Situated Action Generator *Post-hoc* Reconstruction of Plans

Serin Lee
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan
srlee@nnl.isas.jaxa.jp

Takashi Kubota and **Ichiro Nakatani**
JAXA-Institute of Space and Astronautical Science
3-1-1 Yoshinodai, Sagamihara-shi, Kanagawa, Japan
{kubota, nakatani}@nnl.isas.jaxa.jp

Abstract

The approaches to make an agent generate intelligent actions in the field of AI might be very roughly categorized into two ways—the classical planning and situated action system. It is well known that each system has its own strength and weakness. However, each system also has its own application field. In particular, most of situated action systems do not directly deal with the logical problem. This paper proposes a novel action generator to situatedly extract a set of actions, which is likely to help to achieve the goal at the current situation, by the noop first heuristic in the relaxed logical space. After performing the set of actions, the agent should recognize the situation for deciding the next likely set of actions. The empirical result in some planning domains shows that the quality of the resultant path to the goal, *post-hoc reconstruction of plans*, is mostly acceptable as well as deriving the fast response time.

Introduction

The approaches to make an agent generate intelligent actions in the field of AI might be very roughly categorized into two ways—the classical planning and situated action system. The former which comes from logical traditions is to extract the complete path to the given goal by reasoning about the agent's own action and situation. In particular, the planners based on (heuristic) search such as Graphplan(Blum & Furst 1997), HSP(Bonet & Geffner 2001), FF(Hoffman & Nebel 2001), LPG(Gerevini, Saetti, & Serina 2003), and YAHSP(Vidal 2004) have been outperformed in most of classical planning benchmarks.

The latter study on how agents use their circumstances to achieve intelligent actions, rather than reasoning about actions away from its circumstances, actively arose in the 1980s as a reaction against the above classical view(Suchman 1987)(Wilson & Keil 1999). Since this approach is to derive not the complete course to the goal, but only the closer state to the goal (and the action to accomplish it) at every situation, the computational requirement could be reduced compared with the classical planning. Numerous scientific applications adapting this notion have been followed, and usually showed the faster response time in deriving actions for achieving the goal. As a result, this fast runtime makes the situated system be well situated than the classical planner in the dynamic and (partially) unknown environment(Arkin 1998)(Brooks 1986). However, although the closer state to the goal would be easily calculated in spatial reasoning, it might be comparatively difficult to derive it in logical reasoning. Therefore, most of practical applications of the situated action have been restricted to the field such as the navigation of mobile robots, and not directly handled the logical problem which has been dealt by the classical planning.

To solve problems of each approach, the hybrid architecture which simply combines two methods has been proposed. However, most of them still have inherent problems of each approach, and do not provide the fine granularity(Firby 1987)(Arkin 1998).

The objective of this paper is to develop the algorithm which can situatedly solve the logical problem, which might be necessary for the agent to accomplish the complicated task in the dynamic and hazardous environment. Therefore, it might be natural that the situated action generator not to provide the full path to the goal, but to situatedly(immediately) derive only the proper action, which is necessary to achieve the goal of the given logical problem, is first considered. However, the conventional planning is required to exactly extract only the action that needs to achieve goal from currently executable actions.

Therefore, we propose the novel situated action generator which situatedely derives the approximation of the above action from the relaxed logical space, which is similar to the relaxed planning space used in some heuristic planners(Hoffman & Nebel 2001)(Vidal 2004). Because the relaxed logical space for deriving only the set of actions could be built small, and the action would be also extracted very fast, the response of the agent to the environment is consequently expected to be fast. It is believed that this can make the agent be well situated.

By the proposed situated action generator, the action, which is necessary to achieve the goal, is approximately extracted from currently executable actions at every situation. After performing the extracted set of actions, the agent should recognize new situation for deciding the next likely set of actions. By repeating this strategy, the agent is expected to arrive at the goal state. The consequent path to the goal could be considered as Suchman's post-hoc represented

plan(Suchman 1987).

This paper is organized as follows. After defining notations used in this paper, we present how the relaxed logical space can be build. And then, this paper explains the strategy to extract the set of actions from the space which is likely to help to achieve the given goal. This paper finally describes the entire algorithm with its limitations, and provides an empirical result before conclusion.

Definitions

A state S is a finite set of logical atoms (facts). A planning task $\mathcal{P}$ is a triple $\mathcal{P} = (\mathcal{O}, \mathcal{I}, \mathcal{G})$, where $\mathcal{O}$ is the set of actions, and $\mathcal{I}$ (the initial state), and $\mathcal{G}$ (the goals) are set of atoms. An action o is STRIPS action. $o = \{pre(o), add(o), del(o)\}$ denotes the precondition, addition effect, and deletion effect of o. The result of applying o to a state S is the state $Result(S, < o >) = (S \cup add(o)) \backslash del(o)$ if $pre(o) \subseteq S$. Otherwise, it is undefined.

One level of graphplan space(Blum & Furst 1997) includes two sub-levels, the fact and action level. For example, the level 0 includes $\mathcal{F}_0$ (the initial state $\mathcal{I}$) and $\mathcal{A}_0$ (currently applicable actions), where $\mathcal{F}_i$ and $\mathcal{A}_i$ denote the i th fact and action level, respectively.

The set $\mathcal{F}_g$ is the subset of currently achievable facts which is necessary to accomplish $\mathcal{G}$. However, the conventional planner is required to exactly derive $\mathcal{F}_g$. Therefore, the approximation of $\mathcal{F}_g$ is derived from the relaxed logical space $\mathcal{R}$, and the set $\mathcal{F}_{ag}$ denotes it. The set $\mathcal{A}_{ag}$ is the subset of currently executable actions that is required to achieve each fact of $\mathcal{F}_{ag}$. However, because all actions included in $\mathcal{A}_{ag}$ are not always applicable at the same time, a set $\mathcal{L} \subseteq \mathcal{A}_{ag}$ which has a precedence over other subsets of $\mathcal{A}_{ag}$ should be derived. That is, the agent is actually expected to arrive at the goal through applying $\mathcal{L}$ at every situation.

Deriving situatedness

To follow the rapid change of situation, the agent should respond to it as fast as possible. Therefore, if we restrict the meaning of the situatedness to the adaptation in the dynamic environment, then the fast response would be necessary for the agent to be well situated.

However, since the classical planning includes the heavy search for solving the given problem, it would be difficult to obtain the fast response. Therefore, we derive the fast response by roughly considering the problem through the following assumptions.

- The fact is not deleted by the action.[1]
- The fact which could be achieved at a certain time is regarded to be achieved at that time. That is, the case in which delays the time when the fact is achieved is not considered.

[1]Like other AI approaches, this paper assumes that the fact can be represented in predicates, and most of states can be described in the conjunction of facts.

By extracting and performing the set of actions $\mathcal{L}$ at every situation which is derived by the above assumptions, the agent is expected to arrive at the goal.

Relaxed logical space for generating situated actions

The relaxed logical space $\mathcal{R}$ is constructed in similar manner to the relaxed planning graph space used in some heuristic planners such as FF (Hoffman & Nebel 2001). As we defined in the previous section, since it is assumed in building $\mathcal{R}$ that the fact is not deleted by the action, the deletion effects of actions are ignored. Also, since the case in which delays the time when the fact is achieved is also ignored in building $\mathcal{R}$, if the action o is included in the level i of the planning graph space, then o is not appeared in the level j ($j > i$).

As a result, $\mathcal{R}$ can be considered as the graphplan space built by

- $\mathcal{P}' = (\mathcal{O}', \mathcal{S}, \mathcal{G})$, where $\mathcal{S}$ denotes the current situation, $\mathcal{O}' = \{(pre(o), add(o), \emptyset) \mid (pre(o), add(o), del(o)) \in \mathcal{O}\}$.
- If $o \in level_i$, then o is not appeared in the $level_j$ ($j > i$).
- The last fact level includes all of goal facts.

An example on the relaxed logical space for the rocket problem is shown in Figure.1.

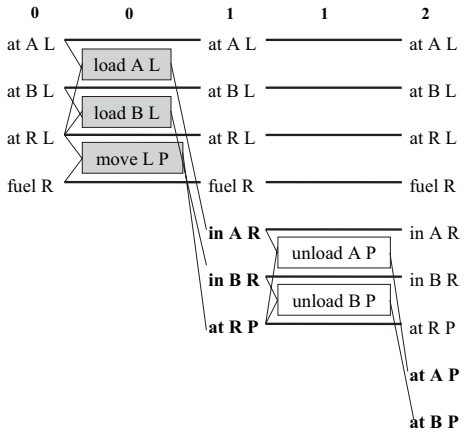

Figure 1: Relaxed logical space for the rocket problem

Extraction of the likely action set

After building the relaxed logical space $\mathcal{R}$, the simple backward chaining is performed from each goal fact to the action level 0. Since this space does not include the deletion (negation) fact, there is no exclusive relation (mutex) between two actions or facts. Therefore, the fact added in the level i is not deleted even after the level i, and would be the precondition of other actions. As a result, the fact extracted during the above backward chaining can be roughly regarded as the necessary fact to achieve the goal. In particular, the extracted fact in the level 1 can be considered to the subset of

currently achievable facts which is likely to help to achieve the goal, and thus defined to $\mathcal{F}_{ag}$. Also, a set of actions which achieves each fact of $\mathcal{F}_{ag}$ can be regarded as the set of action $\mathcal{A}_{ag}$, as we defined before.

Here we note that in building $\mathcal{R}$, the case in which delays the time when the fact is achieved is not considered. For example, the fact f achieved by the action o_i in the level i is regarded to be achieved (transferred) by $noop$ after the level i. Even if the fact f is also achieved by the non-noop action o_j in the $level_j$ ($j > i$), then f is still regarded to be achieved by not o_j, but $noop$.

Therefore, during this backward chaining, if there is a noop that achieves a fact, then the noop is first selected (Hoffman & Nebel 2001). Otherwise, a non-noop is randomly selected because the minimum number of non-noops over the entire level cannot be immediately derived from $\mathcal{R}$.

As a result of this backward chaining, in the case of the rocket problem described in Figure.1, $\mathcal{F}_{ag}$ = {(in A R), (in B R), (at R P)}, and $\mathcal{A}_{ag}$ = $\mathcal{A}_0$ = {(load A L), (load B L), (move L P)}.

This $\mathcal{F}_{ag}$ has some similarities with some heuristics used in some planners(Nebel, Dimopoulos, & Koehler 1997; McDermott 1999; Hoffman & Nebel 2001). However, although this heuristics is to help to select the action in any planning state during the search, $\mathcal{A}_{ag}$ is used to directly extract the proper set of actions $\mathcal{L}$ without searching.

As we mentioned before, since all actions of the extracted $\mathcal{A}_{ag}$ are not always applicable at the same time, a set $\mathcal{L}$ which has precedence over other subsets of $\mathcal{A}_{ag}$ should be derived. Before developing the algorithm to extract $\mathcal{L}$ from $\mathcal{A}_{ag}$, we first consider the case in which $\mathcal{A}_{ag} = \mathcal{A}_0$ and $\mathcal{A}_{ag}$ includes only two actions, o_i and o_j. Then, all possible relations between two actions are as follows,

- $pre(o_i) \cap del(o_j) \neq \emptyset, pre(o_j) \cap del(o_i) = \emptyset,$
  $\forall f \in add(o_i) : \nexists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_j)$
  that is, $o_i \prec o_j$. o_i can be immediately performed, and thus becomes the element of $\mathcal{L}$.
- $pre(o_i) \cap del(o_j) \neq \emptyset, pre(o_j) \cap del(o_i) = \emptyset,$
  $\forall f \in add(o_i) : \exists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_j)$
  Even if o_i and o_j are not consistent, o_i becomes the element of $\mathcal{L}$ expecting that the other action achives the fact f_i ($f_i \in add(o_j) \wedge f_i \in \mathcal{F}_{ag}$) later instead of o_j.
- $pre(o_i) \cap del(o_j) = \emptyset, pre(o_j) \cap del(o_i) = \emptyset,$
  $\forall f \in add(o_i) : \nexists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_j),$
  $\forall f \in add(o_j) : \nexists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_i)$
  that is, $o_i \prec o_j$, and $o_j \prec o_i$. This represents the case in which that there is no order constraint between them, and thus o_i and o_j become the element of $\mathcal{L}$.
- $pre(o_i) \cap del(o_j) = \emptyset, pre(o_j) \cap del(o_i) = \emptyset,$
  $\forall f \in add(o_i) : \exists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_j),$
  $\forall f \in add(o_j) : \nexists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_i)$
  For example, if we assume that $P_1 \in add(o_i)$, $P_1 \in del(o_j)$, and $P_1 \in \mathcal{F}_{ag}$, then $P_1 \in \mathcal{F}_{ag}$ can be achieved by $o_j \prec o_i$ or $o_i \prec o_j \prec o_i$. Therefore, the selection of o_j for $\mathcal{L}$ is expected to extract the shorter path to the goal.
- $pre(o_i) \cap del(o_j) \neq \emptyset, pre(o_j) \cap del(o_i) \neq \emptyset,$
  that is, $o_i \not\prec o_j$, and $o_j \not\prec o_i$. Both of o_i and o_j cannot become the element of $\mathcal{L}$.
- $\forall f \in add(o_i) : \exists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_j),$
  $\forall f \in add(o_j) : \exists f : f \in \mathcal{F}_{ag} \wedge f \in del(o_i)$
  that is, o_i and o_j are not consistent for achieving $\mathcal{F}_{ag}$.

To deal with the case in which all actions in $\mathcal{A}_{ag}$ are exclusive to each other (the fifth and sixth case of the above category), the assumptions suggested in building $\mathcal{R}$ should be dropped, and the decision on which action should be performed later should be carefully made. In this decision, the complete path of actions to the goal should be derived, and eventually the conventional planning is required. If this is not carefully considered, then any necessary fact to accomplish the goal can be deleted, and not be even added to the situation again. That is, the agent can be caught in the deadlock(deadend)[2], and thus cannot accomplish the goal.

Since the conventional planner is required to avoid meeting the deadlock, in this paper, we consider only the deadlock free problem, and deal with the case of $\mathcal{L} = \emptyset$ through randomly selecting an action from $\mathcal{A}_{ag}$.

In the case of $\mathcal{A}_{ag} \neq \mathcal{A}_0$, the proper action can be in not $\mathcal{A}_{ag}$, but $\mathcal{A}_0 - \mathcal{A}_{ag}$. This means that the achievement of $\mathcal{F}_{ag}$ should be all delayed. Therefore, to find the proper action in $\mathcal{A}_0 - \mathcal{A}_{ag}$, the conventional planner would be required. Without this consideration, the agent can be caught in the deadlock, and even in the deadlock free problem, can be captured in *the circular routine*. (e.g. $o_1 \prec o_2 \prec o_1 \prec o_2 ...$)

Since this paper considers only the deadlock free problem, we handle the circular routine by just adding some randomness. This randomness could be provided by randomly selecting an action from $\mathcal{A}_0 - \mathcal{A}_{ag}$.

However, in our experience, the case in which the achievement of $\mathcal{F}_{ag}$ should be all delayed is rarely occurred. Therefore, the random selection from $\mathcal{A}_0 - \mathcal{A}_{ag}$ in the deadlock free problem is performed with very low probability.

In summary, the proposed random selection is as follows,

- When $\mathcal{L} = \emptyset$, an action is randomly selected from $\mathcal{A}_{ag}$ with $1 - \zeta$.
- When $\mathcal{L} = \emptyset$, an action is randomly selected from $\mathcal{A}_0 - \mathcal{A}_{ag}$ with ζ. This case means that the achievement of $\mathcal{F}_{ag}$ should be all delayed, but is rarely occurred. Therefore, $1 - \zeta >> \zeta$.

Consequently, $\mathcal{L}$ is extracted from $\mathcal{A}_{ag}$ or $\mathcal{A}_0 - \mathcal{A}_{ag}$ by the following strategy. We define the candidate set of $\mathcal{L}$ as $\mathcal{L}_{cand}$, and $\mathcal{L}_{cand}$ is initialized to $\mathcal{A}_{ag}$.

- If some action in $\mathcal{L}_{cand}$ deletes one of the precondition of other actions, then the action is removed from $\mathcal{L}_{cand}$. For example, if $o_i \prec o_j$, then o_j cannot become the element of $\mathcal{L}$.
- If some action in $\mathcal{L}_{cand}$ deletes the fact of $\mathcal{F}_{ag}$ which is added by the action $o \in \mathcal{L}_{cand}$, o is removed from $\mathcal{L}_{cand}$.
- If $\mathcal{L}_{cand} = \emptyset$,
  - an action which is randomly selected from $\mathcal{A}_{ag}$ with $1 - \zeta$ becomes the element of $\mathcal{L}_{cand}$.

[2] A state is a deadend if it is reachable and no sequence of actions achieves the goal from it.(Hoffman & Nebel 2001)

  - an action which is randomly selected from $\mathcal{A}_0 - \mathcal{A}_{ag}$ with ζ becomes the element of $\mathcal{L}_{cand}$.
- The noop in $\mathcal{L}_{cand}$ is removed.
- The candidate set $\mathcal{L}_{cand}$ becomes $\mathcal{L}$.

In Figure.1, only `(load A L)` and `(load B L)` of $\mathcal{A}_{ag}$ can be contained in $\mathcal{L}$.

Entire description of the algorithm and its limitations

The proposed situated action generator is briefly described in Figure.2. As we mentioned in the previous section, the algorithm consists of three parts, building $\mathcal{R}$, extracting $\mathcal{A}_{ag}$ and $\mathcal{L}$, respectively. Because it derives only the action that is likely to help to achieve the goal at the current situation, the path to the goal is not known until the goal is accomplished.

This approach would make the agent respond to the environment fast. Therefore the agent that applies this approach is expected to be well situated even in the dynamic environment, and may have the robustness by a high sampling rate of the recognition derived from the fast response in a noisy environment.

Also, since the relaxed logical space to derive the set of actions, $\mathcal{L}$, can be made much smaller than that of the conventional planner, as shown in Figure.1, it could be used to solve the very large problem. Therefore, McCarthy's common sense informatic situation(McCarthy 1989) in which the facts are incomplete and there is no a priori limitation on what facts are relevant, for example, could be also partially considered with respect to the proposed approach.

As well as having the fast response to the environment and the relatively small logical space, the proposed approach can situatedly solve some logical problems which have been handled by the classical planning unlike most of the practical situated systems.

However, as we mentioned before, since the case in which delays the time when the fact is achieved is not considered in deriving $\mathcal{L}$, the proposed approach has the following drawbacks.

- Since the complete path to the goal is only partially considered, its *post-hoc* reconstructed path is usually longer than that of the classical planning. In particular, if there exists a goal ordering, then the path would be significantly longer. However, the approximate goal ordering strategy such as Koehler's goal agenda management(Koehler & Hoffman 2000) might partially help to solve this problem.
- Since the decision on which facts should be delayed requires the same task as the classical planning, it is not considered in the proposed situated action generator. This makes the agent with the situated action generator be caught in the deadlock or circular routine.

These drawbacks would also occur in the navigation of behavior-based robotics, if the action and situation are the same as above. However, unlike the case of the navigation of mobile robots, most of logical problems contain a lot of deadlock states. Therefore, it is believed that the hybrid architecture such as shown in Figure.4 might help to avoid meeting the deadlock.

while $\mathcal{G} \not\subseteq \mathcal{F}_i$ of $\mathcal{R}$ **do**
  building the next level of $\mathcal{R}$ with $i \leftarrow i+1$
endwhile
let $\mathcal{G}_{s1} = \emptyset$
for $level = i, \cdots, 1$ **do**
  if $level = i$ **then**
    $\mathcal{G}_{s2} = \mathcal{G}$
  endif
  else
    $\mathcal{G}_{s2} = \mathcal{G}_{s1}$
  let $\mathcal{A}_{ag} = \emptyset$
  forall $f \in \mathcal{G}_{s2}$ **do**
    let $\mathcal{A}_{ag-cand} = \emptyset$
    forall $o \in \mathcal{A}_{level-1}$ **do**
      if $f \in add(o) \wedge o = noop$ **then**
        $\mathcal{A}_{ag} \leftarrow \mathcal{A}_{ag} \oplus o$
        $\mathcal{G}_{s1} \leftarrow \mathcal{G}_{s1} \oplus pre(o)$
      endif
      if $f \in add(o) \wedge o = non\ noop \wedge \mathcal{A}_{ag} = \emptyset$ **then**
        $\mathcal{A}_{ag-cand} \leftarrow \mathcal{A}_{ag-cand} \oplus o$
      endif
    endfor
    if $\mathcal{A}_{ag} = \emptyset$ **then**
      $\mathcal{A}_{ag} \leftarrow \mathcal{A}_{ag} \oplus$ randomly selected o from $\mathcal{A}_{ag-cand}$
      $\mathcal{G}_{s1} \leftarrow \mathcal{G}_{s1} \oplus pre(o)$
    endif
  endfor
endfor
let $\mathcal{L} = \emptyset$, $\mathcal{L}_{cand} = \mathcal{A}_{ag}$
forall $o_i, o_j \in \mathcal{L}_{cand}$ **do**
  if $o_i \prec o_j$ **then**
    $\mathcal{L}_{cand} \leftarrow \mathcal{L}_{cand} \ominus o_j$
  endif
  if $\forall f \in add(o_i) : \exists f : f \in \mathcal{G}_{s2} \wedge f \in del(o_j)$ **then**
    $\mathcal{L}_{cand} \leftarrow \mathcal{L}_{cand} \ominus o_i$
  endif
endfor
if $\mathcal{L}_{cand} \neq \emptyset$ **then**
  $\mathcal{L} \leftarrow \mathcal{L} \oplus \mathcal{L}_{cand}$
endif
elseif $\mathcal{L}_{cand} = \emptyset \wedge$random variable $\geq \zeta$ **then**
  $\mathcal{L} \leftarrow \mathcal{L} \oplus$ randomly selected o from $\mathcal{A}_{ag}$
endelseif
elseif $\mathcal{L}_{cand} = \emptyset \wedge$random variable $< \zeta$ **then**
  $\mathcal{L} \leftarrow \mathcal{L} \oplus$ randomly selected o from $\mathcal{A}_0 - \mathcal{A}_{ag}$
endelseif
return $\mathcal{L}$

Figure 2: Situated action generator

let $\mathcal{RP} = \emptyset$
while $\mathcal{G} \not\subseteq$ current situation **do**
  generate $\mathcal{L}$
  while $\mathcal{L} \neq \emptyset$ **do**
    apply $o \in \mathcal{L}$
    $\mathcal{RP} \leftarrow \mathcal{RP} \oplus o$
    $\mathcal{L} \leftarrow \mathcal{L} \ominus o$
  endwhile
endwhile
return $\mathcal{RP}$

Figure 3: *Post-hoc* reconstruction of the plan

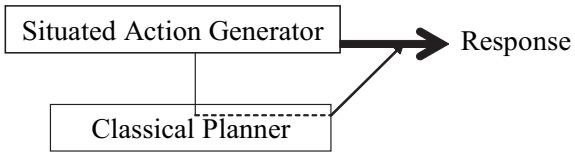

Figure 4: An exemplary hybrid architecture using the situated action generator. Unlike the conventional hybrid architecture, the action(response) is mostly generated by the situated action generator. Only when the agent approaches the deadlock, the classical planning takes charge of providing the proper path(or action).

Empirical results

We compared two classical STRIPS planners—FF v2.3 and LPG-td (speed), and our post-hoc reconstructor. They are all implemented in C, and no other particular library is used.

Although the experimental environment, and languages are the same as each other, their comparison is actually difficult because the objective of each approach is different.

In this paper, we first compared between the runtime which the planners build a plan only once at the initial state, and the average of response times which the situated action generator derives the set of actions at every situation until achieving the goal. And then, the qualities (plan length) of the plan by classical planners, and post-hoc reconstructed plan were compared.

We used four deadlock free domains in our experimentations: the Block's domain, the Logistics domain, the Satellite domain, and the Rovers domain. Only the results of the Block's domain were derived with the goal agenda management(Koehler & Hoffman 2000) because it contains a goal interaction (goal ordering).

We ran these tests on a Pentium IV 1.7GHz machine with 768M of RAM running Linux 2.6. The maximum amount of time which is allowed for each problem was fixed to 200 seconds, and we also assumed that ζ is 0.05. All results are shown in the following figures.

As we have mentioned before, most of experimental results showed that the quality of the reconstructed plan is mostly acceptable in the small problems as well as deriving the fast response time.

Although it still shows the fast response in the large problems, the quality is significantly lower than that of other conventional planners. However, it could deal with the very large problems which could not be solved by other conventional planners because its relaxed logical space could be made small. For example, in the case of some Rover problems, other planners require the memory usage over 70%, but the situated action generator needs only about 30% of the memory.

Conclusion

We presented the novel action generator to situatedly extract a set of actions, which is likely to help to achieve the goal at the current situation. To derive it, two assumptions which the fact is not deleted by the action, and the fact which could be achieved at a certain time is regarded to be achieved at that time were suggested.

By extracting and performing the proper action for achieving the goal at every situation based on these assumptions, the agent could arrive at the goal state as long as the task does not include the deadlock.

The experimental result in deadlock free planning domains showed that the quality of the post-hoc reconstructed plans is relatively acceptable as well as deriving the fast response to the situation.

We believe that the proposed approach might be necessary for the agent to accomplish the complicated task in the dynamic and hazardous environment. Also, if we consider its small logical space and quality of reconstructed plans as well as the fast response, then it could be guessed that the situated action generator shows more noticeable performance in the task which the expected path to the goal is relatively short, and each situation is very complicated and dynamic. Our everyday life may be similar to this situation.

To deal with the deadlock problems, the development of the hybrid architecture and/or the algorithm which can situatedly *image* the lookahead states will be our future works.

References

Arkin, R. C. 1998. *Behavior-Based Robotics*. The MIT Press.

Blum, A. L., and Furst, M. L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90:281–300.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129:5–33.

Brooks, R. A. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2(1):14–23.

Firby, R. J. 1987. An investigation into reactive planning in complex domains. In *Proceedings of the 6th National Conference on AI*.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning through stochastic local search and temporal action graphs in lpg. *Journal of Artificial Intelligence Research* 20:239–290.

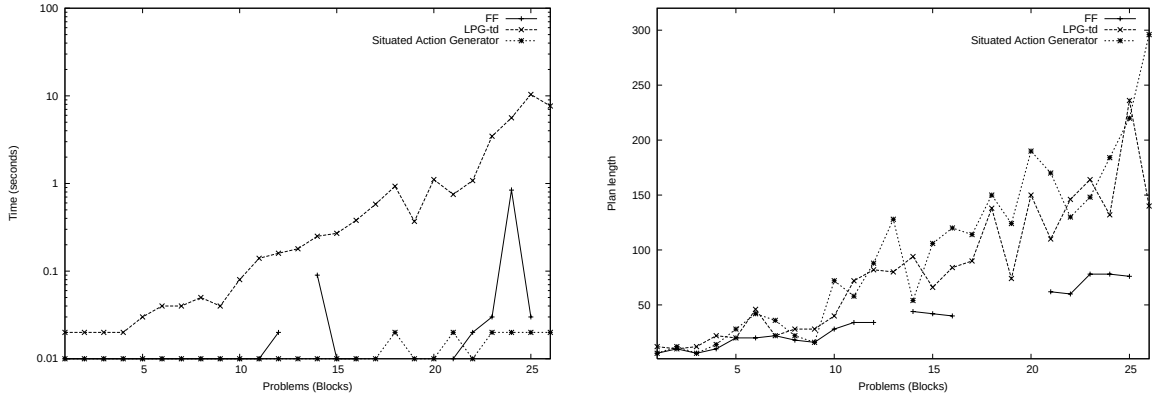

Figure 5: Time(left) and Plan length(right) on Blocks problems

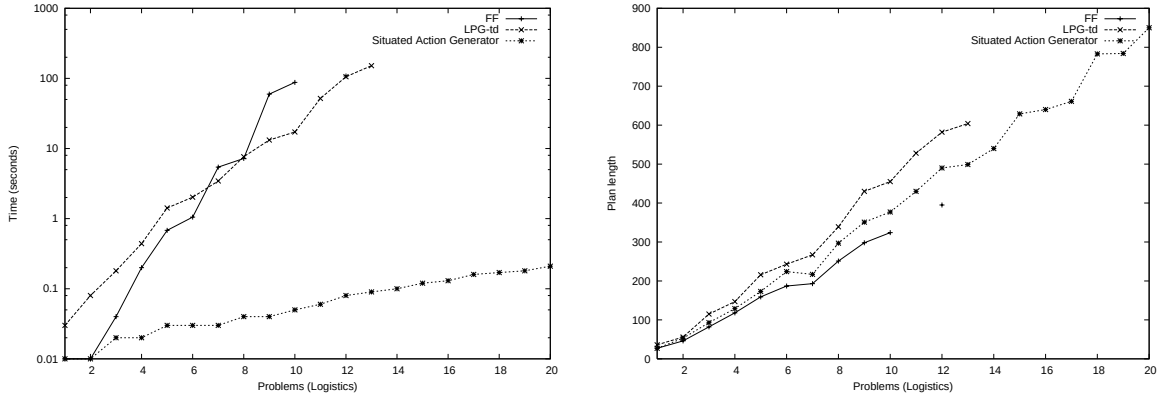

Figure 6: Time(left) and Plan length(right) on Logistics problems

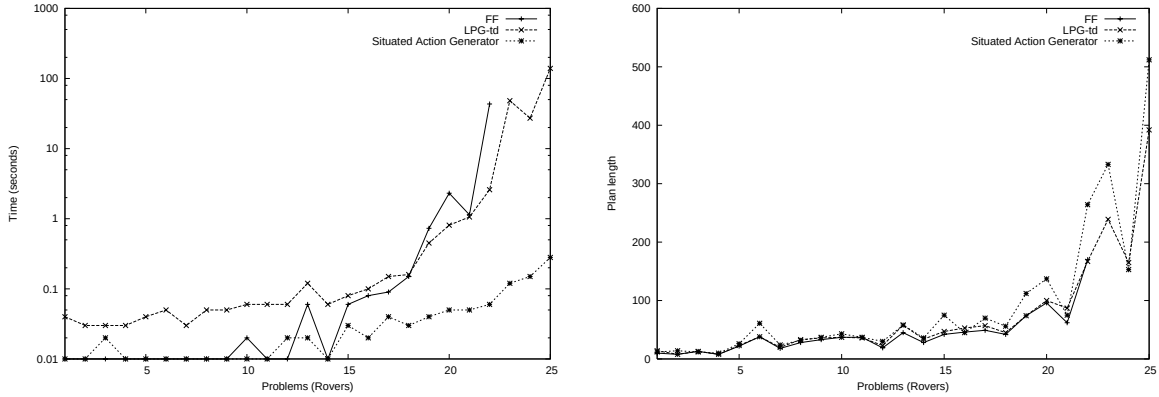

Figure 7: Time(left) and Plan length(right) on Rovers problems

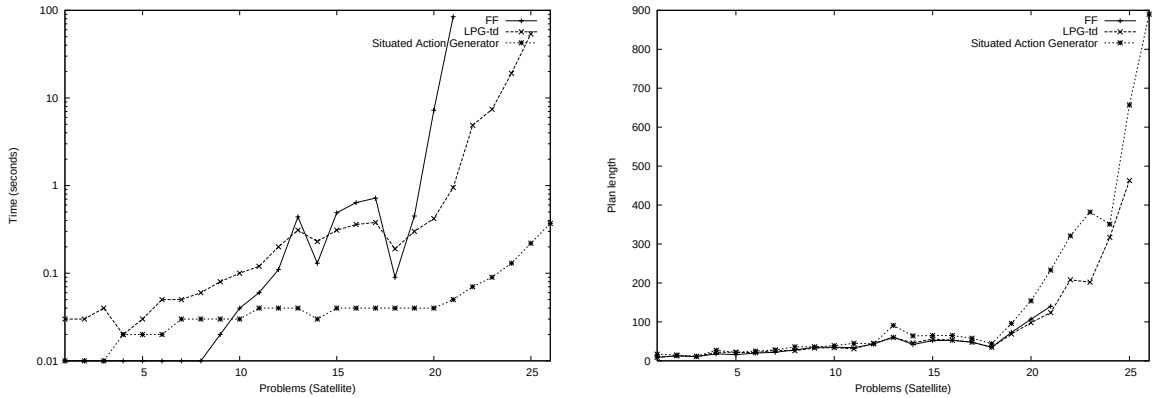

Figure 8: Time(left) and Plan length(right) on Satellite problems

Hoffman, J., and Nebel, B. 2001. The ff planning systems: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Koehler, J., and Hoffman, J. 2000. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research* 12:339–386.

McCarthy, J. 1989. Artificial intelligence, logic and formalizing common sense. In Thomason, R., ed., *Philosophical Logic and Artificial Intelligence*. Kluwer Academic.

McDermott, D. 1999. Using regression-match graphs to control search in planning. *Artificial Intelligence* 109:111–159.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring irrelevant facts and operators in plan generation. In *Proceedings of the 4th European Conference on Planning (ECP-97)*.

Suchman, L. 1987. *Plans and Situated Actions - The Problem of Human-Machine Communication*. Cambridge University Press.

Vidal, V. 2004. A lookahead strategy for heuristic search planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS-2004)*.

Wilson, R. A., and Keil, F. C., eds. 1999. *The MIT Encyclopedia of the Cognitive Sciences*. The MIT Press.