UNIVERSITY OF MISKOLC
FACULTY OF MECHANICAL ENGINEERING AND INFORMATICS

DEEP REINFORCEMENT LEARNING FOR SWARM ROBOTIC SYSTEMS

Booklet of PhD Theses

PREPARED BY:

ALAA ISKANDAR

ENGINEERING OF MECHATRONICS (BSC),

ENGINEERING OF MECHATRONICS (MSC)

ISTVÁN SÁLYI DOCTORAL SCHOOL OF MECHANICAL ENGINEERING SCIENCES
TOPIC FIELD OF BASIC ENGINEERING SCIENCES
TOPIC GROUP OF MECHANICS OF SOLIDS

HEAD OF DOCTORAL SCHOOL
DR. GABRIELLA BOGNÁR
DSC, FULL PROFESSOR

HEAD OF TOPIC GROUP
DR. PÁCZELT ISTVÁN

SCIENTIFIC SUPERVISOR
DR. BÉLA KOVÁCS

Miskolc
2025

JUDGING COMMITTEE

chair:

secretary:

members:

OFFICIAL REVIEWERS

1. INTRODUCTION

Swarm robotics (SR) is the field that focuses on the study and development of multi-robot systems which includes a group of relatively simple and often homogeneous robots. They work together to accomplish tasks. The SR concept was inspired by social insects like ants, bees, and birds, which exhibit remarkable collective behaviors through decentralized and self-organized interactions. Control in swarm robotics is typically distributed among the individual ones. Each robot follows local rules and communicates with nearby peers to achieve group objectives. There is typically no central controller dictating the actions of the entire group. Instead, robots interact with each other and with their environment locally, making decisions based on local information [1]. Collective behavior in SR refers to the interactions of multiple autonomous robots working together as a group to achieve common objectives [2]. Figure 1 shows different types of collective behavior but not limited to aggregation, navigation, and dispersion. Robots in aggregation gather or cluster in a specific location or form a pattern where robots arrange themselves into predefined shapes. Navigation behavior refers to the coordinated movement of a group of autonomous robots to reach specific targets. In other words, they navigate through an environment while adhering to specific objectives such as exploration and avoiding collisions as constraints. This collective behavior is essential in various applications, including exploration, search and rescue missions, and environmental monitoring [3]. In the pursuit of generating collective behavior in swarm robotics, researchers have explored various design methodologies and control strategies. Two prominent approaches that have emerged are behavior-based design methods and automatic design methods [4]. Behavior-based design methods emphasize the modularization of robot control into distinct behaviors or modules. Each module is responsible for a specific aspect of robot behavior. Thus, the collective behavior emerges from the interactions and coordination among these modules. They are often conceptually simpler and more interpretable than automatic design approaches. However, coordinating and tuning multiple behavior modules to achieve desired collective behaviors can be challenging, especially for complex tasks. Automatic design methods involve using optimization algorithms or machine learning techniques to search for and optimize control parameters or policies that govern the behavior of individual robots within a swarm. While evolutionary algorithms like genetic algorithms and Practical Swarm Optimization (PSO) have been traditionally favoured for fine-tuning robot behaviors based on predefined fitness functions, their efficacy is increasingly challenged by the rise of the Reinforcement Learning (RL) approach. Critics argue that RL's trial-and-error approach may offer more dynamic and adaptable learning for robots. PSO and RL are considered effective automatic design methods in the swarm concept with their limitations [5].

Reinforcement Learning (RL) is a decision-making framework where an agent interacts with an environment to maximize cumulative rewards. It is typically modeled as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ), where S is the state space, A is the action space, P is the transition probability, R is the reward function, and γ is the discount factor.

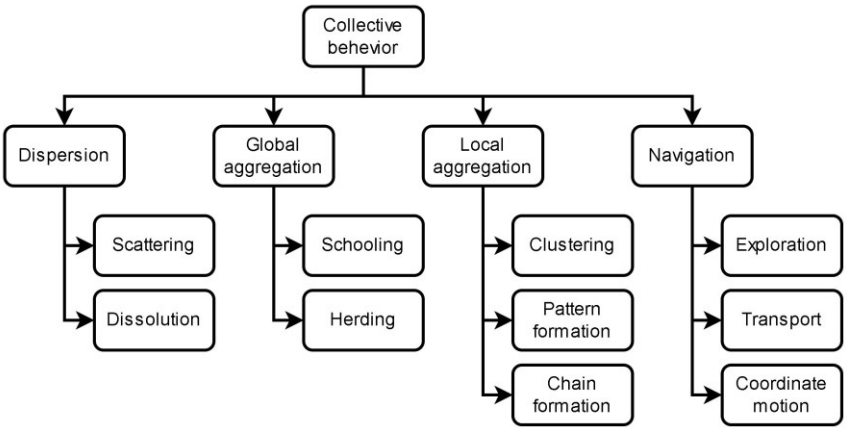

Fig. 1 Classification of collective behaviors in swarm robotics.

The goal is to learn an optimal policy π* that maximizes the expected return G, as Equation 1. Deep Reinforcement Learning (DRL) extends RL by using neural networks to approximate policies, enabling learning in high-dimensional spaces. Adjusting the weights θ based on objective function $J(\theta)$ during the interaction with the environment based on policy$\pi_\theta(s_t|a_t)$, and Advantage $\hat{A}_t$ to enhance learning process, Equation 2. Among DRL methods, Policy Gradient (PG) approaches directly optimize the policy π_θ(s|a) where updates follow as in equation 2, one of the most PG methods is Proximal Policy Optimization (PPO) that improves policy learning by introducing a clipped objective function $L^{CLIP}(\theta)$ that prevents large policy updates, ensuring stable learning as in Equation 3:

$$G = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{1}$$

$$\nabla_\theta J(\theta) = E_t\left[\nabla_\theta log\pi_\theta(s_t|a_t)\hat{A}_t\right] \tag{2}$$

$$L^{CLIP}(\theta) = E_t\left[\min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right] \tag{3}$$

In DRL, the Actor-Critic architecture is a widely used method that combines two components: Actor – Learns the policy π(θ), which maps states s to actions a and decides the best action to take, and Critic – Evaluates the Advantage $\hat{A}_t$, which estimates the expected future rewards for a given state and helps guide the Actor's learning. This framework improves learning efficiency by using the Critic to reduce variance in policy updates while the Actor focuses on policy improvement.

Most of studies have chosen PPO for SR because it ensures stable policy updates, works well in continuous and discrete action spaces, and is computationally efficient, making it ideal for controlling multiple autonomous robots in decentralized environments [6].

Practical swarm optimization (PSO): It is a computational method that optimizes a problem by iteratively improving a candidate solution about a given quality measure. It mimics the social behavior of birds flocking or fish schooling. The collection of particles (robots) moves through the solution space (environment), adjusting their positions based on their updated velocities, equation 4. The particles update their velocities by considering three key factors: their personal best position $P_{i,t}^{d}$, the best-known position of the entire swarm p_g^d , and a current position $\boldsymbol{x_{i,t}^{d}}$. This update process is mathematically represented in Equation (5).

$$\boldsymbol{v_{i,t+1}^{d}} = \omega * \boldsymbol{v_{i,t}^{d}} + c_1 * rand_i * \left(P_{i,t}^{d} - \boldsymbol{x_{i,t}^{d}}\right) + c_2 * rand_i * \left(p_g^d - \boldsymbol{x_{i,t}^{d}}\right) \quad (4)$$

$$\boldsymbol{x_{i,t+1}^{d}} = \boldsymbol{x_{i,t}^{d}} + \boldsymbol{v_{i,t+1}^{d}} \quad (5)$$

$P_{i,t}^{d}$: The best fitness value particle has at the *t* moment. p_g^d: The best fitness value among the swarm.$v_{i,t}^{d}$: The velocity of particle i at t (m/s). $x_i(t)$: The position of particle i at time t. ω: weights. C1, and c2 : cognitive and social contants.

2. METHODOLOGY

The research follows a structured approach:

1. A comparative study between PSO-based and DRL-based swarm navigation methodologies.
2. Development of an enhanced DRL framework, incorporating curriculum learning to improve generalization and adaptability in swarm environments.
3. Proposal of a hybrid modular model, combining DRL and PSO to optimize swarm coordination in foraging and navigation tasks.
4. Investigation of reward structures in DRL, using inverse reinforcement learning to fine-tune reward functions for improved decision-making.

2.1. A comparative study between PSO-based and DRL-based SR navigation behavior.

Swarm intelligence algorithms play a crucial role in optimization and robotics. Enhancements to swarm intelligence generally fall into three categories:

1. Parameter Modifications – Techniques like iSOMA-PPO [7] and RL-LSOP [8] dynamically adjust hyperparameters, improving convergence speed and efficiency. However, they focus on global optimization rather than real-time multi-agent coordination.
2. Algorithm Combinations – Hybrid models such as PSO-GA [9] and ACO-PSO [10] leverage complementary strengths, improving adaptability but increasing computational complexity.
3. Structural Modifications – Approaches like Hierarchical PSO (H-PSO) [11] introduce leader-follower dynamics, enhancing swarm coordination.

While RL-enhanced PSO methods improve performance, they often neglect swarm-level interactions and adaptability 'collective behavior'. This research uniquely evaluates how PPO and PSO influence collective behavior, bridging a major gap in SR literature. By focusing on swarm coordination rather than algorithmic tuning, it advances the understanding of structured control architectures for SRs.

Methodology for Comparing PSO vs. DRL Approaches in Swarm Navigation

The comparison between PSO and DRL for swarm navigation follows a structured methodology:

1. Simulation Environment – The experiments are conducted in a 3D Webots robot simulator with a swarm of E-puck robots. The robots navigate within environments of different sizes (1x1 m^2, 1.3x1.3 m^2, and 1.6x1.6 m^2) with varying obstacle configurations.
2. PSO Approach – Each robot in the swarm is treated as a particle, with its movement determined by PSO velocity and position updates. The fitness function evaluates the distance from the robot to the target, and each robot shares information to update its personal best and global best positions.
3. DRL Approach (PPO-based) – The DRL method is formulated as a MDP. The PPO algorithm trains the swarm using a shaped reward function, Equation 6.

$$Reward = Previous\ distance - Current\ distance + Penalty + Reward_{target} \quad (6)$$

where R_{target} rewards reaching the goal, and a penalty is given for obstacle collisions.

4. Neural Network Architecture – The PPO model consists of:
    - Actor network (10×64×64×2) for selecting actions (motor speeds).
    - Critic network (10×64×64×1) for evaluating action quality.
    - ReLU activation and adding Gaussian noise for exploration.
5. Evaluation Metrics – The comparison is based on three key metrics:
    - Effectiveness – Time taken for the first and last robot to complete the task.
    - Flexibility – Adaptability to different environments.
    - Generalization – Performance in unseen scenarios.

RL is faster and better at coordinating robots, but it works best in conditions similar to where it was trained. RL struggles to adapt to new environments without additional training or more complex training process with heavy structure. On the other hand, PSO may be slower than RL, but it performs consistently well across different environments. This makes PSO a reliable choice for tasks needing stability, especially in unpredictable settings. The slower speed of PSO does not greatly affect its ability to perform steadily. It is suggested to

use RL, where quick reactions and close coordination are essential, mainly in familiar settings.

2.2. Enhanced DRL framework, by incorporating curriculum learning in SR environments

The research introduces an enhanced DRL framework, integrating Curriculum Learning (CL) to improve the generalization and adaptability of SR in navigation tasks. CL is inspired by the pedagogical approach of structuring education, where learners tackle complex topics gradually by beginning with basics and simpler parts until solving the entire task. This concept has been adapted to various machine-learning algorithms and applications. By incorporating CL, models demonstrate improved generalization in new, unseen data. This approach also accelerates the training process, especially in non-convex scenarios where the optimization landscape contains multiple local minima.

This section introduces our significant contribution to the field: a model that integrates CL with DRL to address a navigation challenge for SR. Initially, this model was tested on individual robots before extending its application to a swarm setting. Specifically, we have enhanced the efficiency of the PPO algorithm by incorporating a CL, significantly boosting adaptability and convergence efficiency in complex environments. A comprehensive comparative analysis of three models is conducted to evaluate the effectiveness of the approach: modified PPO (PPO+CL), the standard PPO, and the DDPG. This comparison highlights the improvements the proposed model offers over existing methods.

The methodology consists of several key phases.

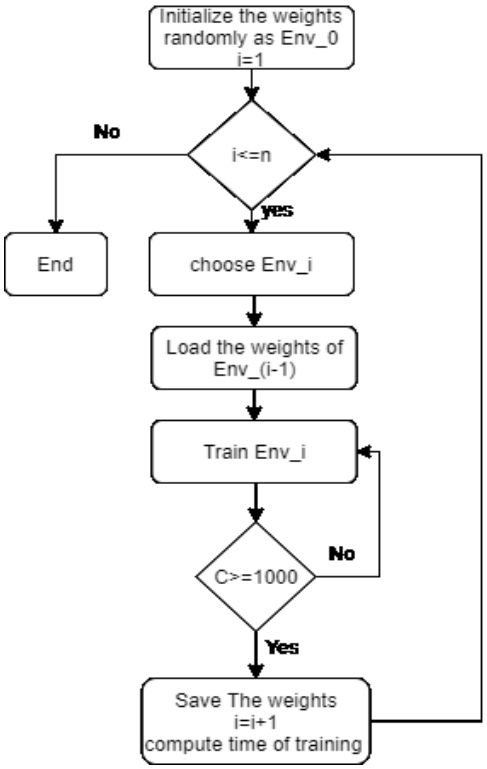

Fig. 2. Curriculum learning- Training procedure.

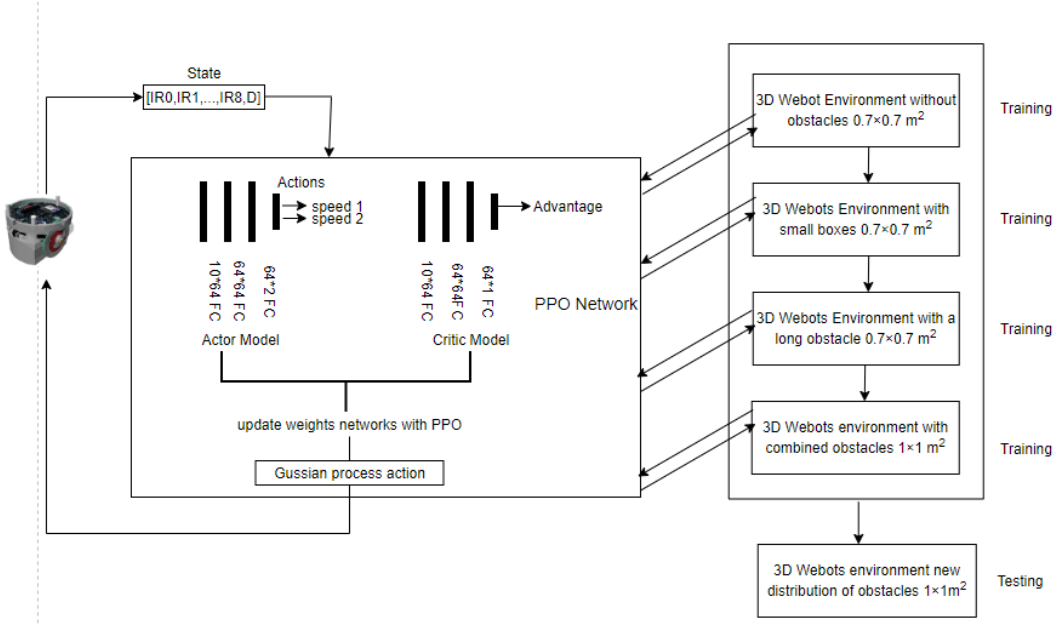

Fig. 4. DRL with CL for SR.

The flowchart in Fig. 3 illustrates the proposed training process of PPO with CL. Firstly, the weights are initialized with random values for (Env_0). Then, the training process continues sequentially as a series of incremental challenging environments i=1,2,...n, (n=4) in the proposed model. Each environment (Env_i) begins the training after transferring the learning from the previous environment by uploading the weights from the previous one (Env_(i-1)). When the robot records 1000 successful attempts to reach the goal, the model is learned, and the weights are saved. It is called C criteria. The training time is computed as $(t_{Env1}, t_{Env2}, t_{Env3,} t_{Env4})$, and the process iterates to the following environment. The cycle continues until the model has been trained among all environments. The convergence efficiency is measured by the time of training as in Equation 7:

$$Training\ time = t_{Env1} + t_{Env2} + t_{Env3} + t_{Env4} \quad (7)$$

As shown in Fig. 4, the training process is iterative, gradually increasing the complexity of the stages and transferring the learning at each stage by uploading the weights from the previous stage. The decomposition process of the training environment is obtained based on three metrics: swarm sizes (2 robots, 3robots, and five robots), collision avoidance complexity (the existence of the obstacle or not), and the distances between the targets and robots (by changing the size of the environment from 0.5×0.5 m^2, 0.7×0.7 m^2,0.1×0.1 m^2, and 1.2×1.2 m^2). We assess the swarm's performance at each stage by measuring the success rate (percentage of targets reached) and collision rate.

The curriculum-based training achieved a higher success rate in reaching targets and reduced collision rates through improved obstacle avoidance tactics. This method also accelerated the learning process, as evidenced by faster convergence times. The swarm trained with CL

demonstrated enhanced performance metrics, robust generalization, and adaptation abilities regarding training and operational efficiency.

2.3. New Hybrid Modular Design with DRL and PSO

The proposed hybrid modular model is designed for swarm robotics applications, focusing on foraging behavior, where a group of robots collaboratively searches, collects, and transports objects to a designated nest. The foraging process mimics the behavior of biological swarms, such as ants and bees, which use decentralized coordination strategies to accomplish tasks efficiently. The simulation environment is built in the Webots 3D simulator, using E-puck robots to form the swarm. The experimental area consists of a 3×3 m^2 space containing small and large objects randomly distributed in addition to dynamics boxes. The swarm operates under a decentralized control strategy, ensuring that no central unit governs the movement of robots. Instead, each robot independently navigates the space based on local perception and swarm intelligence.

The foraging task follows a structured execution cycle:

1. Search Phase – Robots explore the environment to detect objects using light sensors.
2. Gripping Phase – If an object is small, a single robot picks it up; if the object is large, the robot waits for assistance from nearby robot.
3. Transport Phase – Once an object is acquired, the robot (or group) navigates back to the nest, optimizing its path.
4. Release Phase – The object is deposited at the nest location.
5. Return Phase – The robot re-enters the search phase, repeating the process until all objects are collected.

To efficiently manage the foraging task, the hybrid modular model divides the swarm behavior into distinct functional modules, some of them are related to learning-based adaptation and others rule-based optimization as in Table 1.

Table 1. The foraging process.

Module	Optimization Method	Function
Search Module	PSO or PPO	Finds objects using light sensors.
Gripping Module	Rule-based	Picks up small objects or waits for help with large objects.
Transport Module	PPO	Navigates back to the nest.
Release Module	Rule-based	Drops objects at the nest.
Return Module	Rule-based	Re-enters search mode for a new object.

Experimental Setup & Evaluation Metrics

To evaluate the effectiveness of the hybrid approach, two configurations were tested:

1. PSO-PPO Hybrid – Uses PSO for search and PPO for transport.
2. PPO-PPO Hybrid – Uses PPO for both search and transport.

The following performance metrics were analyzed:

- Success Rate – The percentage of objects successfully transported to the nest.
- Efficiency – The time required to complete the foraging task.
- Collision Avoidance – The frequency of obstacle collisions.
- Path Optimization – The total distance traveled by robots.

The hybrid modular reduces demanding computations, such as gripping or releasing and switching between modules because of the behavior-based model. This method boosts computational effectiveness and supports tailored optimization where needed. Several benefits arise from this hybrid modular strategy:

- Specialized Optimization: PPO is integrated into critical modules to enhance task performance, notably in search and transport activities. This ensures optimal use of PPO's capabilities.

- Computational Efficiency: PPO, a resource-intensive algorithm, is selectively applied to manage the computational load effectively. This is essential for controlling numerous robots with limited processing abilities.

- Simplicity in Routine Tasks: Simpler tasks, like gripping or releasing, utilize straightforward control schemes that do not require complex decision-making and facilitate system programming and maintenance.

- Minimized Overfitting Risk: Restricting PPO to complex tasks helps avoid overfitting, keeping the model versatile and suitable for various situations.

- Accelerated Training Periods: Concentrating on specific modules decreases the total time required for training, thus expediting system rollout and adaptation.

- Optimized Reward System: The reward framework is carefully designed to match the objectives of each module, ensuring the primary aims are met and avoiding unintended actions.

- The findings indicated that PPO was more effective, achieving quicker retrieval times and greater overall efficiency due to its superior adaptability and independence. On the other hand, PSO was less effective, showing limitations in both efficiency and autonomous function. Moreover, this study highlights the advantages of a modular design in SRs, laying the groundwork for future innovations that combine operational efficiency with adaptability.

2.4. Investigation of reward structures in DRL

The research explores the impact of reward structures on SR behavior, emphasizing Deep Inverse Reinforcement Learning (DIRL) to automatically infer reward functions from expert demonstrations. This approach eliminates the need for manually designed reward functions, leading to more natural learning and better adaptation to dynamic environments.

Deep RL has significantly advanced the capabilities of SRs, generating complex collective behaviors through decentralized decision-making processes. A critical component in DRL is the design of the reward structure, which guides the learning process and influences the swarm's emergent behavior. In SR, where multiple agents must coordinate, the reward structure often encodes the desired collaborative behaviors, influencing how individual agents contribute to the group's objectives.

Reward shaping, sparse rewards, and Inverse RL (IRL) are three distinct methods used in deep RL to influence SR behavior. Reward shaping modifies the reward function by adding supplementary feedback to encourage specific behaviors, accelerate learning, and guide robots toward desired outcomes more efficiently.

Sparse rewards, awarded only for significant actions like avoiding obstacles or reaching the goal, foster robust strategies without frequent feedback, simplifying reward design but potentially slowing learning and complicating exploration.

Inverse RL derives rewards from observed optimal behaviors, enabling natural and efficient behavior learning without explicit reward programming. However, it relies heavily on the quality of demonstration data and involves greater computational complexity.

This section delves into two primary methods of configuring rewards: Shaping and Sparse methods. The general formula or representation for a sparse reward system can be described in a conditional format, where the occurrence of specific events primarily determines the reward as in Equation (8):

$$R(s, a, \acute{s}) = \begin{cases} x & if\ condition\ met \\ P & otherwise \end{cases} \quad (8)$$

x: Represents the obtained value when the condition is met, typically ranging from xmin to x_{max}, where $x_{min} \geq 0$ and $x_{max} > x_{min}$. P: Represents penalties when the condition is not met, ranging from P_{min} to P_{max}, where $P_{min} \leq P_{max} \leq x_{min}$. The general formula for reward shaping involves modifying the original reward function $R(s, a, \acute{s})$ in equation (4.1). The shaping term is added to provide the robot with additional feedback to encourage specific actions. The modified reward function can be expressed as in Euation (9):

$$\acute{R}(s, a, \acute{s}) = R(s, a, \acute{s}) + F(s, \acute{s}) \quad (9)$$

The shaping function $F(s, \acute{s})$ is carefully designed to align with the task's objectives, while ensuring that it does not change the optimal policy defined by the original reward function. By

generalizing the reward to a variable x and defining the penalty P as a percentage of x as in equation (10)

$$P = -\alpha x \quad x > 0 \qquad (10)$$

So, The aim is to find the optimal balance that maximizes navigation efficiency and safety. It introduces a relationship between penalty percentages and key performance metrics such as average time to reach the goal, number of collisions, and success rate. It showed that a balanced penalty rate, around 0.1 to 0.3, provided the best trade-off between rapid goal attainment and minimal collisions.

Proposed RL-IRL model to reward recovering and collective beahviour generating :

The proposed IRL-RL model is designed to infer rewards by demonstrating for different tasks. It is deployed for two tasks : searching for green boxes by following the light that emerges from them. Second, the navigation task where robots move from initial positions to a target

IRL-RL model consists of two sections: 1- RL is deployed to train the robots to generate the policy based on the reward inferred by IRL. 2- IRL part, we proposed the following structure, The pseudo code 1 explains the steps of this model, :

- Data Loader: This component is a repository for data received from expert demonstrations and training sessions. Data from expert demonstrations is gathered using a pre-trained expert model, while training data is accumulated during the PPO training phase. The data comprises only state frames, which include flags but omit actions. These flags indicate task completion, such as locating a specific item in a search task or arriving at a designated point in navigation tasks. The model is equipped to handle both segmented and continuous state inputs. In segmented mode, sensor readings are first normalized and then categorized into five segments ranging from 0 to 1, each representing a different value.

- Feature Extractor: As illustrated in Table 2, this component details the types of data, and the operations applied to the data received from the data loader. The incoming data is in its raw form, where values from light sensors range between [0, 4095]. Meanwhile, the distance D spans from [0, 3] meters, and the angle θ varies from [−π, π] rad. The function ∅(s) outlined in Equations (11) and (12) transforms these raw states, S, into a feature vector that is more apt for input into the model.

- A shift function is implemented on the normalized states to derive values at $t-1$. These values facilitate the establishment of correlations between states, which is crucial for enhancing the *R* network's efficacy in determining the direction of state changes.

$$\emptyset(S): S \rightarrow [0,1] \tag{11}$$

$$\emptyset(S) = \frac{Max_{Output} - Min_{Output}}{Max_{Value} - Min_{Value}} . (S - Max_{Value}) + Max_{Output} \tag{12}$$

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	$D^{(t)}, \theta^{(t)}, flag\ (Reaching\ P)$	$Normalized\ \left[D^{(t-1)}, D^{(t)}, \theta^{(t-1)}, \theta^{(t)}\right]$, *flag (Reaching P)*

- Reward Network: The reward neural network aims to estimate the underlying reward closely. This estimation is achieved by inputting the feature vector into the neural network, which outputs a scalar reward value. The network is structured with fully connected layers configured as (*length(feature−vectors) ×15×1FC*- ReLU activation function).

In the scenarios described, the feature vectors are typically of length 5, as detailed in Table 2.

- Deep IRL: The backpropagation process in the reward network involves computing losses based on Equation (13), which ensures the reward neural network's weights are updated accordingly. The key loss function used here is the binary cross-entropy loss, which effectively differentiates between the rewards observed from experts and those generated during training. This loss function is used to distinguish expert demonstrations from learned policies. The first term maximizes the probability of expert rewards R_{expert}, while the second term minimizes the probability of learned policy rewards $R_{training}$. The sigmoid function ensures the outputs are in the range (0,1), making this loss similar to binary cross-entropy for classification.

$$loss = -\log\left(sigmoid\left(R_{expert}\right)\right) - log\left(1 - sigmoid\left(R_{training}\right)\right) \tag{13}$$

Algorithm 1: Deep Inverse Reinforcement Learning (DIRL).

Step 1: Collect Expert Demonstrations
Collect expert state-action pairs: $D_{expert} = \{(s_e, a_e)\}$
Step 2: Initialize Reward Function: (length(feature−vectors) ×15×1 FC with ReLU). Same hyperparameter of actor and critic of the PPO neural network.
Initialize reward neural network R_θ with weights w_0
Step 3: Train Initial Policy with RL
Train policy π_0 using RL with initial reward R_θ
Collect agent-generated data: $D_{agent} = \{(s_\pi, a_\pi)\}$
Step 4: Compute Predicted Rewards
Forward propagate through R_θ:
$R_e = R_\theta(s_e, a_e)$ # Predicted rewards for expert actions
$R_\pi = R_\theta(s_\pi, a_\pi)$ # Predicted rewards for agent actions
Step 5: Compute Loss Function
Compute loss $L(\theta)$ based on expert vs. agent rewards (Equation 4.14)
Step 6: Update Reward Function
Backpropagate loss and update reward weights: $w_1 = w_0 - \alpha * \nabla L(\theta)$
Generate new reward function $R_{\theta 1}$
Step 7: Train RL Agent with Updated Rewards
Train new policy π_1 using RL with updated $R_{\theta 1}$
Collect new agent-generated data: $D_{agent} = \{(s_{\pi 1}, a_{\pi 1})\}$
Step 8: Iterate Until Convergence
Repeat Steps 4-7 until policy π converges to optimal behavior
(The condition here is the number of iterations)
Return: Optimized policy π^* and learned reward function $R_{\theta *}$

Finally, the IRL-RL model that utilizes deep IRL to accurately infer the reward function from expert behavior demonstrations was introduced. Rather than directly learning behaviors, IRL aims to comprehend the underlying motivations for specific actions or strategies by estimating the rewards needed to accomplish tasks through generated behaviors. This approach eliminates the necessity for extensive manual adjustment of reward functions and enables more intuitive, demonstration-based learning. The proposed IRL-RL model can manage continuous state spaces and dynamic environments, addressing continuous RL challenges through a deep neural network to represent the reward function R. Additionally, it can recover the reward function using two types of data from the data loader: segmented and continuous features, catering to nuanced strategies. The model was evaluated in two tasks within a simulated swarm robotics environment: navigating to a predefined location and searching for specific items. It proved highly effective in inferring and adapting reward structures crucial for successfully directing autonomous robotic swarms to complete these tasks. Furthermore, The results underscore the model's generalization ability across various scenarios.

3. NEW scientific results – theses

T1. The Swarm intelligence algorithms, particularly PSO and PPO, are widely applied in swarm robotics. While prior research has explored both methods individually, little attention has been given to a direct comparison of their impact on collective swarm behavior, adaptability, and coordination in decentralized robotics. Unlike studies that primarily integrate RL into PSO for parameter tuning and optimization, this research provides a comparative behavioral analysis of PSO and PPO, evaluating their individual strengths, limitations, and potential for structured hybridization. By examining their fundamental role in swarm formation, this study paves the way for more effective hierarchical, structured, and hybrid control strategies. Publications [k1], [k2].

T2. This study presents a method for optimizing mobile robot navigation using DRL by enhancing the PPO algorithm with curriculum learning. The research demonstrates improved convergence efficiency and adaptability. A comparative analysis between the modified PPO, original PPO, and other algorithms highlights the superior performance of the curriculum-augmented PPO, particularly in handling complex, dynamic environments. Additionally, the study investigates swarm robot training, revealing that curriculum learning significantly enhances success rates, collision avoidance, and generalization capabilities in novel scenarios [k3], [k4].

T3. It introduces a hybrid approach combining automatic design methods like DRL or PSO within a modular design to tackle the foraging problem in swarm robotics. The system, implemented in a 3D environment using Webots, involves 8 E-Puck robots equipped with light sensors to search for and transport dynamically moving resources. The modular architecture enhances system manageability and reduces computational demands, making it easier to address complex, non-static foraging tasks. The simulations show that the RL-based model outperforms PSO regarding task efficiency, resource collection, and adaptability to dynamic environments. RL-equipped robots demonstrate superior individual learning and autonomy, contributing to more effective collective swarm intelligence, while PSO relies more on the collective knowledge of the swarm [k5].

T4. The study systematically examines how reward functions can be structured to guide robots in tasks such as efficient resource collection, adaptive navigation, and decentralized decision-making. A key aspect of this research is the balancing of penalties and rewards, ensuring that learning is neither hindered by excessive punishment nor misdirected by overly generous rewards, which could lead to suboptimal behaviors. A major contribution of this thesis is the introduction of a Deep Inverse Reinforcement Learning (RL-IRL) model designed to discover optimal reward structures for guiding swarm behavior in complex and unpredictable environments. Unlike traditional RL methods, which rely on manually defined rewards, IRL extracts implicit reward functions by learning from expert swarm demonstrations. This method is particularly effective in handling continuous state and action spaces, allowing the swarm to develop adaptive collective behaviors based on specific task objectives [k5] ,[k6] ,[k7] ,[k8].

4. LIST OF PUBLICATIONS RELATED TO THE TOPIC OF THE RESEARCH FIELD

[k1] Iskandar A., Kovács B. "A survey on automatic design methods for swarm robotics systems." Carpathian Journal of Electronic & Computer Engineering, v. 14, no. 2, 2021. https://doi.org/10.2478/cjece-2021-0006.

[k2] Iskandar A., Hammoud A., Kovács B. " Swarm Robotics Navigation Task: A Comparative Study of Reinforcement Learning and Particle Swarm Optimization Methodologies " Mekhatronika, Avtomatizatsiya, Upravlenie. v. 25, no. 9, pp. 471-478. https://doi.org/10.17587/mau.25.471-478.

[k3] Iskandar A., Kovács B. "Curriculum learning for deep reinforcement learning in swarm robotic navigation task." Multidiszciplináris Tudományok, v. 13, no. 3, pp. 175-187, 2023. https://doi.org/10.35925/j.multi.2023.3.18.

[k4] Iskandar A., Kovács B. "Investigating the impact of curriculum learning on reinforcement learning for improved navigational capabilities in mobile robots." Inteligencia Artificial, v. 27, no. 73, pp. 163-176, Mar 2024. https://doi.org/10.4114/intartif.vol27iss73pp163-176.

[k5] Hammoud A., Iskandar A., Kovács B. " Dynamic foraging in swarm robotics: a hybrid approach with modular design and deep reinforcement learning intelligence" Informatics and automation, v. 24, pp. 51, 2025. https://doi.org/10.15622/ia.24.1.3.

[k6] Iskandar, B. Kovács, "Analysis of the effects of reward structures in deep reinforcement learning on the path planning of mobile robots." in 5th international black sea modern scientific research congress, Rize, Turkiye. pp. 758, 2023.

[k7] Iskandar A., Rostum H.M., Kovács B. "Using deep reinforcement learning to solve a navigation problem for a swarm robotics system." In 2023 24th International Carpathian Control Conference (ICCC), pp. 185-189, IEEE, 2023. https://doi.org/10.1109/ICCC57093.2023.10178888.

[k8] Iskandar A., Hammoud A., Kovács B. " Implicit understanding: decoding swarm behaviors in robots through deep inverse reinforcement learning" Informatics and automation, v.23, p. 1485,2024. https://doi.org/10.15622/ia.23.5.8.

5. LITERATURE CITED IN THE THESES BOOKLET

[1] Cheraghi, A.R., Shahzad, S., Graffi, K.: Past, present, and future of swarm robotics. In: Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) 3, pp. 190–233,2022. https://doi.org/10.1007/978-3-030-82199-9\13

[2] Majid, M., Arshad, M., Mokhtar, R.: Swarm robotics behaviors and tasks: a technical review. Control Engineering in Robotics and Industrial Automation: Malaysian Society for Automatic Control Engineers (MACE) Technical Series 2018, 99–167,2022. https://doi.org/10.1007/978-3-030-74540-0 5

[3] Shahzad, M.M., Saeed, Z., Akhtar, A., Munawar, H., Yousaf, M.H., Baloach, N.K., Hussain, F.: A review of swarm robotics in a nutshell. Drones 7(4), 269,2023. https://doi.org/10.3390/drones7040269

[4] Iskandar, A., Kov´acs, B.: A survey on automatic design methods for swarm robotics systems. Carpathian Journal of Electronic and Computer Engineering 14(2), 1–5 ,2021. https://doi.org/10.2478/cjece-2021-000614

[5] Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 7, 1–41 (2013) https://doi.org/10.1007/s11721-012-0075-2

[6] Iskandar A., Kovács B. "Investigating the impact of curriculum learning on reinforcement learning for improved navigational capabilities in mobile robots." Inteligencia Artificial, v. 27, no. 73, pp. 163-176, Mar 2024. https://doi.org/10.4114/intartif.vol27iss73pp163-176

[7] Klein L., Zelinka I., Seidl D. "Optimizing parameters in swarm intelligence using reinforcement learning: An application of Proximal Policy Optimization to the iSOMA algorithm." Swarm and Evolutionary Computation, v. 85, p. 101487, 2024. https://doi.org/10.1016/j.swevo.2024.101487

[8] Wang F., Wang X., Sun S. "A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization." Information Sciences, v. 602, pp. 298-312, 2022. https://doi.org/10.1016/j.ins.2022.04.053

[9] Gad A.G. "Particle swarm optimization algorithm and its applications: A systematic review." Archives of Computational Methods in Engineering, v. 29, no. 5, pp. 2531-2561, 2022. https://doi.org/10.1007/s11831-021-09694-4

[10] Niknam T., Amiri B. "An efficient hybrid approach based on PSO, ACO, and k-means for cluster analysis." Applied Soft Computing, v. 10, no. 1, pp. 183-197, 2010. https://doi.org/10.1016/j.asoc.2009.07.001

[11] Janson S., Middendorf M. "A hierarchical particle swarm optimizer for dynamic optimization problems." Applications of Evolutionary Computing: EvoWorkshops 2004, EvoBIO, EvoCOMNET, EvoHOT, EvoISAP, EvoMUSART, and EvoSTOC, Coimbra, Portugal, April 5-7, 2004. Proceedings, pp. 513-524, 2004. https://doi.org/10.1007/978-3-540-24653-4_52