UDK 006.72 **DOI** 10.15622/ia.23.5.8

A. ISKANDAR, A. HAMMOUD, B. KOVÁCS

IMPLICIT UNDERSTANDING: DECODING SWARM BEHAVIORS IN ROBOTS THROUGH DEEP INVERSE REINFORCEMENT LEARNING

Iskandar A., Hammoud A., Kovács B. **Implicit Understanding: Decoding Swarm Behaviors in Robots through Deep Inverse Reinforcement Learning.**

Abstract. Using reinforcement learning to generate the collective behavior of swarm robots is a common approach. Yet, formulating an appropriate reward function that aligns with specific objectives remains a significant challenge, particularly as the complexity of tasks increases. In this paper, we develop a deep inverse reinforcement learning model to uncover the reward structures that guide autonomous robots in achieving tasks by demonstrations. Deep inverse reinforcement learning models are particularly well-suited for complex and dynamic environments where predefined reward functions may be difficult to specify. Our model can generate different collective behaviors according to the required objectives and effectively copes with continuous state and action spaces, ensuring a nuanced recovery of reward structures. We tested the model using E-puck robots in the Webots simulator to solve two tasks: searching for dispersed boxes and navigation to a predefined position. Receiving rewards depends on demonstrations collected by an intelligent pre-trained swarm using reinforcement learning act as an expert. The results show successful recovery of rewards in both segmented and continuous demonstrations for two behaviors – searching and navigation. By observing the learned behaviors of the swarm by the expert and proposed model, it is noticeable that the model does not merely clone the expert behavior but generates its own strategies to achieve the system's objectives.

Keywords: deep inverse reinforcement learning, reward function, demonstrations, searching behavior, navigation behavior.

1. Introduction. Swarm robotics (SR) is a self-organized system with a decentralized control architecture. Robots in SR interact with each other and their environment to achieve the desired behavior collectively. The key features of SR systems rely on autonomy and the local perception of individuals. These features lead to SR with robustness to individual failures, flexibility with changes in environmental conditions, and scalability for different sizes of swarm [1 – 4]. Methods of generating the collective behavior of the SR typically depend on the required task.The methods used to generate collective behavior in swarm robotics (SR) are closely tailored to the specific tasks the robots are designed to perform. For example, in search and rescue missions, SR systems utilize algorithms that optimize area coverage and ensure rapid localization of targets, such as finding survivors in a collapsed building. In navigation tasks, SR might employ pathfinding algorithms that enable the swarm to efficiently move through complex environments, avoiding obstacles and minimizing travel time [5 – 11]. This concept addresses the task-oriented nature of SR. It is challenging because of the unpredictable interactions

within a swarm and the diverse potential applications and environments. There is no generalized, applicable method for designing desired collective behavior, making this a key area of research. Many directions addressed this challenge, like bio-inspired algorithms: Drawing inspiration from nature, these algorithms emulate behaviors observed in biological systems, such as the flocking of birds or the foraging patterns of ants. Such models help in designing decentralized control systems where each robot in the swarm behaves according to simple rules based on local information and interactions [12 – 15], modular design: This approach focuses on creating robots with interchangeable modules, which can dynamically reconfigure based on the task at hand. Modular design enhances the flexibility and scalability of swarm robotics systems, allowing for adaptability to different environments and tasks by rearranging the modules to fit specific needs [16, 17], evolutionary robotics: This method uses evolutionary algorithms to develop the control systems for robots, effectively allowing the robots' behavior to evolve and optimize over time. It mimics natural selection processes to automatically generate solutions that are well-adapted to their environment and task, continually improving as the system encounters new scenarios [18], and machine learning, where reinforcement learning (RL) provides a robust framework for developing SR systems with diverse tasks [19]. RL supplies robots with autonomy and the ability to learn from others and the environment. It can manage the complexity of designing collective behavior by breaking down the learning process into simpler, manageable parts, with dynamic adaptation. The RL approach in SR is represented as a Markov decision process (MDP) as (S, A, R, T, γ), where the robot moves from state S to a new state S_{t+1} by executing an action A. The essential function in RL is the reward R where robots learn to perform the actions that maximize the cumulative received rewards during period T by weighting them by factor γ. Thus, the problem of designing R reflects the main objectives of the given task that correspond to generating the collective behavior of SR [20]. To generate the collective behavior of SR by RL. Firstly, define the environment in which multiple agents can coexist. Secondly, representing the states and actions spaces for policy π representation. Then define the reward function by deciding whether the agents receive rewards based on individual performance, collective outcomes, or a combination of both. Finally, choose an appropriate RL algorithm. Choosing R is critical because the reward function directly shapes the agent's behavior, guiding it towards desired objectives and away from undesirable actions. Inverse reinforcement learning (IRL) is a sophisticated approach that involves learning the underlying reward function based on the observed behavior of experts. Unlike traditional RL, which directly learns a policy based on a pre-defined reward formula, IRL provides a deeper

understanding of complex behaviors by demonstrations instead of explicitly tuning the reward formula that describes these behaviors. This methodology is particularly advantageous in swarm robotics, where explicit reward functions are challenging to formulate due to the interactions and collective dynamics of robots.

2. Related works. Defining an appropriate reward function requires mathematical knowledge and a deep understanding of the operating conditions of the system. Formulating the R equation becomes more complex by increasing the objectives of the given task. Many methods have been used to formulate R which corresponds to generating the collective behavior of SR, like sparse rewards, which are infrequent or only given sparsely throughout the training phase, where rewards or punishment are given for a specific action like when each robot near to its fellow or when the swarm reaches the target, and negatives values for colliding with obstacles [21, 22]. Shaping rewards is another method that depends on providing additional rewards to guide agents toward the desired behavior more efficiently. This involves rewarding each action through each time step during the episode, as opposed to sparse rewards where specific actions are rewarded. Both of sparse and shaping methods were used to solve the foraging SR problem. It demonstrates that the shaping method was able to solve the problem while the sparse one failed. RL was modified with a hierarchal structure to solve the problem with sparse rewards [23].

To eliminate the need for manually crafting rewards and provide more structured learning guidance than traditional methods, IRL introduces a solution to infer the reward function from demonstrations collected by an expert. So, the agent learns a policy that is similar to or better than the expert policy based on the inferred reward function. The obtained policy does not require mathematical experience or a full understanding of the conditions and operations needed to formulate R, thus avoiding human bias and potential suboptimality [24].

The main idea behind IRL is to understand what motivates these behaviors by analyzing the decisions that experts make in various situations. In IRL, the expert is typically an agent (human or robotic) who performs a task with high proficiency. The expert's behavior serves as a benchmark or model that the IRL algorithm attempts to emulate. By observing the expert, the IRL aims to deduce the reward structure that guides the expert's decisions, assuming that the expert's actions are optimized to maximize some form of cumulative reward. Developing an IRL model to be deployed in swarm robotics is an interesting research area due to its ability to facilitate autonomous decision-making in complex, dynamic environments. Many researchers have carried out IRL to generate collective behavior for many missions like [25],

where maximum entropy IRL is used for each agent to infer the birds' reward functions from observed GPS data of pigeon flocks. This approach allowed them to not only simulate flocking behavior but also infer potential leader-follower dynamics within the flock. IRL was also used in SR for area coverage problems, particularly focusing on improving efficiency in unstructured search and rescue scenarios [26]. The solution involved humans in the loop with IRL. Human expert demonstrations are used to train SR, allowing them to learn optimal area coverage strategies. The author in [27] combines IRL with automatic modular design to generate control software for robot swarms based on only the demonstrations, without the need for explicitly defined reward and objective functions. Many methodologies in cited IRL can not handle high-dimensional, continuous state-action spaces and are capable of generalizing across different tasks and dynamic environments. This is crucial for developing adaptive and robust swarm robotic systems that can operate effectively in a wide range of scenarios. Our paper investigates the ability of IRL to generalize across different scenarios and automate reward design, making it robust and efficient, particularly in complex and continuous environments. It introduces an IRL model able to deal with continuous state and action spaces with simplified segmented or continuous demonstrations. This model can be generalized to produce different collective behaviors such as navigation and searching tasks.

3. System Description. This section describes the framework of the SR by testing two tasks: searching for the boxes represented as light sources and navigating from initial positions to a predefined position illustrated as a circular yellow area called (P), as shown in Figure 1.

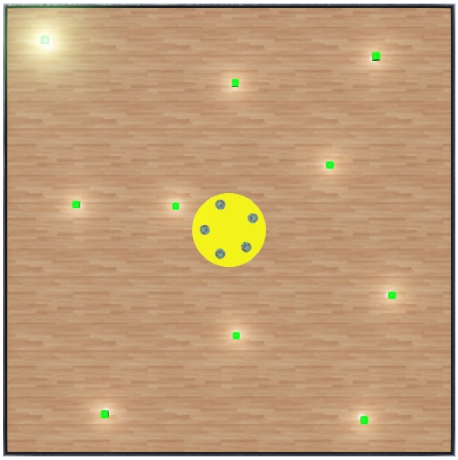

Fig. 1. SR environment

The swarm system was implemented in a 3D robot simulator called Webots where the E-Puck mobile robot was selected to build the swarm. The dimensions of the workspace were defined as $3 \times 3m^2$, forming a square area surrounded by four walls. The parameters of E-Puck robots were set as follows: linear velocity $[0, 0.25]m/s$, angular velocity $[-3.14, 3.14]rad/s$, and light sensors' readings corresponded to the light intensity $[0, 4095]$.

3.1. RL architecture. The proximal policy optimization algorithm (PPO) is used for both searching and navigation tasks. Most of the studies mentioned in the related work section used PPO. It is favored in robotics for its balance between sample efficiency and computational simplicity, avoiding the need for complex calculations like those in Trust Region Policy Optimization (TRPO). PPO's stability is enhanced via a policy gradient method that maximizes an objective function by using a clipped surrogate objective to keep updates stable [28], thus maintaining steady training progress. This makes PPO an adaptable and robust choice for a variety of applications, particularly those involving continuous action spaces and environments with complex dynamics.

The problem is formulated as a MDP represented by the tuple (S, A, T, R, γ). The state space S has two frames, one for the searching task contains light sensor readings, and the other frame for the navigation task includes the distance D, besides the angle θ between the robot and P. The action space A includes the velocities of both the left and right motors. The transition function T describes the dynamics of the system. In continuous states and actions spaces, the transition dynamics function typically cannot be explicitly defined for every possible state and action due to the infinite possibilities. PPO optimizes a policy function that outputs a probability distribution over actions given the current state. The policy is typically parameterized by a neural network where the weights are adjusted to maximize the cumulative reward. The reward function R provides feedback based on the system's behavior. The main architecture of the PPO has two neural networks, actor and critic with fully connected layers, as shown in Figure 2. Table 1 presents the parameters of PPO.

3.1.1. Reward formulating. In the searching task, As we mentioned the PPO receives the light sensor readings as inputs and produces wheel velocities as outputs, Figure 2. The learned velocities attempt to adjust the robot's trajectory toward the light source (boxes). To formulate the equation of the reward function to motivate robots to steer their directions to the light sources where the boxes are located, it is better to measure the intensity of the light between two states at times $t-1$ and t and be rewarded if it is increased.

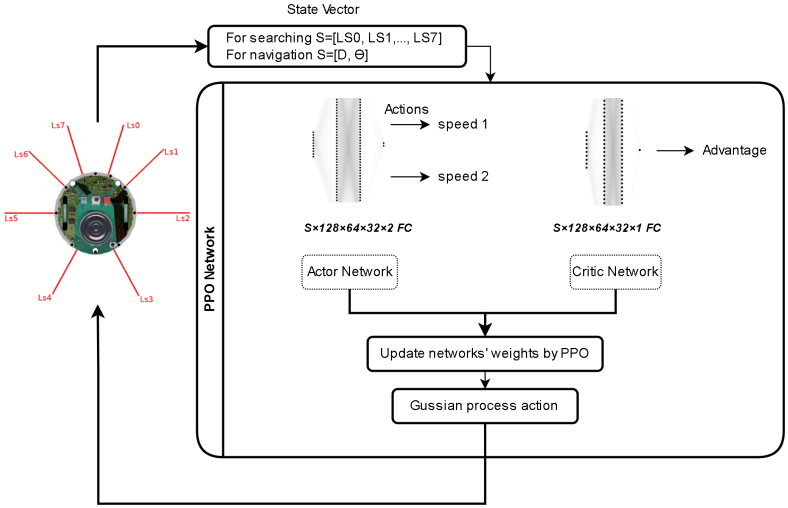

Fig. 2. PPO architecture

Table 1. PPO hyperparameters

Parameter	**Value**
max training timesteps	RL:1000000, IRL-RL:250000
max timesteps per episode	800
state space dimension	Searching:8, Navigation:2
action space dimension	2
discount factor (γ)	0.99
PPO epsilon clip	0.2
PPO K epochs	80
optimizer learning rate actor	0.0003
optimizer learning rate critic	0.001
Layers size	input,128,64,32,output

An additional value 1.1 is given when the robot finds the box, as in Equations 1 and 2. Notably, both shaping and sparse methods were used.

$$r_{\text{box}} = \begin{cases} 1.1 & \text{if } LS_0^{(t)} > \text{FindThreshold}_{\text{searching}} \\ 1.1 & \text{if } LS_7^{(t)} > \text{FindThreshold}_{\text{searching}} \\ 0 & \text{otherwise} \end{cases}, \quad (1)$$

$$R(t)_{searching} = \frac{(LS_0^{(t-1)} - LS_0^{(t)}) + (LS_7^{(t-1)} - LS_7^{(t)})}{2} + r_{\text{box}}, \quad (2)$$

where:
$R(t)_{searching}$ – Reward value at each time step t.
$LS_0^{(t)}, LS_7^{(t)}$ – The current readings of light sensors 0 and 7, respectively, at time t.
$LS_0^{(t-1)}, LS_7^{(t-1)}$ – The previous readings of light sensors 0 and 7, respectively, at time $t-1$.
$\text{FindThreshold}_{\text{searching}}$ – The threshold value for the light sensor where the box is found.
r_{box} – The additional reward when the robot finds the box.

For the navigation task, the inputs of the PPO network are the robot's current distance and angle relative to P, where the outputs modify the wheel velocities to navigate P. Additional reward is sparse for successfully reaching P, as in Equation 3, incorporating the shaping method to speed up the learning process, as in Equation 4.

$$r_{\text{P}} = \begin{cases} 0.1 & \text{if } D_{\text{t}} < \text{FindThreshold}_{\text{navigation}} \\ 0 & \text{otherwise} \end{cases}, \quad (3)$$

$$R(t)_{navigation} = (D_{\text{t-1}} - D_{\text{t}}) + r_{\text{P}} + \frac{\cos(\theta_{\text{t}})}{1000}, \quad (4)$$

r_{P} – The additional reward when the robot reaches P.
$\text{FindThreshold}_{\text{navigation}}$ – The robot is inside P.
D_{t} – The distance between the robot and P at time t.
$D_{\text{t-1}}$ – The distance between the robot and P at time t-1.
θ_{t} – The angle between the robot and P.

3.2. IRL-RL model. In this approach, instead of formulating the reward mathematically train RL to find the policy. IRL is implemented to infer the reward by demonstrations collected via a pre-trained swarm. RL used them to generate the policy to obtain the searching and navigation behaviors, as in Figure 3.

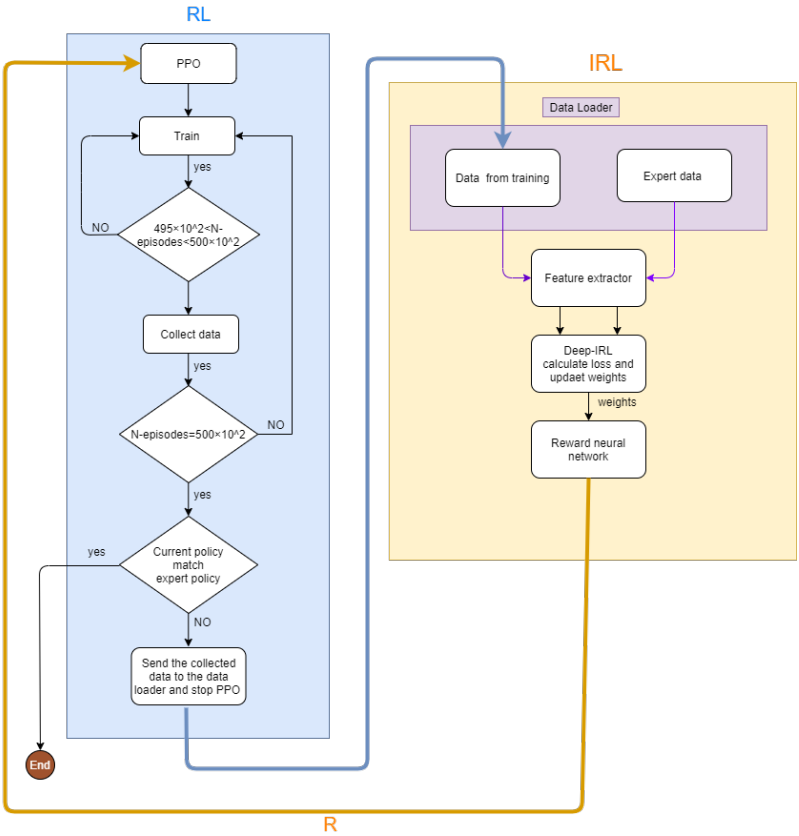

Fig. 3. IRL-RL model

IRL components:

Data loader: it is a container for the data that flows from the expert and training process. Expert data is collected by an expert pre-trained model while training data is collected during the PPO training process. Both of them contain only state frames with flags without actions. The flags are indicators of completing the task like finding a box in the searching task or reaching P in the navigation. The model is able to cope with two types segmented and continuous states. Both types of states were tested in the results section. In segmented mode, the sensors' readings after normalization are divided into five ranges between 0 and 1, each corresponding to a value.

Features extractor: the given Table 2 shows the types of data and functions that are used on the received data from the data loader. The received data is raw, where the values of the light sensors are measured in a range of $[0,4095]$. While the distance D is in the range of $[0,3]m$ and the angle θ is $[-\pi,\pi]rad$, the purpose of $\phi(s)$ in Equations 5, and 6 is to convert raw states S into a feature vector which is more suitable as input for the model. The shift function is applied to the states after normalization to obtain the values in $t-1$. These values are used to produce the correlation between states to encourage the R network to perceive the directions of changes in states.

Table 2. Features Extractor Input and Output for Searching and Navigation Tasks

Task	**Input of features extractor (from the data loader)**	**Output of features extractor**
Searching	$LS_0^{(t)}$, $LS_7^{(t)}$, flag (Finding a box)	Normalized [$LS_0^{(t-1)}$, $LS_0^{(t)}$, $LS_7^{(t-1)}$, $LS_7^{(t)}$], flag (Finding a box)
Navigation	$D^{(t)}$, $\theta^{(t)}$, flag (Reaching P)	Normalized [$D^{(t-1)}$, $D^{(t)}$, $\theta^{(t-1)}$, $\theta^{(t)}$], flag (Reaching P)

$$\phi(s): S \to [0,1], \tag{5}$$

$$\phi(s) = \frac{\mathrm{Max}_{\mathrm{Output}} - \mathrm{Min}_{\mathrm{Output}}}{\mathrm{Max}_{\mathrm{Value}} - \mathrm{Min}_{\mathrm{Value}}} \cdot (s - \mathrm{Max}_{\mathrm{Value}}) + \mathrm{Max}_{\mathrm{Output}}, \tag{6}$$

$\mathrm{Max}_{\mathrm{Value}}$ – the upper value in the raw range of states, for example, in the searching task equals 4095 according to the light sensor reading

$\mathrm{Min}_{\mathrm{Value}}$ – the lower value in the raw range of states, for example, in the searching task equals 0.

$\mathrm{Max}_{\mathrm{Output}}$ – the upper value in the output range of $\phi(s)$, equals 1 as in Equation 5.

$\mathrm{Min}_{\mathrm{Output}}$ – the lower value in the output range of $\phi(s)$, equals 0 as in Equation 5.

Deep IRL – the backpropagation process of the reward network is performed by calculating the losses according to Equation 7, which guarantees updating the weights of the reward neural network. This objective function is the binary cross-entropy loss function applied for distinguishing between expert and training rewards. This loss function is designed to penalize the deviation of the predicted rewards from the "true" rewards indicated by the expert's behavior.

$$\text{loss} = -\log(\text{sigmoid}(\text{R}_{\text{expert}})) - (1 - \log(\text{sigmoid}(\text{R}_{\text{training}}))), \qquad (7)$$

R_{expert} – the output of the reward neural network for states from the expert.
$\text{R}_{\text{training}}$ – the output of the reward neural network for collected states from the training process.

Reward network – The purpose of the reward neural network is to approximate the reward function. This is done by passing the feature vector through the neural network. Then, producing a scalar reward value as an output. It is constructed as fully connected layers of $length(feature - vectors) \times 15 \times 1FC$, where the length of feature vectors in the proposed tasks is 5, as in Table 2.

4. Results and discussion. We have focused on examining the reward that affects swarm behavior in a simulated environment, demonstrating how IRL can approximate the reward function without the need for mathematical formalities. Our implementation involved two tasks: searching for boxes in continuous RL and segmented features, and navigation task to a pre-defined position known as P also in continuous RL but in continuous features. The swarm's performance was evaluated by comparing the rewards between the IRL-based model and an expert-pre-trained RL model, demonstrating the ability to generate behavior to achieve the required tasks. Finally, we analyze the generated behavior of the swarm under both models. Choosing the features plays a major role in recovering a correct reward function. They differ based on the defined problem and objective function of the swarm system as in Table 2. This table illustrates the differences in the chosen features for searching and navigation behavior. In addition, consider the readings in the time step t and $t-1$ to make the R network recognize the difference in the light intensity for the searching task or change in the distance in the navigation task. Using a deep neural network to represent R with binary cross-entropy loss function makes the model able to handle continuous environments. So, in our models, we recovered reward in the continuous and segmented mode of the IRL model to generate the policy in RL with continuous states and action spaces.

4.1. Searching task. In this task, the reward structure is related to the change in the intensity of light detected by sensors. The reward increases as the robots move towards stronger light intensity, collecting a higher reward once the boxes are located.

The training process required three rounds to recover the reward which generates a successful behavior as follows.

In the first round, the reward neural network was initially configured with arbitrary weights, denoted as ω_0. So, RL learned a stochastic policy π_0. Based on this policy, data was collected and forwarded to the data loader. By harnessing both expert and collected data, the IRL part conducted training on the reward network, resulting in an update to the weights, yielding ω_1. Subsequently, the RL part in Figure 2 trained again to generate its policy π_1 by the new weights of reward. The iterative process finished with the weights ω_2, so the robots trained to learn policy $\pi 2$ achieved the required task as shown in Figure 4. The reward function in the expert-RL model, indicated by a red line, maximizes the reward as the robot approaches the light source (box) and maintains peak values upon reaching it. In contrast, the inferred reward by the IRL-RL model, represented by a blue line, captures the increasing light intensity in a segmented fashion, echoing the RL model's behavior but with discrete transitions due to the segmented mode of features. Thus,the data is segmented into specific ranges, such as states from 0 to 0.2 representing darker areas rewarded uniformly.

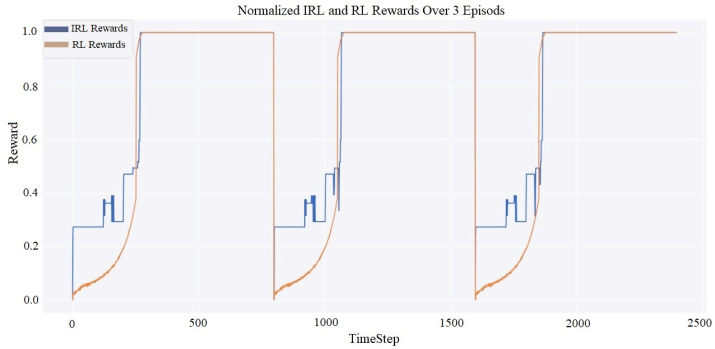

Fig. 4. Normalized IRL and RL rewards over 3 episodes for $\omega 2$-searching task

As shown in Figure 5, the IRL-RL model effectively reconstructs the reward function, when sensor readings LS0 and LS7 in the 0 to 0.2 range result in minimal rewards, increasing as the robot transitions to the 0.2-0.4 range – the reward spikes in the 0.8-1 range, indicating the robot's proximity to the box. The comparative visualization of the reward functions from the expert RL model (pre-trained) and the IRL-RL model highlights differences, with the darker blue associated with the RL model. Nonetheless, a crucial observation is the shared gradient pattern between the models, suggesting a direct correlation of rewards with incremental light intensity, affirming that the robots have learned to search and find boxes, thereby validating the designed

behavior. The IRL-RL model's ability to mimic the decision-making strategy of the pre-trained robot.

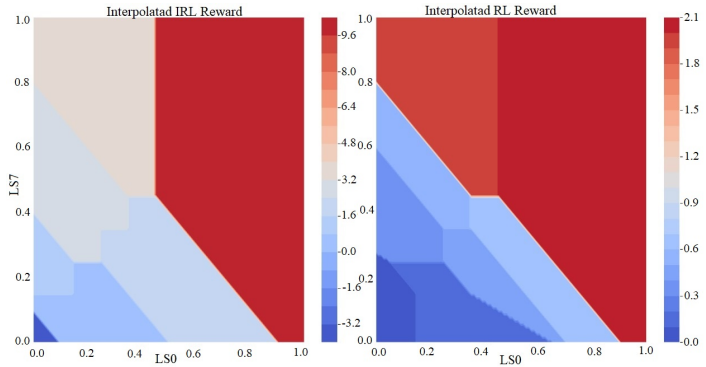

Fig. 5. Heat map of the true reward (right) and the recovered reward (left) for the searching task

For the robot's behavior, the bar chart in Figure 6 highlights a successful behavior of the IRL-RL model compared to pre-trained expert RL across ten incremental collected boxes, The IRL-RL model generates a behavior that enables the robots to collect the boxes in a round-trip manner. However, it is notable that there are differences in the collecting box's times which reflect different behaviors. This means the IRL-RL model does not clone the behavior or actions of RL instead, it learns how to achieve the task with its own generated behavior. This suggests that IRL successfully learns the complex reward structures governing task completion rather than merely copying actions.

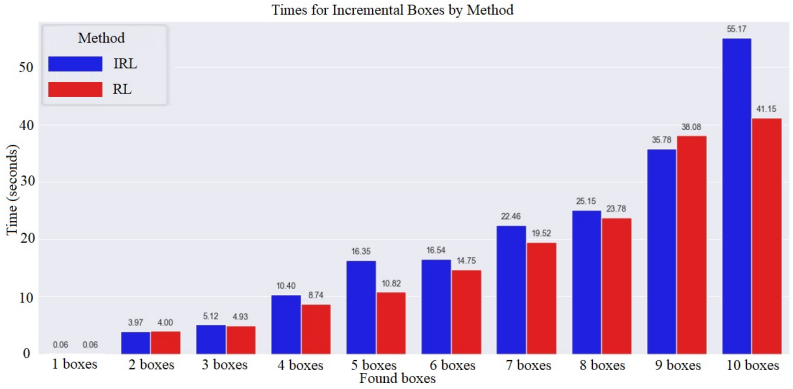

Fig. 6. Swarm searching behavior by IRL and RL

4.2. Navigation task. The same process with the same number of rounds was applied to the navigation task. Initially, the reward neural network was configured with random weights, labeled as $\omega 0$, leading to the production of a stochastic policy $\pi 0$ from which the RL model began its learning process. After the first round, the collected data was used to train the reward network, updating the weights to $\omega 1$, which in turn allowed the RL model to refine its policy to $\pi 1$. The process concluded after a second round of adjustments, resulting in final weights $\omega 2$, enabling the model to successfully execute the required navigation tasks as depicted as in Figure 7.

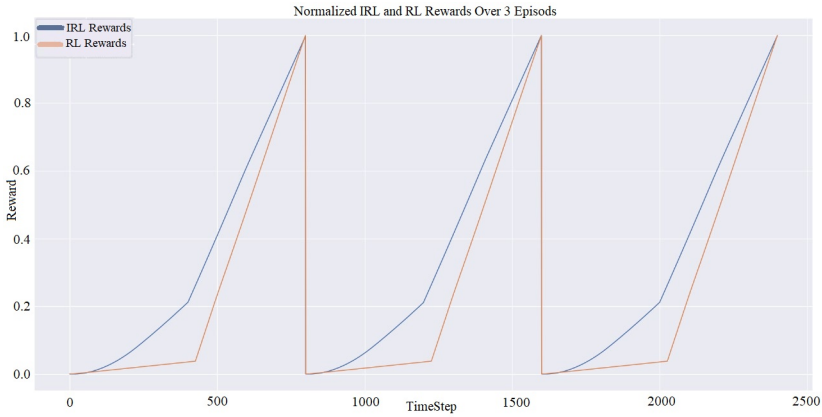

Fig. 7. Normalized IRL and RL rewards over 3 episodes for $\omega 2$- Navigation task

Figure 8 illustrates both the interpolated IRL reward and interpolated RL reward models under these conditions. It shows a gradient of colors from red to blue, indicating varying reward intensities based on the robot's angle and distance to the target.

High rewards are shown in red, corresponding to smaller angles and distances – indicative of the robot directly facing and being close to the target. As the angle increases or the distance increases, the reward diminishes, as shown by the gradient transitioning to blue. Unlike the search, there is no segmentation into discrete state ranges. The continuous nature of the data allows for a smoother gradient in the visual representation and a more nuanced adjustment of the reward based on the robot's proximity and alignment with the target.

The similarities between the paths in both graphs in Figure 9 indicate that the IRL has effectively learned from the RL data, closely replicating the expert RL's behavior. This suggests the successful application of IRL where

the algorithm has inferred the strategies and decisions that the RL considered optimal.

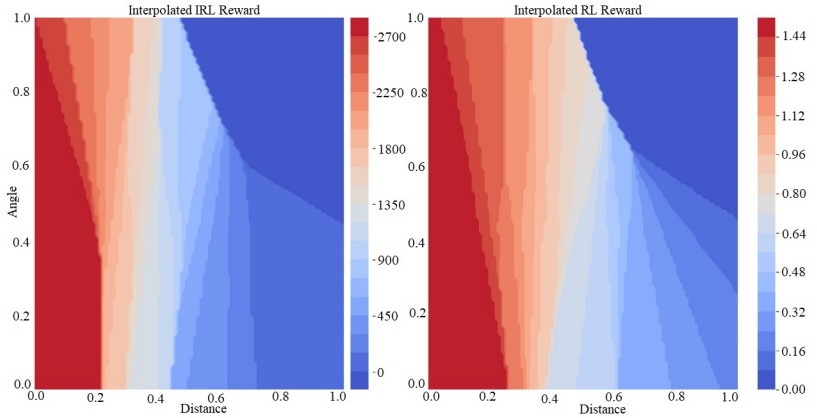

Fig. 8. Heat map of the true reward (right) and the recovered reward (left) for the Navigation task

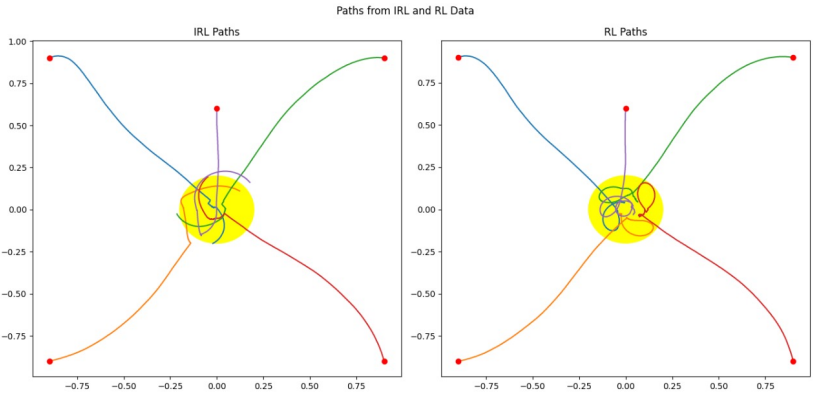

Fig. 9. Robots navigation paths

5. Conclusion. This paper presented an advanced model employing IRL to effectively recover the reward function by demonstrations of expert behaviors. Instead of directly learning the behavior, IRL seeks to understand the reasons behind specific actions or strategies by inferring the reward to solve the task by generating its own behavior. It eliminates the need for extensive manual tuning of reward functions and facilitates a more intuitive

setup via demonstration-based learning. The proposed IRL-RL model has the ability to handle continuous state spaces and dynamic environments to deal with continuous RL problems due to employing a deep neural network for representing R, in addition, to recovering reward function based on two types of data that flow from data loader: segmented features and continous features for naunce stratgies. This model was tested across two tasks, navigating towards a predefined position and searching for specific objects within a simulated swarm robot environment. It demonstrated its robust capability to infer and adapt the reward structures essential for guiding autonomous robotic swarms to accomplish tasks. Moreover, our findings highlight the potential of the proposed model to generalize across different scenarios. For future directions, this model will be developed to achieve composed and more complex tasks like generating foraging collective behaviors and aggregation behavior.

References

1. Shahzad M., Saeed Z., Akhtar A., Munawar H., Yousaf M., Baloach N., Hussain F. A review of swarm robotics in a nutshell. Drones. 2023. vol. 7. no. 4.
2. Schranz M., Umlauft M., Sende M., Elmenreich W. Swarm robotic behaviors and current applications. Frontiers in Robotics and AI. 2020. vol. 7.
3. Cheraghi A., Shahzad S., Graffi K. Past, present, and future of swarm robotics. In Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys). 2022. vol. 3. pp. 190–233.
4. Brambilla M., Ferrante E., Birattari M., Dorigo M. Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence. 2013. vol. 7. pp. 1–41.
5. Nauta J., Van Havermaet S., Simoens P., Khaluf Y. Enhanced foraging in robot swarms using collective lévy walks. In 24th European Conference on Artificial Intelligence (ECAI). 2020. pp. 171–178.
6. Misir O., Gokrem L. Flocking-based self-organized aggregation behavior method for swarm robotics. Iranian Journal of Science and Technology, Transactions of Electrical Engineering. 2021. vol. 45. no. 4. pp. 1427–1444.
7. Sadeghi A., Raoufi M., Turgut A. A self-adaptive landmark-based aggregation method for robot swarms. Adaptive Behavior. 2022. vol. 30. no. 3. pp. 223–236.
8. Lu Q., Hecker J., Moses M. The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016. pp. 3815–3821.
9. Lopes Y., Trenkwalder S., Leal A., Dodd T., Groß R. Supervisory control theory applied to swarm robotics. Swarm Intelligence. 2016. vol. 10. pp. 65–97.
10. Hamann H. Swarm robotics: A formal approach. Cham: Springer, 2018. 210 p.
11. Berlinger F., Gauci M., Nagpal R. Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm. Science Robotics. 2021. vol. 6. no. 50. DOI: 10.1126/scirobotics.abd8668.
12. Zhang J., Lu Y., Che L., Zhou M. Moving-distance-minimized PSO for mobile robot swarm. IEEE Transactions on Cybernetics. 2021. vol. 52. no. 9. pp. 9871–9881.
13. Parhi D., Sahu C., Kumar P. Navigation of multiple humanoid robots using hybrid adaptive swarm-adaptive ant colony optimisation technique. Computer Animation and Virtual Worlds. 2018. vol. 29. no. 2. DOI: 10.1002/cav.1802.

14. Jiang L., Mo H., Tian P. An adaptive decentralized control strategy for deployment and aggregation of swarm robots based on bacterial chemotaxis. Applied Intelligence. 2023. vol. 53. no. 10. pp. 13018–13036.
15. Hu C., Arvin F., Bellotto N., Yue S., Li H. Swarm neuro-robots with the bio-inspired environmental perception. Frontiers in Neurorobotics. 2024. vol. 18.
16. Hasselmann K., Ligot A., Birattari M. Automatic modular design of robot swarms based on repertoires of behaviors generated via novelty search. Swarm and Evolutionary Computation. 2023. vol. 83.
17. Birattari M., Ligot A., Francesca G. AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms. Automated Design of Machine Learning and Search Algorithms. 2021. pp. 73–90.
18. Stolfi D., Danoy G. Evolutionary swarm formation: From simulations to real world robots. Engineering Applications of Artificial Intelligence. 2024. vol. 128. DOI: 10.1016/j.engappai.2023.107501.
19. Blais M., Akhloufi M. Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators. Cognitive Robotics. 2023. vol. 3. pp. 226–256. DOI: 10.1016/j.cogr.2023.07.004.
20. Sutton R., Barto A. Reinforcement learning: An introduction. The MIT press, 2018. 552 p.
21. Iskandar A., Rostum H., Kovacs B. Using Deep Reinforcement Learning to Solve a Navigation Problem for a Swarm Robotics System. Proceedings of the 24th International Carpathian Control Conference (ICCC). IEEE, 2023. pp. 185–189.
22. Wei Y., Nie X., Hiraga M., Ohkura K., Car Z. Developing end to end control policies for robotic swarms using deep Q-learning. Journal of Advanced Computational Intelligence and Intelligent Informatics. 2019. vol. 23. no. 5. pp. 920–927.
23. Jin B., Liang Y., Han Z., Hiraga M., Ohkura K. A hierarchical training method of generating collective foraging behavior for a robotic swarm. Artificial Life and Robotics. 2022. vol. 27. pp. 137–141.
24. Arora S., Doshi P. A survey of inverse reinforcement learning: Challenges, methods and progress. Artificial Intelligence. 2021. vol. 297. DOI: 10.1016/j.artint.2021.103500.
25. Pinsler R., Maag M., Arenz O., Neumann G. Inverse reinforcement learning of bird flocking behavior. ICRA Swarms Workshop. 2018.
26. Chen M., Zhang P. Area Coverage for Swarm Robots Via Inverse Reinforcement Learning. 2023. 9 p.
27. Gharbi I., Kuckling J., Ramos D., Birattari M. Show me what you want: Inverse reinforcement learning to automatically design robot swarms by demonstration. In 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023. pp. 5063–5070.
28. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017.

Alaa Iskandar — Ph.D. student, Faculty of mechanical engineering and informatics (istvan salyi doctoral school of mechanical engineering sciences – mathematic institute), University of Miskolc. Research interests: reinforcement learning for swarm robotics, Navigation, and foraging behaviors. The number of publications — 3. iskandar.alaa@student.uni-miskolc.hu; Egyetemvaros, 3515, Miskolc city, Hungary; office phone: +(36)46-565-111.

Ali Hammoud — Ph.D. student, Faculty of applied informatics, Federal State Budgetary Educational Institution of Higher Education “Kuban State Agrarian University named after I.T. Trubilin”. Research interests: multi-agent systems and decision-making. The number of publications — 2. ali-hammoud@mail.ru; 13, Kalinina St., 350044, Krasnodar, Russia; office phone: +7(861)221-5942.

Béla Kovács — Ph.D., Dr.Sci., Associate professor, Faculty of mechanical engineering and information, institute of mathematics department of analysis, University of Miskolc. Research interests: mechanical engineering and differential equations. The number of publications — 99. matmn@uni-miskolc.hu; Egyetemvaros, 3515, Miskolc city, Hungary; office phone: +(36)46-565-111.

DOI 10.15622/ia.23.5.8

А. Искандар, А. Хаммуд, Б. Ковач

СКРЫТЫЙ СМЫСЛ: ДЕКОДИРОВКА РОЕВОГО ПОВЕДЕНИЯ РОБОТОВ С ПОМОЩЬЮ ГЛУБОКОГО ОБРАТНОГО ОБУЧЕНИЯ С ПОДКРЕПЛЕНИЕМ

Искандар А., Хаммуд А., Ковач Б. **Скрытый смысл: декодировка роевого поведения роботов с помощью глубокого обратного обучения с подкреплением.**

Аннотация. Использование обучения с подкреплением для создания коллективного поведения роевых роботов является распространенным подходом. Тем не менее, формулирование соответствующей функции вознаграждения, которая соответствовала бы конкретным целям, остается серьезной проблемой, особенно по мере увеличения сложности задач. В этой статье мы разрабатываем модель глубокого обратного обучения с подкреплением, чтобы раскрыть структуры вознаграждения, которые помогают автономным роботам выполнять задачи посредством демонстраций. Модели глубокого обратного обучения с подкреплением особенно хорошо подходят для сложных и динамичных сред, где может быть сложно указать заранее определенные функции вознаграждения. Наша модель может генерировать различное коллективное поведение в соответствии с требуемыми целями и эффективно справляется с непрерывными пространствами состояний и действий, обеспечивая детальное восстановление структур вознаграждения. Мы протестировали модель с помощью роботов E-puck в симуляторе Webots для решения двух задач: поиска рассредоточенных коробок и навигации к заданной позиции. Получение вознаграждения зависит от демонстраций, собранных интеллектуальным предварительно обученным роем, использующим обучение с подкреплением в качестве эксперта. Результаты показывают успешное получение вознаграждения как в сегментированной, так и в непрерывной демонстрации двух типов поведения — поиска и навигации. Наблюдая за изученным поведением роя экспертом и предложенной моделью, можно заметить, что модель не просто клонирует поведение эксперта, но генерирует свои собственные стратегии для достижения целей системы.

Ключевые слова: обратное обучение с подкреплением, функция вознаграждения, демонстрации, поисковое поведение, навигационное поведение.

Литература

1. Shahzad M., Saeed Z., Akhtar A., Munawar H., Yousaf M., Baloach N., Hussain F. A review of swarm robotics in a nutshell. Drones. 2023. vol. 7. no. 4.
2. Schranz M., Umlauft M., Sende M., Elmenreich W. Swarm robotic behaviors and current applications. Frontiers in Robotics and AI. 2020. vol. 7.
3. Cheraghi A., Shahzad S., Graffi K. Past, present, and future of swarm robotics. In Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys). 2022. vol. 3. pp. 190–233.
4. Brambilla M., Ferrante E., Birattari M., Dorigo M. Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence. 2013. vol. 7. pp. 1–41.
5. Nauta J., Van Havermaet S., Simoens P., Khaluf Y. Enhanced foraging in robot swarms using collective lévy walks. In 24th European Conference on Artificial Intelligence (ECAI). 2020. pp. 171–178.

6. Misir O., Gokrem L. Flocking-based self-organized aggregation behavior method for swarm robotics. Iranian Journal of Science and Technology, Transactions of Electrical Engineering. 2021. vol. 45. no. 4. pp. 1427–1444.
7. Sadeghi A., Raoufi M., Turgut A. A self-adaptive landmark-based aggregation method for robot swarms. Adaptive Behavior. 2022. vol. 30. no. 3. pp. 223–236.
8. Lu Q., Hecker J., Moses M. The MPFA: A multiple-place foraging algorithm for biologically-inspired robot swarms. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016. pp. 3815–3821.
9. Lopes Y., Trenkwalder S., Leal A., Dodd T., Groß R. Supervisory control theory applied to swarm robotics. Swarm Intelligence. 2016. vol. 10. pp. 65–97.
10. Hamann H. Swarm robotics: A formal approach. Cham: Springer, 2018. 210 p.
11. Berlinger F., Gauci M., Nagpal R. Implicit coordination for 3D underwater collective behaviors in a fish-inspired robot swarm. Science Robotics. 2021. vol. 6. no. 50. DOI: 10.1126/scirobotics.abd8668.
12. Zhang J., Lu Y., Che L., Zhou M. Moving-distance-minimized PSO for mobile robot swarm. IEEE Transactions on Cybernetics. 2021. vol. 52. no. 9. pp. 9871–9881.
13. Parhi D., Sahu C., Kumar P. Navigation of multiple humanoid robots using hybrid adaptive swarm-adaptive ant colony optimisation technique. Computer Animation and Virtual Worlds. 2018. vol. 29. no. 2. DOI: 10.1002/cav.1802.
14. Jiang L., Mo H., Tian P. An adaptive decentralized control strategy for deployment and aggregation of swarm robots based on bacterial chemotaxis. Applied Intelligence. 2023. vol. 53. no. 10. pp. 13018–13036.
15. Hu C., Arvin F., Bellotto N., Yue S., Li H. Swarm neuro-robots with the bio-inspired environmental perception. Frontiers in Neurorobotics. 2024. vol. 18.
16. Hasselmann K., Ligot A., Birattari M. Automatic modular design of robot swarms based on repertoires of behaviors generated via novelty search. Swarm and Evolutionary Computation. 2023. vol. 83.
17. Birattari M., Ligot A., Francesca G. AutoMoDe: a modular approach to the automatic off-line design and fine-tuning of control software for robot swarms. Automated Design of Machine Learning and Search Algorithms. 2021. pp. 73–90.
18. Stolfi D., Danoy G. Evolutionary swarm formation: From simulations to real world robots. Engineering Applications of Artificial Intelligence. 2024. vol. 128. DOI: 10.1016/j.engappai.2023.107501.
19. Blais M., Akhloufi M. Reinforcement learning for swarm robotics: An overview of applications, algorithms and simulators. Cognitive Robotics. 2023. vol. 3. pp. 226–256. DOI: 10.1016/j.cogr.2023.07.004.
20. Sutton R., Barto A. Reinforcement learning: An introduction. The MIT press, 2018. 552 p.
21. Iskandar A., Rostum H., Kovacs B. Using Deep Reinforcement Learning to Solve a Navigation Problem for a Swarm Robotics System. Proceedings of the 24th International Carpathian Control Conference (ICCC). IEEE, 2023. pp. 185–189.
22. Wei Y., Nie X., Hiraga M., Ohkura K., Car Z. Developing end to end control policies for robotic swarms using deep Q-learning. Journal of Advanced Computational Intelligence and Intelligent Informatics. 2019. vol. 23. no. 5. pp. 920–927.
23. Jin B., Liang Y., Han Z., Hiraga M., Ohkura K. A hierarchical training method of generating collective foraging behavior for a robotic swarm. Artificial Life and Robotics. 2022. vol. 27. pp. 137–141.
24. Arora S., Doshi P. A survey of inverse reinforcement learning: Challenges, methods and progress. Artificial Intelligence. 2021. vol. 297. DOI: 10.1016/j.artint.2021.103500.

25. Pinsler R., Maag M., Arenz O., Neumann G. Inverse reinforcement learning of bird flocking behavior. ICRA Swarms Workshop. 2018.
26. Chen M., Zhang P. Area Coverage for Swarm Robots Via Inverse Reinforcement Learning. 2023. 9 p.
27. Gharbi I., Kuckling J., Ramos D., Birattari M. Show me what you want: Inverse reinforcement learning to automatically design robot swarms by demonstration. In 2023 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2023. pp. 5063–5070.
28. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347. 2017.

Искандар Алаа — аспирант, факультет машиностроения и информатики (докторантура машиностроительных наук иштвана сали – математический институт), Университет Мишкольца. Область научных интересов: обучение с подкреплением для роевой робототехники, навигации и поиска пищи. Число научных публикаций — 3. iskandar.alaa@student.uni-miskolc.hu; Эгьетемварош, 3515, Мишкольц, Венгрия; р.т.: +(36)46-565-111.

Хаммуд Али — аспирант, факультет прикладной информатики, Федеральное государственное бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина». Область научных интересов: мультиагентные системы и принятие решений. Число научных публикаций — 2. ali-hammoud@mail.ru; улица Калинина, 13, 350044, Краснодар, Россия; р.т.: +7(861)221-5942.

Ковач Бела — Ph.D., Dr.Sci., доцент, факультет машиностроения и информатики института математики, кафедра анализа, Университет Мишкольца. Область научных интересов: машиностроение и дифференциальные уравнения. Число научных публикаций — 99. matmn@uni-miskolc.hu; Эгьетемварош, 3515, Мишкольц, Венгрия; р.т.: +(36)46-565-111.