Proposal of Situated Planner
Considering on Common Sense Informatic Situation

Serin Lee*  Takashi Kubota**  Ichiro Nakatani**
* The University of Tokyo
** The Institute of Space and Astronautical Science

Abstract
Artificial Intelligence(AI) has been one of the most motivating research area to build intelligent ones since the 1950’s. Many researchers have struggled with it, and people had expected that ”artificially intelligent creatures” would appear soon. However, the problem has become even more and more difficult, and AI approaches to build it are even now in various difficulties.
In this paper, we first discuss the problem that occurs when the conventional AI algorithm, in particular, planner is applied to the common sense informatic situation such as the planetary exploration by the rover. And then, we propose the approach to solve those problems by considering two complementary attitudes to realize the intelligent ones, which are situated action and classical planning approach at the same time.

1 Introduction
Artificial Intelligence(AI) has been one of the most motivating research area to build intelligent ones since the 1950’s. Those intelligent agents have been required from the various fields, and in particular, the agent having the higher intelligence is recently necessary for some fields such as the planetary exploration or the computer game.
If we regard those intelligent agents as the performer which can properly use its own action to solve the problem(or to achieve the goal(s)), then it is natural for many fields of study to concentrate all their efforts on investigating algorithms to generate appropriate(and the best if possible) actions. The AI algorithm can be also viewed as a problem solver which forms the proper actions.
As a good exemplary agent, the human being has been research objective of cognitive psychology. It has tried to explain how this human mind comes to know things about the world around it, about other people, and about itself, and how it uses this knowledge to perform an impressive range of tasks such as remembering, speaking, performing skilled actions, solving problems and reasoning.
However, in view of AI researchers who try to make real agents, the study of cognitive science would be luxurious. The ability of the contemporary computer to process the specific information is not enough to satisfy their whole study.
Therefore, AI researchers choose a little different attitude to realize the algorithm. In particular, they have developed the two complementary approaches to the problem of generating actions as follows,

• Planning : It is appropriate when a number of actions must be executed in a coherent pattern to achieve a goal or when the actions interact in complex ways. Most of these approaches assume the relatively simple environment(e.g. toy world).
• Situated action : It is more based on the cognitive science theory, and appropriate when the best action can easily be computed from the current state of the world, that is, when no look-ahead is necessary because actions do not interfere with each other. And thus, this approach is more proper for the agent in the dynamic and unknown environment, but inherently includes a lack of goal-orientedness.

However, our objective in this paper is to build the goal-oriented intelligent agent operated in the
dynamic, unknown, and real environment (e.g. planetary rovers). And thus, we naturally believe that the following things should be also taken into account to build our aimed and generally desired agent.

- Hybrid architecture: If we think of the merit and demerit of the above two approaches, then it is natural that hybrid architecture systems has been proposed to consider both of purely reactive robotic systems and deliberative systems in the same frame. However, there still have some problems that the nature of the boundary between deliberation and reactive execution is not well understood at this time.[15] and most planners to decide the course of the higher level actions is still based on classical AI planner that assumes the static and simple environment.

- Common sense informatic situation: The ability to handle this situation proposed by John McCarthy is said to be one of the critical differences between the human being and machine[11] [12]. It represents the situation in which the facts are incomplete, and there is no a priori limitation on what facts are relevant. For example, the machine does not know if the fact which there is a chair in this room is necessary to go out of this room before seriously considering the facts. However, if the machine is in the bounded informatic situation (e.g. the location of the door, the path to the door), then the machine can solve the problem quite well.

In this paper, we propose the following approach to consider the above two problems at the same time.

The planner in the dynamic environment and the common sense informatic situation does not need to find the entire course of action. It is enough to decide an action that is now likely to properly achieve goal(s). After performing the action, the situation should be recognized for deciding the next likely action(s).

In the next two sections, we discuss the conventional planning and its problems for the common sense informatic situation. And then, to solve the problems, we describe our situated planning and its detail algorithm with the experimental results, and end with a brief conclusion.

2 Planning

2.1 Notational Conventions

The field of AI planning seeks to build control algorithms that enable an agent to synthesize a course of action that will achieve its goals.

A simple formulation of the planning problem defines the following inputs.

- a description of initial state of the world
- a description of agent’s goal
- a description of the possible actions (domain theory)

The planning output is a course(s) of action. The planning algorithm mentioned in this paper is based on the STRIPS planning task introduced by Fikes and Nilsson. We follow the notation mentioned in [3]

Definition 2.1 (State) A state \( S \) is a finite set of logical atoms.

Definition 2.2 (Strips Action) A STRIPS action \( o \) is a triple

\[
\begin{align*}
& o = \{ \text{pre}(o), \text{add}(o), \text{del}(o) \} \\
& \text{where} \ \text{pre}(o) \ \text{are the precondition of} \ o, \ \text{add}(o) \ \text{is the} \ \text{add list of} \ o \ \text{and del}(o) \ \text{is the delete list of the action}, \ \text{each being a set of atoms. For an atom} \ f \in \text{add}(o), \ \text{we say that} \ o \ \text{achieves} \ f. \ \text{The result of applying a single STRIPS action to a state is Result}(S, < o >) = (S \cup \text{add}(o)) \setminus \text{del}(o) \ \text{if} \ \text{pre}(o) \ \subseteq \ S. \ \text{Otherwise, it is undefined.}
\end{align*}
\]

Definition 2.3 (Planning Task) A planning task \( \mathcal{P} = (O, I, G) \) is a triple where \( O \) is the set of actions, and \( I \) (the initial state), and \( G \) (the goals) are set of atoms.

2.2 Heuristic Search Planning

The heuristic search planning is proposed by Bonet and Geffner[5][6]. In this approach a heuristic function is derived from the specification of the planning instance and used for guiding the search through the state space. Recently this approach is proved to be very competitive, and the planners based on this approach[3][4][7] outperformed in the planning competition.(AIPS-2000, AIPS-2002)

In the above approaches, Fast-Forward (FF) planning system is currently one of the fastest planners. FF first relaxes the planning task \( \mathcal{P} \) to the following task \( \mathcal{P}' \) like [5][6].

Definition 2.4 (Relaxed Planning Task)
Given a planning task \( \mathcal{P} = (O, I, G) \). The relaxation \( \mathcal{P}' \) of \( \mathcal{P} \) is defined as \( \mathcal{P}' = (O', I, G) \), with \( O' = \{ \{ \text{pre}(o), \text{add}(o), \emptyset \} | \{ \text{pre}(o), \text{add}(o), \text{del}(o) \} \in O \} \).

In words, one obtains the relaxed planning task by ignoring the delete lists of all actions. Started on a solvable relaxed problem, Graphplan is known to find a solution plan in polynomial time[3]. Facing a search state \( S \), we therefore run a relaxed version of Graphplan starting out from \( S \), and use the generated output for heuristic evaluation.

The process for extracting the relaxed plan by
using Graphplan results in a relaxed plan $< O_0, \ldots, O_{m-1} >$, where each $O_i$ is the set of actions selected at time step $i$. The FF regard the following solution length estimator as its heuristic.

$$h_{F F}(S) = \sum_{i=0, \ldots, m-1} |O_i|$$

While this heuristic can be computed in polynomial time, heuristic evaluation of states is still costly. It is therefore straightforward to choose hill-climbing as the search method, in the hope to reach the goal by evaluating as few states as possible. \[4\]

Facing a search state $S$, FF evaluates all of its direct successors. If none of those has a better heuristic value than $S$, it goes one step further, i.e., search then looks at the successor’s successors. Although this search process has a possibility to achieve plateaus and local minima, many planning domains tend to have a few them due to the structure of the problem domain.

3 Proposed Situated Planning

3.1 Hybrid Architecture and Common Sense Informatic Situation

As we mentioned in the previous section, what two complementary AI approaches to build an intelligent agent is established in the same frame might be one of natural ways to achieve our goal, which is to build an intelligent agent in the dynamic and unknown environment.

In addition, our agent should be properly operated in the real environment such as the surface of Mars or our home, that is common sense informatic situation. However, although the recent planning algorithm shows a good performance, the planner still assumes the static and toy environment, that is well bounded informatic situation. Therefore, if the agent using the conventional planning system is put in our actual environment, then the agent would not be worked well because our environment is not bounded. For example, given the goal, tt (at home), the agent in the library begins to take open-book-novel walk into consider to achieve the goal. That is, our actual library environment also includes a lot of books, other people, desks, and chairs as well as the path to home and the obstacle. Vast information stimulate a lot of parametrized action, and thus it might be impossible for the conventional planner that consider a possible action throughly to build its fully descriptive plan.

As a result, our agent should satisfy two properties(situated action and planning), and should be ready to put in common sense informatic situation[11][12]. In this paper, we propose the following approach to satisfy them.

The planner in the dynamic environment does not need to find the entire course of action. It is enough to decide an action that is now likely to properly achieve to goal(s). After performing the action, the situation should be recognized for deciding the next likely action(s).

That is, since it is inherently to relax the planning task, $P$, our approach may not derive the most efficient action. However, we believe that it can make the agent operate in the common sense informatic situation where includes vast information(facts) since the proposed approach deals with $P$ lightly.

3.2 New Relaxed Planning Space

We define the new relaxed planning task as follows,

Definition 3.1 (New Relaxed Planning Task)

Given a planning task $P = (O, I, G)$. The proposed relaxation $P'$ of $P$ is defined as $P' = (O', I, G)$, with $O' = \{(pre(o)', add(o), 0) \mid (pre(o), add(o), del(o)) \in O\}$, where $pre(o)' = \{f \mid positive \ fact \in pre(o)\}$. \[1\]

Definition 3.2 (New Plan Space $S$ for $P'$)

The plan space $S$ for $P'$ is built by the graphplan[10] from $O'$, but at least one of atoms $\forall p \in pre(o)' (o \in level_i, i \neq 0)$ should be achieved by non noop action(s). Even though the preconditions are satisfied, the action $o$ is not applicable if all $pre(o)'$ are achieved by noop actions in the previous level.

This plan space $S$ has the following properties.

- Since it ignores the delete lists of all actions, the $S$ does not include the mutex relation. \[3\]
- Any level in $S$ include all facts added in the previous level(s) or maintained by noop action since they cannot be deleted.

By the above relaxation, $S$ cannot provide the careful information for not $P'$, but $P$. And thus, although the relaxed $S$ assumes that the (sub)goal can be achieved, there is a case in which the discovered path to the goal cannot be actually achieved. Let us examine the case. If the actions in the level $i$ to achieve any subset $G_{i+1}$ in the fact level $(i + 1)$ of $S$, which is the given goal or subgoal(precondition) for an action in the next level\[2\], are assumed to be properly applied from the fact level $i$, then all of the actions must be applied, that is, their routines should exist to achieve the goal. However, if any action in the level $i$ satisfy the following conditions, then the routines for performing them cannot exist.

\[1\] From this section, $P'$ means the proposed relaxed planning task.
\[2\] This means that the facts $\in$ subset must exist at the same time.
First let us examine the case in which some of $o_i$ in
the level $i$ can be performed in both orders (that is, there is no order constraint), but the goals cannot be achieved.

- C1: Some pairs of effects of any non noop act
in the level $i$ cannot exist at the same time by
the definition of the actions, and $G_{i+1}$ includes
those facts formed by only the actions.

Second there is the case in which some of $o$ in
the level $i$ can be performed in some order (that is, there is an order constraint), but the goals cannot be achieved.

- C2: Any fact in $G_{i+1}$ achieved by the action $o_i$ is
deleted by other action $o_j$ which achieves some
elements in $G_{i+1}$, and $o_j$ also deletes preconditions
of all routines to achieve pre($o_i$).

Next we consider the case in which some of $o_i$ in
the level $i$ cannot be orderly performed. This case
might include closed sequences by inconsistent order
in the action level $i$. For example, if there are order
constraints $o_1 \prec o_2$, $o_2 \prec o_3$, and $o_3 \prec o_1$, then $o_1$, $o_2$, and $o_3$ form the closed sequence which cannot actually be performed due to its inconsistency.

To perform all actions in any closed sequence, at least
one action $o$ in the closed sequence should be per-
formed after other actions in the closed sequence are
consistently accomplished. And thus, all paths in the
previous level to achieve pre($o$) should not be deleted
with performing other actions in the closed sequence.

- C3: For any closed sequence, preconditions of all
paths to achieve pre($o$) of $o$ in the closed sequence
are deleted by performing other actions except $o$ in the closed sequence.

Until now, we examine the conditions for nonnoop
actions to achieve $G_{i+1}$. However, in the case in which
any fact $\in G_{i+1}$ is transferred by noop action, the
condition is different from cases of non-noop actions since the noop action is not actually performed, but just transfers the fact in the previous level to the next level. Therefore, we should consider only the case in which any fact derived by noop action is deleted.

- C4: Suppose that $G_{i+1}$ includes the atom $p_i$,
which is achieved by only noop action and was
first added in the level $a \leq i + 1$. All of the paths
to achieve $p_j \in G_{i+1}$ includes the $p_i$ in the level
$b(a \leq b \leq i + 1)$, and all of the preconditions in
all paths to achieve $p_i$ are deleted in the level
c($a \leq c \leq b$).

### 3.3 Situated Planner

If we ignore C1-C4 mentioned in the previous section,
then the path to the goal extracted from $S_0$ can be
regarded as one of the guaranteed plans to accomplish it. However, because $P$ might contain C1-C4, the relaxed path(plan) which seems to lead to the goal would be actually blocked by C1-C4 in some level. To avoid to extract those false paths, it seems to be reasonable to find the location(level) where C1-C4 are satisfied. But since the work requires the full consideration(not relaxed) for actions and facts (and their exclusive relation), it would be hard work like other conventional planners.

Instead of full consideration for C1-C4 in the entire
level, in this paper, the exclusive relations including
C1-C4 of actions and facts in the only level 0 are
considered. The following strategy describes it.

1. The actions satisfied with C1-C4 in the level 0
is not be currently performed.

2. If there are order constraints among the actions
extracted in the level 0, for example, $o_i \prec o_j$,
then only the action $o_i$ which have the highest
priority is performed.

3. All actions having the equivalent priority are
performed in random order.

4. If no action is finally selected in the level 0, that is,
all actions are satisfied with C1-C4, then one
action of those actions is randomly selected and
performed.

After performing actions according to the above
strategy, the situation is recognized again, and then
proper actions are extracted over again in the same
manner. This approach has the following merits.

- Since new situation is repeatedly recognized af-
fter performing some actions, the agent is situ-
ated.

- As mentioned before, the relaxed plan which
seems to lead to the goal would be actually
blocked by C1-C4 in some level. By repeat-
edly building the relaxed plan(re-planning), the
agent can avoid to follow only one uncertain re-
axed plan.

- Even if it meets C1-C4, the agent corrects to
other likely path. That is, the agent can consis-
tently modify to the most likely paths by con-
sidering the current situation. Since the agent
can perform the actions included in inefficient
paths, it would not derive the best quality plan.
This is fundamentally the same as behavior-
based robots.

There may be the case in which other all proper
paths are blocked while performing the inefficient
path. We call this latent dead-end. This case make
the agent not to achive the goal. The proposed ap-
proach is failed in this case.

The whole algorithm is described as follows,
while goal \not\subset current situation do
  while goal \not\subset current level \( n \) of \( P' \) do
    building the next level of \( P' \)
  endwhile
  while level \( n, \ldots, 1 \) do
    if \( add(o) \supseteq \forall \) (sub)goal
      if \( \exists o \) is \( \text{noop} \)
        The \( \text{noop} o \) is selected.
      endif
      else The other \( \text{non-noop} \) is randomly selected.
        \( pre(o) \) becomes the next subgoal
    endif
  endwhile
for \( o_i, o_j \subset level_0 \) (\( i, j = 0, \ldots, n, i \neq j \)) do
  if \( o_i \) and \( o_j \) are satisfied with C1-C4
    \( o_i \) and \( o_j \) cannot be selected
  endif
endfor
for \( o_i, o_j \subset level_0 \) (\( i, j = 0, \ldots, n, i \neq j \)) do
  if del(\( o_i \)) \supseteq pre(\( o_j \))
    \( o_i \) cannot be selected
  endif
endfor
randomly perform selected actions
endwhile

The algorithm can be divided into the following three parts.
1. Building the new relaxed plan space
2. Extracting the prospectively necessary actions
   - \( \text{Noop first heuristic}[3] \) with randomly selected non-noop actions
3. Selection of the actions which should be \textit{now} performed by the proposed strategy 1-4.

4 Simple Experimental Results

We examined the proposed approach with two conventional planning problems. The experiment was run on a Intel Pentium4 1.7GHz, 512MB.

First, typical \textit{Blocksworld} problems were experimented. In this domain, due to the inherent structure, the extracted level_0 actions are all exclusive, and thus always randomly selected from the extracted actions. Therefore, this domain can show the worst boundary of our approach.
The detail results are described in Figure.2, and Figure.3. The time to extract only the \textit{currently goal-oriented action} was just about 0.01seconds even in the complex cases. However, in almost all cases, the plan quality was not so good, as we expected.

Figure 1: The proposed situated planning algorithm
Figure 2: Runtime curves on \textit{Blocksworld} instances for five planners
Figure 3: Solution length on \textit{Blocksworld} instances for five planners
And then, we applied our approach to solve the typical logistics problems. As we expected, the plan quality (length) is much better than previous blocksworld domain. The time to extract the currently proper action was still very fast. The detail results were briefly described in the following tables. The previous results, in the table, the runtime of the proposed approach is the average time to extract only the currently goal-oriented action, and one of other planners is the time to look for the entire planning without any repetition for replanning.

<table>
<thead>
<tr>
<th>Situated Planner</th>
<th>log. 10</th>
<th>log. 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>GP-CSP</td>
<td>0.3</td>
<td>2.5</td>
</tr>
<tr>
<td>STAN</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LPG</td>
<td>0.07</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Table 1: Runtime for the log. 10, log. 20

<table>
<thead>
<tr>
<th>Situated Planner</th>
<th>log. 10</th>
<th>log. 20</th>
</tr>
</thead>
<tbody>
<tr>
<td>FF</td>
<td>50</td>
<td>127</td>
</tr>
<tr>
<td>GP-CSP</td>
<td>45</td>
<td>117</td>
</tr>
<tr>
<td>STAN</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LPG</td>
<td>57</td>
<td>159</td>
</tr>
</tbody>
</table>

Table 2: Solution length for the log. 10, log. 20

5 Conclusion and Future Work

In this paper, the new approach for planning has been proposed by adapting the situated action. Although it might not derive the best quality solution, the approach is expected to be well operated with situatedness in the common sense informatic situation since the required space for its algorithm is quite smaller than the conventional planning, and its incompleteness occurred by the fewer information space can be inherently compensated by re-planning. The experimental results showed that although its final plan quality critically depended on the structure of the problem domain, the proposed approach still had the fast response time. The more experiment will be performed, and the simple heuristic or learning algorithm will be applied like the behavior-based robotics to enhance the plan quality.

References

[18] Stefan Edelkamp, and Jörg Hoffman, PDDL2.2: The Language for the Classical Part of the 4th International Planning Competition