The Seventeenth International Symposium on Artificial Life and Robotics 2012 (AROB 17th '12),
B-Con Plaza, Beppu, Oita, Japan, January 19-21, 2012

Importing dynamic planner to BDI agent creating flexible decision-making of policies for selecting robot actions in real world

Megumi Fujita[1], Hiroko Katayama[1], Yuko Ojima[1], and Naoyuki Nide[2]

[1] Graduate School of Nara Women's University, Nara, Nara 630-8506, Japan
[2] Nara Women's University, Nara, Nara 630-8506, Japan
(Tel: +81-742-20-3555, Fax: +81-742-20-3555) (saboten@ics.nara-wu.ac.jp)

Abstract: Our aim is to create a more intelligent form of control for robots that can act autonomously for problem solving in the dynamic environments. The ability to select and modify the action decision policies to achieve the given goals in the most appropriate way as possible, and the easiest and most efficient way of implementation of such policy controls is required. We propose a flexible method for selecting policies of action decision in this paper, using a dynamic planner as the mechanism for determining the policies for action decision making. We proved through experimentation that when a robot cannot achieve its goal using a specific policy, it can modify the given policy to achieve that goal with the use of our method. In particular, for robots in the real world, the error of beliefs due to a false recognition of the sensors may be the reason why a robot cannot achieve its goal, although this situation will not come to light in any simulation. Our method is effective for such situations.

Keywords: Autonomous agent, Intelligent robot, BDI, Real world oriented, Planner

1 INTRODUCTION

With the recent development of advanced robotics, there is a growing expectation for the construction of intelligent robots that can act autonomously by generating plans for achieving their goals in the real world. In particular, such a robot would require the ability to cope with various changes in their environments; for example, when it cannot achieve its goal by using the current plan because of an unexpected change in the real world situation, it must create a new policy of action for achieving its goal.

Our aim is to create such robots using autonomous agents called BDI agents. Since it is now possible to inexpensively and easily obtain small robots, we are currently attempting to equip them with BDI agents and assist them to be able to resolve problems in the real world.

As described in Sec.2, a BDI agent has a native mechanism called "deliberation", which selects a plan to execute immediately from among multiple (currently active) plans for achieving the agent's goals. This mechanism works like the task switching mechanisms of operating systems, and assists the agent in switching goals and gives them a means for easily achieving them. Furthermore, agent designers can separate the codes for switching plans from the codes of the plans themselves. These are part of why the BDI agent is suitable for creating robots that can autonomously solve problems.

However, it would be inadequate to write a deliberation algorithm using the hand-coding manner to achieve a highly flexible switching of plans. A higher-level means to provide a deliberation algorithm is required to infer an appropriate plan depending on the given circumstances of the agents.

In this paper, we propose enhancing the deliberation by introducing a dynamic planner. Human beings normally hold some sort of policy as a higher-level principle for selecting plans for achieving his/her goals, and modify it as necessary to change his/her behavior. Our proposal is to implement this method on an agent using a dynamic planner to decide it's given policy, and therefore, naturally and flexibly enhances the mechanism of the deliberation.

There have been several proposals introducing a planner into a BDI agent ([1, 2, 3]), but they use the planner as a tool that generates plans and provides them to the agents. On the other hand, it is a characteristic point that in our method the dynamic planner works as an effective tool for selecting the action policy of the agent in a dynamic environment. In Sec.4.4, we show that when a robot implemented with this method in our experiment cannot achieve its goal by using the selected plan based on its current action policy, it can change the policy using the dynamic planner and then achieve its goal. Our proposal is effective in that such a property installed in robots is desirable for robots in the real world.

2 BDI AGENTS

A BDI agent[4] is a type of autonomous and rational agent that explicitly has three mental attitudes, beliefs, and desires and intentions (written as B, D and I respectively), and uses their temporal changes in the decision-making process.

When a new goal (desire) arises, a BDI agent selects a usable plan to achieve that goal from the plan library using its belief base ("practical reasoning"), and forms an intention to commit to executing that plan, then adds it to the set

of current intentions (active plans). Next, the agent selects a particular intention to execute right now using the belief base (by using a mechanism called "deliberation"), and takes one step to execute it. Subsequently, the agent perceives the environment to update its beliefs, and then updates its intention set (e.g., intentions that have already been achieved are discarded; and if a goal is judged to be impossible to achieve by the current intention for that goal, then that intention is discarded and an alternative intention to achieve the goal is selected[1]). By repeating this sequence of processes as a loop, the agent attempts to achieve its goals.

We use Jason[5], a well-known platform for implementing BDI agents.

3 COOPERATION WITH DYNAMIC PLANNER

In this section, we explain the design of the dynamic planner we propose for use in cooperation with a BDI agent (hereafter called "BDI cooperative dynamic planner"), and explain how it cooperates with the agent.

The BDI cooperative dynamic planner is created by modifying SHOP2[6], a famous hierarchical planner. It uses the planning engine of SHOP2.

Since SHOP2 is designed for static problem solving, when given a goal, it completely decomposes that goal down to a sequence of atomic actions. However, in a dynamic environment, it is impractical to decompose the goal into atomic actions because once the plan is made it may become difficult to proceed with it due to a change in circumstances.

A human being, in a dynamic environment, first makes a rough plan to achieve his/her goal, and when the time comes, he/she decomposes and substantiates the plan more minutely so that it can be directly executed. The BDI cooperative dynamic planner also does the same thing. So, it stops decomposing the plan in the first step of the decomposition process, and accumulates it as a sequence of subgoals. Then, the subgoal at the top of the sequence of subgoals is a goal for the agent, and at this time, the planner decomposes that subgoal into a set of tasks and gives them to the agent who uses them as an action policy for achieving its goals.

If the agent finishes their goal using this policy, it informs the planner, and the planner gives the next subgoal in the sequence to the agent as the next policy. On the other hand, if the agent cannot accomplish the goal using this policy, the planner re-plans under the current condition to create a new action policy and then gives it to the agent.

We implement our planner as a separate program to Jason. The agent running on Jason invokes a sub-process of the planner that stays active until the agent program terminates, and it interacts with the agent using interprocess communications.

[1] Actually, the alternative intention is not selected in this step, but in the next round of a loop.

4 EXAMPLE USING ROBOT

We explain how our proposal is efficient at controlling a robot in the real world using the experiment results in this section.

4.1 Example problem

The example problem for the robot we used in our experiment is to search for treasure in a maze on a grid (Fig.1). Initially, the agent does not have any knowledge about the locations of walls, so each time the agent perceives the existence of a wall, it adds the information about the position of the wall to its belief base. When the robot finds the treasure, it returns back to the initial position. Note that in this experiment, the robot may occasionally collect erroneous information about the positions of walls due to errors in perception caused by the noise of its sensor.

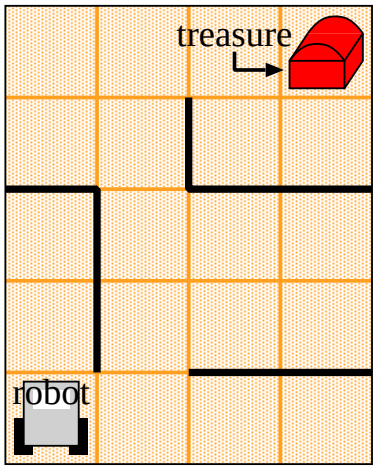

Fig. 1. Map used in our experiment

At first, the robot searches for the treasure while relying on its beliefs about the positions of walls, since it incurs some cost to perceive the walls. However, there are possibilities that the robot is unable to find the treasure due to errors in its beliefs.

If this happens, the robot has to switch to another way of searching while re-perceiving the walls instead of trusting the beliefs about the walls. After that, when the robot finds an unexplored square, it judges that it has escaped the situation of inability to find the treasure, and should switch back to the original way of searching.

4.2 Implementation of atomic actions

The robot we used was a MINDSTORMS NXT developed by the LEGO company. The robot in this experiment is a mobile robot with two front wheels and one rear wheel. It has an ultrasonic sensor in the front to measure the distance to an obstacle (Fig.2).

The NXT fundamentally has only low level commands such as specifying the torque output and receiving an integer signal from the sensor. However, we prepared a higher level of actions, such as "proceed to the next square", "rotate by 90 degrees" and "perceive whether a wall exists", and

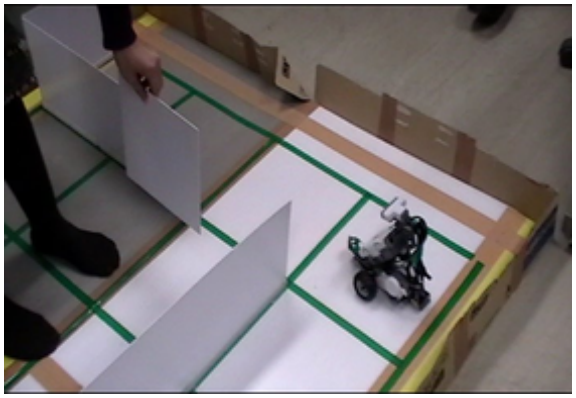

Fig. 2. Schematic layout of our experiment

used them as atomic actions. The merit of this method is that we can separate the low level controls of robots from the plan description of agents. For example, low level robot controls are accompanied by errors due to mechanical inaccuracies with the motors and sensors. If some progress is made with this type of problem, the efficiency of our proposal in this paper improves in conjunction with it, although we do not deal with this topic in this paper.

4.3 Flow of agent program

Let us assume that the agent (robot) has the initial goal of "find the treasure and return to the initial position", and requests an action policy for the subgoal "find treasure" from the planner. The planner creates a "normal search" policy and gives it to the agent. Then, the agent selects a "searching by normal strategy" plan and forms it into an intention.

According to this plan, the agent searches for the treasure while adding (and using) its beliefs about the positions of walls, which is acquired by perception. In this stage, the agent does not repeatedly perceive the places where a wall is believed to exist.

If zero unexplored squares remain, this plan fails. In this case, the agent asks the planner again, and the planner returns a "re-search" action policy. The agent selects a "searching while considering the possibilities of errors with past perceptions" plan, and forms it as a new intention. By following this plan, the agent continues searching while re-perceiving the walls (without relying on the beliefs about the places of walls).

When the agent finds an unexplored square, again it asks the planner for a "normal search" policy. The agent again selects the "searching by normal strategy" plan, makes it an intention, and continues searching in the former way.

After the agent finds the treasure, a "return to the initial position" plan is executed as the next subgoal, which we omit in this paper.

We also omit the actual codes of the agent programs due to space limitations.

4.4 Experimental result

We performed our experiment on a 4 5 grid (Fig.1, 2).

Starting with Fig.1, the agent begins searching using the "normal search" policy described in Sec.4.3.

Meanwhile, to create a situation in which the agent wrongly believes in the existence of a wall, we put an extra wall at a given place in the maze (Fig.3).

Then, the agent continues searching with the erroneous belief in the existence of a wall. When the agent comes that place again, it still believes that there is a wall even after the fake wall has been removed, and does not repeatedly try to perceive it (Fig.4).

The agent goes around all the unexplored squares, and finds that no new routes remain, and thus, it cannot find the treasure. So, the agent asks the planner and switches to the "re-search" policy described in Sec.4.3, and continues searching while re-perceiving in all directions around it (Fig.5).

As a result, the agent finds a new route to the yet unexplored areas (Fig.6). Then, the agent asks the planner, goes back to the "normal search" policy and continues searching until it finds the treasure.

We conducted another experiment in which we place the fake wall on the grid twice. The robot was able to behave properly and find the treasure.

In this regard, we concluded that the robot can deal with the problem of misperception.

5 DISCUSSION

As described in Sec.1, robots in the real world have to change their behaviors according to the various (sometimes unexpected) changes in environments. When an ad-hoc technique, such as the traditional hand-coding method, is used to cope with such problems, there is a tendency for the codes of the plans themselves and the meta-level codes for the switching plans to get mixed up, and the maintainability is then lost.

The BDI agent, as an agent development approach, in comparison with methods such as hand-coding methods, can be regarded as the framework for describing agents using a kind of high-level language. Plans can be described declaratively and in a mutually independent manner, and then explicit codes for switching plans are not necessary. These features are essential for enabling flexible changes in behaviors. Furthermore, the notion of intentions in BDI agents works as a higher-level action decision mechanism and this provides a stable and consistent behavior towards achieving the agent's goals.

In addition, in our method, by leaving the "deliberation" part to the planner, there is no need to procedurally write the deliberation routine, and by only receiving information about the current circumstances from the agent, we can select an

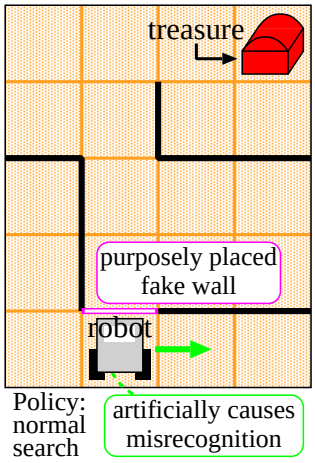

Fig. 3. Producing false recognition of walls

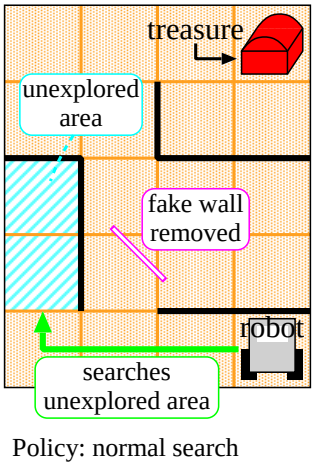

Fig. 4. Continuation of search based on inaccurate belief

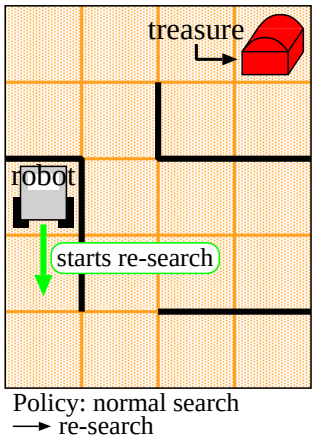

Fig. 5. Being stuck, robot asks planner and switches action policy

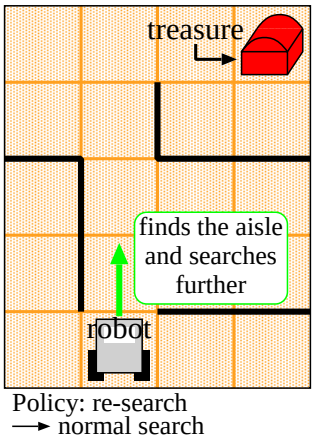

Fig. 6. When unexplored place is found, robot asks planner again, switches policy, and continues searching

appropriate action policy to commit as a plan. Also, when using an efficient planner in large-scale applications, we can expect improvement in the efficiency of selecting plans compared with that from the hand-coding method (though such advantage may be unrealistic in smaller applications such as the experiment discussed in this paper).

Moreover, our method is also a complement to the weakness of Jason as a base of implementation for BDI agents. In Jason, when a plan fails once, the agent can recover by executing a plan for handling that failure, but if the plan for failure fails again, the agent cannot continue to achieve its goal. On the other hand, by using our method, the robot can recover from multiple failures (e.g., misperceptions), as described in Sec.4.4.

Yet another merit of BDI agents is that they have a formalization by using a modal logic named the BDI logic[7], in which not only the demands for the agents but also the behaviors of the agents can be integrally argued. We propose an extension to this by importing the formalization of the planner[8], so there is the future possibility for the formal verification of our method.

6 CONCLUSION

We proposed enhancing the deliberation in BDI agents by introducing a dynamic planner so that an agent can flexibly change its action policy for achieving its goals, and showed that our method is efficient enough for use in robots in the real world through experimentation.

Our future issues include applying this method to a larger task, and introducing the formal verification method introduced in Sec.5.

REFERENCES

[1] A. Walczak, L. Braubach, A. Pokahr, and W. Lamesdorf, "Augmenting BDI Agents with Deliberative Planning Techniques," *The 5th International Workshop on Programming Multiagent Systems (PROMAS-2006)*, pp. 113–127, 2006.

[2] L. de Silva, A. Dekker, and J. Harland, "Planning with Time Limits in BDI Agent Programming Languages," *ACM International Conference Proceeding Series*, vol. 240, pp. 131–139, 2007.

[3] F. R. Meneguzzi, A. F. Zorzo, M. D. C. Móra, and M. Luck, "Incorporating planning into BDI systems," in *Proc. of SCPE 2007*, 2007.

[4] M. P. Singh, A. S. Rao, and M. P. Georgeff, "Formal method in DAI: Logic-Based Representation and Reasoning," in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pp. 331–376, The MIT Press, 1999.

[5] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[6] D. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "SHOP2: An HTN Planning System," *Journal of Artificial Intelligence Research*, vol. 20, pp. 379–404, 2003.

[7] A. S. Rao and M. P. Georgeff, "Modeling Rational Agents within a BDI-Architecture," in *Proc, of International Conference on Principles of Knowlegde Representation and Reasoning*, pp. 473–484, 1991.

[8] N. Nide, M. Fujita, and S. Takata, "Formalizing combination of dynamic planner and BDI agent," in *Proc. of Joint Agent Workshop and Symposium 2010*, 2010. (in Japanese).