by implication paths at the lowest level of the relaxed plan. These are the restricted choices to expand the estimated belief state. We get two successors by expanding the belief state with these actions. Enforced hill-climbing will perform an exhaustive search to select a better successor. In this example we find out that both successors generated by these two actions can give better heuristic evaluations. Thus to reach the subgoal (*on b1 table*) for sure, search is iterated starting from the intermediate state and the other action is also involved into the final plan.

Based on the solving by cases semantics of implication paths, we pursue the idea of considering implication paths for some subgoal as a group of actions. The observation that forms our basic idea is obtained from the back chaining process to determine implication paths during the extraction of relaxed plan. Actually the implication paths in the relaxed plan are back chained exactly to the uncertain part of the estimated belief state and branches of the implication edges are used to deal with different possible world states in current belief state. To construct a conformant plan, all the actions that refer to uncertainty about the current belief state are necessary.

Like helpful actions, we restrict to actions that are part of the implication paths and are at the lowest layer, i.e., those that could be select to start the relaxed plan.

Definition 3 Given the current belief state S, suppose *rplan* is the extracted relaxed plan, $H(S)$ is the set of helpful actions and *min_Impleafs*($g(t)$) is the determined minimal subset of leafs for subgoal g of step t, the set of implication helpful actions achieving subgoal $g(t)$ is defined as follows:

$imp\_H(g(t), S)=\{a \mid a \in H(S)$ and one conditional effect of a is responsible for an edge in *rplan* from *min_Impleafs*($g(t)$) to $g(t)\}$.

We call actions from $imp\_H(g(t), S)$ to be responsible to achieve the subgoal $g(t)$. Selection of $imp\_H(g(t), S)$ is done as a set union operation to avoid any superfluous action. We integrate these actions into an action sequence in an arbitrary order and use this sequence to expand the current search state. However, before making the action sequence, we have to address the problem of the interference between actions. In relaxed planning graph, delete effects of actions are ignored and the interference situation is not visible. Here we focus on the situation that the effect on an edge of the implication path may delete preconditions of action that has an effect on another edge. Entailment of an action precondition is checked against the known propositions, so deleting of one precondition would cause that the action does not work at its point of the sequence. We observe that conditional effects on the implication edges of the first layer refer to different world state. These effects do not interfere with each other generally since the possible current states are usually mutually exclusive. We call $imp\_H(g(t), S)$ to be executable if it can be made into a sequence and each action is executable at the respective time. As $imp\_H(g(t), S)$ is selected from implication paths for subgoal g, to distinguish from the

regular helpful actions, we call them helpful implication paths for $g(t)$.

Definition 4 Given the belief state S, the set of helpful implication paths to S is defined as follows:

$HIP(S)=\{ imp\_H(g(t), S) \mid g$ is a subgoal at time step t of the extracted relaxed plan and $imp\_H(g(t), S)$ is executable$\}$.

What we do in principle is to recognize and pick a group of actions that contribute to achieve a subgoal out of helpful actions. The notion of helpful implication paths shares some similarities with helpful actions. In a nutshell, action sequences in the set of helpful implication paths are usually made up of several helpful actions, which are deemed to have more potential for reaching a state with a much lower heuristic value. Thus helpful implication paths take precedence over other helpful actions during expanding. The enforced hill-climbing procedure that adopts the pruning technique is specified in Fig. 4. The input of the procedure is the planning problem <*A*, *I*, *G* >. The procedure outputs the *plan* if the goal is achieved successfully, otherwise it returns "Fail" if the enforced hill-climbing can not get any better state before the goal state.

The search procedure starts out in the initial belief state. Facing an intermediate search state S, the Search_for_better_state() procedure is invoked. During search, states are kept in a queue. In each iteration process, the first state is removed from the queue and evaluated. If the evaluation is lower than the current heuristic value, the iteration gets a better state. Otherwise, the removed state is expanded. Our implementation puts the successors generated by $HIP(S)$ in the queue with higher priority beside the usual successors obtained by other regular helpful actions.

```
Procedure EhcSearch-HIP(<A, I, G >, plan)
1   Initialize S to be the initial belief state;
2   Initialize the search queue U to be empty
3   plan= <>;
4   while h(S) != 0 do
5       Put S in the queue U;
6       if (Search_for_better_state(S, h(S), S′, h(S′))) then
7           add the path from S to S′at the end of plan;
8           S = S′;
9           reset U to be empty;
10      else
11          output "Fail", stop;
12      endif
13  endwhile

Procedure Search_for_better_state(S, h(S), S′, h(S′))
14  while (true) do
15    Remove the first node S′in the queue;
16    if h(S′) < h(S) then
17        return true;
18    endif
19    Collect HIP(S)
20    for each action sequence p in HIP(S)
21        put the state generated through p to the end of U;
22    for each of other regular helpful action a
23        put the successor generated by a to the end of U;
24  endwhile
```

Figure 4. Enforced hill-climbing with helpful implication paths pruning

Our implementation cuts down the branching factor further. Usually an action sequence from helpful implication paths could reach a search state a few steps away with a strictly better heuristic evaluation, which would be reached in several iterations when using helpful actions only. Thus, helpful implication paths offer some short cuts to better states. This way, the number of evaluated states is also reduced because only the two states that are at the beginning and end point of the action sequence need to get evaluated and all the intermediate states on this path are ignored. To take everything considered, we just leave other helpful actions behind and in this way our idea appears to be a tie breaking preference for the search procedure.

In the above example, the two actions (*move b1 b2 table*) and (*move b1 b3 table*) come from helpful implication paths for the subgoal (*on b1 table*)(1) of the relaxed plan. Fig. 5 illustrates our implementation of expanding process. Grayed parts are computations that are saved by our pruning. The two helpful actions are integrated into an action sequence to expand the initial belief state S_I. A better belief state S_3 is generated immediately by this sequence. Our pruning cuts down the branching factor from two to one and reaches the subgoal (*on b1 table*) within one search iteration, which will be accomplished by the normal helpful actions in two iterations. Moreover, the state evaluation of S_1 is avoided.

As our specific implementation collects helpful implication paths amongst the helpful actions, we can get the helpful implication paths totally for free, as another important side effect of the basic heuristic method, more strictly, as a side effect of the helpful actions for conformant setting. Moreover, just like helpful actions, helpful implication paths pruning do not preserve completeness as well, so the pruning technique is only

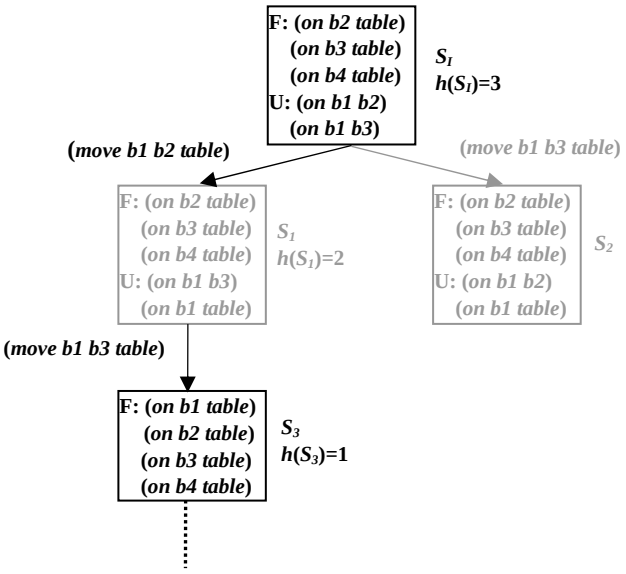

Figure 5. The expanding procedure for the initial belief state

adopted in the enforced hill-climbing search, which is not complete anyway, and the complete best-first search procedure is unchanged. In all of our test domains, the tasks can be solved successfully in the enforced hill-climbing procedure with our new pruning technique.

IV. Experimental Results

We implemented the heuristic pruning technique presented in the previous sections and evaluated it on a number of conformant planning benchmarks. The experiments were run on a PC running Ubuntu 8.0 at 1.90 GHz with 1 GB main memory. There are three heuristic setups in Conformant-FF and most of the time they yield similar performance. We select to use the state-formula heuristic function which gives the most precise computing. It is enough to evaluate the power of the pruning technique under one heuristic function setup. The individual domains are described below respectively. For each problem we provide the runtime in seconds, the number of evaluated states during search and the length of the found plan of the planning system that adopts our helpful implication paths pruning and the original helpful actions pruning, respectively.

A. The Uts domain

The Uts domain describes a series of tasks of universal transversal sequences. In this domain, one is initially located at any node on a graph with *n* nodes. Some nodes on the graph are connected by edges. One can execute the action "*start*" to switch to a "started" phase and make the current node visited. After started, one can "*travel*" through edges to change the locations and visit other nodes. The goal is to visit all the nodes with the initial location uncertain. There are three different test suites, k-*n*, l-*n*, r-*n*, corresponding to tasks with different extents to which the graph is connected, where the number of nodes in 2*n*. Tab. 1 provides our experimental results in the Uts domain. "US" indicates that the problem is unsolvable.

From a quick glance, our idea improves the helpful actions pruning significantly. To make every node visited eventually, a subgoal that gets to a "started" phase is required at first. Since the initial location is unknown, one should apply a *start* action to each possible initial location. Intuitively, the number of the required *start* actions is equal to the number of nodes in a task. Conformant-FF recognizes these helpful actions and to assure the "started" subgoal, these actions are applied one after another until all of them are included into the plan. For a task with 2*n* nodes, the subgoal becomes true after 2*n* search iterations. In our implementation, all these actions form helpful implication paths for the "started" subgoal. Using helpful implication path pruning technique to expand the initial belief state, the subgoal becomes true in the second search iteration in despite of the number of nodes in the task.

Our pruning technique of helpful implication paths leads to a decrease of evaluated states which refers to the size of search space. Most strikingly, the number of evaluated states is even a lot smaller than the plan length in some problems. This means that helpful implication paths contribute to find a fragment of the plan within one search iteration and don't have to compute the heuristic values for all the states that are along the search path corresponding to the plan. With the number of evaluated states decreased, the efficiency is improved dramatically. Regarding runtime, our implementation is much superior and scales up more easily. We find that sometimes the plan length of our idea is a little longer due to the order of actions in the plan sequence. This, however, does not affect the plan quality seriously, since Conformant-FF does not guarantee to find an optimal plan anyway.

TABLE I.
RESULTS IN THE UTS DOMAIN

	Helpful implication paths			Helpful actions		
	T (s)	S	L	T (s)	S	L
k-01	0.00	2	4	0.01	4	4
k-02	0.00	4	10	0.01	10	10
k-03	0.01	6	16	0.04	16	16
k-04	0.01	8	22	0.22	22	22
k-05	0.02	10	28	0.83	28	28
k-06	0.02	12	34	1.81	34	34
k-07	0.04	14	40	4.45	40	40
k-08	0.07	16	46	10.25	46	46
k-09	0.12	18	52	23.58	52	52
k-10	0.20	20	58	45.95	58	58
l-01	0.00	2	4	0.00	4	4
l-02	0.01	7	12	0.02	14	11
l-03	0.01	17	24	0.03	28	22
l-04	0.03	26	35	0.11	41	34
l-05	0.08	40	51	0.39	56	48
l-06	0.20	57	66	1.23	73	64
l-07	0.42	76	86	3.03	92	82
l-08	1.25	104	105	6.29	113	102
l-09	2.48	129	129	16.95	136	124
l-10	4.04	171	152	41.28	161	148
r-01	0.00	1	US	0.00	1	US
r-02	0.00	1	US	0.00	1	US
r-03	0.00	8	8	0.02	17	16
r-04	0.01	20	32	0.14	34	25
r-05	0.04	29	37	3.23	1021	37
r-06	0.08	28	47	18.18	4071	41
r-07	0.26	52	53	2.63	74	45
r-08	0.27	36	55	7.24	83	50
r-09	0.17	40	56	13.59	112	58
r-10	0.78	64	72	40.18	156	69

B. The Safe domain

In the Safe domain, there are several possible combinations for a safe. The right combination is one out of these combinations which is initially unknown. The goal is to get the safe door open. To assure the goal, one must "*try*" all combinations exhaustively. Tab. 2 and Tab. 3 provides the experimental results for instances with *n* possible combinations. "-" indicates time-out, with a runtime cutoff 1200s.

The plan for any problem in this domain is to try all combinations in some order. In Conformant-FF's implementation, the number of evaluated states is equal to the plan length, i.e., the number of possible combinations in the task. For each visited state, the goal proposition *safe-open* appears at the first layer of the relaxed planning graph, providing all the *try* actions at first layer that have not been applied as helpful actions.

TABLE II.
RESULTS IN THE SAFE DOMAIN-1

	Helpful implication paths			Helpful actions		
	T (s)	S	L	T (s)	S	L
s5	0.00	1	5	0.00	5	5
s10	0.01	1	10	0.03	10	10
s30	0.14	1	30	3.87	30	30
s50	1.34	1	50	72.87	50	50
s70	6.06	1	70	368.74	70	70
s100	28.22	1	100	-	-	-

TABLE III.
RESULTS IN THE SAFE DOMAIN-2

	Helpful implication paths			Helpful actions -D		
	T (s)	S	L	T (s)	S	L
s5	0.00	1	5	0.00	5	5
s10	0.01	1	10	0.01	10	10
s30	0.14	1	30	0.22	30	30
s50	1.34	1	50	2.18	50	50
s70	6.06	1	70	8.92	70	70
s100	28.22	1	100	35.01	100	100

Executing one of these helpful actions will get to a state closer to the goal. Such iterations are repeated until the goal is reached. Using our pruning technique, all the try actions are selected as helpful implication paths in the first search iteration. After executing the helpful implication paths in the initial belief state, the goal becomes known in the second iteration and the task is solved successfully. For tasks in different sizes, our idea can find plans by evaluating only one state.

Regarding runtime, our implementation behaves much faster. The poor runtime performance of Conformant-FF is also due to the special structure of the Safe domain that no proposition becomes known until the task is solved. As the number of possible combinations increases, Conformant-FF gets in trouble with repeated states checking. Helpful actions -D refers to the option that turns the repeated states checking off, at this time the original planner can solve the tasks quickly. Generally, our pruning technique is still competitive in this domain with the helpful actions of Conformant-FF without repeated states checking.

C. The Dispose domain

The Dispose domain is a variation of Grid family, which is about picking objects and dropping them into a trash. Initially the locations of all the objects are uncertain and the location of the trash is given. One starts from a given location and can "*move*" between two adjacent locations. One can "*pickup*" an object with a condition that the object is at the current location. One can also "*drop*" an object into a trash if the trash is at the current location. The goal is to get all the objects disposed of in the trash. Tab. 4 shows our results in the Dispose domain.

Once again, the old helpful action pruning technique evaluates much more states and performs a lot worse. To drop an object into a trash, one has to guarantee that the object is held in hand. Given that the initial locations of the objects are unknown, a conformant plan should execute a *pickup* action for each object at all the possible

TABLE IV.
RESULTS IN THE DISPOSE DOMAIN

	Helpful implication paths			Helpful actions		
	T (s)	S	L	T (s)	S	L
2-1	0.00	9	9	0.00	11	9
2-2	0.00	10	14	0.01	16	14
2-3	0.01	11	19	0.02	21	19
2-4	0.01	12	24	0.03	26	24
2-5	0.02	13	29	0.04	31	29
2-6	0.02	14	34	0.08	36	34
2-7	0.04	15	39	0.13	41	39
2-8	0.07	16	44	0.23	46	44
2-9	0.10	17	49	0.32	51	49
2-10	0.13	18	54	0.49	56	54
3-1	0.00	25	20	0.01	36	24
3-2	0.04	26	30	0.07	46	34
3-3	0.15	27	40	0.30	56	44
3-4	0.41	28	50	0.80	66	54
3-5	0.66	29	60	1.84	76	64
3-6	1.15	30	70	2.55	86	74
3-7	2.11	31	80	5.53	96	84
3-8	3.26	32	90	10.98	106	94
3-9	5.17	33	100	18.11	116	104
3-10	8.77	34	110	32.24	126	114
4-1	0.06	61	43	0.07	61	39
4-2	0.78	62	60	0.79	78	56
4-3	1.94	63	77	2.37	95	73
4-4	5.59	64	94	7.94	112	90
4-5	18.09	65	111	27.82	129	107
4-6	25.84	66	128	46.22	146	124
4-7	47.63	67	145	93.87	163	140
4-8	73.42	68	162	156.43	180	158
4-9	189.02	69	179	462.63	197	175
4-10	207.30	72	183	508.43	202	180

locations. We observed that our pruning technique explores a much smaller search space. As the size of the problem increases, heuristic evaluation becomes more costly and the runtime spent mostly goes into the computation of heuristic estimations, thus the reduction of search space is quite significant. Regarding plan length, our implementation is somewhat longer in a few cases. But still the quality of the found plan is satisfying.

D. The Logistics domain

Logistics we consider in this section is a classical planning domain enriched with uncertainty. A package can be loaded onto a truck if the package is at the same location as the truck. Packages are transported by trucks between different positions of a city. Airplanes fly between different cities. Instances in Tab. 5 are generated by choosing the initial positions of the trucks and airplanes, the initial positions and goal positions of packages randomly. The uncertainty lies in the initial position of each package within its origin city.

Results from Tab. 4 show that the pruning of helpful implication paths is generally competitive or faster, compared to helpful actions. Our pruning technique avoids the computation of heuristic values for some search states which helps to save the runtime considerably. In the relatively small cases, the behavior of helpful implication path pruning is not obvious that much. Large cases with 10 packages, cities, trucks and airplanes reveal the power of the pruning strategy and show that our implementation scales better to large instances. With respect to the plan quality, our technique

TABLE V.
RESULTS IN THE LOGISTICS DOMAIN

	Helpful implication paths			Helpful actions		
	T (s)	S	L	T (s)	S	L
2-2-2	0.00	16	16	0.00	23	16
2-2-4	0.00	23	26	0.00	34	26
2-3-2	0.01	19	17	0.00	37	17
2-3-3	0.01	36	24	0.01	46	24
2-10-10	0.89	211	83	3.59	324	83
3-2-2	0.01	23	20	0.01	29	20
3-2-4	0.01	29	33	0.02	36	33
3-3-2	0.01	36	28	0.02	55	28
3-3-3	0.01	32	32	0.07	72	34
3-10-10	3.09	427	112	10.95	435	108
4-2-2	0.00	15	19	0.01	19	19
4-2-4	0.01	36	40	0.08	59	40
4-3-2	0.00	24	23	0.01	31	23
4-3-3	0.02	48	37	0.07	70	37
4-10-10	2.51	281	127	11.45	356	121

is a little longer only in two problems. In most of the time, our pruning of helpful implication paths finds exactly the same conformant plan as the situation of helpful actions pruning.

All in all, from the experimental results above, we conclude that our pruning technique has the potential to reduce the size of search space and consequently improve the runtime efficiency. In this aspect we consider that our idea is clearly superior to helpful actions technique.

V. CONCLUSION

In this paper, we addressed the Conformant-FF planner which solves conformant planning problem by belief state space search. The size of search space has been a bottleneck to this method which could be ameliorated by using heuristic function and pruning. Based on our analysis of the implication paths in the relaxed planning heuristic function, we proposed the pruning technique of helpful implications paths to reduce the search space further. We run a number of conformant benchmarks to evaluate our idea and the experimental results indicate that our heuristic technique has two advantages:

1) An action sequence that integrates several helpful action branches together usually cuts down the branching factor of a search state. At the same time, considering helpful implication paths over other branches often finds a better state faster.

2) Executing a helpful implication path can get to a better state a few steps away within one search iteration, which relates with recent tread on obtaining long sequence of actions instead of applying one by one. Thus the evaluations of intermediate states on the helpful implication path are avoided, which leads to a considerable improvement of runtime efficiency.

The planner Conformant-FF still has inherent limitations due to its implicit representation of belief states and incomplete information adding to the relaxation. These make the planner provide inaccurate heuristic sometimes and get trouble in situations where an action may contain many conditional effects or takes more complicated forms. It will be significant to propose promising ideas to overcome those disadvantages.

In future, we will treat nondeterministic action effects and explore to use a similar pruning idea to solve the more general setting of contingent planning, i.e. to handle partially observable planning problems. We also plan to investigate the search spaces of other non-classical planning settings that can be formalized as search problems, in particular probabilistic planning and temporal planning.

ACKNOWLEDGMENT

The authors wish to thank Professor Dantong Ouyang for her comments, which was helpful to improve the paper. This work was Supported by the National Natural Science Foundation of China under Grant No. 61272208,61133011,60973089,61003101,61170092;
Jilin Province Science and Technology Development Plan under Grant No. 20101501,20100185,201101039;
Doctoral Fund of Ministry of Education of China under Grant No.20100061110031;

REFERENCES

[1] P. Bertoli, M. Pistore, P. Traverso, "Automated Composition of Web Services via Planning in Asynchronous Domains," *Artificial Intelligence*, vol. 174, pp. 316-361, 2010, doi: 10.1016/j.artint.2009.12.002.

[2] A. E. Gerevini, P. Haslum, D. Long, A. Saetti, Y. Dimopoulos, "Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners," *Artificial Intelligence*, vol. 173, pp. 619-668, 2009, doi:10.1016/j.artint.2008.10.012.

[3] A. L. Blum, M. L. Furst, "Fast Planning through Planning Graph Analysis," Artificial Intelligence, vol. 90, pp. 279-298, 1997, doi: 10.1016/S0004-3702(96)00047-1.

[4] D. Bryce, W. Cushing, S. Kambhampati, "State Agnostic Planning Graphs: Deterministic, Non-deterministic, and Probabilistic Planning," *Artificial Intelligence*, vol. 175, pp. 848-889, 2011, doi:10.1016/j.artint.2010.12.002.

[5] Y. Chen, R. Huang, Z. Xing, W. Zhang, "Long-distance mutual exclusion for planning," Artificial Intelligence, vol. 173, pp. 365-391, 2009, doi: 10.1016/j.artint.2008.11.004.

[6] N. Meuleau, E. Benazera, R. I. Brafman, E. A. Hansen, Mausam, "A Heuristic Search Approach to Planning with Continuous Resources in Stochastic Domains," *Journal of Artificial Intelligence Research*, vol. 34, pp. 27-59, 2009, doi: 10.1613/jair.2529.

[7] T. De la Rosa, S. Jimenez, R. Fuentetaja, D. Borrajo, "Scaling up Heuristic Planning with Relational Decision Trees," *Journal of Artificial Intelligence Research*, vol. 40, pp. 767-813, 2011, doi: 10.1613/jair.3231.

[8] J. Hoffmann, B. Nebel, "The FF Planning System: Fast Plan Generation through Heuristic Search," *Journal of Artificial Intelligent Research*, vol. 14, pp. 253-302, 2001, doi: 10.1613/jair.855.

[9] B. Bonet, "Conformant Plans and Beyond: Principles and Complexity," *Artificial Intelligence*, vol. 174, pp. 245-269, 2010, doi: 10.1016/j.artint.2009.11.001.

[10] D. Bryce, S. Kambhampati, D. E. Smith, "Planning Graph Heuristics for Belief Space Search," *Journal of Artificial* Intelligence Research, vol. 26, pp. 35-99, 2006, doi: 10.1613/jair.1869.

[11] A. Albore, M. Ramirez, H. Geffner, "Effective Heuristics and Belief Tracking for Planning with Incomplete Information," in *AAAI*, F. Bacchus, C. Domshlak, S. Edelkamp, M. Helmert, Eds. 21st International Conference on Automated Planning and Scheduling, 2011, pp. 2-9.
[12] H. Palacios, H. Geffner, "Compiling Uncertainty Away in Conformant Planning Problems with Bounded Width," *Journa*l of Artificial Intelligence, vol. 35, pp. 623-675, 2009, doi: 10.1613/jair.2708.
[13] D. Tran, H. Nguyen, E. Pontelli, T. C. Son, "Improving performance of conformant planners: Static analysis of declarative planning domain specification," in *Springer*, Practical Aspects of Declarative Languages, 11th International Symposium, 2009, pp. 239-253, doi: 10.1007/978-3-540-92995-6_17.
[14] J. Hoffmann, R. Brafman, "Conformant Planning via Heuristic Forward Search: A New Approach," *Artificial Intelligence*, vol. 170, pp. 507-541, 2006, doi: 10.1016/j.artint.2006.01.003.

Wei Wei was born Dec. 28, 1984, in Jilin, China. She received the Master degree at College of Computer Science and Technology of Jilin University, China, in 2007.

She is a Ph.D candidate at College of Computer Science and Technology of Jilin University. Her main research interests include automated planning and reasoning.

Dantong Ouyang was born in 1968, Jilin, China. She received the Ph. D degree at College of Computer Science and Technology of Jilin University, China, in 1998.

She is Professor, Ph.D. supervisor, Deputy Dean of College of Computer Science and Technology of Jilin University. Her main field of expertise is model-based diagnosis and automated reasoning.

Tingting Zou was born Nov. 24, 1984, in Jilin, China. She received the Master degree at School of Computer Science and Information Technology of North East Normal University, China, in 2007.

She is a Ph.D candidate at College of Computer Science and Technology of Jilin University. Her main research interest is description logic.

Shuai LU was born Jul. 11, 1981, in Jilin, China. He received the Ph.D degree at College of Computer Science and Technology of Jilin University, China, in 2010.

He is a lecturer at College of Computer Science and Technology of Jilin University. Her main research interests include automated planning and reasoning.