Multi-Agent Planning Method Using Affordances from Environment

Sawako Tajima
Graduate School of Sci. and Tech.
Keio University
Yokohama, Kanagawa, Japan
email: sawako_tajima_0727@keio.jp

Daiki Takamura
Graduate School of Sci.and Tech.
Keio University
Yokohama, Kanagawa, Japan
email: d.takamura.1102@keio.jp

Daiki Shimokawa
Graduate School of Sci. and Tech.
Keio University
Yokohama, Kanagawa, Japan
email: jump1204@keio.jp

Reo Kobayashi
Graduate School of Sci. and Tech.
Keio University
Yokohama, Kanagawa, Japan
email: reokoba@keio.jp

Reo Abe
Faculty of Sci. and Tech.
Keio University
Yokohama, Kanagawa, Japan
email: reo1431@keio.jp

Satoshi Kurihara
Faculty of Sci. and Tech.
Keio University
Yokohama, Kanagawa, Japan
email: satoshi@keio.jp

***Abstract*—For autonomous actors, such as robots to achieve their goals while operating adaptively in a dynamically changing environment, they need to accurately recognize changing conditions from moment to moment. Planning research for finding a sequence of actions to achieve a goal has a long history, and in recent years, machine learning has become the mainstream method for finding the optimal sequence of actions. However, it is difficult to deal with unexpected situations using this method, and in such cases, the real-time performance is lost due to the blind search for a sequence of actions that will achieve the goal. Living organisms, such as ourselves, have learned how to avoid blind search by perceiving environmental affordances. In this paper, we propose a method for robots to use affordances to recognize their situation accurately and to seek a sequence of actions to achieve their goals efficiently. Affordances are common sense in an individual situation, i.e., tacit knowledge, and conventionally can only be constructed manually, which has the limitation that they cannot be scaled. However, large-scale language models that have recently emerged may contain tacit knowledge, and we have developed a method for extracting this tacit knowledge. In this study, we incorporated this method into a multi-agent planning system that is highly adaptive to dynamic environmental changes. We confirmed that a sequence of actions can be efficiently obtained to achieve a goal by using affordances.**

***Keywords*—multi-agent planning; action selection; affordance.**

I. INTRODUCTION

Planning is an essential area of Artificial Intelligence (AI) research and continues to be actively studied. In particular, planning, which is highly adaptive to dynamic environments, is a core technology for realizing next-generation AI with high autonomy and versatility.

To realize this capability, it is necessary to distinguish between immediate planning, which enables instantaneous action selection in response to changes in the environment, and deliberative planning, which outputs a highly optimal action sequence even if it takes time to achieve a goal that requires deliberation, given to a robot equipped with the planning system.

A method to construct the planner as a multi-agent type has been proposed to achieve immediacy and deliberateness. In multi-agent planning, one agent is in charge of one operant in Stanford Research Institute Problem Solver (STRIPS) [1], and the agents cooperate to generate a sequence of operants that transitions from the initial state to the goal state. Subsumption Architecture (SA) proposed by Brooks [2] is well known as an existing study. SA can hierarchically connect lower-level reflective functional modules and higher-level deliberative functional modules to use immediacy and deliberateness as appropriate. However, SA exclusively selects between immediate and deliberative planning, and the two cannot cooperate or be more adaptive to the environment.

On the other hand, Agent Network Architecture (ANA) proposed by Maes [3] [4] is an activation propagation-based behavioral network architecture. This method allows more fine-grained coordination between immediate response and deliberation than SA by allowing agents to interact with each other in the form of activation propagation.

However, in planning, a search is conducted to obtain action sequences, and the more versatile the planning is, the more the search for the optimal sequence that achieves the objective from among a vast number of combinations of action sequences becomes burdensome and real-time performance declines. In planning, many methods have been proposed in recent years to find the optimal action sequence by machine learning. However, even with this method, unexpected situations are difficult to deal with. In such cases, blind search is achieved for a sequence of actions that achieves the objective, which results in a loss of real-time performance.

Living organisms, such as ourselves, can avoid blind search by perceiving environmental affordances. Affordance is 'the opportunity for action that an organism receives from a particular object or environment' [5]. For example, if a human perceives a door with a handle, he/she can obtain the affordance of pulling from this door to open it. Affordances are common sense in an individual situation, i.e., tacit knowledge, and conventionally can only be constructed manually, which has the limitation that they cannot be scaled.

However, we believe large-scale language models that have recently emerged may contain tacit knowledge. If this can

 ISBN: 978-1-68558-053-7

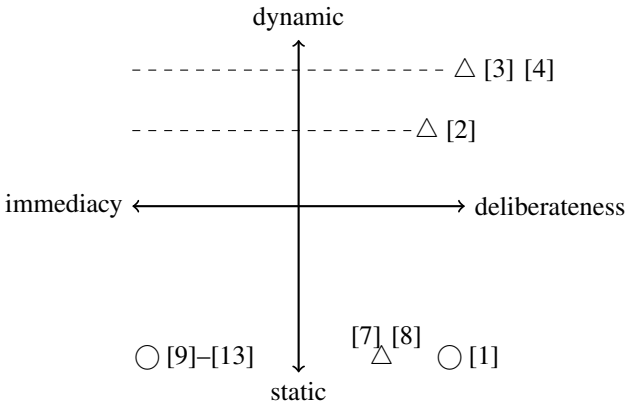


Figure 1. Comparison of related works.

be automatically extracted, affordances can be easily used. Our research aims to incorporate affordances into multi-agent planning and use affordances to obtain a sequence of actions to achieve a goal efficiently.

Naturally, robots cannot receive affordances from the environment, so common sense and tacit knowledge have needed to be manually set for each task given to the robot. Therefore, the greater the generality of planning, the greater the amount of tacit knowledge to be set, and the more difficult it becomes to set by hand.

To address this issue, we focus on large-scale language models such as Generative Pre-trained Transformer 3 (GPT-3) [6] that have recently emerged. Since this model was constructed by learning a huge amount of linguistic information, it can be said to contain tacit knowledge and common sense. In this study, we propose a method for automatically extracting affordances from this large-scale language model in accordance with the situation and using them for planning. The proposed method enables the robot to understand the dynamically changing environment by understanding its situation and to plan adaptively to achieve its goals while avoiding unnecessary searches for sequences of actions.

In Section 2, we compare related works in planning, and in Section 3, we introduce our proposed method. Section 4 describes the evaluation conducted to confirm the significance of the proposed method, and Section 5 describes the results and the discussion. Finally, Section 6 presents the conclusion and future work.

II. Related Work

In this section, we present related work in planning. Figure 1 plots each method based on the characteristics. The horizontal axis in the figure represents immediacy and deliberateness, and the vertical axis represents the ability to adapt to a dynamic environment. The plotted symbols are marked ◯ for methods that take a long time but output an optimal action sequence and △ for methods that output a suboptimal action sequence within a limited time (i.e., with ”anytime” property).

The most typical method of classical planning is STRIPS [1]. STRIPS outputs a sequence of actions to transform an initial state into a goal state. Although STRIPS can output an optimal action sequence, it is problematic because it requires a significant amount of time for search. Research to improve the efficiency of the search is still ongoing. Shen et al. [7] introduced the concept of hypergraphs to output a suboptimal action sequence in a limited amount of time. Domshlak et al. [8] detected state symmetries within A* cost-optimal planning, allowing them to prune a larger portion of the search space. However, these methods are difficult to adapt to dynamic environments because they require planning to be redone from scratch when the environment changes, which is computationally very expensive.

In recent years, machine learning has been actively used to find optimal action sequences. Many methods have been proposed in deep reinforcement learning based on DQN [14], such as Rainbow [9], APE-X [10], R2D2 [11], and NGU [12]. In particular, Badia et al. [13] have outperformed standard human performance on all Atari games. The advantages of such machine learning-based methods are that they can automatically extract features from large amounts of observed data and achieve high accuracy in the environment and task for which they were trained. However, because learning is based on observed data, they are not good at dealing with dynamic environments, such as changing the priority of tasks in accordance with changes in the environment when there are multiple objectives. Therefore, learning priorities by considering the immediacy and deliberateness of multiple objectives in every situation is not practical because the learning load is too high.

A further method is to construct the planner as a multi-agent type. In a Multi-Agent Planner (MAP), each agent is responsible for an individual operant in STRIPS [1] and generates a sequence of operants that transition from the initial state to the target state through agent coordination. SA [2] can connect lower-level reflective functional modules and higher-level deliberative functional modules hierarchically to use immediacy and deliberateness as appropriate. However, SA exclusively selects between immediate and deliberative planning, and the two cannot cooperate or be more adaptive to the environment. Maes [3] [4] proposed an activation-propagating Agent Network Architecture (ANA). In ANA, an agent will attempt to act if the state of the environment matches the conditions under which it will act. Each goal of the robot tries to activate an agent to achieve it, and the agent in turn tries to activate another agent to activate itself further. Therefore, ANA can use immediacy and deliberateness more flexibly than SA by allowing agents to interact in activation propagation. However, ANA has the disadvantage of requiring manually designed agents, making it difficult to scale. MAP continues to be studied today, with research using ANA to make conversational decisions with users [15] and research combining state-based and action-based frameworks to control dynamic behavior [16]. There are also studies using multi-agents for natural disaster modeling [17] and inventory modeling [18] to increase prediction accuracy while reducing reliance on the experience of experts. Thus, although MAP is

 ISBN: 978-1-68558-053-7

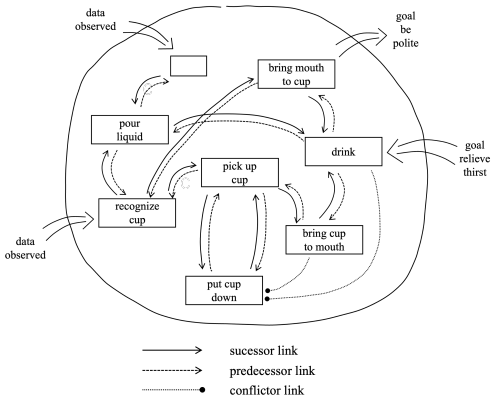


Figure 2. The network of ANA [3].

inferior to other methods in terms of optimality, it is superior at responding to dynamic environments.

III. Proposed Method

We consider that using affordances can solve the disadvantage of the MAP. Humans similarly use affordances to reduce the search space, reducing the brain's load. This is the inspiration for the proposed method. In this study, we propose extending the MAP method that utilizes affordance information. We also propose a method for extracting affordance information from large-scale language models.

A. Agent Network Architecture (ANA)

In this paper, ANA [3] [4] is used as the baseline for MAP because ANA can use immediacy and deliberateness flexibly. This section describes the architecture of the ANA and the algorithm for action selection.

ANA has agents equivalent to STRIPS [1] operants and consists of a network connecting them. Each agent has a condition list, add list, and delete list. The condition list contains states necessary for execution, the add list contains states to be added to the current environment after execution, and the delete list contains states to be deleted from the current environment.

Figure 2 shows the ANA network, consisting of a network of agents connected to each other by three links: predecessor, successor, and conflictor. A predecessor link is a link between two agents when an agent satisfies the preconditions of an agent (has a condition list state in the add-list). A successor link is a link between two agents when there is an agent that can be executed after the execution of an agent (has an add list state in the condition list). A conflictor link is a link between two agents when an agent becomes non-executable (has a condition list state in the delete list) after an agent is executed.

Through these links, activation propagation from the environment, goal, and protected goal (external spreading) and activation propagation between agents (internal spreading) are performed. The amount of stimulus accumulated by activation propagation is called the activation level. Each agent has a threshold, the minimum activation level required for activation. The agent is activated and executed when the activation level exceeds the threshold. This sequence of events is called selection. The agent that can be executed from the current environment is selected and executed by repeating the selection process. The following is a formula for the activation level of agent y at time t.

$$\begin{aligned}\alpha(y,t) = &\, decay(\alpha(y,t-1)(1-active(y,t-1)) \\ &+ input\_from\_state(y,t) \\ &+ input\_from\_goals(y,t) \\ &- taken\_away\_by\_protected\_goals(y,t) \\ &+ \sum_{x,z}(spreads\_bw(x,y,t) \\ &\quad + spreads\_fw(x,y,t) \\ &\quad - takes\_away(z,y,t)))\end{aligned} \tag{1}$$

The $decay()$ that runs throughout (1) is a function that normalizes the activation level of all agents in the network. $\alpha(y,t-1)(1-active(y,t-1))$ in the first line indicates whether the activation level of agent y at time $t-1$ is used at time t. The agent active at time $t-1$ is calculated with $active(y,t-1)=1$, and the agent not active at time t is calculated with $active(y,t-1)=0$.

The parameters used in the formulas after the second line of (1) are shown below.

- ϕ, the amount of activation energy injected by the state per true proposition
- γ, the amount of activation energy injected by the goals per goal
- δ, the amount of activation energy taken away by the protected goals per protected goal
- $S(t)$, the set of states that are true at time t
- $M(j)$, the set of agents with state j in condition list
- $A(j)$, the set of agents with state j in add list
- $U(j)$, the set of agents with state j in delete list
- c_y, condition list of agent y
- a_y, add list of agent y
- d_y, delete list of agent y

The second line of (1) represents the amount of stimuli that agent y receives from the environment, which is calculated by the following equation. In (2), # represents the size of the set or list.

$$input\_from\_state(y,t) = \sum_j \phi \frac{1}{\#M(j)} \frac{1}{\#c_y} \tag{2}$$

The third line of (1) represents the amount of stimulus that agent y receives from the goal and is calculated by the following equation.

$$input\_from\_goals(y,t) = \sum_j \gamma \frac{1}{\#A(j)} \frac{1}{\#a_y} \tag{3}$$

 ISBN: 978-1-68558-053-7

The fourth line of (1) represents the amount of stimuli that agent y receives from the protected goal, which is calculated by the following equation. Protected goal refers to a goal that has already been achieved or should be protected.

$$taken\_away\_by\_protected\_goals(y,t) = \sum_j \delta \frac{1}{\#U(j)} \frac{1}{\#d_y} \quad (4)$$

The fifth line of (1) represents the amount of stimuli sent from agent x to agent y via the predecessor link and is calculated by the following equation. If agent x is not executable at time t (the current environment does not satisfy the condition list of agent x), the stimulus is sent to agent y.

$$spreads\_bw(x,y,t) = \sum_j \alpha_x(t-1) \frac{1}{\#A(j)} \frac{1}{\#a_y} \quad (5)$$

The sixth line of (1) represents the amount of stimuli sent from the successor link from agent x to agent y, which is calculated by the following equation. If agent x is executable at time t (the current environment satisfies the condition list of agent x), then stimuli are sent to agent y.

$$spreads\_fw(x,y,t) = \sum_j \alpha_x(t-1) \frac{\phi}{\gamma} \frac{1}{\#M(j)} \frac{1}{\#c_y} \quad (6)$$

The seventh line of (1) represents the amount of stimuli sent from the conflictor link from agent z to agent y and is calculated by the following equation. Stimuli are sent to agent y when $\alpha_x(t-1) > \alpha_y(t-1) \vee (\exists i \notin S(t) \cap c_y \cap d_x)$ is satisfied.

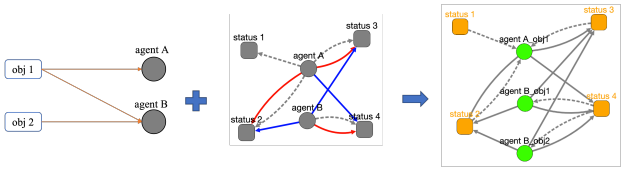
$$takes\_away(z,y,t) = max(\sum_j \alpha_x(t-1) \frac{\delta}{\gamma} \frac{1}{\#U(j)} \frac{1}{\#d_y},\ \alpha_y(t-1)) \quad (7)$$
$$where\ j \in c_x \cap d_y \cap S(t)$$

B. Use of Affordances

Agents are expressed as a combination of a verb and a noun, as in “pick up cup”, but it is common to describe a noun as a variable, as in “pick up X”. In the MAP we are building, there are agents such as “wash X with Y (in right hand) in Z” (where X is the object, Y is the way, and Z is the location variable).

Since the noun “cup” affords verbs such as “drink”, “grab by hand” and “wash” an agent such as “throw X” can be excluded from activation candidates from the beginning. However, if it does not use affordance, all verbs, including the object, must be considered candidates for activation.

Thus, if an agent such as “X” is “cup” and “pick up cup (on Z) with right hand” tries to activate it, it will have to consider all possible candidates for activation of Z if it does not use affordance. However, by using affordances, the agent can try to activate only those Zs that are appropriate to the situation, such as “table” or “rack”, thus avoiding unnecessary search.



(a) Affordance Network (b) Variable Agent Behavior Network (c) Agent Behavior Network

Figure 3. The architecture of the proposed method.

As described above, an agent containing variables such as X, Y, and Z is defined abstractly in advance, and actual concrete objects are assigned to the variable parts by using the affordance information described below. In this process, a generic agent with variables dynamically generates an agent representing each concrete action applicable to planning, and then planning is executed.

Figure 3 shows the architecture of the proposed method. Figure 3(a) shows Affordance Network, the network of objects and agents afforded by those objects connected by edges. Figure 3(b) shows Variable Agent Behavior Network, the MAP network of agents and states with noun parts as variables. Figure 3(c) shows Agent Behavior Network, MAP generated by connecting (a) and (b).

C. Affordance Extraction Methods

The sentences output from a large-scale language model such as GPT-3 express the knowledge contained in the large number of sentences humans have spun. In other words, outputting sentences related to a particular object is equivalent to outputting knowledge. In this study, we analyzed the output of the sentence by a large-scale language model grammatically. We call this network the affordance network.

An affordance network is constructed from a large-scale language model as follows.

1) Output sentences related to nouns using GPT-3.
2) Perform dependency parsing using CoreNLP [19] to extract dependency relations between verbs and nouns.
3) Build an affordance network from the extracted dependency relations.

In Step 1, GPT-3 outputs sentences using the following template of imperative sentences (prompts). By assigning the target nouns to {noun} in the prompt template, GPT-3 outputs a sentence containing knowledge about each noun. For example, a prompt with “cake” assigned to {noun} would produce a sentence like “He cut a cake with a knife.”

- Please come up with some story.
  Keyword: {noun}
- Talk about your memories.
  Keyword: {noun}
- Please write a diary with yourself as the subject.
  Keyword: {noun}

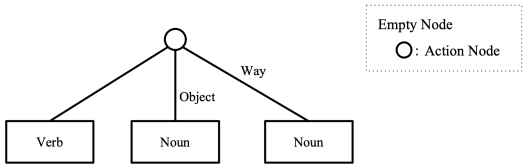


Figure 4. Affordance Network Structure.

In Step 2, the output of the sentence in Step 1 is subjected to dependency parsing using CoreNLP. By performing the dependency parsing, it is possible to identify the relationship between the verb and its object noun and the relationship between the verb and its way noun. For example, “cake” and “knife” are identified as the object and way of the verb “cut,” respectively.

In Step 3, the network is constructed on the basis of the verb-noun affiliation relations identified in Step 2. Figure 4 shows the network.

IV. Evaluation

This section evaluates the introduction of the concept of affordance into MAP.

A. The Way to Use Affordance

Compared with SyntagNet [20], the proposed method better handles affordances inherent in the combination of multiple objects. For example, consider the output verb set V_{AB} when simultaneously observing object O_A and object O_B.

SyntagNet takes an object as input, outputs the verb set recalled from that object alone and outputs the verb sets V_A and V_B from O_A and O_B, respectively. On the other hand, the combination of O_A and O_B cannot directly be used to output a verb set, but only a union set $V_A \cup V_B$ or an intersection set $V_A \cap V_B$ can be given as an approximation of V_{AB}, for example.

The proposed method better outputs the set of verbs recalled when observing multiple objects simultaneously. The output of GPT-3, which has acquired a vast amount of human experience as knowledge, must include sentences that describe the experience of performing actions on one object using other objects as tools. In other words, GPT-3 also contains knowledge about actions specific to combinations of objects. Our proposed method is a networked output of GPT-3, which can directly output a set of verbs V_{AB} recalled for objects O_A and O_B.

Assume that SyntagNet outputs “cut” and “eat” from “apple”, and “cut” and “stab” from “knife”. The affordances of “apple” + “knife” would be defined as a union set (“cut”, “eat”, “stab”) or an intersection set (“cut”). On the other hand, the proposed method can derive cut directly from the combination “apple” + “knife” rather than combining the affordances of individual objects.

B. Evaluation of MAP Incorporating Affordances

In this section, we built an ANA with built-in affordances and evaluated whether the proposed method can efficiently obtain a sequence of actions to achieve the goal.

TABLE I. COMPARISON OF RESULTS.

	With affordances	Without affordances
agent	16	90
link	73	346
activation propagation	273	1315

For this reason, we conducted experiments on simulations with and without the affordances extracted for ANA and compared them in terms of (a) the number of agents, (b) the number of links, and (c) the number of activation propagations required to achieve the goal. In (a), the greater the number of agents that do not contribute to activation for goal attainment, the more sure we can be that useless agents are provided. When useless agents are prepared, links are also prepared to connect them. Thus, in (b), the greater the number of links connecting the wasted agents, the more wasted activation propagation may occur. In (c), we counted the number of activation propagations until the goal set in the experiment was achieved. The smaller this number is, the more sure we can be that we can reduce the computational cost.

The experiment was conducted in a scenario that could occur in daily life, and the goal was to keep the windows clean. The scenario used two types of nouns (window, towel). The parameters in (1) were set to $\phi = 20, \gamma = 70, \delta = 50$ described in the original paper [4].

V. discussion

a) The number of agents was 16 with affordances and 90 without affordances. In this experiment, only 6 agents contributed to the activation to achieve the goal, and most prepared agents were not used in the case of no affordances. Therefore, the with-affordance case is more efficient without unnecessarily activating many agents.

(b) The number of links was 73 with affordances and 346 without affordances. It can be seen that the number of links increased as the number of wasted agents was prepared in (a).

(c) From Table I, the number of activation propagations required to achieve the goal is clearly smaller in the case with affordances, thus reducing the computational cost. This result confirms that activation propagation is more compact and less wasteful with affordance.

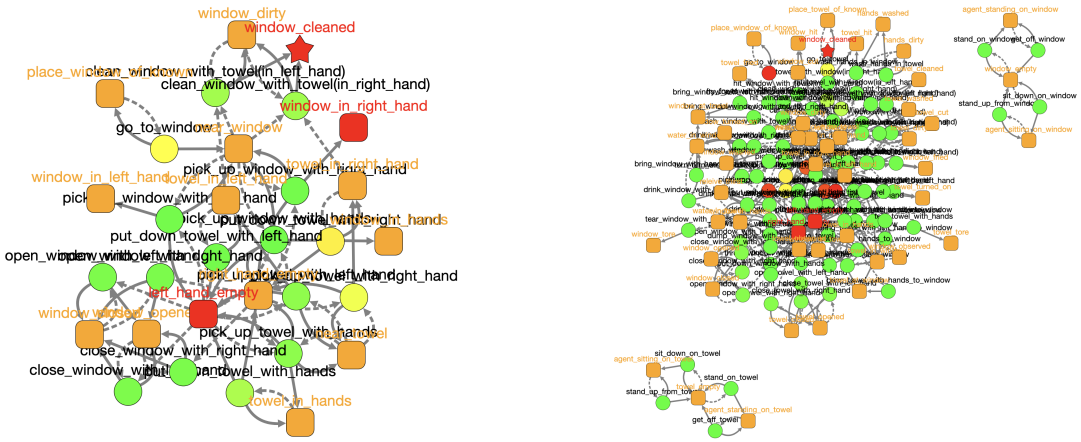
Figure 5 compares agent behavior networks consisting of agents and links. The fact that the goal is achieved regardless of affordances confirms that appropriate action sequences can be obtained even with compact network configurations such as those with affordances.

VI. Conclusion and Future Work

In this paper, we proposed a method of automatically extracting affordances from a large-scale language model in accordance with the situation and using them in a MAP. The experiments demonstrated the effectiveness of using affordances, and we plan to address three points.

The first is to experiment in an unknown dynamic environment. This study conducted experiments in a known environment on the simulation. To verify the effectiveness of

 ISBN: 978-1-68558-053-7



(a) With affordances

(b) Without affordances

Figure 5. Comparison of Agent Behavior Network.

the proposed MAP, its adaptability needs to be tested in a real-world, dynamically changing environment. In the future, we will test the MAP in a 3D simulation environment that simulates the real world.

The second point is to verify the object's affordance by considering its attributes. The organisms are able to associate the behavior of eating with red apples but not with brown apples. Thus, even for objects labeled with the same name, the fordable behavior should vary depending on attributes such as color, size, weight, shape, and texture. By considering the objects' attributes, we believe we can generate agents that are more appropriate to the situation.

The third point is to verify the results in several scenarios. The experiments conducted in this study were conducted under a single scenario. Experiments with several scenarios are necessary to verify the efficiency of the proposed method.

ACKNOWLEDGMENT

This work was supported by grant NEDO/Technology Development Project on Next-Generation Artificial Intelligence Evolving Together With Humans "Constructing Interactive Story-Type Contents Generation System".

REFERENCES

[1] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[2] R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[3] P. Maes, "The agent network architecture (ANA)," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 115–120, 1991.

[4] P. Maes, "How to do the right thing," *Connection science*, vol. 1, no. 3, pp. 291–323, 1989.

[5] J. J. Gibson, "The theory of affordances," *Hilldale, USA*, vol. 1, no. 2, pp. 67–82, 1977.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan *et al.*, "Language Models are Few-Shot Learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[7] W. Shen, F. Trevizan, and S. Thiébaux, "Learning domain-independent planning heuristics with hypergraph networks," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, pp. 574–584, 2020.

[8] C. Domshlak, M. Katz, and A. Shleyfman, "Enhanced symmetry breaking in cost-optimal planning as forward search," in *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

[9] M. Hessel, J. Modayil, H. V. Hasselt, T. Schaul, G. Ostrovski *et al.*, "Rainbow: combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, vol. 32, 2018.

[10] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel *et al.*, "Distributed prioritized experience replay," *arXiv*, 2018.

[11] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *International conference on learning representations*, 2019.

[12] A. P. Badia, P. Sprechmann, A. Vitvitskyi, D. Guo, B. Piot *et al.*, "Never give up: learning directed exploration strategies," *arXiv*, 2020.

[13] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi *et al.*, "Agent57: outperforming the atari human benchmark," in *International Conference on Machine Learning*, 2020, pp. 507–517.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou *et al.*, "Playing Atari with deep reinforcement learning," *arXiv*, 2013.

[15] O. J. Romero, "Cognitively-inspired agent-based service composition for mobile and pervasive computing," in *International Conference on AI and Mobile Services*, 2019, pp. 101–117.

[16] P. Allgeuer and S. Behnke, "Hierarchical and state-based architectures for robot behavior planning and control," *arXiv*, 2018.

[17] J. Simmonds, J. A. Gómez, and A. Ledezma, "The role of agent-based modeling and multi-agent systems in flood-based hydrological problems: a brief review," *Journal of Water and Climate Change*, vol. 11, no. 4, pp. 1580–1602, 2020.

[18] J. H. Park and S. C. Park, "Agent-based merchandise management in business-to-business electronic commerce," *Decision Support Systems*, vol. 35, no. 3, pp. 311–333, 2003.

[19] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard *et al.*, "The Stanford CoreNLP natural language processing toolkit," *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 55–60, 2014.

[20] M. Maru, F. Scozzafava, F. Martelli, and R. Navigli, "SyntagNet: challenging supervised word sense disambiguation with lexical-semantic combinations," *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3532–3538, 2019.

 ISBN: 978-1-68558-053-7