Creative Problem Solving by Robots Using Action Primitive Discovery

Evana Gizzi[1], Mateo Guaman Castro[2] and Jivko Sinapov[1]

***Abstract*— Humans and many other species have the remarkable ability to innovate and creatively problem solve on-the-fly. Inspired by these abilities, we propose a framework for action discovery in problem solving scenarios similar to puzzle-boxes used to evaluate intelligence in animal species. The proposed framework assumes that the robot starts with a knowledge base including predicates and actions, which, however, are insufficient to solve the problem faced by the robot. We describe a method for discovering new action primitives through object exploration and action segmentation, which is able to iteratively update the robot's knowledge base on-the-fly until the solution becomes feasible. We implemented and evaluated the framework using a 3D physics-based simulated object retrieval task for the Baxter bi-manual robot. Results suggest that action segmentation is one viable path towards enabling autonomous agents to adapt on-the-fly and in short amounts of time to new situations that were unforeseen by their programmers and engineers.**

I. Introduction

Creative problem solving is a fascinating skill harbored by both human and non-human species alike [1], [2], [3], [4]. These abilities enable us to find new ways of solving old problems as well as to find solutions to new problems. In day to day lives, humans often augment the way they achieve a goal – for example, to open a tightly closed jar, we may put a towel between our hand and the lid before twisting. In other cases, we find new ways of using available objects, as is the case of using a mug, an object for storing liquids, as a paperweight when an actual paper weight is not present. Research has documented such abilities in a variety of animals other than humans, ranging from primates to birds, indicating that the ability to vary behavior and explore objects is crucial for adaptation to novel problems [5].

In robotics, problem solving is often achieved through the use of planning over a knowledge base with specified predicates, actions, pre-conditions, and effects [6], [7]. While such methods can often find counter-intuitive (from a human's point of view) ways to achieve a goal, they implicitly assume that the existing actions are sufficient to find a solution. In other words, these methods lack the ability to discover new actions at the lowest level (what we call, action primitives) and instead can only sequence and compose existing actions. Other approaches to solving novel tasks involving object manipulation include reinforcement learning [8] which, despite showing impressive and promising results, require long-lasting interaction with the environment that is prohibitively expensive in many real-world domains.

[1]Department of Computer Science, Tufts University, Medford MA, 02155, {Evana.Gizzi|Jivko.Sinapov}@tufts.edu

[2]Department of Electrical and Computer Engineering, Tufts University, Medford MA, 02155, Mateo.Guaman@tufts.edu

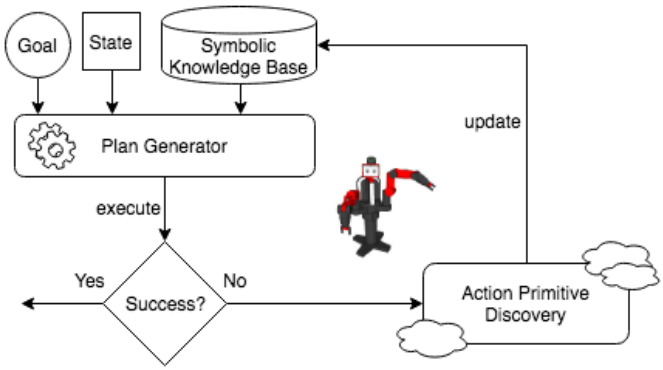

Fig. 1. **System diagram of the Action Primitive Discovery Framework.** The agent starts with a goal, state information, and a knowledge base that includes information about its own actions. Next, a plan is generated and executed. If the execution is not successful, the agent attempts to expand and update its knowledge base through action primitive discovery. This process is repeated until execution is successful.

Recently, a few studies have formalized creative problem solving by relating the ability to that of MacGyver, a character from a 1985 series, known for his ability to synthesize creative solutions to challenging problems through the ingenious use of materials in his environment. Nair *et al.* [9] introduce the term "MacGyvering" as the process of creating or repairing something in an inventive or improvised way by utilizing available objects. Sarathy and Scheutz [10] formalize a MacGyver problem as one in which a solution exists, but is not immediately available to a planning agent given its knowledge base of actions and state information.

To illustrate the MacGyver scenario, suppose a service robot is taught how to press a button to call for an elevator. In the demonstration, the robot learns that the "press_button" action consists of moving its gripper to the button, pushing down, and retracting its arm. Suppose that in a new situation the robot encounters a different elevator, one that requires the button to be held upon pressing instead of released. In this novel setting, there is no human demonstrator available to teach the robot the new action "hold_button." The question becomes, how can the robot discover this action on its own — to gain the ability to reach a once unreachable goal?

Research with humans and other species suggests that two key mechanisms enable agents to creatively problem solve in novel situations: 1) object exploration, and 2) behavioral variation [11], [5]. Inspired by these lines of work, we propose a framework for problem solving through action discovery in robotic agents which uses autonomous exploration and action segmentation. The proposed framework, illustrated in Figure 1, leverages symbolic logical predicate information to

develop meaningful representations of newly formed actions so that the discovered actions can easily be used for planning when attempting to find a solution to the problem. When faced with an unsolvable problem, the proposed method for action discovery leverages change-point detection to break existing actions into sub-parts, each of which can be used independently in subsequent exploration. The framework was evaluated in an object retrieval task similar to the button example described earlier in this section, in which a robot had to learn several action primitives (e.g., holding the button once initially pressed) in order to make the target object accessible. Results suggest that action segmentation is one viable path towards enabling autonomous agents to adapt on-the-fly and in short amounts of time, to new situations that were unforeseen by their programmers and engineers.

II. Related Work

A. Innovation and Creativity in Humans and Animals

Humans and many other species have the remarkable ability to innovate and creatively problem solve on-the-fly [1], [2]. The problem solving skills of birds and primates are often studied in scenarios where the organism has to solve a "puzzle-box", sometimes using tools, to retrieve a food item out of the box that is not immediately accessible. Reader and Laland [4] define innovative behavior as "a new or modified learned behaviour not previously found in a population". Research in animal psychology postulates that the ability to exhibit novel behaviors, or familiar behaviors in novel settings, is central to physical problem solving [12]. Motor diversity has been identified as a strong predictor for problem solving skills in several species [13], [14]. Inspired by these findings, the work in this paper proposes a mechanism for discovering new actions that an agent can subsequently use when searching for a solution to a puzzlebox-like problem.

Research into the underpinnings of human creativity span cognitive science, neuroscience, psychology and philosophy [2], [15], [16]. Physical creativity and innovation are tightly linked to object exploration and play [11]. In studies that compare human problem solving to that of other species (e.g., [17]), humans express the most creativity and innovation, with one possible explanation being that our longer developmental period fosters such abilities [18]. While for a long time, it was thought that human creativity and innovation is unique and separate from that of animals, a large body of research indicates that the individual mechanisms for creativity in humans are also present in other problem-solving species [3]. Physical object exploration, in particular, is a key mechanism for human and animal problem solving. In this paper, the proposed framework also relies on the robot exploring its environment with the actions it currently knows, in order to discover new actions that can potentially solve the object retrieval task it is faced with.

B. Creative Problem Solving in Robotics and AI

In robotics and AI, several recent works have brought attention to the notion of creative and innovative problem solving. Sarathy and Scheutz [10] formulate the MacGyver problem as one in which a solution exists but is not immediately available given the robot's knowledge base of predicates, actions and their preconditions and effects. Similar problems have also been investigated in the context of tool use [19], [20] and tool construction [9].

Reinforcement learning remains a popular approach for learning complex manipulation sequences [8]. While human feedback over the course of learning (e.g., [21]) and learning by demonstration [22] can substantially reduce training time, the amount of interaction needed to learn a good policy can be prohibitively expensive for a physical robot operating in a real-world setting without direct human assistance.

Other related research has proposed segmenting actions learned by demonstration into *dynamic movement primitives (DMP)*, which are partial motion trajectories of the original action [23], [24]. Segmentation alone limits the context of creative problem solving because it lacks symbolic knowledge representations of the actions being discovered. Techniques for bootstrapping low-level primitives with high-level symbolic information have been explored in recent years. These representations take the form of finite state machines, skills trees, mappings to sub-goal abstractions, and finding repeated substructures for generalization and stronger representational value [25], [26], [27], [28]. Perhaps, most related to our work is that of Baisero *et al.* [29] which developed a method for segmentation of actions by using relational information of both the objects in the scenario and of their affordance information.

Although progress has been made toward segmentation and representation of action primitives, existing methods fall short of generating representational forms, which are often utilized in cognitive applications that operate at higher levels of abstraction. We address this shortcoming by developing a novel framework that makes use of logical predicate information to inform and develop action primitive discovery. Our framework is able to not only learn newly discovered actions, but is also capable of using newly learned actions to inform the representation of its old actions. To our knowledge, this is the first action segmentation framework which leverages predicate information to build newly formed action representations, and one of the first attempts at illustrating a solution to MacGyver Problems [10] through implementation.

III. THEORETICAL FRAMEWORK

A. Notation and Problem Formulation

We frame our problem in the context of a MacGyver Problem, adopting a similar notation as that described in [10]. We define our planning domain as $\Sigma = (S, A, \gamma)$, where S is a set of finite states of the system, A is a finite set of action primitives, and γ is the transition function over actions and states (which we denote $A_\Sigma, S_\Sigma, \gamma_\Sigma$ for a given domain Σ). We define a set of known objects in the domain $O = \{o_1, o_2, \dots o_{|O|}\}$ and a set of predicates,

$$F = \{f_1(\cdot), \dots f_{|F|}(\odot)\}, \odot \subset O,$$

where each predicate operates over a subset of objects in the domain. Any state $s \in S$ is composed of a set of predicate values F_s that hold true. A problem in the domain is defined as $P = (\Sigma, s_0, s_g)$, where s_0 is an initial state, s_g is a goal state, and $s_0, s_g \in S$. A plan $\pi = [a_1, a_2, \dots a_{|\pi|}]$ is a solution to a problem P. Suppose a plan π to accomplish a goal state s_g is successfully generated, but fails on execution. At this point, the problem becomes how to transform the domain Σ to another domain, Σ', such that π' generated from Σ' will not fail. Note that in our framework, failure occurs in the execution phase, rather than the planning phase, which differentiates our method from [10].

B. Action Discovery

To handle the execution failure of π, we propose a method that generates a refined and expanded action set A' through exploration and action segmentation. As a first step to this discovery, the proposed algorithm generates a list of all action-object combinations in Σ, which we call

$$C_\Sigma = C\begin{pmatrix} a \\ o \\ (|R_a|) \end{pmatrix}, \forall a \subset A_\Sigma, o \subset O_\Sigma$$

where R_a are the parameters to action a. Next, the agent executes each action combination in $c \in C_\Sigma$, and generates a set of change points $\Delta_i = [t_1, \dots t_k], t \in \mathbb{Z}$, where Δ is a time series list of ordered time stamps corresponding to change points. The input to the change point detection algorithm is a set of time series trajectories that represent motor data (e.g., joint efforts, or end-effector position) for the arm that executed the action. That is, for an action a_i, a set of n trajectories $d_1, \dots d_n$ are represented by a matrix $M \in \mathbb{R}^{n \times d_i}$. These change points are used to generate candidate action segmentations $\hat{a_1}, \dots \hat{a_{k-1}}$ where each $\hat{a_j}$ corresponds to the limb trajectory from time step t_j to t_{j+1}. Each candidate $\hat{a_j}$ is executed, generating effects $E_{\hat{a_j}}$. A is refined to include candidate actions that generate a non-empty set of effects, where the new action set $A' \leftarrow A \cup \{\hat{a_j}\}, \forall a_j$ s.t. $E_j \neq \emptyset$. The transition function γ is implicitly updated with this refinement. Finally, in line 28, the preconditions of the failure action a_f are refined to include the effects of the action that succeeded to accomplish s_g.

It should be noted that some algorithmic components are agnostic to our framework, including the choice of planner used to find π, the change point detection algorithm used to find Δ, and motion data collected in M. Likewise, although we describe the algorithm in this section within the context of a *breadth-first search (BFS)* method for finding new actions, the method choice is also agnostic to our framework. In our results section, we discuss findings for both breadth-first search and *depth-first search (DFS)* approaches.

Algorithm 1 Action Primitive Discovery Framework

1: **procedure** PLANEXECUTOR(s_g, Σ)
2: s_g : predicates needed to accomplish goal
3: $\delta \leftarrow \emptyset$: list of candidate action primitives
4: $a_f, a_s = \varnothing$
5: **while** $s_g \not\subset s_C$ **do**
6: $\pi \leftarrow generatePlan(s_C, s_g, \Sigma)$
7: **for** $a \in \pi$ **do**
8: **if** $execute(a) = FAIL$ **then**
9: $a_f \leftarrow a$
10: **break**
11: **if** $\delta = \emptyset$ **then**
12: $C_\Sigma \leftarrow getAllActionCombos(A_\Sigma, O_\Sigma)$
13: **for** $a \in C_\Sigma$ **do**
14: **while** $execute(a)$ **do**
15: $M \leftarrow MotionData(a^{limb})$
16: $\Delta \leftarrow getChangePoints(M)$
17: **for** $j \in range(|\Delta| - 1)$ **do**
18: $\hat{a} \leftarrow newAction(\Delta[j], \Delta[j+1])$
19: $execute(\hat{a})$
20: **if** $E_a \neq \emptyset$ **then**
21: $A_\Sigma \leftarrow A_\Sigma \cup \{\hat{a}\}$
22: $\delta \leftarrow \delta \cup \{\hat{a}\}$
23: **if** $\delta \neq \emptyset$ **then**
24: $a_s \leftarrow \delta[randInt(0, |\delta|)]$
25: $execute(a_s)$
26: $F_{a_s} \leftarrow b$
27: $\delta \leftarrow \delta \setminus a_s$
28: $I_{a_f} \leftarrow I_{a_f} \cup E_{a_s}$

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

We ran a proof-of-concept of our framework in the Gazebo simulation environment, which utilizes a high performance physics engine. This experimental setup consisted of five parts: a Baxter robot, a table with a sliding wall mechanism, two red buttons, and a green block, which is shown in Figure 2. In this environment, the red buttons are readily accessible to the robot's arms, while access to the block is obstructed by the wall. The wall can be actuated by pressing one of the buttons, but the robot lacks domain knowledge on how to operate these buttons to cause the wall to slide out of the way to enable the robot to be able to pick up the block.

The Baxter robot used Pyperplan[1] as its planner, operating over actions in a *Partial Domain Definition Language (PDDL)* format. This format was most conducive to the predicate representations employed by our framework. Action primitives in our framework, following the PDDL format, were equipped with a set of *preconditions* and *effects*, which describe the predicates which must hold true prior to execution, and as a consequent of execution, respectively.

For the change point detection component, the agent used Bayesian Change Point (BCP) detection in conjunction with agglomerative bottom-up clustering. The BCP method uses a Bayes factor to estimate the position of change points in trajectories [30]. The agglomerative bottom-up clustering method then grouped change points in an iterative manner based on distance metrics. A final set of change points were

[1]https://bitbucket.org/malte/pyperplan

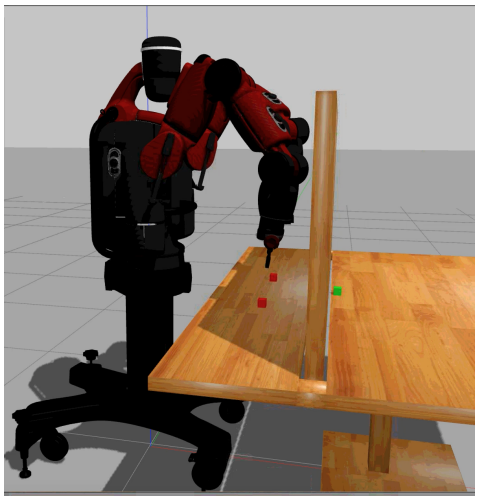

Fig. 2. proof-of-concept Gazebo simulation environment, including Baxter robot in starting position (grippers above table), wooden table with wall, two red blocks representing left button and right button, and green block. Pressing and holding the left button results in lowering the wall which blocks the robot from retrieving the target object.

generated by extracting the minimum, maximum, and mean of each cluster that had a cardinality greater than 10, which we set as our minimum grouping value in order to consider only the most significant groupings.

B. Illustrative Example

In our proof-of-concept task, the goal of the Baxter robot was to obtain the block located at the other side of the sliding wall, which is initially inaccessible. The domain Σ and problem formulation P of the task are shown in Table I. Using this information, the Baxter robot generates a plan $\pi = [\texttt{obtain\_object(l\_gripper, block)}]$. Due to the wall obstructing the path of the grippers to the object, π fails on execution. At this point, the Baxter robot generates a list of action-object combinations, shown in section C of Table I, to be used for segmentation. The table shows samples executed by the left gripper but not by the right gripper. This is because in our proof-of-concept, actions performed by the left gripper and right gripper were reflective of one another, generating the same predicate results. In other applications, this assumption may not be favorable or appropriate.

Next, the robot executes each action-object combination in Table I to find candidate action primitives. For our motion trajectory data, we recorded the 3-dimensional coordinate (*x*-*y*-*z*) position of the gripper performing the action. Using the data collected during exploration, change points were generated using our BCP/clustering algorithm, with motion trajectory matrix M as input. This process is illustrated in Figure 3, shown over the execution of $a = \texttt{press\_button(l\_gripper, l\_button)}$, where nine change points were detected $[t_1, \ldots t_9]$, resulting in eight trajectories $\hat{a_1}, \ldots \hat{a_8}$. Figure 3 shows corresponding simulation image frames of the execution steps for each trajectory. It should be noted that some sub-trajectories had very small positional differences in the gripper, and some

Planning Domain: Σ	
A	`obtain_object(`**`:gripper, :location`**`)` `press_button(`**`:gripper, :button`**`)`

F	`pressed(`**`:button`**`)`
	`obtained(`**`:obj`**`)`
	`at(`**`:object`**`)`
	`is_visible(`**`:obj`**`)`

O	**`:obj -`** `wall, block` **`:gripper -`** `l_gripper, r_gripper` **`:button`** `- l_button, r_button` **`:object`** `- :obj, :gripper, :button` **`:location -`** `loc_a, loc_b, loc_c, loc_d, loc_e`
Problem: P	
s_g	`obtained(block)`

s_0	`at(left_gripper, loc_a)`
	`at(r_gripper, loc_b)`
	`at(l_button, loc_c)`
	`at(r_button, loc_d)`
	`at(block, loc_e)`
	`is_visible(l_button)`
	`is_visible(r_button)`

Action-Object Combinations: C	

	`press_button(l_gripper, l_button)`
	`press_button(l_gripper, r_button)`
	`obtain_object(l_gripper, wall)`
	`obtain_object(l_gripper, table)`
	`obtain_object(l_gripper, block)`

TABLE I

DOMAIN KNOWLEDGE, INITIAL PROBLEM FORMULATION, AND ACTION-OBJECT COMBINATIONS

images reflect more than one sub-trajectory due to non-visible trajectory differences.

After generating change points, the Baxter robot executes each sub-trajectory, keeping the sub-trajectories that result in non-empty effects lists. For our proof-of-concept, we omitted location information from effects list, and only considered actions with state-based predicates in its effects list (`pressed(), obtained() or is_visible()`). From Figure 3, it can be seen that the only sub-trajectory with non-empty state effects list is $\hat{a_2}$, with $E_{\hat{a_2}} = \texttt{is\_visible(block)}, \texttt{pressed(l\_button)}$, corresponding to the sub-trajectory from t_2 to t_3. In this particular example, none of the other sub-trajectories produced a change in state-based predicates in their effects lists.

The robot was able to use $\hat{a_2}$ to successfully obtain the object by first holding the left button (via $\hat{a_2}$), and then using the `obtain_object` action primitive to move the object with its right gripper, generating a new plan,

$\pi' = [\ \hat{a_2}\texttt{(l\_gripper, l\_button)},$
`obtain_object(r_gripper, block)`$]$.

Furthermore, it refined the preconditions of the `obtain_object` action primitive to contain $E_{\hat{a_2}}$.

C. Results

We implemented and tested our framework under the conditions described in our illustrative example[2]. We define an experimental trial to be an independent sequence of re-plan attempts that the agent uses to get from its initial problem formulation to its goal. *Execution phases* are algorithmic

[2]The code for our framework and experiments is available at https://github.com/Evana13G/RAPDR

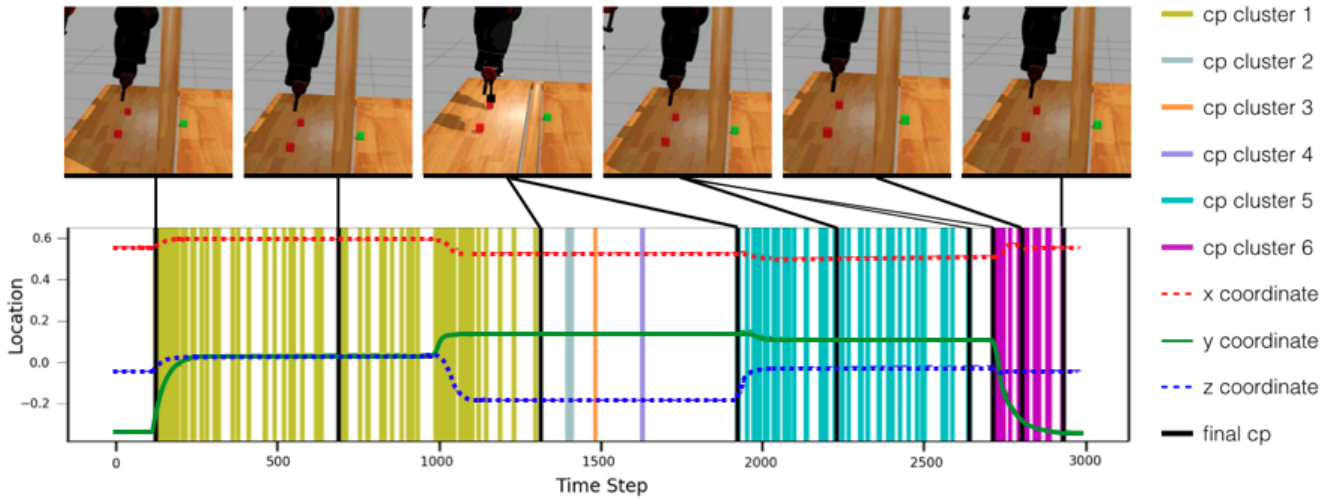

Fig. 3. **Change point detection for press_button(right_gripper, left_button)**. The image on the bottom shows the x-y-z location of the left gripper over time steps during action execution, separated into three separate traces. All vertical lines represent change points in the motion data, grouped based on clustering, with a total of 6 clusters. The black vertical lines show the change points that got used for segmentation, consisting of the min/max/mean of clusters that include more than 10 change points.

TABLE II
EVALUATION RESULTS

Method	# of Discovered Action Primitives	Execution Attempts	Exploration Attempts
BFS	2.62 ± 1.12	3.74 ± 3.48	1.04 ± 0.22
DFS	1.76 ± 1.03	6.18 ± 0.22	6.18 ± 0.22

steps which consist of plan generation and plan execution, corresponding to lines 6-10, 23-27 of our algorithm. *Exploratory phase* are steps which consist of executing an action-object combination or of executing a candidate action segmentation, corresponding to lines 11-22 of our algorithm. Any re-plan attempt that includes an exploratory phase is referred to as an *exploratory attempt*, whereas all other attempts are referred to as *execution attempts*.

We consider a trial to be successful if the agent succeeds in accomplishing its goal after some unbounded number of execution and exploratory phases. We define *time to solution (TTS)* of a successful trial as the total time in seconds elapsed from the initial problem formulation to completion of the goal. For successful trials, we use the TTS and number of execution attempts as evaluation performance measures.

We ran a total of 38 trials to evaluate and test our framework, with 21 trials in BFS mode and 17 trials in DFS mode. The number of trials between modes differed because of server issues which rendered some trials to be removed from analysis. In all trials, the agent was successful in accomplishing its goal. Figure 4 shows the TTS for all trials, split across BFS and DFS modes. This figure shows higher value mean TTS of BFS trials than the DFS trials, but with a lower variance. We believe that this is because of the differing nature of the exploration attempts of the BFS and DFS approaches. In the BFS approach, exploration attempts require the agent to iterate over all action-object combinations, finding all viable segmentations, before returning to an execution step. In the DFS approach, exploration attempts include the attempted discovery of action primitives for just one action-object combination, and the problem re-plan following that potential exploration phase. This implies that in the case of DFS, a lucky trial could find a successful segmentation on the first try, from its first action-object combination exploration, whereas an unlucky case could find it after iterating over every action-object combination possibility. This phenomena is further reflected in Table II, where the mean value of execution attempts for BFS mode is larger than that of DFS, but the mean value of exploration attempts for BFS is smaller than that of DFS.

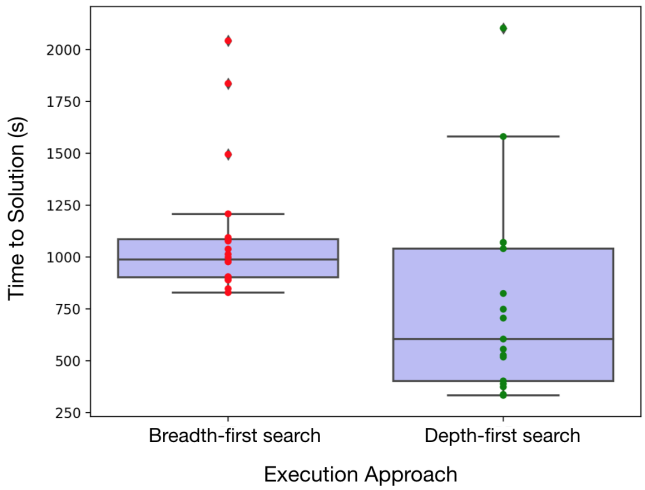

Fig. 4. **Total execution time for each execution type.** Although the time to solution for the BFS approach was higher than the median time to solution of the DFS approach, it had less variance. The BFS approach thus generates more consistent times to solution, whereas the DFS approach has a wider spread of times to solution. The difference is statistically significant ($p = 0.0221$) under an unpaired t-test.

Table II also shows the mean value of the number of newly discovered action primitives in BFS and DFS trials. BFS mode had a larger mean value and standard deviation for this metric than DFS mode. We do not believe that this data is distinguishable enough to represent a general trend, and it is unclear to us if this finding can be generalized to all tasks which employ our framework, but we do believe this to be application specific behavior.

V. Conclusion and Future Work

The ability to innovate and problem solve on-the-fly is a hallmark of human and animal intelligence. This paper proposed a framework for such problem solving in the context of manipulation tasks in which the agent's starting knowledge base of predicates and actions is insufficient to successfully execute the agent's plan. The key mechanism behind our approach consists of action segmentation during exploration so that the agent can increase the diversity of its behavioral repertoire. We evaluated the framework with a simulated robot that faced an object retrieval task inspired by puzzle-box domains used to evaluate the problem solving abilities of biological organisms. Results suggest that action segmentation is one viable means towards enabling autonomous agents to adapt on-the-fly and in short amounts of time, to new situations that were unforeseen by their programmers and engineers.

In future work, we hope to use the segmented action primitive in reinforcement learning settings, and in more complicated domains. We also would like to extend the framework to include additional mechanisms for action discovery (e.g., varying control parameters of an action's controller) in addition to segmentation. Furthermore, in addition to discovering novel actions, novel problems may also require the discovery of new predicates and new sensorimotor means to infer their state. Finally, we are working towards the development of a set of scenarios for creative problem solving that grow in complexity (e.g., tool creation and use [19], [9]) relative to the problem domain used in our evaluation.

References

[1] M. A. Boden, *The creative mind: Myths and mechanisms*. Routledge, 2004.

[2] A. Fink, R. H. Grabner, M. Benedek, G. Reishofer, V. Hauswirth, M. Fally, C. Neuper, F. Ebner, and A. C. Neubauer, "The creative brain: Investigation of brain activity during creative problem solving by means of eeg and fmri," *Human brain mapping*, vol. 30, no. 3, pp. 734–748, 2009.

[3] S. J. Shettleworth, "Clever animals and killjoy explanations in comparative psychology," *Trends in cognitive sciences*, vol. 14, no. 11, pp. 477–481, 2010.

[4] S. M. Reader and K. N. Laland, *Animal innovation*. Oxford University Press Oxford, 2003, vol. 10.

[5] A. S. Griffin and D. Guez, "Innovation and problem solving: a review of common mechanisms," *Behavioural Processes*, vol. 109, pp. 121–134, 2014.

[6] S. Zhang, M. Sridharan, and F. S. Bao, "ASP+POMDP: Integrating non-monotonic logic programming and probabilistic planning on robots," in *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*. IEEE, 2012, pp. 1–7.

[7] P. Khandelwal, F. Yang, M. Leonetti, V. Lifschitz, and P. Stone, "Planning in action language BC while learning action costs for mobile robots," in *International Conference on Automated Planning and Scheduling (ICAPS)*, June 2014.

[8] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[9] L. Nair, J. Balloch, and S. Chernova, "Tool MacGyvering: Tool Construction Using Geometric Reasoning," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019.

[10] V. Sarathy and M. Scheutz, "MacGyver problems: Ai challenges for testing resourcefulness and creativity," *Advances in Cognitive Systems*, vol. 6, pp. 31–44, 2018.

[11] T. G. Power, *Play and exploration in children and animals*. Psychology Press, 1999.

[12] S. M. Reader, J. Morand-Ferron, and E. Flynn, "Animal and human innovation: novel problems and novel solutions," 2016.

[13] H. M. Manrique, C. J. Völter, and J. Call, "Repeated innovation in great apes," *Animal Behaviour*, vol. 85, no. 1, pp. 195–202, 2013.

[14] A. S. Griffin, M. Diquelou, and M. Perea, "Innovative problem solving in birds: a key role of motor diversity," *Animal Behaviour*, vol. 92, pp. 221–227, 2014.

[15] A. Dietrich, "The cognitive neuroscience of creativity," *Psychonomic bulletin & review*, vol. 11, no. 6, pp. 1011–1026, 2004.

[16] G. A. Wiggins, "A preliminary framework for description, analysis and comparison of creative systems," *Knowledge-Based Systems*, vol. 19, no. 7, pp. 449–458, 2006.

[17] L. Rat-Fischer, A. Kacelnik, K. Plunkett, and A. von Bayern, "Development of physical problem-solving competences in human infants and corvids," in *2016 Joint IEEE Intl. Conf. on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2016, pp. 101–102.

[18] L. Fogarty, N. Creanza, and M. W. Feldman, "Cultural evolutionary perspectives on creativity and human innovation," *Trends in ecology & evolution*, vol. 30, no. 12, pp. 736–754, 2015.

[19] J. Sinapov and A. Stoytchev, "Detecting the functional similarities between tools using a hierarchical representation of outcomes," in *7th IEEE International Conference on Development and Learning*. IEEE, 2008, pp. 91–96.

[20] M. Stilman, M. Zafar, C. Erdogan, P. Hou, S. Reynolds-Haertle, and G. Tracy, "Robots using environment objects as tools the macgyver-paradigm for mobile manipulation," in *2014 IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2568–2568.

[21] W. B. Knox and P. Stone, "Tamer: Training an agent manually via evaluative reinforcement," in *2008 7th IEEE International Conference on Development and Learning*. IEEE, 2008, pp. 292–297.

[22] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[23] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[24] A. Ude, A. Gams, T. Asfour, and J. Morimoto, "Task-specific generalization of discrete and periodic dynamic movement primitives," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 800–815, 2010.

[25] S. Niekum, S. Osentoski, G. Konidaris, S. Chitta, B. Marthi, and A. G. Barto, "Learning grounded finite-state representations from unstructured demonstrations," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 131–157, 2015.

[26] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[27] V. Chu, R. A. Gutierrez, S. Chernova, and A. L. Thomaz, "The role of multisensory data for automatic segmentation of manipulation skills," in *RSS Workshop on Empirically Data-Driven Manipulation*, 2017.

[28] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2015.

[29] A. Baisero, Y. Mollard, M. Lopes, M. Toussaint, and I. Lütkebohle, "Temporal segmentation of pair-wise interaction phases in sequential manipulation demonstrations," in *2015 IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, 2015, pp. 478–484.

[30] D. L. Ensign and V. S. Pande, "Bayesian detection of intensity changes in single molecule and molecular dynamics trajectories," *The Journal of Physical Chemistry B*, vol. 114, no. 1, pp. 280–292, 2009.