# Ökolopoly: Case Study on Large Action Spaces in Reinforcement Learning*

Raphael C. Engelhardt[1][0000−0003−1463−2706], Ralitsa Raycheva, Moritz Lange[2][0000−0001−7109−7813], Laurenz Wiskott[2][0000−0001−6237−740X], and Wolfgang Konen[1][0000−0002−1343−4209]

[1] Cologne Institute of Computer Science, Faculty of Computer Science and Engineering Science, TH Köln, Gummersbach, Germany
{Raphael.Engelhardt,Wolfgang.Konen}@th-koeln.de
[2] Institute for Neural Computation, Faculty of Computer Science, Ruhr-University Bochum, Bochum, Germany
{Moritz.Lange,Laurenz.Wiskott}@ini.rub.de

**Abstract.** Ökolopoly is a serious game developed by biochemist and pioneer of networked thinking Frederic Vester with the goal of promoting the understanding of interactions in complex systems. Since the strategy game has a huge action space, it presents a challenge to Deep Reinforcement Learning (DRL). In this article we explain how the board game is made available as a reinforcement learning environment, we compare different methods of making the large spaces manageable as well as different reward functions, and we shed light on the conditions under which DRL agents in this case study are able to learn the game from self-play.

**Keywords:** Deep Reinforcement Learning · Large Action Space · Cybernetics · Serious Games.

## 1 Introduction

Despite the overwhelming success of Deep Reinforcement Learning (DRL) in the last decade, large action spaces can still pose a challenge for DRL algorithms [6]. The serious game *Ökolopoly* [17] is an example of an environment exhibiting such a large action space. The game has its roots in cybernetics as it aims at teaching the players how to steer circular causal processes. The game in its internationalized computer simulation version *Ecopolicy* is cited as an example for training systemic and long-term thinking in complex, interconnected systems opposed to linear thinking in immediate, linear cause-effect terms. International competitions in schools were held (*"Ecopoliciade"*) to train pupils the art of thinking holistically [1, Sect. 2.2] [10, Sect. 3.2].

* This research was supported by the research training group "Dataninja" (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

The combinatorial explosion of choices quickly leads to a large number of possible game states and a very large action space, making this game an interesting test case for DRL algorithms [6].

In this paper we describe how the board game can be formalized as a Reinforcement Learning (RL) environment in OpenAI Gym [2]. Given this implementation, we investigate the following research questions:

RQ 1 Is it possible for RL agents to learn the game Ökolopoly from self-play?
RQ 2 Which components are essential for learning success (if any)?
RQ 3 Can the agent learn to propose only valid actions or is it necessary that the environment transforms invalid actions to valid ones?

We explain and experimentally test different methods of approaching such large action spaces. We will show which of these methods are essential for a DRL agent to successfully learn the game and to what extent the agent can develop an "understanding" of the underlying game mechanics. Our hope is that the results from this Ökolopoly case study are also useful for other RL problems with large action spaces.

The remainder of the paper is structured as follows: Section 2 will discuss related work. In Sect. 3 we briefly describe the game of Ökolopoly. Section 4 contains technical information about how the game is translated from a board game to the domain of RL as well as methods to treat the large action space and different reward functions. Section 6 presents the experimental outcomes. In Sect. 7 we answer the research questions and give a short conclusion.

## 2 Related Work

Large action spaces have been identified as one of the main challenges in RL [6]. Proposed solution techniques may factorize the action space into binary or ternary subspaces [11], embed the discrete action space in a continuous one [5] or use the technique of action elimination [18]. In our work we will use the first two techniques as well. Instead of action elimination we use action normalization (projection on valid actions, see Sect. 5.2). While the above-mentioned papers investigate action spaces of size $10^2 - 10^4$, the application studied in this work has an action space of size $10^6 - 10^8$ (see Sect. 3.3, depending on whether we use action normalization or not).

The game of Ökolopoly has – to the best of our knowledge – not been solved successfully by RL methods before.

Serious games in biology [15] and ecology [10] have a long tradition and are often used for educational purposes. The use of RL for serious games is an emerging research topic [4]. Dobrovsky et al [3] use interactive DRL to balance in serious games the transfer of knowledge and the entertainment based on the context information from gameplay. Another example are rehabilitation serious games [9] where an RL-based approach is used to modify the difficulty of the rehabilitation exercises.

## 3 The Game of Ökolopoly

Designed by Frederic Vester and made available as board game in 1984 [16,17], the game of Ökolopoly aims to raise awareness for and deepen the understanding of acting in systems of complex interdependencies.

The game is conceived as a single-player turn-based strategy game. It models the state of the imaginary country of *Kybernetien* with scores on eight interacting departments or fields (such as Population, Quality of Life or Environment). The player's task is to lead the country to success by cleverly distributing available action points to the fields and developing an understanding of the underlying interdependencies. The fields are described in Tab. 1 with their minimal, maximal, and starting values.

**Table 1.** The eight fields, number of rounds played, and available action points determining the state of the game. Five of these fields are directly *actionable*, i.e., they may receive action points.

| **Field** | **min** | **max** | **start value** | **actionable** |
|---|---|---|---|---|
| Sanitation | 1 | 29 | 1 | yes |
| Production | 1 | 29 | 12 | yes |
| Environment | 1 | 29 | 13 | |
| Education ($e$) | 1 | 29 | 4 | yes |
| Quality of Life ($q$) | 1 | 29 | 10 | yes |
| Population Growth ($g$) | 1 | 29 | 20 | yes |
| Population ($b$) | 1 | 48 | 21 | |
| Politics ($p$) | $-10$ | 37 | 0 | |
| Rounds ($r$) | 1 | 30 | 1 | |
| Action Points | 1 | 36 | 8 | |

### 3.1 One Turn of the Game

In each turn (or timestep in RL terms) the player chooses how to distribute the available action points among the five fields Sanitation, Production, Education, Quality of Life, and Population Growth, so that the respective field values are incremented by those action points. Only for the field Production the player may also choose to diminish its value; this, however, costs action points as well. There is no minimum value of action points to use, i.e., the player may choose to save some or all action points for the next round.

Once the action points are distributed, certain interdependency functions between the fields, e.g., $G_i(x), i = 1, \ldots, 4$ for field population growth $g$, where $x$ is any of the other fields,$^{\text{WK}}$ give rise to a number of automatic adjustments (feedback effects). For example, if Education is $e = 19$, then Population Growth changes by $G_1(e) = +3$. The interdependency functions are deterministic, but complex and hard to memorize for a human player. Finally, the action points available for the next round are assigned following the interdependency functions, the round counter is increased by one, and the round ends.[3]

For higher values of Education $e$, the interdependency function $G_1(e)$ may exhibit a number preceded by $\pm$: in those cases the player can choose to diminish or increase the field Population Growth $g$ in the given range (e.g., if $G_1(e) = \pm 3$ then any choice of $\Delta g \in \{-3, -2, -1, 0, 1, 2, 3\}$ is allowed). The reasoning behind this rule is that a sufficiently educated population is able to steer its growth.

## 3.2 End of the Game

When one or more fields leave the allowed range (either due to the distribution of action points or due to the automatic adjustments thereafter) or when 30 rounds are played, the game ends. At the end of the game, the balance score $B$ is computed as a function of the field Politics $p$, the value of the interdepency function $D(q)$, which is monotonically rising with Quality of Life $q$, and the number of played rounds $r$:

$$B(p, q, r) = \begin{cases} \frac{10\,[p+3D(q)]}{r+3} & \text{if } 10 \leq r \leq 30 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

This means that a balance score of 0 is given, if the condition $10 \leq r \leq 30$ is not met. The game instructions define a score of over 20 as exceptionally good.

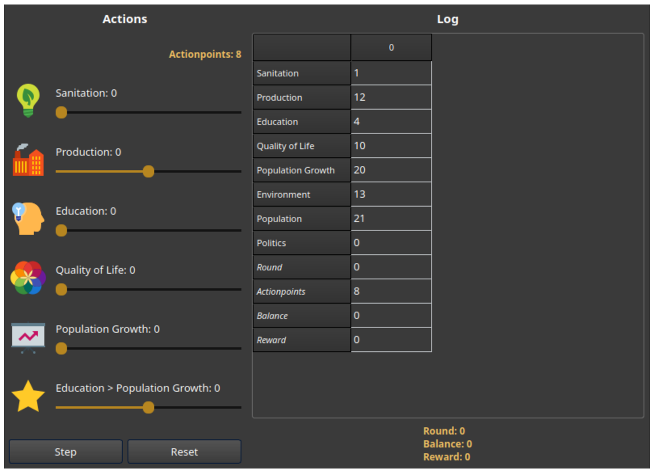
## 3.3 Observation and Action Spaces

Given Tab. 1, the observation space allows for $29^6 \cdot 48^2 \cdot 36 \cdot 30 \approx 1.48 \times 10^{14}$ different states.

When distributing $a \in \{1, \ldots, 36\}$ available action points to the five fields, there are in principle $(a+1)^5$ possible combinations. But since a player cannot distribute more action points than available, the number of valid combinations is much smaller. Counting the number of valid and the number of possible combinations for all values of $a$, we find that there are $9.7 \times 10^6$ valid combinations, which are only 1.1% of all $9.1 \times 10^8$ possible combinations.

This poses two challenges for any DRL agent: Firstly, even when restricting the agent to valid actions (e.g., by sum normalization, see Box action wrapper in Sect. 5.2), there is still a large number of $9.7 \times 10^6$ options. Secondly, if we give the agent no information whether a possible action is valid or not (no action wrapper), it has to learn by reinforcement feedback to suggest only valid combinations (otherwise the episode will terminate immediately). This is a demanding task given the small percentage of only 1.1% valid combinations.

[3] An advanced version of the game provides optional "event cards" to be drawn every five rounds. We ignore this advanced version in our implementation.



Actionpoints: 8

Sanitation: 0

Production: 0

Education: 0

Quality of Life: 0

Population Growth: 0

Education > Population Growth: 0

| | 0 |
|---|---|
| Sanitation | 1 |
| Production | 12 |
| Education | 4 |
| Quality of Life | 10 |
| Population Growth | 20 |
| Environment | 13 |
| Population | 21 |
| Politics | 0 |
| *Round* | 0 |
| *Actionpoints* | 8 |
| *Balance* | 0 |
| *Reward* | 0 |

Round: 0
Balance: 0
Reward: 0

**Fig. 1.** GUI for the RL environment

## 4 Implementation of the Game

In this section we briefly describe how the board game was implemented as an OpenAI Gym [2] compatible RL environment. The code and GUI (Fig. 1) for human-play were adapted from [13] and are available on Github[4].

### 4.1 Representation of the Observation Space

The observation space is internally represented as a ten-dimensional object of class `MultiDiscrete` containing the values of the eight fields, the number of rounds played, and the currently available action points. The agent has therefore full access to all information visible to the human player of the board game. As different ranges are allowed in the different dimensions of the observation space (the field Politics can even contain negative values which are not supported by `MultiDiscrete`), allowed states are shifted accordingly (see first row of Tab. 2).

### 4.2 Representation of the Action Space

In a similar way the action space is encoded as a six-dimensional `MultiDiscrete` object containing the number of action points assigned to each of the five fields. The sixth number accounts for the possibility of modifying Population Growth by up to $\pm 5$ points according to the value displayed in the field $G_1(e)$ at no extra action point costs (see end of Sect. 3.1).

[4] [Github link]

### 4.3 Reward Functions

The basic reward function merely implements Eq. 1. This requires the agent to perform a long streak of profitable actions, which is difficult to find by exploration, before receiving any learning signal (only after the terminal step and if it occurs after at least ten rounds). For this reason, we implemented and tested different auxiliary reward structures, which additionally assign a return after each step. These dense reward functions are described in detail in Sect. 5.3.

## 5 Methods

To assess the impact of different ways to handle the large observation and action spaces and different reward structures on the success of training, we performed experiments with different combinations of DRL algorithms and wrappers we describe in the following. A summary of the different spaces and their implementation is given in Tab. 2.

### 5.1 Observation Wrappers

We consider three different observation wrapper choices that should enable the algorithms to digest the huge observation space.

**None** We treat the observation space as a `MultiDiscrete` object.

**Box Observation Wrapper** In the case of `MultiDiscrete` observations, the DRL agent does not have an intrinsic concept of distance between possible values in each dimension of the observation space. To mitigate this problem, the Box observation wrapper represents each of the eight fields, the current round and the available action points as a value in a continuous `Box` reaching from the minimum to the maximum of the respective observation.

**Simple Observation Wrapper** This observation wrapper subdivides each dimension into just three possible values: *low*, *medium*, and *high*. This way each field, the current number of rounds played, and the available action points are represented each by one value in $\{0, 1, 2\}$. The state space is thereby reduced to $3^{10} = 59\,049$ different states. The observation wrapper is implemented as a ten-dimensional `MultiDiscrete` object.

### 5.2 Action Wrappers

Similarly, we implemented and tested three different action wrapper choices to simplify the action space.

**None** The action space is the unaltered `MultiDiscrete` object from Sect. 4.2. Since there is no mechanism translating actions into legal moves, the validity of an action is not ensured. In fact the overwhelming majority of points in this action space do not correspond to valid moves.

**Box Action Wrapper** This wrapper transforms the discrete action space into a continuous one of type `Box`. The elements of this six-dimensional vector range from $[0,-1,0,0,0,-1]$ to $[1,1,1,1,1,1]$ and have the meaning described in Sect. 4.2. If more than the available action points should be distributed according to the tentative action vector $\mathbf{a}'$, the values are normalized by their absolute sum, multiplied with the currently available action points $n$, and rounded to the next integer:

$$a_i = \left\lfloor \frac{a'_i}{\sum_{i=0}^{4} |a'_i|} \cdot n + 0.5 \right\rfloor \qquad \text{for} \quad i = 0, \ldots, 4$$

If, due to rounding effects, the sum of action points $a_i$ still exceeds $n$, the highest element of vector $\mathbf{a}$ is decreased by one.[5]

**Simple Action Wrapper** Analogous to the Simple observation wrapper, this action wrapper reduces the number of available discrete actions by dividing the number of available action points in three equal or near-equal parts. These blocks of action points can then be distributed among the five different fields in the game. This distribution is encoded as a six-digit string: The first five digits from the left assume values in $\{0,1,2,3\}$ which represent the number of action point partitions assigned to the respective field (the sum of the first five digits may therefore not exceed 3); the rightmost digit encodes whether action points are added (0) or deducted (1) from Production[6]. In total there are 77 such strings to represent legal moves. Thus, the number of available actions is largely reduced and makes the problem better learnable because the agent has fewer actions to disentangle. On the other hand, it is also less precise, because the agent might distribute to a given field, e.g., no or at least one third of the action points. In certain situations, this can cause episodes to break off, which could have been avoided by a finer-grained distribution. The Simple Action Wrapper mimicks a possible human strategy of being less precise but actionable in unknown complex environments.[WK]

The action space is implemented as two-dimensional `MultiDiscrete` object of which the first element contains the index of one of the 77 legal moves while the second one contains the possibility of modifying Population Growth by up to $\pm 5$. The total number of possible actions is therefore reduced to $77 \cdot 11 = 847$.

## 5.3 Reward Functions

We trained DRL agents using the score defined by the rules of the board game (see Sect. 3.2) as well as two other reward functions that alleviate mentioned problems related to this sparse reward structure.

**Balance** At intermediate timesteps no reward is given. Only the final step yields a reward $B$ computed according to Eq. 1.

**PerRound** This reward wrapper issues an additional constant reward $R_c$ after each intermediate step. After the episode's last step the known reward $B$

[5] The sixth dimension $a'_5 \in [-1,1]$ (modifier for Population Growth $g$, Sect. 3.1) is multiplied by 5, rounded and then appended to $\mathbf{a}$.

[6] As an example the string 020101 encodes the following distribution of action points: one third of the action points are added to Education, two thirds of the action points are deducted (rightmost digit is 1) form Production.

of Eq. 1 is added to the return. This should be an incentive for the agent to reach a higher number of rounds, as the rules of the game suggest that between 10 and 30 rounds should be played. Different values for $R_c$ will be investigated. If not stated otherwise, we use $R_c = 0.5$,

**Heuristic** A heuristic that keeps Production and Population at healthy, intermediate values (15 and 24 respectively) empirically seemed to be a good strategy. To this end, the following auxiliary reward $R$ is assigned after each intermediate step:

$$\begin{aligned} R &= s \cdot (R_{\text{prod}} + R_{\text{pop}}) \\ \text{with} \quad R_{\text{prod}} &= 14 - |15 - V_{\text{prod}}| \\ R_{\text{pop}} &= 23 - |24 - V_{\text{pop}}| \end{aligned}$$

After the last step of each episode, the usual balance $B$ from Eq. 1 is given as reward. The scaling factor $s$ allows to steer the agent to maximize the auxiliary reward $R$ and therefore the number of rounds (for higher values such as $s = 1$) or to maximize the balance $B$ (for smaller values of $s$, see Fig. 7). Where not explicitly stated otherwise we use $s = 1$.

**Table 2.** Overview of action and observation space, wrappers and their representation

| **Space** | **Wrapper** | **Object** |
|---|---|---|
| Observation | None | `MultiDiscrete([29,29,29,29,29,29,48,48,31,37])` |
| | Box | `Box(low=[1,1,1,1,1,1,1,-10,0,0], high=[29,29,29,29,29,29,48,37,30,36])` |
| | Simple | `MultiDiscrete([3,3,3,3,3,3,3,3,3,3])` |
| Action | None | `MultiDiscrete([29,57,29,29,29,11])` |
| | Box | `Box(low=[0,-1,0,0,0,-1], high=[1,1,1,1,1,1])` |
| | Simple | `MultiDiscrete([77,11])` |

## 5.4 DRL Algorithms

During the experiments we trained agents with the on-policy algorithm PPO [14] and the off-policy algorithms TD3 [7] and SAC [8] using their implementation provided by the Python DRL framework Stable-Baselines3 [12]. We use the default hyperparameters of SB3; for the TD3 agents, Gaussian action noise with $\sigma = 0.1$ is applied. While all mentioned algorithms are suited to handle `MultiDiscrete` and `Box` observation spaces, not all are compatible with `MultiDiscrete` action spaces (see Tab. 3).

## 5.5 Experimental Setup

As a consequence of Tab. 2 and Tab. 3, not every combination of wrapper and algorithm is possible: we have $3 \cdot 3 \cdot 3 = 27$ combinations for PPO, but only 9

**Table 3.** Compatibility of mentioned algorithms and used action and observation space representations

| | Observation Space | | Action Space | |
|---|---|---|---|---|
| **Algorithm** | `MultiDiscrete` | `Box` | `MultiDiscrete` | `Box` |
| PPO | ✓ | ✓ | ✓ | ✓ |
| SAC | ✓ | ✓ | ✗ | ✓ |
| TD3 | ✓ | ✓ | ✗ | ✓ |

combinations for SAC and TD3 (only `Box` action wrappers). For our experiments we train DRL agents with all available combinations of observation and action wrappers and reward functions for 800 000 timesteps while logging the balance $B$ (Eq. 1), the number of played rounds $r$, and the reward as seen by the DRL agent. After each completed training episode the agent plays one evaluation episode using deterministic predictions according to its current training state. As a measure of the agents' performance we compute the averages and standard deviations of balance $B$ and played rounds $r$ of the last 1 000 evaluation episodes.

We investigate the stability of the training process with respect to initialization of the agents' networks, the role of the different methods in handling observation and action space, the impact of reward functions, and the strategy of single agents.

## 6 Results

After examining the training history we examine our results regarding the role of different representations of observation and action space as well as different reward functions. We touch upon the possibility of steering the agents' focus by suitable choices of rewards and provide more insights regarding the agents' performance in terms of balance and played rounds.

### 6.1 Stability of Training

To asses how critical the random initialization of the DRL agent's networks is, as an example we trained 5 PPO agents on the combination Box observation wrapper, Box action wrapper, and PerRound reward $R_c = 1$ under identical conditions except for a different seed of the DRL algorithm. As can be seen in Fig. 2, the initialization has at most only marginal effect on the training process and especially on the outcome.

### 6.2 Impact of Wrappers

To compare the training outcome of different combination of wrappers, we examine the average and standard deviation of agents' performance across the last 1 000 training episodes in terms of played rounds and balance for all reward

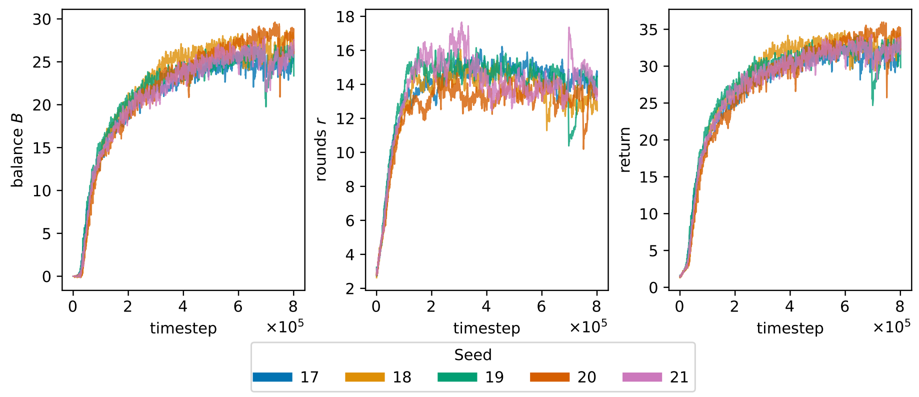


**Fig. 2.** Training curves of 5 differently seeded PPO agents (Box observation, Box action wrapper, PerRound reward $R_c = 0.5$) encoded by color. (For better visualization a Savitzky-Golay smoothing filter was applied.)

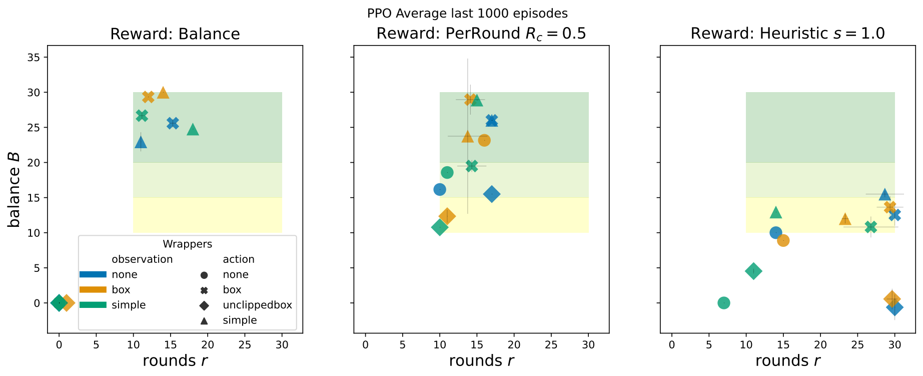


**Fig. 3.** Average and standard deviation of number of played rounds and balance for the last 1 000 evaluation episodes of **PPO** agents. Different reward functions are shown in separate plots, while observation and action wrapper are encoded by color and marker. The areas marked by colored backgrounds represent scores considered as increasingly good by the rules of the game. (n.b. at $(0, 0)$ the combination without observation wrapper and UnclippedBox action wrapper, as well as all combinations without action wrapper are hidden below the green diamond marker)

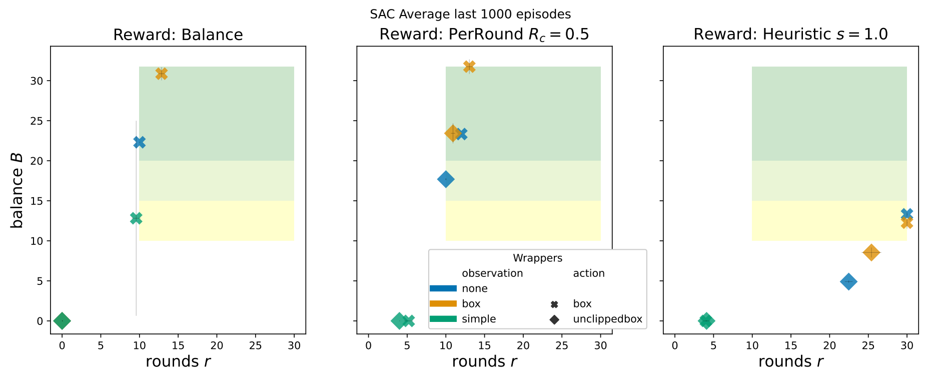


**Fig. 4.** Average and standard deviation of number of played rounds and balance for the last 1 000 evaluation episodes of **SAC** agents. The plots are structured as Fig. 3. Tables 2 and 3 show why this plot contains fewer results. (n.b. the three diamond markers at $(0, 0)$ are overlapping in the leftmost plot)

functions that were used for training. The results are shown in Fig. 3 (PPO) and Fig. 4 (SAC).

Without ensuring valid actions by the use of action wrappers and with only sparse reward, the agents don't receive a guiding learning signal and consequently fail (three overlaying round markers at $(0, 0)$ in the left plot of Fig. 3). With action wrappers intrinsically allowing only valid moves, the agents accumulate in a specific region of the multiobjective plot with exceptionally high score after having played between 10 and 20 rounds. If PPO agents are rewarded with a fixed positive amount $R_c$ after each intermediate step, they are able to cope with even large multidiscrete observation and action spaces, learn legal moves, and find a suitable strategy, as shown in the central plot. While a clear clustering by color or marker could not be observed here, the right plot shows the agents' difficulties to find suitable actions by the accumulation of round markers in the lower left side.

While PPO agents exhibit an overall remarkably good performance (Fig. 3), SAC agents using Simple observation wrapper seem to struggle (Fig. 4). We trained also TD3 agents with default hyperparameters and Gaussian action noise with $\sigma = 1$. Since their results were inferior to PPO and SAC (only some TD3 combinations resulted in a balance above 10; most combinations have a balance $B \approx 0$ and rounds $r < 10$), we do not show these results in a separate figure.

The action wrappers serve a dual purpose: not only do they reduce the number of available actions, but they also ensure only valid actions can be performed. To investigate which effect is predominant, all PPO and SAC agents were also trained using a modified Box action wrapper. This *UnclippedBox* action wrapper still represents the discrete action space as a continuous box space. But it does not ensure that the intended distribution of action points is possible (i.e. does not exceed the number of available action points). Figures 3 and 4 show how

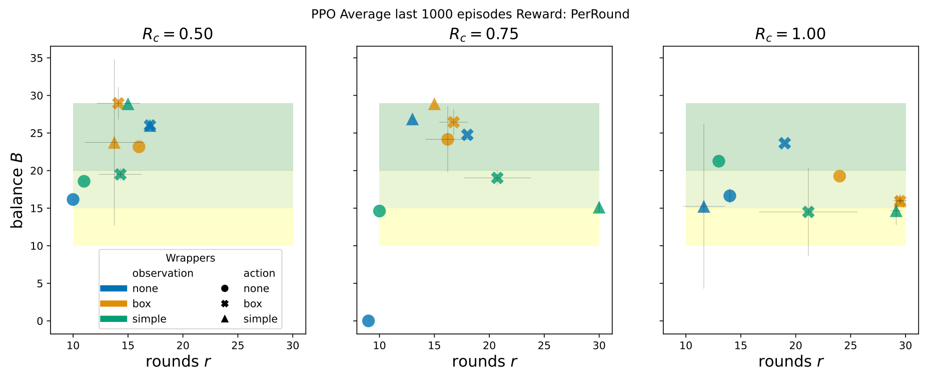


**Fig. 5.** Effect of different values of constant per-round reward $R_c$ on PPO agents

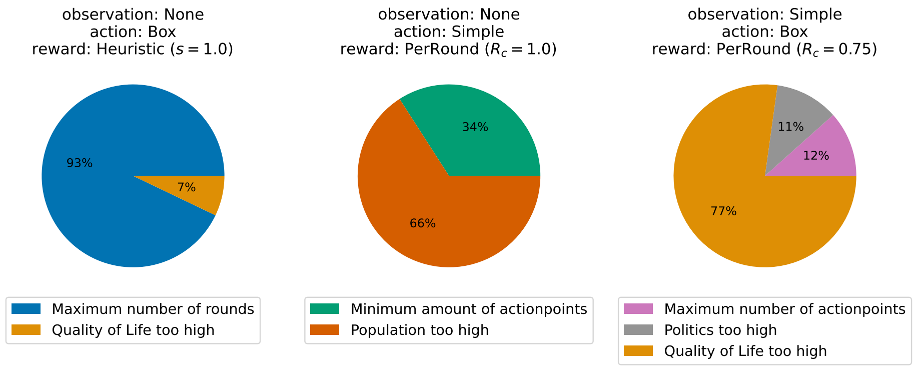


**Fig. 6.** Reason for episodes' termination of the last 1 000 evaluation episodes during training. Shown is a small subset of PPO agents from the set of about 140 pie charts (see text).

the additional task of learning valid distributions of action points negates any success in the case of a sparse reward. With dense rewards, *UnclippedBox* leads to somewhat reduced performance but it is noteworthy that the agents learn to perform mostly valid actions.

Investigating the agents' gameplay further, Figure 6 shows statistics of reasons for termination of the last 1 000 evaluation episodes during training. Due to space restrictions, we can show here only 3 of about 140 pie charts from possible combinations[7]. We can see that a variety of reasons leads to the termination of episodes. What we want to demonstrate with Figure 6 is that many of them are "positive" in the sense that they leave the range of possible values in a direction that would commonly be considered desirable (e.g., "Quality of Life too high").

[7] The full set of pie charts is available at [GitHub-link].

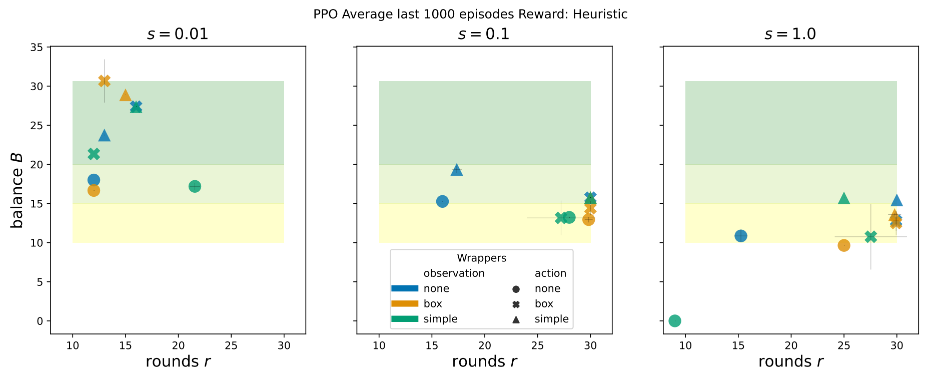


**Fig. 7.** Effect of different scaling of Heuristic reward on PPO agents.

### 6.3 Reward Shaping

The importance of a suitable reward function is well-known among RL practitioners. This holds true also in the case of Ökolopoly. Not only does the reward function have arguably the biggest effect on overall performance (see Fig. 3), but also do the parameters of the single reward function itself play an important role. In Fig. 5 we show the results of comparing three different values of the constant per-round reward $R_c$. By changing this value we can tune the agent towards surviving more rounds $r$ (in case of higher values $R_c$) on one side or towards prioritizing a higher balance $B$ on the other.
Similarly, Fig. 7 shows the results for different values of scaling factor $s$.

### 6.4 Optimal Balance and Rounds

Looking at Fig. 5, one might ask why in the case $R_c = 0.5$ (where the incentive is more on balance) the high balance results are not possible together with a higher number of rounds: The average episode length does not exceed 17 rounds. A similar observation is made in the 'Reward: Balance' plots of Fig. 3 and Fig. 4.

The cause is visualized in Fig. 8: First, the balance $B$ (for rounds $r < 10$ only hypothetical) shows a local optimum roughly at $r \approx 15$. Secondly, the number of episodes with $r \in [10, 15]$ is about 2000 times higher than the number of episodes with $r \in [20, 25]$, meaning that the average in Fig. 5, $R_c = 0.5$, is likely to be below $r \approx 15$. The agent has learned that it is not detrimental to stop an episode after $10-15$ rounds, because a longer lasting episode would not improve (or even diminish) $B$. This is why the agent often stops quite early (after surpassing the required 10 rounds) unless a special incentive (e.g., $R_c = 1$) tells it otherwise. Missing: short mentioning of computation times, maybe small table$^{\mathrm{RE}}$

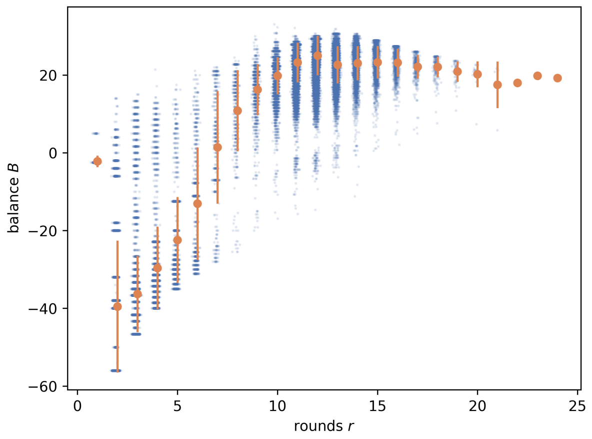


**Fig. 8.** Correlation of rounds $r$ and balance $B$. Blue dots mark every pair $(r, B)$ encountered by the PPO agent during training for the combination (Box observation wrapper, Box action wrapper, reward Balance). Orange dots mark average for every number of rounds.

## 7 Conclusion

In this article we described how we made cybernetic board game Ökolopoly accessible as a RL environment. Various ways of simplifying the large observation and action spaces as well as different reward functions have been investigated. We were able to show that in many of our combinations (wrapper, reward), the DRL agents could successfully learn how to deal with the rather complex interdependencies of the game. Our results show that DRL agents can consistently learn the game from self-play (RQ 1). However, one of the following conditions have to be met in order to be successful (RQ 2): (i) either an action wrapper that inherently maps to only valid actions (Box or Simple) or (ii) a dense reward function (PerRound or Heuristic).

If neither (i) nor (ii) is present, no learning occurs in this large action space. We have shown that within the Box action wrapper not the mapping to continuous space but the clipping to valid actions is essential. If, without any action wrapper, only (ii) is present, we got the interesting result that the agent is able to learn just by self-play to propose only valid actions (RQ 3).

We invite everyone to use our new open source RL environment Ökolopoly, which is made available as a benchmark for researchers to test their methods for handling large observation and action spaces.

## References

1. Bosch, O., Nguyen, N., Sun, D.: Addressing the Critical Need for 'New Ways of Thinking' in Managing Complex Issues in a Socially Responsible Way. Business Systems Review **2**, 48–70 (2013)

2. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym (2016). https://doi.org/10.48550/arXiv.1606.01540
3. Dobrovsky, A., Borghoff, U.M., Hofmann, M.: Improving adaptive gameplay in serious games through interactive deep reinforcement learning. Cognitive infocommunications, theory and applications pp. 411–432 (2019)
4. Dobrovsky, A., Wilczak, C.W., et al.: Deep reinforcement learning in serious games: Analysis and design of deep neural network architectures. In: Moreno-Díaz, R., et al. (eds.) Computer Aided Systems Theory – EUROCAST 2017. pp. 314–321 (2018). https://doi.org/10.1007/978-3-319-74727-9_37
5. Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., Coppin, B.: Deep reinforcement learning in large discrete action spaces (2015). https://doi.org/10.48550/arXiv.1512.07679
6. Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning **110**(9), 2419–2468 (2021). https://doi.org/10.1007/s10994-021-05961-4
7. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: Dy, J., Krause, A. (eds.) Proc. 35th Int. Conf. on Machine Learning, PMLR. vol. 80, pp. 1587–1596 (2018), https://proceedings.mlr.press/v80/fujimoto18a.html
8. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Dy, J., Krause, A. (eds.) Proc. 35th Int. Conf. on Machine Learning, PMLR. vol. 80, pp. 1861–1870 (2018), https://proceedings.mlr.press/v80/haarnoja18b.html
9. Hornak, D., Jascur, M., Ferencik, N., Bundzel, M.: Proof of concept: Using reinforcement learning agent as an adversary in serious games. In: 2019 IEEE International Work Conference on Bioinspired Intelligence (IWOBI). pp. 111–116 (2019)
10. Nguyen, N.C., Bosch, O.J.H.: The art of interconnected thinking: Starting with the young. Challenges **5**(2), 239–259 (2014). https://doi.org/10.3390/challe5020239
11. Pazis, J., Parr, R.: Generalized value functions for large action sets. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. pp. 1185–1192 (2011)
12. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021), http://jmlr.org/papers/v22/20-1364.html
13. Raycheva, R.: Erstellung eines Custom Environments in OpenAI Gym für das Spiel Ökolopoly. Tech. rep., TH Köln (2021), http://www.gm.fh-koeln.de/~konen/research/PaperPDF/PP_Oekolopoly_Raycheva_final.pdf
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). https://doi.org/10.48550/arXiv.1707.06347
15. Teixeira, J.d.S., Angeluci, A.C.B., Junior, P.P., Martin, J.G.P.: 'Let's play?' A systematic review of board games in biology. Journal of Biological Education pp. 1–20 (2022). https://doi.org/10.1080/00219266.2022.2041461
16. Vester, F.: Der blaue Planet in der Krise. Gewerkschaftliche Monatshefte **39**(12), 713–773 (1988)
17. Vester, F.: Ökolopoly: das kybernetische Umweltspiel. Studiengruppe für Biologie und Umwelt (1989)
18. Zahavy, T., Haroush, M., Merlis, N., Mankowitz, D.J., Mannor, S.: Learn what not to learn: Action elimination with deep reinforcement learning. In: Bengio, S., et al. (eds.) Advances in Neural Information Processing Systems. vol. 31 (2018)

## General discussion items

Other title suggestions: Case study for large action spaces – or – Ökolopoly: A Cybernetic Strategy Game as Reinforcement Learning Environment$^{RE}$

Since the paper is written in English for an English audience: Why don't you call it Ecolopoly/Ecopolicy and simply explain that it's based on the original German Ökolopoly?$^{ML}$

That's a good point. We discussed this in one meeting early on and decided to stick to the German original title because Ecopolicy is a commercially distributed computer game. (On the other hand we also decided to not show any GUI (as it adds no scientific value and is but a clone of a commercially available computer game) and this decision was reverted.)$^{RE}$

Just as in idea: we write multiple times how the absence of an action wrapper creates the additional challenge to discover valid actions. Maybe it would be nice to show that. $^{RE}$

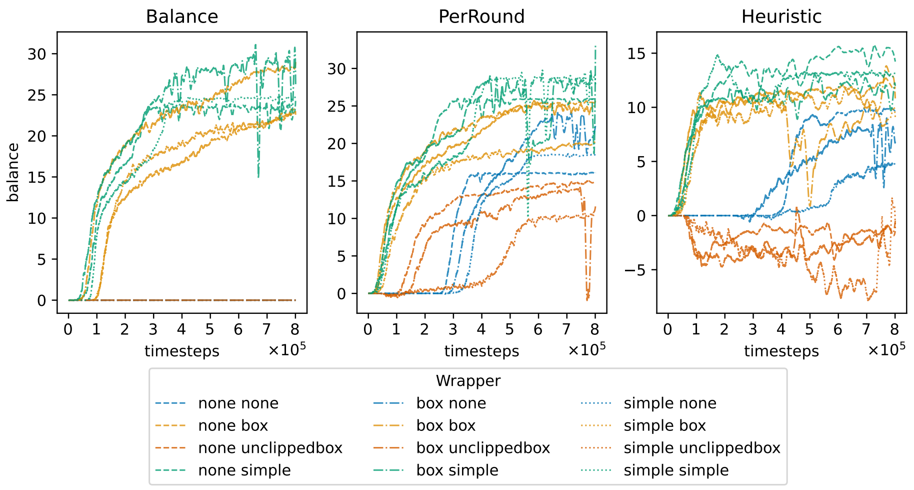


**Fig. 9.** Training curves to show delay in the absence of action wrappers that enforce valid actions. First word in legend refers to observation, second one for action wrapper Nice idea, but I think the plots are a little too complex for the average conference reader (and we probably have no space left) (and it is hard to disentangle all the curves w/o color). – But since you show them, I am just curious: Any idea why the red curves (unclipped box) are so bad in case 'Heuristic'? Even worse than 'None'? $^{WK}$ Now that I have more or less the final text there is definitely not the space to show this plot. I tried a lot of versions to encode the three family of wrappers but I fear this was the best. The colors show best how the different combination of wrappers give rise to different characteristics in the learning curve (these cluster way better than the results). Of course I could leave out unclippedbox but decided to not do that because of the interesting results. I have no explanation for the even worse performance of the unclippedbox$^{RE}$

To preserve the conversations:
4.1 Representation of the Observation Space The observation space is internally represented as a ten-dimensional object of class `MultiDiscrete` containing the values of the eight fields, the number of rounds played, and the currently available action points. The observation space is therefore fully visible to the agent, who has the same information as a human player of the board game. / The agent has therefore full access to all information visible to the human player of the board game / All information available to the human player of the board game is therefore fully visible for the agent.$^{RE}$ I would actually say that you could briefly state what this "according" translation actually is. I suspect you shift the politics points to be 0-47? I think you have the space to state that explicitly and maybe it helps someone. Otherwise, if you want to omit the implementation detail but have more than one sentence you could reason why it is important to provide the last two dimensions of the observation in addition to the eight fields (although I guess that's kind of obvious). Or you could explain, and that might actually be relevant, why you opted for MultiDiscrete rather than Box (which, as I've stated many times, would have been my intuitive choice).$^{ML}$ As different ranges are allowed in the different dimensions of the observation space (the field Politics can even contain negative values which are not supported by `MultiDiscrete`), allowed states are shifted accordingly (see first row of Tab. 2).$^{RE}$

To preserve conversation on Simple Action Wrapper:
I could imagine that restricting action point distribution to these partitions has an impact on how well the game can be played - perhaps moves that would be smart at a certain time step are not available anymore. In essence you restrict yourself to a subset of regular available moves. So there are two effects that impact your experimental performance results: One, which you actually mean to study, is that this wrapper makes the problem easier to understand for an algorithm. The other is what I described, that the achievable performance might be better/worse for e.g. random strategies or best strategies due to these restrictions. It's probably hard to disentangle these two effects based on your experimental outcomes. Still, is there anything you can say about this?$^{ML}$This is a very interesting and valid point. Actually this wrapper does three things: ensuring valid moves, reducing the number of actions, and removing the ability to play with fine-grained actions. I don't see a way to disentangle these effects now. We do something similar for the Box action wrapper, when we report results for the UnclippedBox wrapper. (Now that I think about it, the latter was meant as a quick sidenote-investigation but in the end all combinations (except TD3, which we don't show anyway) have been trained with UnclippedBox, too. Maybe it's confusing we don't explain it here...)$^{RE}$ I like the points brought up here. I would further argue that the simple action wrapper represents in itself a possible human strategy, which disengages the possibility of being too precise while choosing an action so that the exploration is made easier for the agent while he navigates in a more complex observation space.$^{RR}$ I suggest the following wording to summarize the interesting discussion brought up here:$^{WK}$ Thus, the number of available actions is largely reduced and makes the problem bet-

ter learnable because the agent has fewer actions to disentangle. On the other hand, it is also less precise, because the agent might distribute to a given field, e.g., no or at least one third of the action points. In certain situations, this can cause episodes to break off, which could have been avoided by a finer-grained distribution. The Simple Action Wrapper mimicks a possible human strategy of being less precise but actionable in unknown complex environments.