Understanding the Behavior of Reinforcement Learning Agents

Jörg Stork[1](✉), Martin Zaefferer[1], Thomas Bartz-Beielstein[1], and A. E. Eiben[2]

[1] TH Köln, Steinmüllerallee 1, 51643 Gummersbach, Germany
{joerg.stork,martin.zaefferer,thomas.bartz-beielstein}@th-koeln.de
[2] Vrije Universiteit Amsterdam, De Boelelaan 1105,
1081 HV Amsterdam, Netherlands
a.e.eiben@vu.nl

Abstract. Reinforcement Learning (RL) is the process of training agents to solve specific tasks, based on measures of reward. Understanding the behavior of an agent in its environment can be crucial. For instance, if users understand why specific agents fail at a task, they might be able to define better reward functions, to steer the agents' development in the right direction. Understandability also empowers decisions for agent deployment. If we know why the controller of an autonomous car fails or excels in specific traffic situations, we can make better decisions on whether/when to use them in practice. We aim to facilitate the understandability of RL. To that end, we investigate and observe the behavioral space: the set of actions of an agent observed for a set of input states. Consecutively, we develop measures of distance or similarity in that space and analyze how agents compare in their behavior. Moreover, we investigate which states and actions are critical for a task, and determine the correlation between reward and behavior. We utilize two basic RL environments to investigate our measures. The results showcase the high potential of inspecting an agents' behavior and comparing their distance in behavior space.

Keywords: Reinforcement Learning · Behavior · Understandable AI

1 Introduction

In Reinforcement Learning (RL), agents are learning policies to solve a specific task. For example, we can consider a robot as an agent who has to navigate a particular environment and react to certain obstacles. At first, a user is interested in these robots' performance, which is commonly evaluated by their ability to solve the task and further based on a user-defined reward function. Besides this performance assessment, the trained robot's behavior, such as the action it takes for individual states, is the only observable part, as the internals of the policy remain indistinguishable by an external observer. Thus, users desire to analyze

B. Filipič et al. (Eds.): BIOMA 2020, LNCS 12438, pp. 148–160, 2020.
https://doi.org/10.1007/978-3-030-63710-1_12

and compare the behavior to exploit how the robot reacts in certain situations or if it behaves as intended. Even a well-performing robot may have developed a specialized behavior not intended by the user, such as only driving backward.

This paper compares agents based on their behaviors, which span a new space, the behavior space. This paper's primary motivation is to create a better understanding of this behavior space and develop useful measures for the comparisons of agents without knowing the inner details of their policies. Moreover, these measures could allow us to identify how agents in a learning set differ, not concerning their reward, but with regard to their behavior. It is particularly interesting to identify situations (states) in which an agent behaves differently than expected. As this is a broad topic, we will start by tackling the following research questions:

Q-1. How does an agent's behavior with good performance compare to similarly performing agents or inferior agents?
Q-2. Which input states are important or problematic for the task?
Q-3. Is there a correlation between an agent's reward and behavior, and how do changes in the behavior affect their reward?

Comparing agents in the behavior space has some prerequisites: For most RL environments, individual agents will not visit the entire state space and thus not learn the optimal action for these unobserved states. Unobserved states are, for instance, present in environments with continuous state spaces or exclusive paths. Nevertheless, as we investigate agents that map a policy from state to action space (i.e., Artificial Neural Network (ANN) policy controllers), we can compute an agent's behavior to any state, even if not observed or observable by the agent itself. This property allows us to compare two agents in the behavior space on a mutual state set and investigate differences. However, state sets are usually not initially known but based on processing the RL tasks and discovered during each agent's learning process. Thus, the individual state sets' contents are based on the state trajectory each agent follows, for example, an absolute path for a robot in a maze. Each agent in a learning set will likely have different trajectories, which renders it challenging to select input states to compute a useful behavior space to compare many agents.

Behavior spaces have previously been investigated in the RL literature. Most frequently, they were utilized to measure diversity and enforce explorative search strategies. For instance, Doncieux and Mouret used behavioral similarity measures to encourage the diversity of evolved agents in an evolutionary search [1]. Ollion and Doncieux suggested to measure and enforce exploration in the behavioral space [12]. Meyerson et al. [9] investigated how behavior characterizations can be learned automatically for novelty search. Quality diversity algorithms also depend on effective behavior comparison [13]. Similar directions have been investigated in the field of surrogate model-based optimization. Here, the term *phenotypic* space has been used, defining a space that encompasses behaviors and outcomes of individuals, rather than their encoding (genotype). Distances in the phenotypic space are used to train surrogate models. For instance, this has been investigated in the context of tree-coded genetic programming [5,11,16].

Similar work focused on graph-coded representations of neural networks. Here, phenotypic spaces and distance measure have been investigated for tasks like classification, reinforcement learning, or for evolving neural network controllers for robotic navigation [4,14].

Unlike these previous investigations, we aim to look at the behavior space not primarily to improve the performance of optimization or modeling algorithms. Instead, we aim for the understandability of agents' behavior. To do so, we utilize two RL environments, a designed maze with different mutual exclusive paths and the inverted pendulum, with a large real-valued state space.

2 Methods

2.1 Behavior Space in Reinforcement Learning

The *behavior* of an RL agent encompasses its (re-)actions, based on its environment and observed input states. The actions an agent takes for a specific state $s \in S$ is defined by a *policy* $\pi : S \to B$. The agents get a reward $r \in R$ for each state transition. The discussed methods apply to a wide range of RL agents. The only prerequisite is their ability to calculate a behavior for states that have not been observed by those agents themselves. More precisely, we define the behavior as the set of actions for a set of input states. For an agent A, we denote its behavior as B_A, with $B_A = \pi_A(\mathbf{S})$. Here, $\mathbf{S}$ is a set containing n input state vectors, $\pi_A(\mathbf{S})$ is the policy function computing the actions of agent A for all states in $\mathbf{S}$. Consequently, the behavior space $\mathcal{B}$ is the set of all possible behaviors (or the behavior of all possible agents) for a RL task, that is, $B_A \in \mathcal{B}$.

2.2 Behavior Comparison and State Importance

For the comparison of two agents A and A', we can calculate the distance of their behaviors, which can then be denoted by $\mathrm{d}(B_A, B_{A'})$. Because the distance depends on the state space, we consider the distance of two behaviors concerning the same state set $\mathbf{S}$. The employed distance function can be chosen according to the data type of $B_A, B_{A'}$. That is, if they contain continuous values, we might use the Euclidean distance. If they are ordinal integers, we can choose the Manhattan distance instead, with $\mathrm{d}(B_A, B_{A'}) = \sum_i^n |\pi_A(\mathbf{S_i}) - \pi_{A'}(\mathbf{S_i})|$. The comparison of actions for individual states can provide interesting insights into the specific behavior of an agent and further the importance of that state for the task. In particular, we analyze the effects of unobserved states (UOS), which are not present in the state set of a specific agent, and the influence of states with degrees of freedom (DFS), where several actions lead to the same or similar reward. In general, we consider a state as important (or problematic), if the correct action for this state is essential for getting a good reward (or challenging to learn, e.g., a majority of agents in a learning set fails to learn the correct action). For the impact of states on the reward, we utilize the *Action Reward Rank*: For each state, all performed actions of the agents are compared, and the best ranking agent who took this action is outlined. Hence, this action is related to the final best performing agent in the set.

2.3 Reward Behavior Correlation

To understand the benefits of comparisons in the behavior space, the correlation between reward distance and behavior distance is interesting. Therefore, we investigate a set of m agents $\{A_1, ..., A_m\}$, their behaviors $\{B_{A_1}, ..., B_{A_m}\}$, and their accumulated rewards $\{R_{A_1}, ..., R_{A_m}\}$. We compute the Reward Behavior Correlation (RBC) for all pairwise comparisons

$$\mathrm{RBC}_{\mathrm{all}} = \mathrm{cor}\Big(\mathrm{d}\big(\{B_{A_1}, ..., B_{A_m}\}\big), \mathrm{d}\big(\{R_{A_1}, ..., R_{A_m}\}\big)\Big).$$

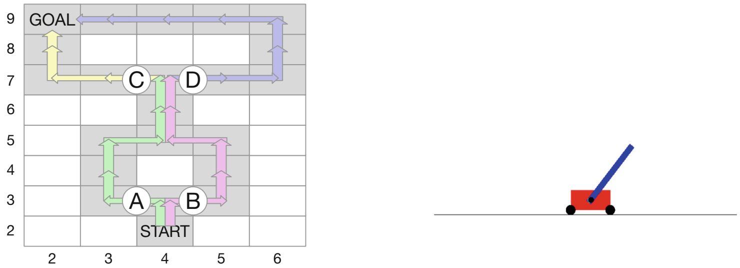
Here, $\mathrm{d}\big(\{B_{A_1}, ..., B_{A_m}\}\big)$ calculates all pairwise distances of the present agents using the behavioral distance $\mathrm{d}(B_{A_i}, B_{A_j})$. Correspondingly, $\mathrm{d}\big(\{R_{A_1}, ..., R_{A_m}\}\big)$ calculates all pairwise distances of the present agents using a distance of their accumulated rewards $\mathrm{d}(R_{A_i}, R_{A_j})$. The correlation $\mathrm{cor}(.,.)$ may be computed rank-based, if desired, or with standard linear correlation (Pearson correlation). Similarly to $\mathrm{RBC}_{\mathrm{all}}$, we can also compare each agent to the optimum agent A_{opt} (the agent with the largest reward), instead of performing all pairwise comparisons. We denote this as $\mathrm{RBC}_{\mathrm{opt}}$. A large RBC means that small/substantial differences in reward coincide with small/significant differences in behavior. Hence, a large RBC is a good indicator that the behavior space is easier to traverse for search algorithms and easier to learn for surrogate models.

This property has a close connection to the Fitness Distance Correlation (FDC) used in evolutionary computation to rate problem difficulty [6]. There, differences in fitness are correlated with distances in the search space. $\mathrm{RBC}_{\mathrm{all}}$ considers all pairwise distances while $\mathrm{RBC}_{\mathrm{opt}}$ and FDC consider distances only between candidate solutions and the global optimum (or best-known solution [7]).

3 Experiments

3.1 Deterministic Maze

The deterministic maze was designed with *mazelab* [17] as a comprehensible problem where correct actions are known, and behavior is manually rateable. The environment, visualized in Fig. 1(a), consists of a 10 × 7 matrix (shown in figures as 9 × 6, excluding external walls) with different encoding for accessible ways, walls, the agent and goal. The target is to find the shortest path to the goal. The agent is allowed to take only deterministic actions for each observed agent position in each cardinal direction. Thus they can get stuck against a wall. Agents get a small negative reward for each movement, a larger negative reward if running against a wall or moving backward, and a positive reward for reaching the goal. The maximum step size of each agent is fixed to 30, whereas only 11 are needed to follow the shortest path. We manually designed the maze to feature DFS and UOS: The maze has a total of four paths to the goal and 22 unique agent positions, but these are partly exclusive, and successful agents have always UOS. Moreover, the lower fork is a DFS, while the upper one is not. The intention was to construct a problem where agents with the same reward



(a) Deterministic Maze

(b) Inverted Pendulum

Fig. 1. Environments. For the maze environment, external walls are not displayed. Different ways: A and B are equal in reward, while D is slightly worse than C.

can have different behavior, cause of the forks, and different paths. Moreover, to analyze the effect of different exclusive paths and the UOS on the pairwise behavior comparison.

3.2 Continuous Inverted Pendulum

The inverted pendulum is a time-dependent physics simulator with a continuous input space (Fig. 1(b)). The target is to balance the pendulum on a car in the upright position for most time steps, starting at a random downwards position by moving the car. The environment is evaluated over 500 timesteps but discontinues if the base car moves out of designated limits. The action space was made deterministic for more comprehensible behavior comparisons. The pendulum environment has no exclusive paths, i.e., all states are observable, but agents will have an enormous number of UOS because of the real-valued input space. We also consider the environment to include multiple DFS, e.g., multiple correct behaviors are possible. The environment allows a large number of behaviors and different sized sets of observed states per agent.

3.3 Generating Reinforcement Learning Agents by Neuroevolution

The RL agents' policies are created and trained by utilizing Neuroevolution to learn ANN policies in an evolutionary process. The underlying algorithm is the graph-based *cartesian genetic programming* CGP by A. Turner[1] [8,15]. For the maze problem, ANNs with 70 inputs and 4 outputs were evolved, where the softmax function computes the resulting action. The pendulum ANN has 6 inputs (5 + 1 bias) and a single output. For an output value >0.5, the action is drive left, otherwise, drive right. The ANNs are evolved in terms of connection weights and structure, i.e., the number and placing of connections, nodes, and transfer functions. The maximum number of nodes and connections for each ANN was set to 100 (maze) and 1000 (pendulum). This leads to a vast amount

[1] http://www.cgplibrary.co.uk - v2.4 - accessed: 2018-01-12.

Table 1. Parameters and results of the neuroevolution run for both environments

	maze	pendulum		maze	pendulum
repeated runs	12	1	total agents	48e3	4020
evaluations per run	4020	4020 × 30	unique agents	43	3648
observed states per agent	30	max 15e3	unique states	22	30e6

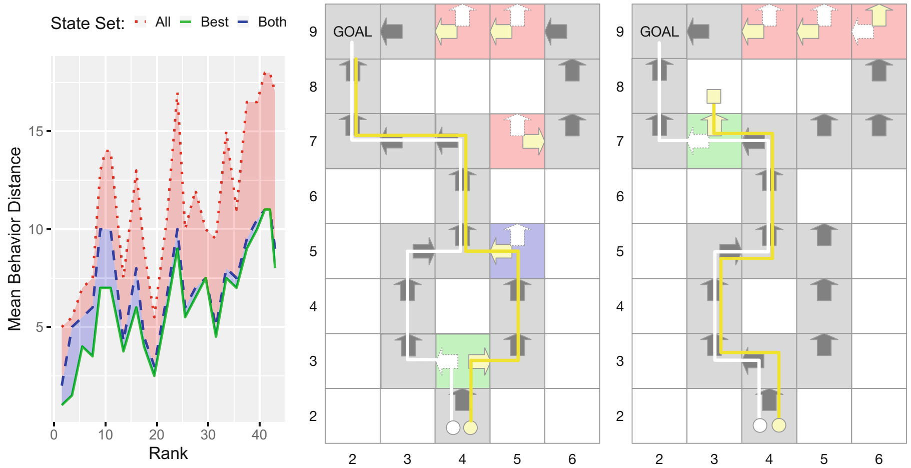
of different ANN topologies. The inner workings of the ANNs are complex and very difficult to compare [3,14]. Thus, only the reward and the behavior of the agents using these ANNs are considered observable.

Table 1 summarizes the parameters and outcomes of the Neuroevolution. The pendulum agents' rewards were aggregated over 30 different instances for reducing the impact of the random start positions; all states and actions from these instances are included in the agents' state sets. The agents of each environment were merged into one data set. Agents with equal state-input sets (i.e., those following precisely the same path) were filtered to acquire a feasibly sized data set. Due to the small number of input states for the maze environment, its amount of agents is significantly reduced. Conversely, the majority of trajectories in the pendulum experiment is unique. The cleaned-up data for each environment consist of all unique agents; the input states they observed, the corresponding actions, and their rewards. The agents were ranked, where equal performance leads to a shared rank. The maze problem has two best-ranked (rank 1) agents. For the following experiments, we arbitrarily chose one of these two as a reference agent (denoted as "best agent").

3.4 Experimental Setup for Analyzing the Behavior Measures

Behavior Comparison: First, explorative data analysis is conducted to analyze the behaviors and visualize them in the environment. We analyze the behavior for individual input states in particular for the maze problem, as we can manually identify wrong actions and understand their impact on the reward. Furthermore, we use a one-to-one comparison of agents with similar rewards to see the influence of UOS and DFS. For the pendulum problem, we analyze and reveal different behavior based on specific inputs and compare the influence of using different state sets as input for the behavior comparison. The denoted "best agent" for this problem is the best found.

State Importance: The maze environment has designed DFS and UOS, i.e., the forks with different importance and different exclusive paths to reach the goal. The goal of the importance analysis is to discover these states by comparing the behavior of all agents. We take a best-ranked agent as the reference for correct actions and calculate the percentage of different actions for each state by one-to-many comparisons, weighted by the difference in rank for these agents, by $\mathrm{d}(B_A, B_{A'}) \times \mathrm{d}(rank_A, rank_{A'})/sum(rank_A, rank_{A'})$. Further, we calculate and visualize the action reward rank (Sect. 2.2) for each state.



(a) Behavior Distance, Aggregated Mean by Rank

(b) 2x Rank 1 Comparison, d= All 5, Both 2, Best 1

(c) Best vs Rank 13, d= All 4, Both 1, Best 1

Fig. 2. One-to-one behavior comparison of the best agent against all agents, computed with different state input sets (a) and two selected examples (b,c) Green: differences on Input set C (BEST). Blue: differences on Input set D (BOTH, includes C). Red: differences on Input set B (ALL, includes C and D). Grey cells: agents act the same. a) Summary of behavior distance of best against all. Shaded areas illustrate the input set differences. b) Trajectories: best rank 1 (white) vs. another rank 1 (yellow) c) Trajectories: best (white) vs. Rank 13 (yellow) (Color figure online)

Reward Behavior Correlation: The main challenge in computing the $\mathrm{RBC}_{\mathrm{all}}$ and $\mathrm{RBC}_{\mathrm{opt}}$ is selecting a suitable state set to compare the behavior. With the previous experiments' experience, we defined different options to select a suitable state set and analyze which of them leads to the best overall RBC:

- Input set A: Random states sampled from all known states of all agents.
- Input set B: All known states of an environment.
- Input set C: The observed states of the best agent.
- Input set D: The observed states of both compared agents.
- Input set E: The observed states of one compared agent.

For the pendulum problem, we calculate the $\mathrm{RBC}_{\mathrm{all}}$ on an equidistant sampled (each 15th) subset of agents to significantly reduce the computation time.

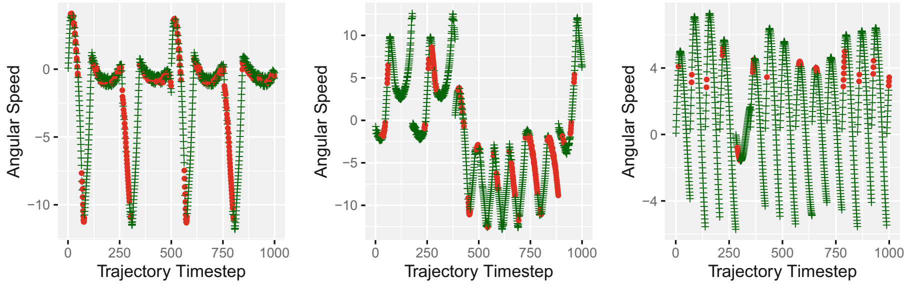
4 Results and Discussion

4.1 Behavior Comparison

The comparison of the best agent against all and selected inferior agents for the maze environment on different state sets (B, C, D) is illustrated in Fig. 2.

Interestingly, the best agent does not choose the best action in all states. It only chooses correctly for the states it observed by itself. The agent would run into walls if placed in certain positions (e.g., 5,5 or 4,9).

The behavior distance is amplified by different actions for states that were not observed by the compared agents, i.e., UOS lead to a larger distance, in particular visible in Fig. 2(b) and (c), where red cells highlight the UOS. If the state input set of both agents are used instead of all, the influence of UOS is smaller, as at least one of the agent has observed these states (blue line/cells). However, it is still evident for the higher ranks. A remarkable observation is shown in Fig. 2(c), for a comparison between the best agent and a medium-rank agent (rank 13). They have a behavior distance of only 1 if compared on their mutual state set and 4 with UOS considered. Their behavior on their mutual state set is nearly identical, despite the significant difference in rank.



(a) Trajectory from best, total difference is 33%

(b) Trajectory rank 2000, total difference is 24%

(c) Trajectory rank 3500, total difference is 4%

Fig. 3. Behavior comparison for the pendulum. The behavior difference from best versus rank 1000 is shown (Green cross = same action, red dot = different action) for the *angular speed* value over the first 1000 states of state sets from best, rank 2000, and rank 3500. As visible, the behavior difference is influenced by the state sets. Particularly, the dissimilarity in (c) is smaller. (Color figure online)

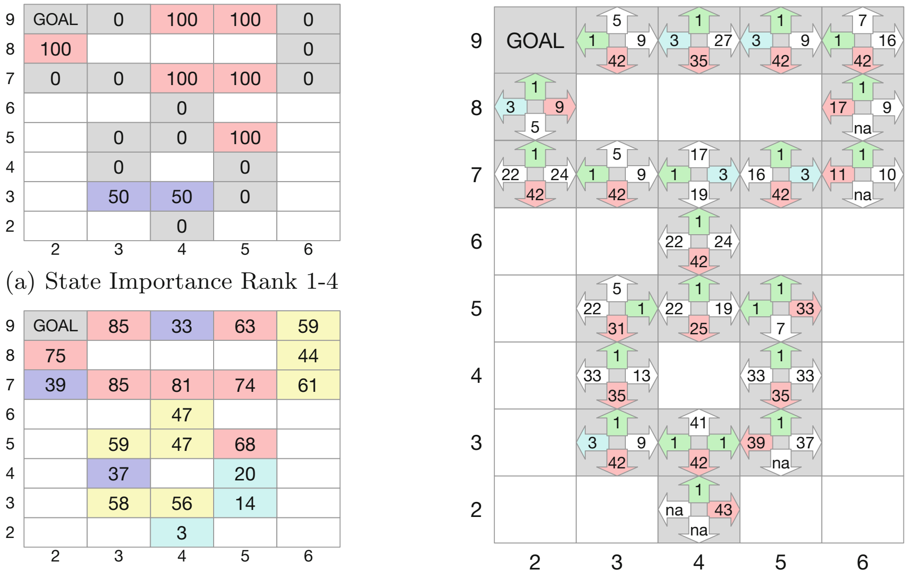
The number of acquired states for the pendulum is enormous and not suitable for complete comparisons as we visualized for the maze. However, we computed behavior differences of smaller state subsets and visualized them using a selected input, the *angular speed* of the pendulum, which is nearly zero if the pendulum is balanced in the upright position. Fig. 3(a) shows the behavior of the best agent against the rank 1000 (of 3648) agent by calculating it on best, as well as on rank 2.000 (b) and rank 3500 (c) input sets. Fig. 3(a) shows that for time-steps 250–300 and 750–800, the rank 1000 agent behaves consequently differently. These time-steps illustrate a situation of a falling pendulum, shortly after it was balanced. While the best agent countersteers this movement, the rank 1000 agent accelerates it. Consequently, we were able to identify a situation where the lower-ranked agent fails to learn the correct actions. However, as the actions are based on all inputs and the angular speed is just one of them, finding these situations manually remains challenging. Figure 3(c) shows what happens if the behavior of the two agents is compared on the input set of a distant ranked agent. The state input set of the rank 3500 is considerably different: Each recorded trajectory is only some time steps long, presumably caused by the agent quickly driving the base car to the horizontal limit, which leads to termination. For such extreme situations, both compared agents (best vs. rank 1000) seem to behave similarly. Conversely, their difference in reward seems to be related to smaller differences in critical situations. We can summarize these observations to identify some properties of the behavior space:

I) Agents with the same reward/rank can have a considerable behavior distance, mainly if compared on state input sets with UOS and DFS.
II) Small behavior differences (e.g., $d < 3$) can cause significant rank changes.
III) The input set has a huge impact on the behavioral distance comparison.

These observations reflect a central challenge of behavioral comparisons: We need to find important states and a suitable state set for conducting behavior comparisons. We argue that comparing the behavior on input sets with UOS can help distinguish between agents of similar reward, but is presumably overestimating their behavior distance on task level and further influenced by significant variances due to random actions in UOS. Moreover, comparing agents on state sets of other agents, even without considering the influence of UOS, might not reveal useful behavior distances, as these states represent situations not suitable for telling apart good behavior.

4.2 State Importance

For the state importance, we illustrate the percentage of agents with behavior differing from the best agent for each state, weighted by their differences in rank. In case of the maze, Fig. 4(a) shows this statistic only for agents reaching the goal, while Fig. 4(b) concerns all agents. Here, highly valued states are considered to be more important, as most agents behave dissimilarly to the presumed ‘correct‘ action. For the comparison between the best agents, many states show no importance, i.e., similar behavior in this set. The maze was designed such that the importance of the DFS fork in (4,3), should be less than the no-DFS fork in (4,7). This is represented by our importance measure, as (4,7) has a twice as high value in Fig. 4(a) and (b). However, the importance measure also provides other states with a high importance value, particularly visible in the maze's upper part. This can be explained by the type of behavior comparison (all states) and the influence of UOS for each agent. Agents can have ‘wrong’ behavior for these states, even if they can solve the environment. This is observable in Fig. 4(c), where for each state, the best agent choosing a specific action is shown. While for the state (4,3) and also for (4,7), we see a correct identification of different ways, (5,5), (4,9), and (5,9) give the wrong idea of correct actions, as the supposedly best-outlined action is surprisingly to run against a wall. This effect of UOS is amplified if all agents are considered. For example, the lowest-ranked agent runs directly against a wall. However, we compute and



	2	3	4	5	6
9	GOAL	0	100	100	0
8	100				0
7	0	0	100	100	0
6			0		
5		0	0	100	
4		0		0	
3		50	50	0	
2			0		

(a) State Importance Rank 1-4

	2	3	4	5	6
9	GOAL	85	33	63	59
8	75				44
7	39	85	81	74	61
6			47		
5		59	47	68	
4		37		20	
3		58	56	14	
2			3		

(b) State Importance all Ranks

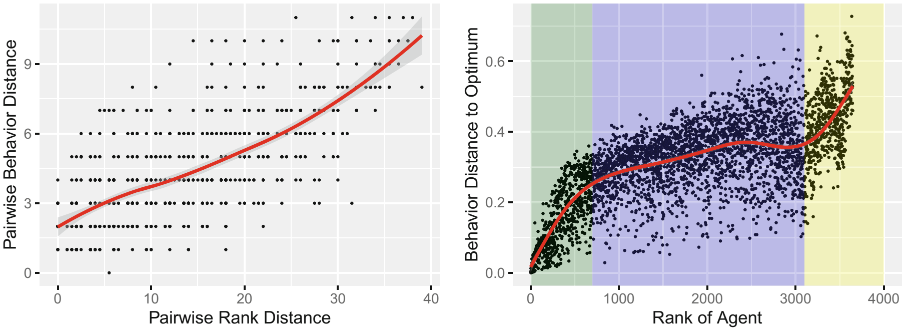
(c) Ranked Actions per State

Fig. 4. (a) State importance calculated using either the best agents or (b) all agents. Higher values depict higher importance, colored by value quarters for easier comparison. (c) Action reward rank. Shows the best rank choosing each action for each state. Green = rank 1, blue = rank 3, red = worst action rank. The two rank 1 agents choose different actions in (4,3) and (5,5). (4,3) is DFS, and (5,5) an UOS for the best agent. (Color figure online)

compare its behavior (intensified by its low rank) on all states. We assume that if we compare a broad set of agents, the UOS, with their presumably random actions, do not affect the importance as strongly. Thus, the shown importance is presumably higher in the states of the upper part of the maze, as only a minority of agents reach this part of the maze. For the rest, we are just comparing their behavior on UOS. Thus, our importance measures could also help identify states of an environment rarely reached by any agent in a set.

4.3 Reward Behavior Correlation

For the RBC analysis, the previous results have shown that it is essential to choose a suitable state set for each pairwise comparison. The results are displayed in Fig. 5 and Table 2. Figure 5(a) and (b) shows the resulting RBC_{all} and RBC_{opt} values, respectively. For both, the overall correlation is notably positive. In 5(b) it differs between good agents (rank 1–800), medium agents (800–3100), and poor agents (3100–3600). The other input sets (A, B, C, and E) lead to an inferior RBC_{opt} for both problems. Moreover, set D and E lead to the highest RBC_{all}, with a very significant difference for the maze problem. We assume that



(a) Pairwise RBC_{all} for maze, Correlation=0.72, Input set E

(b) Pairwise RBC_{opt} for pendulum, Correlation=0.70, Input set D

Fig. 5. (a) RBC_{all} plot for the maze environment (b) RBC_{opt} plot for the pendulum problem. Computed on input set E and D, respectively. A decent correlation is visible.

the best correlation would be achieved if agents are compared in their mutual behavior space. The presumed cause for the higher RBC_{all} is the reduction of the influence of unobserved states in the comparison. In particular, the maze environment agents have less UOS where they likely act random, if set D or E are considered. Including UOS in a comparison does thus not lead to a more detailed behavior distance, but one with higher overall variance, thus leading to an inferior RBC. This is visible for the pendulum, which for all sets, has a large amount of UOS due to the continuous state space, which leads to a smaller difference in the variants to compute the RBC_{all}. The overall positive RBC_{all} outlines the high potential of agent comparisons in the behavior space to improve the search for good ranking agents.

Table 2. RBC_{all} of all agents for different input sets

environment	A) random	B) all	C) reference	D) both observed	E) one observed
maze	0.27	0.29	0.28	0.62	**0.72**
pendulum	0.36	na	0.34	**0.45**	0.36

5 Conclusion

In this work, we investigated the properties of the behavior space of RL agents and how this space can help to compare agents in learning sets to gain valuable insights. Regarding our research questions, we can conclude for Q-1, that even small changes in the behavior can have considerable effects on the reward. At the same time, agents achieving the same reward can show quite different behavior. We believe that focusing only on the reward of an agent might not be the optimal choice. Instead, the agents' behavior can give valuable insights into how agents achieve that reward. This can reveal agents with surprising behavior or help to improve the learning process. For instance, reward functions can be designed to enforce or suppress specific behavior.

The analysis of Q-2 has shown that accessing the variable importance is challenging and highly dependent on the underlying set of agents and the environment. These challenges are mainly caused by comparing an agent on states, which were not observed by it, or are even not observable by this agent due to environment restrictions, e.g., mutually exclusive paths. For these cases, an agent's behavior can be random, even for the ones with the best reward. A comparison of behavior on these states might deliver misleading results. Only if multiple agents observed states, we could access their real importance.

This finding is further stressed when considering Q-3. The RBC is highest if we consider pairwise behavior comparisons on those states that have been observed by both compared agents. The reasonable positive RBC shows that the behavior space is a promising concept. We suggest that searching in that space may be beneficial.

For future work, we aim to take a close look at how the understanding of behavioral spaces can be exploited, e.g., by new reward measures, direct search in the behavior-space, and specialized search operators:

Reward Measures: Ideally, reward measures help to steer the search into desirable areas of the search space. Understanding which states are critical to receiving a good match between behavior and reward may help design better reward measures. The importance of developing useful reward measures for RL is stressed in a review by Doncieux and Mouret [2].

Search in Behavior Space: The usage of agents' behavior distance as an additional search criterion seems very attractive. It can be used to preserve diversity in evolutionary search procedures [1]. Further, the search for a specific behavior may be of interest, independent or in addition to reward-driven search, e.g., by modeling the reward to behavior space with surrogate models. An example application would be *inverse reinforcement learning* [10]. The search in behavior space allows the use of completely different agent topologies or even comparing agents trained by different algorithms.

Search Operators: Finally, a good understanding of the latent, behavioral space may help to define better search operators. For instance, search operators could be designed to search directly in the behavior space, rather than the policy or topology space.

References

1. Doncieux, S., Mouret, J.: Behavioral diversity measures for evolutionary robotics. In: IEEE Congress on Evolutionary Computation, pp. 1–8 (2010)
2. Doncieux, S., Mouret, J.-B.: Beyond black-box optimization: a review of selective pressures for evolutionary robotics. Evol. Intell. **7**(2), 71–93 (2014). https://doi.org/10.1007/s12065-014-0110-x

3. Gaier, A., Asteroth, A., Mouret, J.-B.: Data-efficient neuroevolution with kernel-based surrogate models. In: Genetic and Evolutionary Computation Conference (GECCO) (2018)
4. Hagg, A., Zaefferer, M., Stork, J., Gaier, A.: Prediction of neural network performance by phenotypic modeling. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion - GECCO 2019, Prague, Czech Republic, pp. 1576–1582. ACM (2019)
5. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evol. Comput. **23**(3), 343–367 (2015)
6. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1995, pp. 184–192. Morgan Kaufmann (1995)
7. Kallel, L., Schoenauer, M.: Fitness distance correlation for variable length representations. Technical Report 363, CMAP, Ecole Polytechnique (1996)
8. Khan, M.M., Khan, G.M., Miller, J.F.: Evolution of neural networks using cartesian genetic programming. In: IEEE Congress on Evolutionary Computation, pp. 1–8, July 2010
9. Meyerson, E., Lehman, J., Miikkulainen, R.: Learning behavior characterizations for novelty search. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO 2016, pp. 149–156. Association for Computing Machinery, New York (2016)
10. Ng, A.Y., Russell, S.J., et al.: Algorithms for inverse reinforcement learning. In: Icml vol. 1, pp. 663–670 (2000)
11. Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. IEEE Trans. Cybern. **47**(9), 2951–2965 (2016)
12. Ollion, C., Doncieux, S.: Why and how to measure exploration in behavioral space. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, GECCO 2011, pp. 267–274. Association for Computing Machinery, New York (2011)
13. Pugh, J.K., Soros, L.B., Stanley, K.O.: Searching for quality diversity when diversity is unaligned with quality. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) PPSN 2016. LNCS, vol. 9921, pp. 880–889. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45823-6_82
14. Stork, J., Zaefferer, M., Bartz-Beielstein, T., Eiben, A.E.: Surrogate models for enhancing the efficiency of neuroevolution in reinforcement learning. In: Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2019, Prague, Czech Republic, pp. 934–942. ACM (2019)
15. Turner, A.J., Miller, J.F.: Cartesian genetic programming encoded artificial neural networks: a comparison using three benchmarks. In: Proceedings of the GECCO 2013, pp. 1005–1012. ACM (2013)
16. Zaefferer, M., Stork, J., Flasch, O., Bartz-Beielstein, T.: Linear combination of distance measures for surrogate models in genetic programming. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 220–231. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99259-4_18
17. Zuo, X.: mazelab: a customizable framework to create maze and gridworld environments (2018). https://github.com/zuoxingdong/mazelab