Proceedings of the 6th WSEAS International Conference on Robotics, Control and Manufacturing Technology, Hangzhou, China, April 16-18, 2006 (pp159-164)

Depth First Random Walk Based on Symbolic Situated Action

SERIN LEE[1], TAKASHI KUBOTA[2] and ICHIRO NAKATANI[1,2]
1. Department of Electronic Engineering
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, JAPAN
2. Institute of Space and Astronautical Science (ISAS)
Japan Aerospace Exploration Agency (JAXA)
3-1-1 Yoshinodai, Sagamihara-shi, Kanagawa, JAPAN

Abstract: The approaches to make an agent decide the proper actions for achieving the goal might be roughly categorized into two groups-the classical planning and situated action system. It is well known that each system has its own strength and weakness with its own application areas. In particular, most of situated action systems do not directly deal with general problems given in propositional logic. However, to build an embodied intelligent agent that can perform complicated tasks as well as navigate, we believe its design should be based on both the situated action approach and symbolic processing, which seem to be antithetic to each other. This paper first briefly mentions a novel action selector to situatedly extract a set of actions, which is likely to help to achieve the goal at the current situation, from the relaxed propositional space. After applying the set of actions, the agent should recognize the new situation for deciding the next proper set of actions. By repeating this procedure, the agent is expected to arrive at the goal state. However, since those actions are derived from the relaxed space in which roughly considers the planning problem, this method can be applied only in the deadlock free domain where fatally wrong decisions cannot be made. To solve the deadlock problems, some of subsequent states after applying an action should be considered to avoid meeting the deadlocks. This paper proposes a novel method to make the agent with the situated action selector solve most of deadlock problems without the help of the conventional planner, which situatedly images the lookahead states. If the agent is caught in a deadlock state during imaginarily walking on the lookahead states, then before meeting the actual deadlock, the agent could avoid meeting it. The experimental results of the proposed approach show that the agent can handle most of deadlock problems, and the quality of the resultant path to the goal is mostly acceptable as well as deriving much faster response time than the classical planning.

Key-Words: Action Selection, Planning, Situated Action

1 Introduction

The approaches to make an agent decide the proper actions might be roughly categorized into two groups-the classical planning and situated action system. The former which comes from the symbolic tradition of the field of AI is to extract the complete path to the given goal state by reasoning about the agent's own action and situation. Although this approach can derive the highly goal-driven course of actions to the goal, it is well known that this task requires much time and substantial resource. Due to this reason, the planning approach has not been used in an embodied agent, which requires other resource-consuming task such as recognition and learning.

On the other hand, the situated action approach is to study on how agents use their circumstances to achieve intelligent actions, rather than reasoning about actions away from its circumstances[3][4][5]. Since this approach is generally to derive not the complete course to the goal, but only the currently 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. Consequently, the faster response in deriving currently proper actions for achieving the goal can be achieved, and therefore an agent based on situated action is expected to be better situated than the classical planner, in particular in the dynamic and unanticipated environment. Furthermore, since the remaining resource derived by the fast response can be used in other important tasks such as learning and recognizing the situation, most of situated action systems have been actually applied to an embodied agent. However, as mentioned before, since the congeries of situated action have denied the abstract reasoning away from

its real circumstances, the use of symbols and its processing have been also considered to be unnecessary.

Although the conventional symbol representation method might not be optimal to describe our reasoning, over the past 35 years, a substantial number of symbol systems have been constructed and tested successfully, for their ability to simulate human thinking and learning over a wide range of task domains[6]. We also believe the conventional symbol representation can solely provide rich expressiveness to describe the general problems.

However, since the situated action approach has regarded the use of symbols and their processing to be unnecessary, and furthermore has not applied the symbol. Therefore, the situated action approach, which opposes the use of symbolic expression necessary to describe problems given to the agent, has been applied to only a restricted problem.

We believe the design of an embodied agent which can deal with more general problems should be based on both the situated action approach and symbolic processing, which seem to be antithetic to each other.

Therefore, we first briefly mention a novel method which situatedely derives the approximation of the above action from the relaxed propositional space, which roughly considers the given planning problem. Since this relaxed propositional 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 in solving the complicated tasks.

By the proposed method for deriving the situated action, the action, which is necessary to achieve the goal, is approximately extracted from currently executable actions at every situation. After applying the extracted set of actions, the agent should recognize the new situation to decide the next proper set of actions. By repeating this procedure, the agent is expected to arrive at the goal state.

However, since those actions are just derived from the relaxed space in which roughly considers the planning problem, this method can be applied only in the deadlock free domain where fatally wrong decisions cannot be made. To solve the deadlock problems, some of subsequent states after applying an action should be considered to avoid meeting the deadlocks. This paper proposes a novel method to make the agent with the situated action selector solve most of deadlock problems without the help of the conventional planner, which situatedly images the lookahead states. If the agent is caught in a deadlock state during imaginarily walking on the lookahead states, then before meeting the actual deadlock, the agent could avoid meeting it.

This paper is organized as follows. After defining notations used in this paper, we briefly present how the situated action can be derived from the relaxed propositional space. And then, this paper explains a novel strategy based on the situated action selector to avoid meeting the deadlock. This paper finally provides its experimental results before conclusion.

2 Definitions

A state **S** is a finite set of logical atoms (facts). A planning task **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. An action **o** is STRIPS action. *{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** becomes $\text{Result}(S, <o>) = (S \bigcup add(o)) \setminus del(o)$ if $pre(o) \subseteq S$. Otherwise, it is undefined.

A relaxed propositional space **R** is built by the two assumptions: (1) The deletion effect of actions is ignored. That is, a fact is not deleted by performing actions. (2) A fact which can be achieved at some time is regarded to be achieved at the time. That is, the case in which the time to achieve the fact is delayed is not considered.

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

Then, as mentioned before, the agent with only the above situated action selector can be caught in the deadlock. Therefore, to make the agent avoid meeting the deadlock through imaging the lookahead states, two ques are used. One is the primary imagination queue, $\mathbf{Im_q}$, that contains at most *n* elements. Each of the elements corresponds to the initial state or situations derived by the situated action selector. Like the primary imagination queue, the secondary

imagination queue, $\mathbf{Im_{sq}}$, also contains at most *n* elements, and is used instead of $\mathbf{Im_q}$ when the situated action selector meets the deadlock during filling $\mathbf{Im_q}$.

3 Classical Planning

Since 1960's, various algorithms have been proposed to make an agent build the plan for itself. In particular, the search-based planner has been one of the most interesting approaches. The well-known graphplan is also one of those state space search-based planners, and has been dealt from various aspects[1].

The planning space of the graphplan is a directed graph, and includes two types of nodes, proposition nodes and action nodes, arranged into levels. Nodes in odd numbered levels correspond to action instances; there is one such node for each action instance whose preconditions (facts) are present, and are mutually consistent at the previous even numbered level. All of the facts are transferred to the next level by the maintain action (noop). Since the specific level denotes a relative time, *mutex* (mutually exclusion relation) represents the exclusive relation in the same level. After the planning graph is built until the last fact level includes all of goal facts that do not include mutex, the graphplan performs a backward-chaining search on the space to look for the plan.

In this paper, $\mathbf{A_i}$ and $\mathbf{F_i}$ denotes the *i*-th action and fact level, respectively. For example, $\mathbf{A_0}$ denotes the set of action that is currently executable.

Fig. 1 shows an example of the planning graph for the rocket problem described in PDDL.

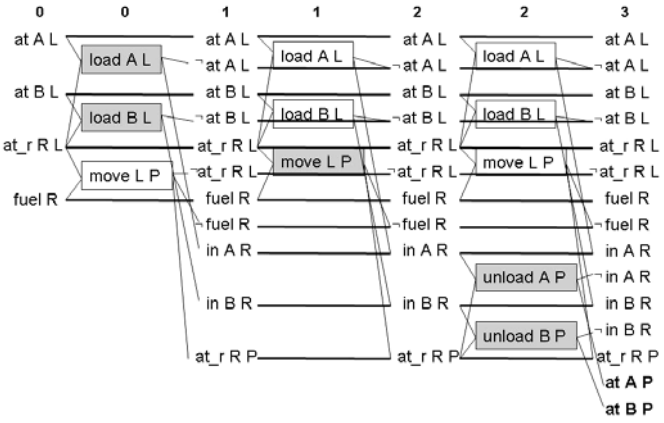

Fig. 1 Example of the planning graph

4 Situated Action Selector

4.1 Relaxed Propositional Space

The relaxed propositional space **R** is made in similar manner to the relaxed planning graph space used in some heuristic planners such as FF[2]. As defined in the previous section, since it is assumed in building **R** that a fact is not deleted by the action, the deletion effects of actions are ignored. Also, since the case in which delays the time when a fact is achieved is ignored in building **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, **R** can be considered as the graphplan space built by

- **P'=(O', S, G)** , where $O' = \{pre(o), add(o), \varnothing \mid (pre(o), add(o), del(o)) \subset O\}$, and **S** denotes the current state.
- 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 propositional space for the rocket problem is shown in Fig. 2.

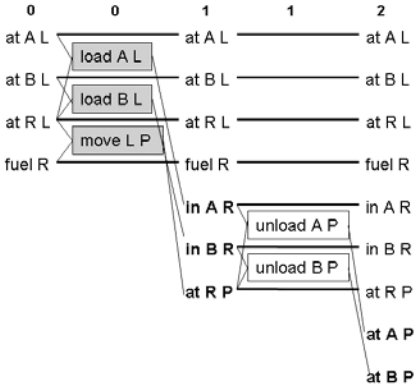

Fig. 2 Example of the relaxed propositional space

4.2 Extraction of A_{ag} and L

After building the relaxed propositional space **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.

Note that in building **R**, the case in which delays the time when the fact is achieved is not considered. Therefore, during this backward chaining, if there is a noop for achieving a fact, then the noop is first selected[2]. Otherwise, a non-noop is randomly selected because the minimum number of non-noops

over the entire level cannot be immediately derived from **R**.

In this paper, we regard the element extracted from $\mathbf{F_1}$ and $\mathbf{A_0}$ by the above backtracking strategy through *noop first heuristic with randomly selected non-noops* as $\mathbf{F_{ag}}$ and $\mathbf{A_{ag}}$, respectively. This $\mathbf{A_{ag}}$ has some similarities with heuristic actions proposed in various planners such as the helpful action of FF[2].

After deriving $\mathbf{A_{ag}}$ from the relaxed propositional space **R**, **L** which has a precedence over other subsets of $\mathbf{A_{ag}}$ is directly extracted from $\mathbf{A_{ag}}$ by the following algorithm.

(1) The candidate set of **L**, $\mathbf{L_{cand}}$ is initialized to $\mathbf{A_{ag}}$.

(2) If some action in $\mathbf{L_{cand}}$ deletes one of the precondition of other actions, then the action is removed from $\mathbf{L_{cand}}$. For example, if $o_i \prec o_j$, then $\mathbf{o_j}$ cannot become the element of **L**.

(3) If some action in $\mathbf{L_{cand}}$ deletes the fact of $\mathbf{F_{ag}}$ which is added by the action **o** in $\mathbf{L_{cand}}$, **o** is removed from $\mathbf{L_{cand}}$

(4) The noop in $\mathbf{L_{cand}}$ is removed.

(5) The candidate set $\mathbf{L_{cand}}$ becomes **L**.

In Fig. 2, $\mathbf{A_{ag}} = \mathbf{A_0}$ = {(load A L), (load B L), (move L P)}, and **L** = {(load A L), (load B L)}.

4.3 Limitations of the Situated Action Generator

In building **R**, the case in which delays the time when the fact of $\mathbf{F_{ag}}$ is achieved is not considered. However, there is some case in which the achievement of some facts of $\mathbf{F_{ag}}$ must be delayed.

The decision on which facts should be delayed requires the same task as the classical planning. Nevertheless, if this is not decided, then the agent with the situated action generator could be caught in the deadlock.

Even if the problem is deadlock free, there can be the case in which the agent only with the situated action generatoror is caught into a *cyclic routine* (e.g. $\mathbf{o_1} \prec \mathbf{o_2} \prec \mathbf{o_1} \prec \mathbf{o_2} \prec \mathbf{o_1} \prec \mathbf{o_2}$ ...). And thus, to escape the routine, the randomness should be added to the situated action generator. This paper deals with it by randomly selecting one action out of $\mathbf{A_{ag}}$ with probability ζ, or out of $\mathbf{A_0}$ with probability $1-\zeta$. Since, in our experience, the agent is rarely caught in the circular routine, $\zeta >> 1-\zeta$.

5 Situated Imagination of Lookahead States

5.1 Detecting Deadlocks

From the definition of the deadlock, the agent cannot arrive at the goal state from the deadlock state. That is, it means that some goal facts are not added to the subsequent states after meeting the deadlock. If we consider the assumptions proposed to build the relaxed propositional space, then the situated action selector can be considered to meet the deadlock when **R** contains successively equivalent fact levels (i.e. $\mathbf{F_i} = \mathbf{F_{i+1}}$).

5.2 Algorithm Sketch

As long as the problem is deadlock free where fatally wrong decisions cannot be made, the situated action selector mentioned in the previous section is complete. However, all subsequent states after applying an action should be considered to solve the deadlock problems, and thus the conventional planner is eventually required to derive them.

This section proposes a new method to make the agent with the situated action selector solve most of deadlock problems without the help of the conventional planner, which situatedly images the lookahead states. If the agent itself in the imagination is caught in a deadlock state, then before meeting the actual deadlock, the agent could avoid meeting it.

As defined before, two queues, $\mathbf{Im_q}$ and $\mathbf{Im_{sq}}$, are provided to make the agent image the lookahead states. The situations derived by *imaginarily* applying actions of **L**s are written to $\mathbf{Im_q}$. If $\mathbf{Im_q}$ is full of those situations (that is, *n* situations), then the sets of actions which achieved each situation of $\mathbf{Im_q}$ are *actually* performed by the agent. If the situated action selector meets the deadlock during filling $\mathbf{Im_q}$, then one of the situations written to $\mathbf{Im_q}$ is randomly selected, and then written to the other queue $\mathbf{Im_{sq}}$. Like the case of $\mathbf{Im_q}$, the situations derived by imaginarily applying **L**s from the randomly selected state are written to $\mathbf{Im_{sq}}$. If $\mathbf{Im_{sq}}$ is full of the situations derived by the situated action selector, then $\mathbf{Im_{sq}}$ is substituted for $\mathbf{Im_q}$, and then the sets of actions which achieved each state of $\mathbf{Im_q}$ (that is, $\mathbf{Im_{sq}}$) are performed by the agent.

Similar to the previous case, if the situated action selector meets the deadlock even during filling $\mathbf{Im_{sq}}$, then one of the situations written to $\mathbf{Im_q}$ or $\mathbf{Im_{sq}}$ is randomly selected, and then begin to fill $\mathbf{Im_{sq}}$ from the selected state again. What the agent cannot fill $\mathbf{Im_q}$ ($\mathbf{Im_{sq}}$) even after enough trials means that the agent having an *imagination depth*, *n* could not escape from the deadlock.

In fact, this algorithm is to roughly perform *n* depth first state space search walking on the lookahead states.

Therefore, as we shall see, if the deadlock is beyond *n* depth, and the agent cannot avoid meeting it, then the agent can be caught in the deadlock.

5.3 Experimental Results

We compared two classical STRIPS planners-FF v2.3[2] and LPG-td (speed)[7] with our algorithm for the situated imagination, which realizes the depth first random walk on the lookahead states. They are all implemented in C language, and no other particular library is used. To compare how fast the proposed approach can extract the currently proper action in solving deadlock or deadlock free problems, we examined the runtime which the classical planners build a plan only once at the initial state and the average of *latent* response times which the situated action selector fills the imaginary queue, $\mathbf{Im_q}$, until achieving the goal. And then, the qualities (plan length) of the plan by classical planners, and post-hoc represented plan by the algorithm for the situated imagination were compared.

Various planning domains were used in our experiments, and all of the experimental results were similar to each other. In this paper, we introduce the experimental results performed on three domains: the Airport domain, the Schedule domain, and the Freecell domain. The problems included in each domain can be deadlock or deadlock free.

The results are shown in Fig. 3 – Fig. 8.

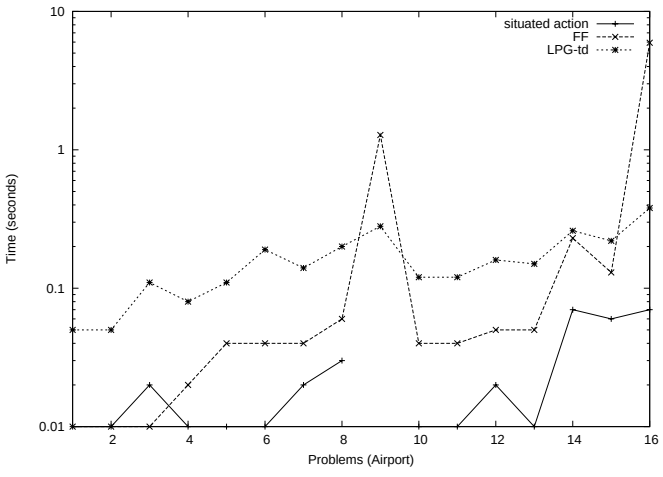

Fig. 3 Time on Airport problems

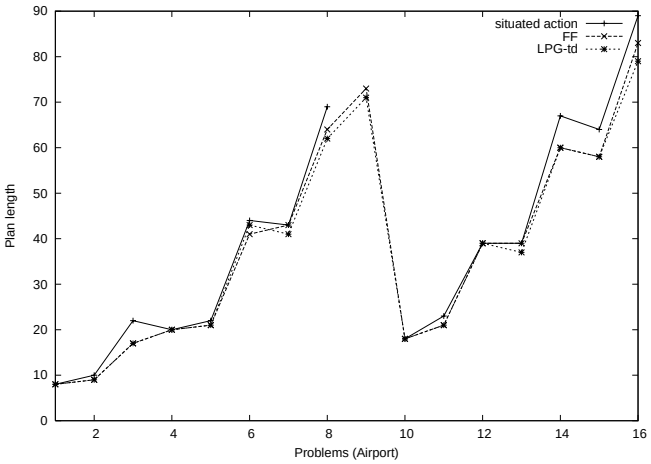

Fig. 4 Plan length on Airport problems

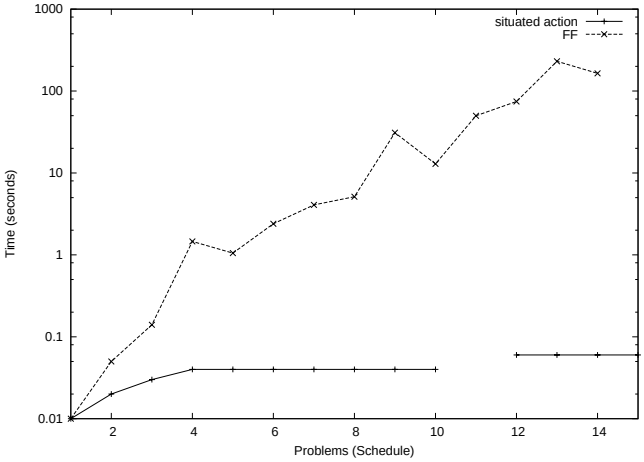

Fig. 5 Time on Schedule problems

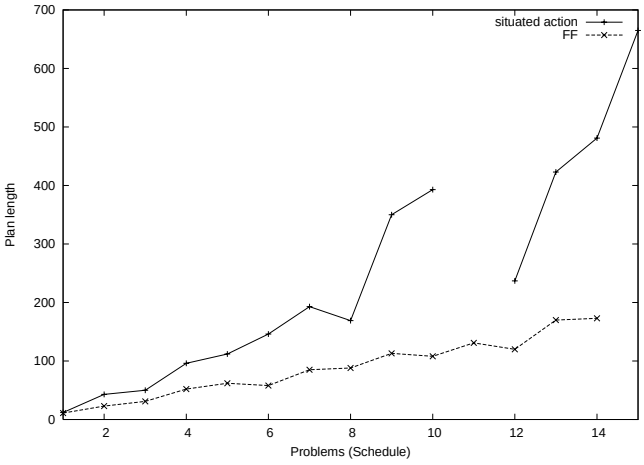

Fig. 6 Plan length on Schedule problems

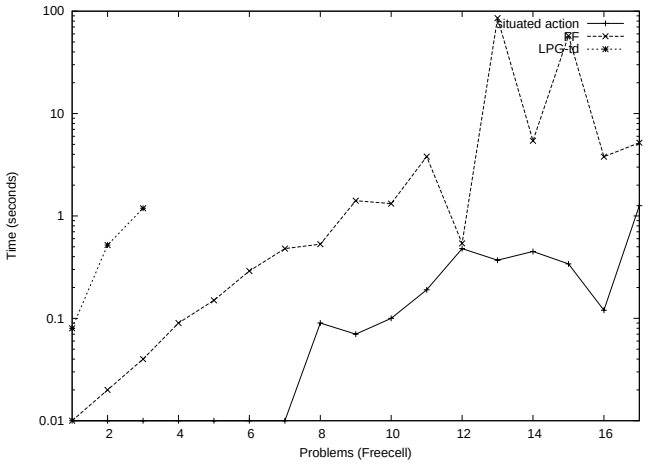

Fig. 7 Time on Freecell problems

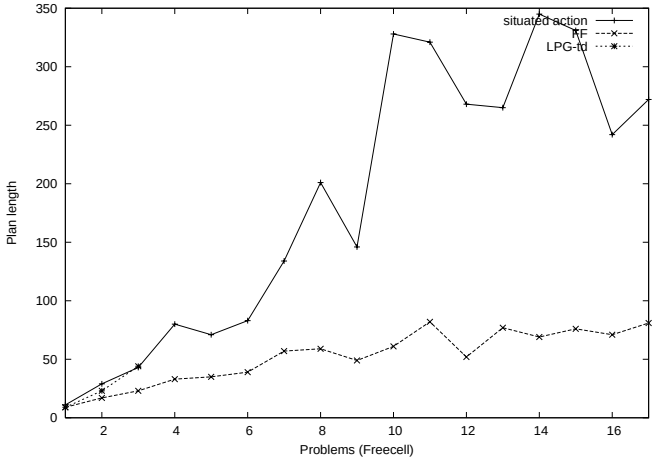

Fig. 8 Plan length on Freecell problems

5.4 Discussion

Although the proposed approach for situatedly imagination of lookahead states could solve much of the deadlock problems, some of the problems could not be solved by the proposed approach. These cases where the proposed approach could not solve the problem were caused by the following:

- Unavoidable Deadlocks: Since the agent has the limited imaginary depth, if the deadlock is beyond the depth at certain situation, and the agent cannot avoid to meeting it from the situation, then the agent can be caught into the deadlock.
- Excessively Time Consuming: In the case where the poor paths of actions, which are not included in the optimal path, include a lot of deadlocks, it could take much time to avoid meeting them (i.e. requires much imagination for finding safe lookahead states), even than conventional planners.

However, in the case where the proposed approach could derive the result, the response time was much faster than the runtime of the conventional planner, and its quality was mostly acceptable.

6 Conclusion

We first presented the novel action selector 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 were proposed: a fact is not deleted by the action, and a fact which could be achieved at a certain time is regarded to be achieved at that time. Although these assumptions make the agent derive the situated action even in symbolic processing, the pure situated action selector cannot deal with the deadlock domain where fatally wrong decisions can be made. To allow the situated action selector to solve more general problems including deadlocks, we proposed the algorithm, which can situatedly images the lookahead states. If the agent itself in the imagination is caught in a deadlock state, then before meeting the actual deadlock, the agent could avoid meeting it. The experimental results showed the very fast response to the situation solving various planning problems. We believe learning of the problem structure can enhance the quality of the proposed method, and moreover could also give a clue to understand intelligence itself.

References:

[1] Avrim L. Blum and Merrick L. Furst “Fast Planning Through Planning Graph Analysis”, *Artificial Intelligence*, vol. 90, 1997

[2] Jörg Hoffman and Bernard Nebel, “ The FF Planning Systems: Fast Plan Generation Through Heuristic Search”, *Journal of Artificial Intelligence Research*, vol.14, 2001

[3] Ronald C. Arkin, Behavior-Based Robotics, The MIT Press, 1998

[4] Lucy Suchman, Plans and Situated Actions - The Problem of Human-Machine Communication, Cambridge University Press, 1987

[5] Rodney A. Brooks, “A robust layered control system for a mobile robot”, *IEEE Journal of Robotics and Automation*, vol. 2, 1986

[6] Alonso H. Vera and Herbert A. Simon, Situated Action: A Symbolic Interpretation, *Cognitive Science*, vol. 17, 1993

[7] Alfonso Gerevini, Alessandro Saetti, and Ivan Serina, “Planning through Stochastic Local Search and Temporal Action Graphs in LPG”, *Journal of Artificial Intelligence Research*, vol. 20, 2003