The Mapping Trick: Leveraging RoboSoccer Obstacle Avoidance and Navigation for Advanced Tasks Management Solutions

Seyed Omid Azarkasb[1]✉, Seyed Hossein Khasteh[2]

[1] Visiting Professor and Ph.D. Student of Artificial Intelligence and Robotics, K.N. Toosi University of Technology, Tehran, Iran
Seyedomid.azarkasb@email.kntu.ac.ir
ORCiD: 0009-0003-3322-2216

[2] Assistant Professor of Artificial Intelligence, K.N. Toosi University of Technology, Tehran, Iran
Khasteh@kntu.ac.ir
ORCiD: 0000-0003-2227-4507

Abstract

This paper addresses the mapping of tasks partitioning and scheduling problem based on reinforcement learning to the problem of obstacle avoidance and navigation in RoboSoccer through feature engineering. By conducting a comparative analysis of these two problems and identifying their similarities and differences, innovative methods and techniques from RoboSoccer are applied to solve task scheduling and partitioning issues within the context of the foggy Internet of Everything (IoE). This research aims to illustrate and analyze the impacts of this mapping on the performance of various algorithms and the resulting improvements. The mapping and resolution of diverse problems in the RoboSoccer environment offer several advantages, including the management of dynamic environments and rapid changes, the simplified handling of agent interaction complexities, knowledge transfer and reuse of algorithms, the high diversity of RoboSoccer simulators, enhanced learning processes, and optimized energy consumption. These benefits can be considered as innovative techniques for researchers. Initially, traditional and modern algorithms in the domain of task scheduling are tested and evaluated, and subsequently, the results and performance of the algorithms are analyzed and compared before and after the mapping process. Performance evaluation of algorithms is conducted using a wide range of metrics. The findings indicate that mapping the problem to the RoboSoccer environment not only does not degrade algorithm performance but also results in significant improvements in the efficiency and effectiveness of task scheduling and partitioning systems. The ability to leverage RoboSoccer's facilities, tools, and advancements to address various issues constitutes a major contribution of this study.

Keywords: Task Scheduling, RoboSoccer, Fog Computing, Reinforcement Learning, Feature Engineering, Internet of Everything (IoE), Energy Optimization.

1- Introduction

With rapid advancements in artificial intelligence and robotics technologies, diverse applications for intelligent systems have emerged, particularly in dynamic and complex environments. A significant and challenging research area in this domain is the mapping of various problems to new and complex environments, which can enhance the performance and efficiency of algorithms and intelligent systems. This paper explores a specific example of such mappings: the alignment of task partitioning and scheduling problems with obstacle avoidance and navigation issues in RoboSoccer. This alignment aims to find a suitable correspondence to leverage the facilities, developments, and advancements in the field of RoboSoccer. Task partitioning and scheduling, especially within the foggy Internet of Everything (IoE) context, are key issues in intelligent systems and resource management systems. This problem involves allocating limited resources to various tasks in an optimal manner, based on constraints that minimize task completion times and improve system performance. These constraints include network bandwidth, device processing power, latency, and energy consumption. The Internet of Everything (IoE), an advanced evolution of the Internet of Things (IoT), was introduced by Cisco and concurrently by several renowned technology companies such as Gartner and Qualcomm [1]. In IoE, not only are objects, devices, processes, and data interconnected, but individuals also interact seamlessly with these entities. This technology facilitates better

communication and collaboration between different system components, leading to enhanced productivity, cost reduction, and increased efficiency. Due to the potential for rapid response near edge components, the use of fog computing for IoE applications has grown rapidly [2]. Fog computing is an intelligent computational system where nodes in the fog can independently address computational and processing requests from end devices, and can also connect with each other for collaboration or delegate certain tasks to cloud servers. Management and collaboration procedures in fog nodes are implemented for task management and control. Collaboration between fog nodes can occur through remote or local communications [3]. It is important to note that this collaboration is not observed in edge computing, which is a key difference between fog computing and edge computing [4]. Other differences can be found in reference [5], which provides ten key comparisons between fog computing and edge computing. At the same time, the issue of obstacle avoidance and navigation in robots addresses challenges related to robot movement and routing in complex environments with diverse obstacles. This problem includes optimal path planning for navigating obstacles and reaching specific goals, which aids the precision and efficiency of robotic systems [6]. Mapping the problem to the RoboSoccer environment, as a dynamic and complex setting, provides an appropriate opportunity to examine and analyze the improvements resulting from this transfer. RoboSoccer robots, due to their need for fast and optimal interactions in variable and complex environments, are recognized as suitable models for evaluating various algorithms and techniques in scheduling and navigation domains. These robots must make rapid decisions in unpredictable environments, avoid obstacles, and select the most optimal path to achieve their goals [7]. In this context, numerous advanced simulators have been developed to address RoboSoccer problems [8], which, if the problem can be mapped to the RoboSoccer environment, could benefit from these advancements. This paper is part of the future work from our previous study titled "Eligibility Traces in an Autonomous Soccer Robot with Obstacle Avoidance and Navigation Policy" [9]. This paper examines the mapping of task partitioning and scheduling problems to obstacle avoidance and navigation in the RoboSoccer environment and analyzes the performance of various algorithms, including reinforcement learning algorithms, under real and dynamic conditions. The primary research question is whether mapping task partitioning and scheduling problems to the RoboSoccer environment can improve algorithm performance and optimize the system. Our hypothesis is that this mapping not only enhances the efficiency and accuracy of intelligent systems but also provides benefits such as better management of dynamic environments, more complex interactions between agents, and optimized energy consumption. Reducing energy consumption is a critical factor in modern computing [10]. The research methodology of this paper includes detailed analysis and feature engineering of both datasets, establishing suitable correspondences between features, and evaluating algorithm performance in both non-mapped and mapped scenarios. For these evaluations, real datasets from the task scheduling and RoboSoccer domains are utilized. The results indicate significant improvements in algorithm performance after mapping to the RoboSoccer environment. The main innovations of this paper include the creative alignment of seemingly disparate problems in a dynamic and complex environment and a comprehensive analysis of the impact of this mapping on the performance of intelligent algorithms. The findings of this study can significantly contribute to the development and improvement of problem-solving in real-world applications.

Following the introduction in Section 1, the structure of the paper is organized as follows: Section 2 provides a comprehensive review of the literature relevant to the study, specifically focusing on task partitioning and scheduling, as well as obstacle avoidance and navigation in RoboSoccer robots. At the end of this section, two tables are presented that summarize the findings, strengths, and weaknesses of the reviewed sources. Section 3 details the proposed methodology, including implementation specifics and experimental procedures. This section includes thorough analyses of the datasets employed, the performance of both classical and advanced algorithms, and the evaluation criteria used. Subsequently, Section 4 offers an integrated analysis of the results, with a particular focus on the key features of each algorithm. The paper concludes with Section 5, which presents the conclusions and outlines future research directions and development opportunities in Section 6.

2- **Literature Review**

The issue of task partitioning and scheduling, particularly in fog computing, and the challenge of obstacle avoidance and navigation in RoboSoccer robots are two critical research domains in artificial intelligence and robotics. Both problems exhibit unique complexities, and various methods have been proposed to address them. Among these, reinforcement learning has emerged as a prominent approach in the field of

artificial intelligence and machine learning due to its high capabilities in solving complex and dynamic problems. This method is particularly relevant in the contexts of task partitioning and scheduling, as well as obstacle avoidance and navigation in RoboSoccer robots. This section reviews studies and research related to each of these issues.

2-1- Tasks Partitioning and Scheduling

In recent years, the problem of task scheduling in cloud computing has garnered significant attention from researchers. With the advent of fog computing and the emergence of foggy environments such as the Internet of Everything (IoE), this issue has gained even more prominence, leading to extensive research in this area, as highlighted in studies [11] and [12]. Efficient task partitioning and scheduling are crucial for optimal utilization of computational and communication resources. These tasks involve allocating resources to different devices, scheduling task execution, and managing resources to maximize system performance [13]. The heterogeneity of tasks, the unknown and dynamic nature of foggy IoE environments, and the involvement of human components in this context [14] create specific conditions, such as critical states [15], which have led researchers to focus on reinforcement learning methods for solving this problem. In this approach, the learning agent receives rewards or penalties based on its actions in the environment. Consequently, the agent interacts with the environment through trial and error and learns to choose optimal actions to achieve its goals. Reinforcement learning-based systems enable the learning agent to perform various actions in different states and receive varying rewards, thus learning the appropriate actions for each state based on the cumulative rewards received [16]. High online performance is required for such systems as evaluation often occurs simultaneously with learning [17]. A model presented in [18] considers each fog node as an agent that decides at each moment whether to forward a task to other nodes or handle it itself. The model aims to minimize long-term task delay and energy consumption, employing Q-learning to achieve this objective. Authors in [19] address the task partitioning problem in fog computing with the goal of minimizing the weighted average task completion time and the average number of requested resources. They propose a Q-network method for solving this problem, which incorporates multiple memories for storing experiences with small interactions that are applied separately. In [20], adding a backup queue to the algorithm yielded satisfactory results. In this architecture, tasks are first placed in the backup queue before entering the main waiting queue. The agent can only allocate resources to tasks in the waiting queue, not the backup queue. This paper argues that if each task were to correspond to an action for the agent, the action space would become very large and complex, as the number of tasks in the system could be very large. Therefore, some tasks are placed in the backup queue and others in the main waiting queue. When a slot in the waiting queue becomes empty, a task from the backup queue is transferred to this queue. [21] formulates the resource allocation problem in fog computing using a Markov decision process. This paper employs several reinforcement learning algorithms, including Q-learning, SARSA, and Monte Carlo methods, to learn optimal policies in IoE environments. [22] introduces a fog architecture based on a software-defined networking (SDN) controller. In this architecture, a central controller, with its knowledge of the overall network status, decides on the load offloading operations for each fog node. The ultimate goal of this system is to minimize task delay and node overload. The controller finds optimal actions using reinforcement learning algorithms, enabling each node to find an optimal neighboring node to send tasks to, thereby reducing execution delay and node overload. However, this method is limited in scalability due to its reliance on a central controller. [23] addresses the task partitioning problem based on constraints in fog nodes and the various requirements of IoE applications, considering delay constraints. The problem is modeled as a Markov process, and solutions are obtained using Q-learning, SARSA, and Monte Carlo methods. A significant advantage of reinforcement learning is its ability to learn from previous actions to address the scheduling challenges, even in the absence of a mathematical model of the environment, achieving optimal scheduling [24]. Here, scheduling refers to the allocation of shared resources over time to efficiently complete tasks within a specified period. This term is used separately for tasks and resources in the context of task scheduling and resource allocation [25]. The heterogeneity of nodes and their positions in distributed systems, and the order of tasks in a directed acyclic graph of dependencies, can be used as a scheduling policy to achieve better execution times [26]. Similarly, a platform described in [26] offers scheduling solutions as a service based on machine learning agents and proposes a global reinforcement learning model for scheduling. This platform, known as a machine learning toolbox, facilitates further development of scheduling algorithms and easy integration processes. The practical model can be easily mapped onto parallel systems to improve overall efficiency. Due to the complex nature of distributed systems,

reinforcement learning models face limitations and complexities, and as more nodes are added to the system, the world expands, and the reinforcement learning agent may struggle to learn an optimal policy [26]. The study presented in [27] introduces an innovative task scheduling approach tailored for cloud environments, leveraging meta-reinforcement learning techniques. This method aims to optimize the learning process and minimize the adaptation time to novel tasks. The research demonstrates that the proposed approach outperforms traditional methods under various conditions and has the potential to enhance the overall efficiency of cloud systems. In [28], a model for optimization based on precise forecasting of workload and future demands in edge and fog computing environments is proposed. By employing deep learning techniques and time series forecasting, this model achieves high accuracy in predicting future needs and enhances resource allocation. This approach can contribute to reducing network latency and extending the lifespan of user devices. Nevertheless, deep reinforcement learning is an emerging technology that has dynamically addressed task optimization and resource scheduling [29]. However, methods relying solely on reinforcement learning often converge slowly and perform relatively poorly in real-time applications in foggy IoE environments with large task volumes. Therefore, the innovative idea of this paper is to map the problem to the Soccer robot problem. According to studies conducted by the authors, the characteristics of the dataset and, in particular, the obstacle avoidance and navigation problem in Soccer robots have a close symmetry with the task partitioning and scheduling problem. If feature engineering [30] is performed well and accurately, advances and tools in the Soccer robot domain can be optimally utilized to address the task partitioning and scheduling problem. A summary review of the research and studies discussed in this section, in chronological order of publication, is presented in Table 1. The chronological arrangement in the table provides a clear depiction of the evolutionary trajectory of these methods, offering insights into their development over time.

Table 1: Summary Review of the Research and Studies Reviewed in This Section on Tasks Scheduling and Partitioning

Ref	Year	Methodology and Advantages	Drawbacks, Key Points, and Considerations
[30]	2013	Utilizes a Markov Decision Process for feature selection by modeling the feature space as a decision-making process, employing temporal difference methods to traverse the feature space and select the optimal subset of features.	Fails to incorporate the best features and exclude the worst features adaptively, relies on a fixed number of iterations without considering convergence conditions, and does not utilize multi-class datasets.
[26]	2018	Introduces a platform-as-a-service framework based on reinforcement learning agents, termed the Machine Learning Box, which facilitates the development and integration of scheduling algorithms.	Aims to reduce the number of states while achieving policies for task acceptance or rejection.
[16]	2019	Approximation of branch weights in task flow graphs based on estimated task execution times, avoiding tasks with a low success probability as predicted by failure forecasts.	Enables more informed scheduling and minimizes task failures.
[11]	2019	Provides a comprehensive review of scheduling strategies, relevant criteria for cloud computing environments, and limitations in determining which features should be included or excluded in a system.	Examines three distinct perspectives: methods, applications, and parameter-based criteria.
[23]	2019	Formulates the resource allocation problem in fog computing as a Markov Decision Process and employs various reinforcement learning methods, including Q-learning, SARSA, SARSA (λ), and Monte Carlo methods.	Formulates the resource allocation problem as a Markov Decision Process but does not address dynamic resource allocation with heterogeneous service times or shared resource blocks among

			multiple fog nodes.
[15]	2020	Develops a Multi-Resource Scheduling and Routing Problem (MRSRP) framework for emergency recovery. Six methods, including Monte Carlo, are evaluated for resource delivery efficiency. Monte Carlo is found to be the most effective for large instances.	The study shows that while all methods are effective for small instances, their performance in large-scale scenarios varies. Further validation of Monte Carlo's effectiveness in very large or complex situations might be needed.
[25]	2020	Highlights reinforcement learning as an optimal approach for dynamically and effectively addressing task scheduling and resource optimization issues based on documented evidence.	Reinforcement learning traverses a prolonged path with improvements and developments, ultimately providing significant research value in this domain.
[12]	2020	Conducts a systematic literature review on computational resource management approaches in fog computing across six major areas: task placement, resource scheduling, job loading, load balancing, resource allocation, and provisioning.	Addresses resource limitations, heterogeneity, dynamic nature, and unpredictability of the fog computing environment.
[18]	2020	Develops a multi-agent Q-learning algorithm where each edge user acts as a learning agent observing its local network environment to make optimal resource allocation decisions.	Formulates the mechanism as a stochastic game, enabling independent policy learning at the edge (end-user) and reducing computational load on gateways.
[19]	2020	Demonstrates improved convergence performance of the proposed algorithm, with simulation results showing that the learned policy outperforms compared policies in terms of long-term average reward, achieving lower weighted sums of average task completion time and average requested resources.	Formulates the resource allocation problem in IoT edge computing as a Markov Decision Process.
[22]	2020	Utilizes deep reinforcement learning techniques for resource and cost management through network virtualization architectures, employing SDN controllers to reduce the complexity of edge computing architectures and improve resource utilization efficiency.	Faces issues with scalability and lacks implementation and validation of fog computing architectures.
[13]	2021	Selects the most suitable task set to maximize resource utilization.	Employs genetic algorithms to enhance convergence speed and achieve global optimization.
[20]	2021	Optimizes the task execution sequence and allocation jointly using a neural network for feature extraction, considering task diversity and resource heterogeneity.	Does not address collaborative task execution in edge devices and communication delays.
[21]	2021	Applies a deep Q-network approach to approximate optimal value functions, useful for task offloading in fog nodes by selecting appropriate nodes and managing resources while ensuring quality of service	Emphasizes the integration of virtualized servers within networks and formulates the approach based on Markov Decision Processes.

		requirements.	
[17]	2022	Implements in a real-world vehicular edge computing environment.	Lacks consideration of concurrent multi-node management.
[14]	2022	Divides the working area into logical segments to optimize customer order preparation times, considering task execution uncertainties and human user involvement, and critical time windows.	The algorithm dynamically recalculates solutions when an action cannot be performed due to unexpected events, allowing for efficient and timely updates.
[24]	2022	Balances workload, reduces service time, and decreases task failure rates using deep Q-network algorithms to address complexity and high-dimensional workload scheduling, modeling based on previous actions to achieve optimal scheduling in the absence of a mathematical model of the environment.	Analyzes the system model into three sub-models: task model, network model, and computational model, considering the architecture of edge computing.
[27]	2023	Introduces MRLCC, a task scheduling method for cloud environments based on meta reinforcement learning. MRLCC improves adaptability to new environments and sample efficiency. It performs better in reducing makespan and maintains high utilization rates after minimal gradient updates.	Requires comprehensive experimentation to validate effectiveness across diverse environments. The performance in highly dynamic or rapidly changing environments may need further exploration.
[28]	2024	Proposes DRLIS, a Deep Reinforcement Learning-based scheduling algorithm for edge and fog computing environments. DRLIS optimizes system load and response time, significantly reducing execution costs of IoT applications in load balancing, in response time, and in weighted cost. Implemented in the FogBus2 framework for server less computing integration.	Faces challenges with limited computational resources on edge/fog servers, which may affect the application of computationally demanding techniques. The dynamic nature of edge/fog environments requires continuous evaluation of algorithm performance in varying conditions.
[29]	2024	Provides a comprehensive review of Deep Reinforcement Learning (DRL)-based methods for resource scheduling in cloud computing. Highlights the effectiveness of DRL due to its integration of deep learning and reinforcement learning, which offers significant improvements in resource management over classic algorithms. Analyzes theoretical formulations and RL frameworks, and discusses future directions in DRL-based cloud scheduling.	Classic algorithms such as heuristics and meta-heuristics may struggle with complex scenarios where direct evaluation of scheduling performance is difficult. The review highlights the need for further research to address challenges and explore future directions for DRL in cloud scheduling.

2-2- Obstacle Avoidance and Navigation in Robotic Soccer

The RoboCup, an international initiative founded in 1997, has emerged as a leading platform for advancing research in robotics and artificial intelligence [31]. Its core mission is to develop robotic teams capable of competing against human teams in soccer matches. This competition has not only facilitated the development of cutting-edge robotics technologies and AI algorithms but has also served as a comprehensive testing ground for evaluating and optimizing these technologies. Over time, RoboCup has evolved into a significant research platform, enabling researchers to refine and enhance navigation and machine learning algorithms. It stands as one of the most prestigious robotics competitions globally, driving numerous innovations in the

field .In the context of RoboCup, robotic soccer players are one of the most challenging and engaging aspects, demonstrating capabilities such as dynamic movement, obstacle avoidance, and precise navigation [32]. Therefore, advancing sophisticated algorithms to improve the performance of these robots is crucial . One prominent method in this area is Reinforcement Learning (RL), which empowers robots to learn optimal policies for obstacle avoidance and navigation through continuous interaction with their environment and by leveraging past experiences [33]. RL, inspired by learning theories in neuroscience and psychology, is a powerful AI technique that autonomously optimizes complex behaviors in dynamic environments without direct supervision. Applying RL to robotic soccer can lead to significant improvements in navigation accuracy and obstacle avoidance .A pioneering RL algorithm is Q-Learning, introduced by Watkins and Dayan in 1992. This algorithm enables robots to utilize their past experiences to optimize future actions. In Q-Learning, robots maintain and update a Q-value table, which records the value of various actions in different states of the environment. The use of Q-Learning has resulted in notable improvements in navigation accuracy and obstacle avoidance in robotic soccer [34]. With advancements in deep neural networks, Deep Reinforcement Learning (DRL) has become a popular research domain [35]. DRL algorithms, such as Deep Q-Network (DQN), integrate RL with deep neural networks to estimate value functions and optimal policies [36]. The use of DQN and other DRL algorithms, like Proximal Policy Optimization (PPO) [37] and Soft Actor-Critic (SAC) [38], has demonstrated superior performance in complex applications, including robotic navigation, compared to traditional methods. Moreover, implementing RL algorithms in simulated environments allows researchers to test and optimize their algorithms without incurring the costs and risks associated with real-world environments [39]. Simulated environments, such as Gazebo and V-REP, enable researchers to accurately replicate RoboCup environments for testing and refining navigation algorithms [40]. Research has shown that employing algorithms like Deep Deterministic Policy Gradient (DDPG) in simulated environments significantly enhances the accuracy and effectiveness of robotic navigation [41]. These algorithms use deep neural networks to learn continuous policies for navigating dynamic and complex environments. One of the primary challenges of using RL in robotic soccer is the requirement for large amounts of training data and extended training times. To address this challenge, researchers have utilized optimization techniques such as Transfer Learning and Multi-Agent Reinforcement Learning. Transfer learning allows agents to apply experiences from similar environments, thereby reducing training time [42]. Studies have demonstrated that Transfer Learning in robotic soccer leads to substantial improvements in training efficiency and algorithm performance [43]. Multi-Agent Reinforcement Learning enables agents to enhance overall system performance through collaboration and interaction. In this approach, each agent operates as an independent player interacting with others, learning optimal policies through these interactions [44]. Research indicates that this method can significantly improve the performance of robotic soccer players [45]. For instance, studies have shown that Multi-Agent Reinforcement Learning enhances team coordination and efficiency [46]. Reinforcement Learning has applications beyond navigation and obstacle avoidance, extending to strategy formulation and team collaboration. For example, applying RL to develop attack and defense strategies can enhance the effectiveness of robotic teams [47]. Distributed Reinforcement Learning is another advanced approach, allowing agents to use their own and others' experiences in parallel to improve their policies. Research has indicated that distributed RL algorithms significantly improve learning efficiency and speed in robotic soccer [48]. In summary, substantial scientific progress has been made in robotic soccer. By carefully engineering features and addressing challenges within this environment, these advancements can greatly contribute to the development of innovative solutions for existing problems. Table 2 provides a chronological summary of the research and studies examined in this section, organized by their publication dates. The sequential ordering of the years in the table effectively illustrates the progression and refinement of these methods, tracing their evolution through time.

Table 2: Summary Review of the Research and Studies Reviewed in This Section on Obstacle Avoidance and Navigation in Robotic Soccer

Ref	Year	Methodology and Advantages	Drawbacks, Key Points, and Considerations
[31]	1999	Introduces RoboCup for integrating AI and robotics, showcasing research in a competitive, dynamic environment.	Focuses on early RoboCup achievements, with less emphasis on practical implementation challenges.

[40]	2004	Introduces Gazebo, an advanced open-source 3D simulator for multi-robot environments, enabling precise control and high-fidelity simulations that closely replicate complex real-world scenarios.	While providing robust simulation capabilities, it necessitates substantial computational power and may present a steep learning curve for new users.
[45]	2012	Proposes a multi-agent reinforcement learning approach for robot soccer, leveraging a probabilistic neural network (PNN) for action prediction and a sharing policy mechanism to enhance learning speed and coordination among agents.	While effective in improving coordination, the method's reliance on joint-state and joint-action complicates scalability and may limit its application in highly dynamic or unpredictable environments.
[32]	2016	Provides a retrospective overview of RoboCup's 20-year history, focusing on developments across various leagues. Highlights RoboCup as a successful and influential robotics competition, emphasizing its role as a model for similar events.	The article is a retrospective and does not focus on specific methodologies or technical advancements but rather on the overall success and impact of RoboCup over two decades. Further detailed analysis of individual league advancements might be needed.
[42]	2017	Introduces a scalable transfer learning methodology in reinforcement learning, focusing on model-based approaches that support efficient online learning and adaptation in heterogeneous, dynamic environments. The approach is validated with regret bounds and demonstrates quick convergence to near-optimal policies.	Despite its scalability, the method's computational complexity may still be prohibitive for some real-world applications, and the variant formulation of Markov decision processes may require extensive computational resources in highly dynamic settings.
[48]	2019	Applies Distributional Reinforcement Learning (RL) to the Half Field Offense testbed for multi-agent RL. Focuses on learning the distribution of possible returns rather than just the expected value, aiming to improve learning efficiency and explore transfer learning potentials.	The application of Distributional RL in multi-agent tasks is novel and not extensively explored yet. This initial work is a starting point, with future plans to integrate it with multi-agent transfer learning. The approach may require further development to address practical implementation challenges..
[36]	2020	Proposes the Hierarchical Movement Grouped Deep-Q-Network (HMG-DQN) to enhance training efficiency in RoboCup soccer by focusing on high-level action hierarchies.	Shows improved performance in cooperative scenarios but the focus is primarily on high-level movement groups.
[38]	2021	Employs DRL with a Kinect sensor for grasping moving objects. Integrates Soft-Actor-Critic and object detection, enabling the system to generalize from simulations to real robots. Demonstrates effective autonomous grasping with various object trajectory.	The approach depends on the integration of multiple technologies, which may introduce complexity in system implementation and calibration. Further validation might be needed to assess the system's performance in highly dynamic or diverse real-world scenarios.
[43]	2022	Presents novel neural network architectures for semantic segmentation and object detection in RoboCup soccer using the Nao robot. The models leverage synthetic transfer learning to improve performance with limited labeled data. These architectures provide	Focuses on the computational limitations of the Nao robot and aims to enhance vision system performance despite these constraints. Challenges include ensuring high accuracy with minimal computational resources and a small amount of labeled data.

		low-cost inference while outperforming state-of-the-art methods like Tiny YOLO.	
[46]	2022	Uses Reinforcement Learning (RL) to unify control and strategic decision-making in a multi-agent robotic soccer environment. The approach involves self-play to enable RL agents to learn both control and strategy, achieving high win rates (up to 93%) against heuristic strategies. A simulated environment was developed for the study.	Focuses on combining control and strategic decision-making into a single RL framework. Potential challenges include the need for extensive training to achieve high performance and the generalization of strategies against diverse opponents.
[47]	2022	Uses an improved SARSA algorithm for strategic planning in robot soccer games, specifically in the RoboCup2D platform. Introduces heuristic information to enhance learning by sharing Q-values between agents. The algorithm improves both offensive and defensive capabilities of the team.	Focuses on augmenting SARSA with heuristic information and Q-value sharing. Potential challenges include the complexity of integrating heuristic methods and ensuring the effectiveness of the algorithm across diverse scenarios.
[39]	2022	Quantitatively compares four widely-used mobile robot simulators: CoppeliaSim, Gazebo, MORSE, and Webots. The comparison is based on accuracy of motion using data from real-world experiments with a Husky A200 mobile robot. CoppeliaSim is identified as the best performer, with Gazebo as a close alternative.	The study is focused on comparing simulators based on motion accuracy, which might not cover all aspects of simulation needs such as ease of use, community support, or additional features. The results are specific to the terrains and robot model used.
[33]	2023	Explores scaling Multi-Agent Reinforcement Learning (MARL) to a full 11 vs. 11 soccer game using a custom 2D simulator. Introduces improvements to the Proximal Policy Optimization (PPO) algorithm, including attention mechanisms, network sharing, and Polyak averaging for stable training. Achieves competent strategies for a full 22-player game with limited computational resources.	The study relies on a 2D simulator rather than a more complex 3D environment, which might limit generalization. The improvements to PPO are specific to the context of soccer simulation and might not be broadly applicable to other MARL problems.
[34]	2023	Presents a shooting strategy for RoboCup Soccer 3D League that incorporates Q-Learning to adjust robot movement parameters dynamically while walking. This method uses a curved path for optimal positioning and applies Inverse Kinematics (IK) for shooting once the robot is correctly positioned. Demonstrates superior performance in RoboCupSoccer and Iran's Open RoboCup3D leagues.	The approach is tailored to RoboCup3D's specific challenges and may not generalize to other robotic environments or soccer simulations. The effectiveness of Q-Learning in real-world noisy environments may differ from simulation results.
[37]	2023	Utilizes Proximal Policy Optimization (PPO) and Generative Adversarial Networks (GANs) to enable humanoid	The complexity of implementing PPO and GANs may require substantial computational resources and expertise.

		robots to autonomously learn and select optimal gait patterns for different environments. The approach reduces the need for manual retraining and allows robots to adapt to various situations effectively.	The performance of the approach in highly dynamic or unstructured environments may need further validation.
[41]	2023	Utilizes a Bi-Channel Q-Value Evaluation with MADDPG (Multi-Agent Deep Deterministic Policy Gradient) for cooperative offensive decision-making in soccer robots. This approach enhances the ability of robots to make effective decisions in both discrete and continuous action spaces. The inclusion of a shooting angle reward helps to improve the scoring rate. The method is shown to be robust and scalable based on simulation results.	The approach may require sophisticated implementation and tuning of the MADDPG algorithm. The performance in real-world scenarios and the efficiency of the shooting angle reward need further investigation. The complexity of handling both discrete and continuous action parameters might pose challenges in practical applications.
[35]	2024	Utilizes Deep Q-Network with real-time sentiment feedback delivered through the trainer's speech to enhance cooperative robot learning. The system includes a novel reward function incorporating sentiment feedback and improved reward shaping techniques. This approach allows robots to learn interactive tasks more effectively and engage in natural human-robot interactions.	The implementation of real-time sentiment feedback may require advanced natural language processing capabilities. The effectiveness of the approach in various real-world scenarios and its adaptability to different types of tasks and feedback needs further exploration. The method's dependency on real-time interaction might present practical challenges in dynamic environments.
[44]	2024	Comprehensive survey of Multi-Agent Reinforcement Learning (MARL), including its historical evolution, challenges, applications, and benchmark environments. Highlights MARL's adaptability and relevance in various fields such as intelligent machines, chemical engineering, and healthcare.	Addresses the complexity of hyper parameter tuning and computational requirements. The survey indicates the need for continued research to overcome practical challenges and explore new applications.

2-3- Key Algorithms for Problem Solving

To further strengthen the foundation of our research, it is essential to explore the algorithms that have gained significant attention in the domains of task partitioning, scheduling, obstacle avoidance, and navigation in RoboSoccer. In this study, we focus on the algorithms that are most relevant to these areas and have demonstrated substantial effectiveness in addressing the challenges within these fields. Given the complexity and dynamic nature of the problems at hand, the selection of algorithms for experimentation is crucial. For clarity and better understanding, these algorithms are divided into two main categories: advanced reinforcement learning algorithms and classical algorithms. This categorization not only reflects the diversity of approaches but also highlights the evolution of methods from traditional strategies to more sophisticated, learning-based techniques. The reinforcement learning algorithms considered include:

1- **Deep Q-Network (DQN):** DQN is a reinforcement learning algorithm that utilizes deep neural networks to estimate the Q-value function. By leveraging deep neural networks to predict Q-values, DQN is capable of learning optimal policies in environments with high-dimensional state spaces. The accuracy of Q-value estimates is enhanced using techniques such as experience replay and target network stabilization [49].

2- **Double DQN (DDQN):** Double DQN addresses the overestimation bias in Q-value predictions by employing two separate neural networks: one for action selection and the other for value estimation. This separation enhances the accuracy of value predictions and improves the learning process [50].
3- **Prioritized Experience Replay (PER):** PER is an advanced technique that prioritizes the replay of experiences with larger prediction errors, thus accelerating the learning process and improving performance by focusing on more informative experiences [51].
4- **Soft Actor-Critic (SAC):** SAC is an algorithm that employs soft value functions and stochastic policies for optimal learning in continuous action spaces. By concurrently learning both the policy and the value function, SAC enhances the stability and efficiency of the learning process [52].
5- **Trust Region Policy Optimization (TRPO):** TRPO is a policy optimization algorithm that ensures safe and reliable updates to the policy by imposing trust region constraints [53]. This method prevents drastic policy changes, thereby contributing to more stable learning in robotic navigation [54].
6- **Proximal Policy Optimization (PPO):** PPO improves upon traditional policy optimization methods by simplifying the update process and incorporating constraints, resulting in increased stability and efficiency in learning [55].
7- **Deep Deterministic Policy Gradient (DDPG):** DDPG is tailored for continuous action spaces, using neural networks to learn both the policy and the value function [56]. This algorithm is particularly well-suited for problems involving continuous actions [57].
8- **Rainbow DQN (RDQN):** Rainbow DQN integrates various improvements on the standard DQN, including action distribution, prioritized experience replay, and other techniques. This combination of enhancements significantly boosts the performance of the DQN algorithm [58].
9- **Asynchronous Advantage Actor-Critic (A3C):** A3C facilitates parallel learning across multiple agents, offering substantial improvements in the efficiency and stability of policy and value function learning through asynchronous updates [59]. This method accelerates the learning process and enhances robustness [60].

These algorithms were chosen for their widespread use and effectiveness in research, making them ideal for the experimental evaluation of segmentation and scheduling problems in mapping to robotic Soccer environments.
The second category comprises classical algorithms, which are widely utilized in resource allocation and scheduling problems and frequently employed by researchers. These algorithms include:

1- **First-Come, First-Served (FCFS):** FCFS is the simplest scheduling method, operating based on the order in which tasks arrive in the system. Tasks that enter the system earlier are executed first. This algorithm does not prioritize tasks, treating all with equal importance regardless of their specific requirements. However, it offers specialized applications tailored to the specific conditions of the problem at hand [61].
2- **Round Robin (RR):** The Round Robin algorithm executes each task for a fixed time slice (quantum) before moving on to the next task. Once the time slice is completed, the task is returned to the end of the queue, where it awaits its next turn for execution [62].
3- **Shortest Job Next (SJN):** Also known as Shortest Job First (SJF), this algorithm prioritizes tasks that require the least execution time. By focusing on the shortest tasks first, SJN aims to minimize the average waiting time for all tasks in the system [63].
4- **Priority Scheduling (PS):** In Priority Scheduling, tasks are executed based on their assigned priorities. Tasks with higher priorities are scheduled before those with lower priorities. These priorities can be manually assigned or automatically determined by the system based on specific criteria [64].
5- **Earliest Deadline First (EDF):** EDF schedules tasks according to their deadlines, with tasks having the nearest deadlines being executed first. This algorithm is particularly useful in real-time systems where meeting deadlines is critical [65].
6- **Shortest Remaining Time (SRT):** SRT operates similarly to SJN but focuses on the remaining execution time rather than the total execution time. Tasks with the shortest remaining time are given priority, ensuring that those closest to completion are finished first [66].

7- **Multilevel Queue Scheduling (MQS):** This algorithm divides tasks into multiple queues, each governed by a different scheduling policy. Tasks within each queue are scheduled according to the queue's specific policy, while the queues themselves are prioritized based on overall system needs [67].
8- **Multilevel Feedback Queue (MFQ):** This dynamic algorithm allows tasks to move between queues based on their behavior and actual processing time. It adapts to task requirements, providing a flexible and responsive scheduling strategy [68].
9- **Rate-Monotonic Scheduling (RMS):** RMS is designed for real-time systems, scheduling tasks based on their periodicity. Tasks with higher rates (more frequent repetitions) are given higher priority, ensuring that critical tasks are executed on time [69].
10- **Lottery Scheduling (LS):** This probabilistic algorithm assigns lottery tickets to tasks, with the task to be executed next chosen randomly based on the number of tickets it holds. Lottery Scheduling helps in fairly distributing resources among tasks, providing an equitable system for task execution [70].

To provide a clearer understanding based on the review in this literature, the applications of each of these algorithms within the dual domains discussed in this paper are illustrated in Tables 3 and 4. These tables demonstrate the relevance and effectiveness of each algorithm in the specific contexts under consideration.

Table 3: The Applications of First Category within the Dual Domains

Algorithm	Applications in Task Scheduling and Resource Allocation	Applications in Obstacle Avoidance and Navigation in RoboSoccer
Deep Q-Network (DQN)	Solving complex scheduling and task partitioning problems.	Training robots to navigate and avoid obstacles in cluttered environments.
	Learning optimal resource allocation policies.	Optimizing movement paths.
	Managing job queues.	
Double DQN (DDQN)	Enhancing resource allocation and scheduling through reduced overestimation.	Improving obstacle avoidance and navigation strategies with reduced action value overestimation.
	Learning better task partitioning policies.	
Prioritized Experience Replay (PER)	Accelerating learning in resource allocation and task management by prioritizing important experiences.	Focusing on key experiences to improve obstacle avoidance and navigation in complex environments.
	Optimizing task scheduling.	
Soft Actor-Critic (SAC)	Optimizing task management and resource allocation in dynamic environments.	Enhancing flexibility in navigation and obstacle avoidance in varying environments.
	Learning soft policies for improved scheduling.	Adapting to changes.
Trust Region Policy Optimization (TRPO)	Learning stable policies for complex resource scheduling and task management.	Developing stable and effective navigation strategies for dynamic and variable environments.
	Ensuring reliable policy updates for scheduling.	
Proximal Policy Optimization (PPO)	Efficiently learning policies for scheduling and resource allocation.	Enhancing navigation and obstacle avoidance through stable policy updates.
	Utilizing stable updates for task management.	Adapting to dynamic changes.
Deep Deterministic Policy Gradient (DDPG)	Managing continuous action spaces in resource allocation and scheduling.	Optimizing continuous path adjustments and obstacle avoidance in complex environments.
	Achieving precise task management.	Fine-tuning movement strategies.

Rainbow DQN (RDQN)	Improving scheduling and task partitioning with advanced DQN techniques.	Enhancing navigation and obstacle avoidance with advanced techniques in DQN.
	Optimizing resource allocation using integrated features.	Adapting to complex environments.
Asynchronous Advantage Actor-Critic (A3C)	Accelerating learning of resource scheduling policies through parallel updates.	Rapidly learning optimal navigation and obstacle avoidance strategies through concurrent updates.
	Enhancing task management with simultaneous learning.	Adapting to dynamic conditions.

Table 4: The applications of second category within the dual domains

Algorithm	**Applications in Task Scheduling and Resource Allocation**	**Applications in Obstacle Avoidance and Navigation in RoboSoccer**
First-Come, First-Served (FCFS)	Useful for environments requiring simple task management.	Not suitable for dynamic environments where quick response to changes is needed.
	Easy to implement and understand.	Limited in prioritizing tasks.
Round Robin (RR)	Suitable for multi-tasking environments requiring fair distribution of processing time.	Can manage robot movement timings but may not handle rapid environmental changes effectively.
	Good for managing time slices.	
Shortest Job Next (SJN)	Optimizes response time and reduces waiting time in predictable environments.	Useful in environments where tasks have predictable durations, less suitable for dynamic scenarios.
	Effective for tasks with known execution times.	
Priority Scheduling (PS)	Helps in allocating resources to high-priority tasks.	Effective for addressing critical tasks but requires careful priority management.
	Suitable for environments needing priority management.	
Earliest Deadline First (EDF)	Ideal for real-time systems needing timely execution of tasks.	Helps robots make timely decisions near obstacles, choosing appropriate strategies at critical times.
	Ensures deadlines are met.	
Shortest Remaining Time (SRT)	Optimizes response times in environments where remaining times are variable.	Effective in managing tasks with changing time requirements, optimizing robot movement management.
	Useful for tasks with fluctuating durations.	
Multilevel Queue Scheduling (MQS)	Manages tasks with varying characteristics and scheduling needs.	Helps manage tasks with different priorities and navigation strategies.
	Suitable for multi-tasking environments.	
Multilevel Feedback Queue (MFQ)	Optimizes scheduling and increases system efficiency.	Useful for managing tasks with variable behaviors and requirements in robotics.
	Responds to tasks with varying needs and behaviors.	
Rate-Monotonic Scheduling (RMS)	Ideal for real-time systems with tasks of varying frequencies.	Helps manage tasks requiring frequent updates or actions, such as avoiding obstacles frequently.
	Manages tasks based on their repetition rates.	
Lottery	Helps in fair distribution of	Assists in fairly distributing time for

Scheduling (LS)	resources between tasks.	robot tasks but may be less effective for complex navigation strategies.
	Suitable for environments needing equitable resource allocation.	

The algorithms detailed in Tables 3 and 4 highlight a comprehensive spectrum of approaches to task scheduling, resource allocation, and robotic navigation, each offering unique strengths and applications. The advanced reinforcement learning algorithms, as illustrated in Table 3, exemplify cutting-edge techniques designed to handle complex, dynamic environments with enhanced precision and adaptability. These algorithms are instrumental in advancing robotic capabilities through improved learning efficiencies and sophisticated decision-making processes. Conversely, the classical algorithms presented in Table 4 offer time-tested methodologies for managing task scheduling and resource allocation. While these algorithms may seem simplistic compared to their modern counterparts, they provide fundamental principles that remain relevant in less dynamic or more predictable contexts. These methods are particularly effective in scenarios where tasks are well-defined and require straightforward management, demonstrating their enduring value in specific applications. The juxtaposition of these two categories underscores the evolution from traditional scheduling strategies to contemporary reinforcement learning approaches, reflecting the ongoing advancements in algorithmic design. As we integrate these diverse methods into practical systems, the insights gained from both classical and advanced algorithms will be crucial in addressing the complexities of modern robotics and achieving optimal performance in dynamic environments. This dual approach not only bridges historical and modern computational strategies but also lays a robust foundation for future innovations in robotic systems and intelligent automation.

3- **Proposed Method and Implementation Details**

Our proposed method addresses the problem of task partitioning and scheduling within the context of navigation and obstacle avoidance for soccer-playing robots, utilizing reinforcement learning techniques. To achieve this, it is crucial to conduct a thorough examination and analysis of the features of both datasets to establish an appropriate correspondence between the features and definitions for obstacles and goals. In essence, this requires meticulous feature engineering. This process necessitates a comprehensive understanding of both the problem domain and the datasets involved. Building upon the prior work of the authors [9] and a detailed review of relevant literature, as discussed in the previous section, we proceed to the next steps. Consequently, we must first ascertain the features contained within our datasets. The following section introduces the two datasets utilized in this study.

3-1- Datasets Utilized

To evaluate and analyze the effectiveness of the proposed method, two datasets were employed. These datasets include data on task scheduling and partitioning, as well as data on obstacle avoidance and navigation in soccer robots. The details of each dataset are described below.

3-1-1- Task Scheduling and Partitioning Dataset

This dataset is sourced from **Kaggle** and is available here. It consists of a trace of workloads executed on eight Google Borg scheduling programs in May 2019. The trace includes job submissions, scheduling decisions, and resource usage data for the jobs run within these programs. The dataset can be categorized into the following domains:

1- Cloud Computing & CloudSim
2- Job & Task Scheduling
3- CPU Usage, Memory Usage Information & Resource Usage Data
4- Scheduling Decisions
5- Job Submissions

The dataset encompasses 32 primary features across 405,894 samples. These features are as follows:

1- time: The timestamp of each record, represented in microseconds.
2- instance_events_type: The type of events occurring in the instances.
3- collection_id: Identifier for collections, assigned to groups of samples or jobs.
4- scheduling_class: Classification of job scheduling.
5- collection_type: Type of collection, indicating differences between collections.

6- priority: Relative priority or urgency of jobs.
7- alloc_collection_id: Identifier for resource allocation collections.
8- instance_index: Index or position of an instance within a collection.
9- machine_id: Identifier for individual machines in the system.
10- resource_request: Specifications or requirements for resources for a job or task.
11- constraint: Constraints or conditions on resource allocation.
12- collections_events_type: Type of events occurring at the collection level.
13- user: User or entity responsible for job submission.
14- collection_name: Name of the collection.
15- collection_logical_name: Logical or symbolic name of the collection.
16- start_after_collection_ids: Identifiers for dependencies or conditions for starting the collection.
17- vertical_scaling: Scaling of computational resources to improve performance.
18- scheduler: Scheduler or scheduling algorithm used.
19- start_time: Start time of the event, job, or task.
20- end_time: End time of the event, job, or task.
21- average_usage: Average resource usage.
22- maximum_usage: Maximum resource usage.
23- random_sample_usage: Resource usage data for randomly sampled instances.
24- assigned_memory: Amount of memory allocated.
25- page_cache_memory: Memory used for page caching.
26- cycles_per_instruction: Average number of processor cycles per instruction.
27- memory_accesses_per_instruction: Average number of memory accesses per instruction.
28- sample_rate: Sampling rate.
29- cpu_usage_distribution: Distribution of CPU usage.
30- tail_cpu_usage_distribution: Distribution of tail-end CPU usage.
31- event: Event or action occurring within the system.
32- failed: Indicator of success or failure within the system.

After data collection, two primary stages of data analysis are Data Cleaning and Exploratory Data Analysis (EDA). Data Cleaning addresses issues such as missing data, outliers, inconsistencies, duplicate entries, irrelevant variables, and managing imbalanced data. EDA involves understanding the initial data structure, size, and types, calculating descriptive statistics (mean, variance, median, etc.), visualizing the data, and identifying patterns and trends. Since one of the core approaches of this paper is precise feature engineering, extensive operations were performed on the task scheduling and partitioning dataset from Kaggle for data cleaning and EDA. The results, including relevant pre- and post-processing charts, visualization of EDA, output cleaning file (final dataset), and source codes are available in the **GitHub repository** provided here. Additional explanations for researchers are included in the appendix. Based on this extensive analysis and careful review of related literature (some of which are mentioned in the research background section), we determined that not all features were necessary for our small-scale implementation of task scheduling and partitioning. The essential features identified were:

1- **Task ID:** A unique identifier for each task.
2- **Task Priority:** The importance and priority of each task as determined by system requirements.
3- **Arrival Time:** The time when the task enters the system.
4- **Completion Time:** The time when the task is completed.
5- **Waiting Time:** The duration the task waits before processing begins.
6- **Processing Time:** The time required to complete the task.
7- **Resource Requirements:** The type and amount of resources needed for task execution (e.g., processors, memory, and storage).
8- **Task Status:** The current state of the task (e.g., waiting, processing, completed).
9- **Delay:** The difference between the predicted completion time and the actual completion time.

While many required features could be directly extracted from the Kaggle dataset, certain features such as Arrival Time, Waiting Time, and Delay necessitated additional processing or indirect estimation. Other features like Task Priority, Completion Time, and Resource Requirements were directly aligned with the available data. The processing steps for each feature are as follows:

1- **Task ID:** The features collection_id and instance_index from the Kaggle dataset are used as unique identifiers for tasks or jobs.
2- **Task Priority:** The priority feature in Kaggle represents the relative importance or urgency assigned to a task, making it suitable for use as Task Priority.
3- **Arrival Time:** This feature is not explicitly present in the Kaggle dataset. However, it can be estimated using the time feature, which includes various timestamps of events as they occur in the system.
4- **Completion Time:** The end_time feature in Kaggle indicates the time at which a task is completed and is used directly for this purpose.
5- **Waiting Time:** This feature is not directly available in the Kaggle dataset. It is calculated as the difference between the estimated Arrival Time and the start_time.
6- **Processing Time:** This feature is not directly provided in the dataset. It can be derived by calculating the difference between start_time and end_time.
7- **Resource Requirements:** The resource_request, assigned_memory, and page_cache_memory features in Kaggle provide information regarding the resources required for task execution.
8- **Task Status:** The features failed, instance_events_type, and event in Kaggle can be used to analyze the task status (e.g., successful, waiting, processing).
9- **Delay:** This feature is not explicitly available in the dataset. It is computed by comparing the actual end_time with the predicted end time, which may require additional processing.

3-1-2- Dataset for Obstacle Avoidance and Navigation

To achieve precise engineering of the features and attain the objectives of this paper, it is essential to thoroughly understand the problem definition of the Soccer-playing robot, specifically with respect to obstacle avoidance and navigation, as well as the associated dataset. The problem definition has already been comprehensively addressed in a previous study by the authors of this paper [9]. Consequently, in this section, our focus will shift to the dataset available on the **Kaggle** platform, which is openly accessible to researchers and scholars through the link provided here. The dataset comprises data from simulations created using robotic simulation software capable of generating dynamic environments and realistic conditions. The data is extracted from various experiments conducted in simulation environments with diverse obstacles and complexities. This dataset is specifically designed for evaluating obstacle avoidance and navigation algorithms under different conditions. Given the previous successful experience of the authors with robot soccer and obstacle avoidance issues [9], not all features of this dataset were used. The selected features were chosen to provide a deeper analysis of reinforcement learning algorithms' performance in the presence of dynamic obstacles and environmental changes, while also enhancing algorithm convergence speed. The features included are:

1- **Path ID:** A unique identifier for each navigation path.
2- **Obstacle Position:** The spatial location of obstacles in the simulation environment (x, y, and z coordinates).
3- **Obstacle Type:** The type of obstacle (e.g., wall, moving object, or other obstacles).
4- **Robot Movement Type**: The type of robot movement (e.g., direct movement, rotation, or a combination).
5- **Navigation Time:** The total time required to traverse the path.
6- **Navigation Start Time:** The exact time when the robot begins navigating the path.
7- **Navigation End Time:** The exact time when the robot completes the navigation and reaches the goal.
8- **Delay Time:** The duration of any delay in executing the navigation relative to the scheduled time.
9- **Navigation Requirements:** The amount and type of resources required for robot navigation.
10- **Obstacle Avoidance Success Rate:** The percentage of successful obstacle avoidance by the robot.
11- **Distance Traveled:** The total distance covered by the robot.
12- **Navigation Error:** The positional error relative to the target upon reaching the end of the navigation.

13- **Final Status:** The final status of the robot (e.g., successfully reached the goal, collision with obstacle, or other issues).

3-2- Specific Preprocessing of Data

After collecting and understanding the dataset, preprocessing is crucial for improving data quality for accurate modeling and evaluation. In this study, preprocessing was performed twice: a general preprocessing covered earlier and a specialized preprocessing detailed here. The following sections outline the specific preprocessing steps applied to both datasets used in this study.

3-2-1- Preprocessing of Task Scheduling and Assignment Dataset

1- **Removal of Incomplete Data:** Records with missing values for key features such as arrival time, completion time, or processing time were removed. Incomplete data can lead to distorted model results and incorrect analyses.
2- **Correction of Invalid and Outlier Values:** Incorrect and unrealistic values (e.g., negative times or values outside acceptable ranges) were corrected. Invalid values can cause model inaccuracies and unreliable results.
3- **Scaling:** Numerical features such as processing time and waiting time were standardized to a z-score. This step reduces the impact of large or small values on modeling, helping to ensure a balanced influence of all features.
4- **Normalization:** Features were scaled to the [0,1] range to minimize differences between features with varying ranges. Normalization allows the model to handle features with different ranges more effectively.
5- **Principal Component Analysis (PCA):** PCA was used to reduce feature dimensions, identifying the principal features with the highest variance. Dimensionality reduction simplifies the model and improves its performance.
6- **Data Segmentation:** Task scheduling data was segmented into different categories based on type and resource requirements. This segmentation enhances modeling accuracy and analysis by grouping similar data.
7- **Encoding:** Categorical features were encoded numerically using binary encoding or One-Hot Encoding. This encoding facilitates better processing of categorical data by the model.
8- **Removal of Duplicate Data:** Duplicate records were removed to prevent their negative impact on the model. Duplicates can lead to biased results and an overemphasis on certain samples.
9- **Removal of Irrelevant Variables:** Features deemed irrelevant for analysis and modeling were removed. Eliminating irrelevant features reduces model complexity and increases its accuracy.
10- **Management of Imbalanced Data:** Techniques such as Oversampling and Undersampling were used to balance the data. Imbalanced data can lead to modeling discrepancies and incorrect results.

It is crucial to briefly examine the status of the data before and after preprocessing, given that feature engineering is a primary focus of this study. This status is illustrated in Tables 5 and 6.

Table 5: Example of Feature Status in the Task Scheduling and Assignment Dataset before Preprocessing

Task ID	Task Priority	Arrival Time	Completion Time	Waiting Time	Processing Time	Resource Requirements	Task Status	Delay
1	2	12:00	12:40	10	42	Storage, GPU	Waiting	8
2	4	12:10	12:50	12	24	CPU, GPU	Waiting	6
3	1	12:23	13:03	13	20	CPU, Storage	Completed	-1
4	1	12:33	13:13	8	20	RAM, Storage	In Process	-1
5	1	12:47	13:22	5	38	CPU, RAM	Waiting	5
6	1	13:01	13:38	5	49	CPU, RAM	Completed	1
7	3	13:16	13:57	14	21	CPU, GPU	Waiting	0
8	1	13:28	14:12	8	39	RAM, Storage	In Process	3
9	4	13:39	14:28	11	45	Storage, GPU	Completed	4
10	5	13:55	14:50	6	36	RAM, Storage	In Process	2

11	1	14:11	14:48	7	23	CPU, RAM	Waiting	7
12	3	14:23	15:05	5	35	CPU, Storage	In Process	4
13	3	14:38	15:21	9	28	RAM, Storage	Completed	1
14	1	14:51	15:45	5	33	CPU, GPU	Waiting	0
15	5	15:05	16:05	12	20	RAM, Storage	Completed	3
16	1	15:19	16:12	5	45	CPU, Storage	Waiting	-2
17	3	15:30	16:30	5	49	RAM, Storage	In Process	-4
18	2	15:42	16:35	6	25	Storage, GPU	Waiting	5
19	4	15:56	16:55	6	44	RAM, Storage	Completed	2
20	3	16:12	17:00	10	21	CPU, Storage	Waiting	4

Table 6: Example of Feature Status in the Task Scheduling and Assignment Dataset after Preprocessing

Task ID	Task Priority (Standardized)	Arrival Time (Normalized)	Completion Time (Normalized)	Waiting Time (Normalized)	Processing Time (Normalized)	Resource Requirements (PCA)	Task Status (Encoding)	Delay (Standardized)
1	0.01	0.200	0.600	0.555556	0.793103	2.5	1 (Waiting)	1.40
2	1.47	0.233	0.633	0.777778	0.172414	2.0	1 (Waiting)	1.02
3	-1.45	0.267	0.667	0.888889	0.000000	2.5	0 (Completed)	-0.22
4	-1.45	0.300	0.700	0.333333	0.000000	3.0	2 (In Process)	-0.22
5	-1.45	0.333	0.733	0.000000	0.620690	1.5	1 (Waiting)	0.66
6	-1.45	0.367	0.767	0.000000	1.000000	1.5	0 (Completed)	0.18
7	0.01	0.400	0.800	1.000000	0.206897	2.0	1 (Waiting)	0.00
8	-1.45	0.433	0.833	0.333333	0.655172	3.0	2 (In Process)	0.62
9	1.47	0.467	0.867	0.666667	0.965517	3.5	0 (Completed)	0.78
10	2.93	0.500	0.900	0.111111	0.586207	3.0	2 (In Process)	0.44
11	-1.45	0.533	0.933	0.222222	0.103448	1.5	1 (Waiting)	1.14
12	0.01	0.567	0.967	0.000000	0.551724	2.5	2 (In Process)	0.78
13	0.01	0.600	1.000	0.444444	0.310345	3.0	0 (Completed)	0.18
14	-1.45	0.633	1.033	0.000000	0.448276	2.0	1 (Waiting)	0.00
15	2.93	0.667	1.067	0.777778	0.000000	3.0	0 (Completed)	0.62
16	-1.45	0.700	1.100	0.000000	0.965517	2.5	1 (Waiting)	-0.38
17	0.01	0.733	1.133	0.000000	1.000000	3.0	2 (In Process)	-0.78
18	0.01	0.767	1.167	0.111111	0.172414	3.5	1 (Waiting)	0.66
19	1.47	0.800	1.200	0.111111	0.931034	3.0	0 (Completed)	0.44
20	0.01	0.833	1.233	0.555556	0.206897	2.5	1 (Waiting)	0.00

In Table 4, the values within certain columns warrant further explanation, which are elaborated upon as follows:

1- **Delay (Standardized):**

The "Delay (Standardized)" feature quantifies the deviation of task completion times from their expected deadlines. A negative value in this column indicates that the task was completed ahead of schedule. For instance, consider Task A, which is expected to be completed at 14:00, but the actual completion occurs at 13:50. The delay is calculated as follows:

Delay=Actual Completion Time−Expected Completion Time

Delay=13:50−14:00=-10 minutes

Thus, a negative delay value reflects the task's earlier-than-anticipated completion, suggesting enhanced efficiency, reduced wait times, and a performance that exceeds initial expectations.

2- **Resource Requirements (PCA):**

The "Resource Requirements (PCA)" feature represents the resource demands of each task after applying Principal Component Analysis (PCA). This feature is computed through the following steps:

- Conversion of Textual Data to Numerical Values: Initially, textual data pertaining to resources are encoded into numerical values. For example, CPU = 1, RAM = 2, GPU = 3, and Storage = 4. Subsequently, the combination of resources required for each task is represented as the sum of these values. For example, if a task requires both CPU and GPU, its corresponding value will be 1+3=4.
- Construction of Resource Data Matrix: A matrix is created where each row represents a task, and each column represents a type of resource. The numerical values from the previous step are populated in this matrix.
- Summation of Resource Values: The numerical values for each task are summed to yield a single value representing the total resource requirements.
- Execution of PCA: Principal Component Analysis (PCA) is applied to the aggregate resource values. PCA serves to reduce the dimensionality of the data while capturing the most significant variations. This analysis provides the new values for the "Resource Requirements (PCA)" column.
- Updating the Column with PCA Results: The original resource requirement values are replaced with the new values derived from PCA, thereby reflecting the influence of principal components in the data.

3- Task Priority (Standardized):

The "Task Priority (Standardized)" feature indicates the priority of tasks following a standardization process. This process transforms the priority values so that they exhibit a mean of zero and a standard deviation of one. Consequently, negative values in this column represent tasks with priorities lower than the mean, while positive values denote tasks with priorities higher than the mean. For example, if a task initially assigned a priority of 1 is standardized to -1.45, it signifies that this task has a lower priority relative to others. This standardization is employed to facilitate more precise comparisons and analyses of task priorities across the entire dataset.

These detailed explanations aim to provide readers with a deeper understanding of how the data was processed and how the results should be interpreted.

3-2-2- Preprocessing the Obstacle Avoidance and Navigation Dataset

The preprocessing steps for the obstacle avoidance and navigation dataset follow the same approach as outlined for the task scheduling and assignment dataset. Therefore, details of the preprocessing operations are not repeated here. The status of the data before and after preprocessing is presented in Tables 4 and 5. Due to the extensive number of features and their detailed descriptions, incorporating the full explanations in the headers of Tables 4 and 5 would result in tables that are excessively large, thereby diminishing their readability. This would hinder the ability to quickly comprehend the contents of these tables at a glance. Therefore, Tables 7 and 8 will be presented in a manner distinct from Tables 5 and 6, with features listed as Features 1 through 13 in the header to ensure clarity and facilitate ease of reference.

Table 7: Example of Feature Status in the Obstacle Avoidance and Navigation Dataset before Preprocessing

Feature Number												
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	(5, 10, 2)	Wall	Direct	15	08:00	08:15	0	Medium	85%	200m	1.5m	Success
2	(12, 5, 0)	Moving Object	Turning	20	08:10	08:30	2	High	75%	180m	2.0m	Failure
3	(7, 8, 3)	Wall	Combined	18	08:05	08:23	1	High	90%	210m	1.0m	Success
4	(6, 9, 1)	Barrier	Direct	17	08:12	08:29	1	Medium	80%	190m	1.8m	Success
5	(11, 7, 2)	Wall	Turning	22	08:15	08:37	3	Low	70%	185m	2.2m	Failure
6	(8, 6, 4)	Wall	Direct	16	08:20	08:36	0	Medium	88%	205m	1.6m	Success

7	(9, 11, 2)	Moving Object	Turning	19	08:25	08:44	2	High	78%	190m	2.1m	Failure
8	(4, 7, 1)	Barrier	Combined	14	08:18	08:32	1	High	85%	195m	1.2m	Success
9	(5, 10, 3)	Wall	Direct	21	08:30	08:51	1	Medium	82%	210m	1.7m	Success
10	(10, 5, 1)	Moving Object	Turning	24	08:35	08:59	3	Low	65%	185m	2.4m	Failure
11	(6, 8, 4)	Wall	Direct	18	08:15	08:33	0	Medium	90%	200m	1.3m	Success
12	(12, 6, 2)	Moving Object	Turning	20	08:40	09:00	2	High	77%	190m	2.2m	Failure
13	(7, 9, 3)	Wall	Combined	17	08:45	09:02	1	High	88%	205m	1.5m	Success
14	(8, 10, 1)	Barrier	Direct	16	08:50	09:06	1	Medium	83%	190m	1.8m	Success
15	(11, 7, 2)	Wall	Turning	22	08:55	09:17	3	Low	72%	185m	2.3m	Failure
16	(8, 6, 4)	Wall	Direct	18	09:00	09:18	0	Medium	87%	205m	1.6m	Success
17	(9, 11, 2)	Moving Object	Turning	19	09:05	09:24	2	High	79%	190m	2.1m	Failure
18	(4, 7, 1)	Barrier	Combined	14	09:10	09:24	1	High	86%	195m	1.2m	Success
19	(5, 10, 3)	Wall	Direct	21	09:15	09:36	1	Medium	81%	210m	1.7m	Success
20	(10, 5, 1)	Moving Object	Turning	24	09:20	09:44	3	Low	66%	185m	2.4m	Failure

In the table presented, the features are enumerated in the following sequence:

1- **Feature 1:** Path ID
2- **Feature 2:** Obstacle Position (x, y, z)
3- **Feature 3:** Obstacle Type
4- **Feature 4:** Robot Movement Type
5- **Feature 5:** Navigation Time
6- **Feature 6:** Navigation Start Time
7- **Feature 7:** Navigation End Time
8- **Feature 8:** Delay Time
9- **Feature 9:** Navigation Requirements
10- **Feature 10:** Obstacle Avoidance Success Rate
11- **Feature 11:** Distance Traveled
12- **Feature 12:** Navigation Error
13- **Feature 13:** Final Status

Table 8: Example of Feature Status in the Obstacle Avoidance and Navigation Dataset after Preprocessing

Feature Number												
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
1	(0.3, 0.5, 0.1)	1 (Wall)	0 (Direct)	0.6	0	0.5	0	1 (Medium)	0.8	0.7	0.5	1 (Success)
2	(0.7, 0.2, 0.0)	2 (Moving Object)	1 (Turning)	0.8	0.2	0.8	1	2 (High)	0.6	0.6	0.7	0 (Failure)
3	(0.4, 0.4, 0.2)	1 (Wall)	2 (Combined)	0.7	0.1	0.65	0.5	2 (High)	0.9	0.8	0.4	1 (Success)
4	(0.5, 0.6, 0.1)	3 (Barrier)	0 (Direct)	0.65	0.15	0.72	0.5	1 (Medium)	0.7	0.65	0.6	1 (Success)
5	(0.6, 0.4, 0.2)	1 (Wall)	1 (Turning)	0.75	0.25	0.85	1.5	0 (Low)	0.5	0.6	0.7	0 (Failure)
6	(0.4, 0.3, 0.3)	1 (Wall)	0 (Direct)	0.65	0.35	0.6	0	1 (Medium)	0.85	0.72	0.55	1 (Success)
7	(0.45, 0.55, 0.15)	2 (Moving Object)	1 (Turning)	0.75	0.4	0.7	1	2 (High)	0.65	0.68	0.65	Failure
8	(0.2, 0.35, 0.05)	3 (Barrier)	2 (Combined)	0.6	0.3	0.55	0.5	2 (High)	0.8	0.71	0.4	1 (Success)
9	(0.25, 0.5, 0.15)	1 (Wall)	0 (Direct)	0.82	0.5	0.95	0.5	1 (Medium)	0.75	0.8	0.58	1 (Success)
10	(0.5, 0.25, 0.05)	2 (Moving Object)	1 (Turning)	0.9	0.55	1	1.5	0 (Low)	0.55	0.7	0.8	0 (Failure)
11	(0.3, 0.4, 0.3)	1 (Wall)	0 (Direct)	0.7	0.25	0.68	0	1 (Medium)	0.9	0.7	0.48	1 (Success)
12	(0.7, 0.3, 0.15)	2 (Moving Object)	1 (Turning)	0.8	0.6	0.9	1	2 (High)	0.62	0.68	0.72	Failure
13	(0.35, 0.45, 0.2)	1 (Wall)	2 (Combined)	0.65	0.65	0.81	0.5	2 (High)	0.88	0.72	0.6	1 (Success)
14	(0.4, 0.5, 0.1)	3 (Barrier)	0 (Direct)	0.6	0.7	0.85	0.5	1 (Medium)	0.83	0.66	0.62	1 (Success)
15	(0.55, 0.35, 0.2)	1 (Wall)	1 (Turning)	0.75	0.75	1	1.5	0 (Low)	0.72	0.7	0.77	0 (Failure)
16	(0.4, 0.3, 0.3)	1 (Wall)	0 (Direct)	0.7	0.8	0.9	0	1 (Medium)	0.87	0.72	0.55	1 (Success)
17	(0.45, 0.55, 0.15)	2 (Moving Object)	1 (Turning)	0.75	0.85	1	1	2 (High)	0.79	0.68	0.65	0 (Failure)
18	(0.2, 0.35, 0.05)	3 (Barrier)	2 (Combined)	0.6	0.9	1	0.5	2 (High)	0.86	0.71	0.4	1 (Success)
19	(0.25, 0.5, 0.15)	1 (Wall)	0 (Direct)	0.82	0.95	1	0.5	1 (Medium)	0.81	0.8	0.58	1 (Success)
20	(0.5, 0.25, 0.05)	2 (Moving Object)	1 (Turning)	0.9	1	1	1.5	0 (Low)	0.66	0.7	0.8	0 (Failure)

In the provided table, the features are delineated as follows:

14- **Feature 1:** Path ID
15- **Feature 2:** Obstacle Position (x, y, z) (Normalized)
16- **Feature 3:** Obstacle Type (Encoded)
17- **Feature 4:** Robot Movement Type (Encoded)
18- **Feature 5:** Navigation Time (Normalized)
19- **Feature 6:** Navigation Start Time (Normalized)
20- **Feature 7:** Navigation End Time (Normalized)
21- **Feature 8:** Delay Time (Standardized)
22- **Feature 9:** Navigation Requirements (Encoded)

23- **Feature 10:** Obstacle Avoidance Success Rate (Standardized)
24- **Feature 11:** Distance Traveled (Normalized)
25- **Feature 12:** Navigation Error (Standardized)
26- **Feature 13:** Final Status (Encoded)

The data preprocessing procedures, as illustrated in Tables 4 and 5, enable us to refine the dataset, ensuring it is meticulously prepared for sophisticated analyses and advanced modeling. This preprocessing is crucial for optimizing data quality and standardizing it to meet the stringent requirements of machine learning algorithms and statistical analysis, thereby enhancing the reliability and accuracy of the outcomes.

3-3- **Implementation of the Proposed Methodology:**

The core of our approach involves establishing a precise correspondence between the features of the two datasets. The results of this alignment are detailed as follows:

1- **Task ID and Path ID:** The Task ID from the task scheduling dataset is aligned with the Path ID from the obstacle avoidance and navigation dataset. Both features signify a unique entity within the system, used for tracking and managing tasks or paths.
2- **Task Priority and Robot Movement Type:** The Task Priority in the task scheduling dataset can be associated with the Robot Movement Type in the navigation dataset. Higher priority tasks may necessitate more complex and rapid robot movements, potentially influencing the selection of movement types and navigation strategies.
3- **Arrival Time and Navigation Start Time:** The Arrival Time in the task scheduling dataset is linked to the Navigation Start Time in the navigation dataset. Both features represent the initiation of activities and can impact the overall system scheduling.
4- **Completion Time and Navigation End Time:** Completion Time in the task scheduling dataset corresponds to Navigation End Time in the navigation dataset. These features indicate the successful completion of activities and contribute to evaluating system efficiency.
5- **Waiting Time and Delay Time:** Waiting Time in the task scheduling dataset is associated with Delay Time in the navigation dataset. Both features reflect periods of waiting or delay within the system, potentially affecting overall system performance.
6- **Processing Time and Navigation Time:** Processing Time in the task scheduling dataset is linked to Navigation Time in the navigation dataset. These features represent the duration required to complete activities and are crucial for assessing system efficiency.
7- **Resource Requirements and Navigation Requirements:** Resource Requirements in the task scheduling dataset correspond to Navigation Requirements in the navigation dataset. Both features reflect the specific system needs for carrying out activities, impacting resource allocation and overall system performance.
8- **Task Status and Final Status:** Task Status in the task scheduling dataset is associated with the Final Status in the navigation dataset. Both features indicate the current or final state of activities, aiding in evaluating the system's success.
9- **Delay and Navigation Error:** Delay in task completion in the task scheduling dataset may correlate with Navigation Error in the obstacle avoidance dataset. Delays in task completion could lead to increased navigation errors due to reduced planning time for robots to navigate and avoid obstacles.
10- **Obstacle Position:** To align this feature, we can combine the Resource Requirements and Processing Time features: Resource Requirements + Processing Time. The combination of these two features can represent the complexity of tasks and the resources needed to complete them, potentially related to the obstacle positions in the simulated environment.
11- **Obstacle Type:** To align this feature, we can combine the Task Status and Task Priority features: Task Status + Task Priority. This combination can represent the type and importance of different tasks, potentially linked to the types of obstacles in the simulated environment.
12- **Obstacle Avoidance Success Rate:** To align this feature, we can combine the Processing Time and Delay features: Processing Time + Delay. The combination of these two features can represent the efficiency and success in task execution, potentially related to the success rate in avoiding obstacles.
13- **Distance Traveled:** To align this feature, we can combine the Processing Time and Waiting Time features: Processing Time + Waiting Time. The combination of these two features can represent the overall time spent and effort exerted, potentially linked to the distance traveled by robots.

These comprehensive mappings and associations between the datasets facilitate a coherent integration, ensuring that the system operates efficiently and effectively, with a clear understanding of the interplay between various system components. Table 9 presents the established correspondence between the features of the task scheduling and obstacle avoidance and navigation datasets.

Table 9: Mapping of Features between Task Scheduling and Obstacle Avoidance & Navigation Datasets

Features	**Task Scheduling Dataset**	**Obstacle Avoidance & Navigation Dataset**
Unique Identifier	Task ID	Path ID
Priority	Task Priority	Robot Movement Type
Start Time	Arrival Time	Navigation Start Time
End Time	Completion Time	Navigation End Time
Waiting Time	Waiting Time	Delay Time
Processing Time	Processing Time	Navigation Time
Resource Requirements	Resource Requirements	Navigation Requirements
Status	Task Status	Final Status
Delay	Delay	Navigation Error
Obstacle Position	Resource Requirements + Processing Time	Obstacle Position
Obstacle Type	Task Status + Task Priority	Obstacle Type
Obstacle Avoidance Success	Processing Time + Delay	Obstacle Avoidance Success Rate
Distance Traveled	Processing Time + Waiting Time	Distance Traveled

This table effectively demonstrates how the features from the task scheduling dataset are aligned with those in the obstacle avoidance and navigation dataset, ensuring a coherent and structured integration between the two datasets. When considering the four derived features—obstacle position, obstacle type, obstacle avoidance success rate, and distance traveled—readers may question whether their combination introduces redundancy into the data. However, if the feature combination is performed judiciously, it should not result in unnecessary redundancy. Here are several justifications for this assertion:

- **Purpose of Feature Combination:** The primary goal of combining features is to create a more comprehensive representation of the data, rather than to introduce redundancy. The intent is to derive new features that significantly enhance the correlation and analysis between different datasets.
- **Enhancement of Analytical Accuracy:** Combining features can improve the precision of the analysis by extracting additional information from the existing features. For instance, combining processing time and wait time can provide a more accurate representation of the total time consumed for a given task.
- **Logical Relationships between Features:** The combination of features is typically based on logical relationships. These relationships are usually grounded in well-established theories and hypotheses relevant to the specific domain. For example, combining required resources with processing time to reflect task complexity is a logically sound approach.
- **Avoidance of Unnecessary Redundancy:** When done thoughtfully, feature combination yields meaningful and useful information rather than introducing unnecessary redundancy. The combination should be designed to add analytical value and avoid meaningless data repetition.
- **Reduction of Multicollinearity:** In cases where features independently contribute little value, their combination can reduce multicollinearity, thereby enhancing the accuracy of statistical models and machine learning algorithms.
- **Improved Interpretability:** Feature combination allows researchers to interpret analytical results more effectively. For instance, combining task priority with robot movement type can provide deeper insights into task types and their execution methods.

- **Adherence to Standard Practices:** Feature combination is a widely accepted and standard method in data analysis, utilized across many scientific and industrial fields. It is commonly employed to enhance model performance and result interpretation.
- **Practical Justifications:**
  Processing Time and Routing Time: Combining these two features helps in understanding the total time required for task completion and how this time impacts the robot's navigation and routing.
  Required Resources and Obstacle Avoidance Success Rate: By combining these features, one can assess how resources contribute to obstacle avoidance and how optimal resource allocation can enhance the success rate.

Thus, when features are combined thoughtfully and logically, they do not create redundancy; rather, they contribute to more accurate analyses and improved model performance. This approach allows researchers and analysts to consider more comprehensive and meaningful information, leading to better results and more precise data interpretation. The alignment between the features of task scheduling and obstacle avoidance datasets demonstrates that task scheduling problems can effectively be mapped onto the obstacle avoidance and navigation problem in a robotic soccer player, with practical benefits such as improved management of dynamic environments, better handling of complex agent interactions, knowledge transfer, and reuse of algorithms, and enhanced energy optimization and reinforcement learning.

3-4- Experimental Analysis

This section of the paper delves into the problem of segmentation and scheduling within the context of a robotic Soccer environment, considering scenarios both with and without features mapping operation. As we know, the primary objective of a Soccer-playing robot is to reach the goal within an optimal timeframe, employing the best strategies while ensuring obstacle avoidance and efficient navigation [9]. To achieve this, various methods are utilized, and depending on the specific problem definition, a range of criteria is applied to assess the success in reaching the desired outcome. The algorithms under evaluation are categorized into two primary groups: reinforcement learning algorithms and classical algorithms. The emphasis on reinforcement learning algorithms is justified by their demonstrated proficiency in addressing complex and dynamic environments, as previously mentioned, and their potential application in the authors' future work.
All algorithms tested, from both categories, are frequently used and represent state-of-the-art methods in the current research landscape. The reinforcement learning algorithms considered include:

- **Deep Q-Network (DQN)** is detailed in reference [49] from 2023.
- **Double DQN (DDQN)** is detailed in reference [50] from 2024.
- **Prioritized Experience Replay (PER)** is detailed in reference [51] from 2024.
- **Soft Actor-Critic (SAC)** is detailed in reference [52] from 2022.
- **Trust Region Policy Optimization (TRPO)** is detailed in reference [53] from 2015.
- **Proximal Policy Optimization (PPO)** is detailed in reference [55] from 2023.
- **Deep Deterministic Policy Gradient (DDPG)** is detailed in reference [57] from 2021.
- **Rainbow DQN (RDQN)** is detailed in reference [58] from 2018.
- **Asynchronous Advantage Actor-Critic (A3C)** is detailed in reference [60] from 2024.

These algorithms were selected due to their extensive application and proven effectiveness in research, making them well-suited for the experimental assessment of segmentation and scheduling issues within the context of robotic Soccer environments. The second group consists of traditional algorithms that are commonly used in resource allocation and scheduling challenges and are regularly employed by researchers. These algorithms include:

- **First-Come, First-Served (FCFS)** is detailed in reference [61] from 2018.
- **Round Robin (RR)** is detailed in reference [62] from 2018.
- **Shortest Job Next (SJN)** is detailed in reference [63] from 2015.
- **Priority Scheduling (PS)** is detailed in reference [64] from 2022.
- **Earliest Deadline First (EDF)** is detailed in reference [65] from 2022.
- **Shortest Remaining Time (SRT)** is detailed in reference [66] from 2021.

- **Multilevel Queue Scheduling (MQS)** is detailed in reference [67] from 2023.
- **Multilevel Feedback Queue (MFQ)** is detailed in reference [68] from 2012.
- **Rate-Monotonic Scheduling (RMS)** is detailed in reference [69] from 2009.
- **Lottery Scheduling (LS)** is detailed in reference [70] from 2021.

It is important to highlight that the source code for all evaluated algorithms has been comprehensively documented and is publicly accessible on **GitHub** through **the repository** provided here.

3-4-1- Solving Task Partitioning and Scheduling Issues without Mapping to the Soccer Robot Environment

In this phase, the task partitioning and scheduling problem is addressed independently of the soccer robot environment. The primary objective is to optimize task scheduling, resource allocation, and minimize delays. The source code utilized for each algorithm is available at the specified address. The evaluation criteria are as follows:

- **Average Task Completion Time (ATCT)**: This metric represents the mean duration required to complete each task from its entry into the system until its processing is finished. It is derived from the difference between 'Completion Time' and 'Arrival Time' for each task. This metric evaluates the efficiency of the scheduling algorithm in managing task completion and highlights its impact on overall system performance.
- **Resource Allocation Accuracy (RAA)**: This metric measures the precision in allocating resources to tasks based on their actual requirements. It is calculated by comparing the 'Resource Requirements' of each task with the resources actually allocated during processing. This metric is crucial for assessing how well the algorithm optimizes resource distribution, aiming to avoid both over-allocation and resource shortages.
- **Processing Speed (PS)**: This metric indicates the rate at which tasks are processed and is typically represented as the average time required to handle each task. It can be derived from the 'Processing Time' for each task. Processing Speed assesses the algorithm's capability to complete tasks efficiently and contributes to overall system performance by minimizing the time each task requires.
- **Average Waiting Time (AWT)**: This metric represents the average duration that tasks spend in the queue before processing begins. It is derived from the 'Waiting Time' feature. Average Waiting Time helps analyze how well the algorithm manages the queue, impacting both user experience and overall processing time. It provides insights into the system's efficiency in handling task queues.
- **Average Response Time (ART)**: This metric measures the average time elapsed from a task's entry into the system until processing begins. It is calculated using the difference between 'Waiting Time' and 'Arrival Time'. Average Response Time is essential for evaluating the system's responsiveness to new tasks and its influence on performance and response times.
- **Delay Time (DT)**: This metric captures the additional time incurred due to delays in resource allocation or scheduling. It is derived from the 'Delay' feature, which reflects the difference between the predicted and actual completion times. Analyzing Delay Time helps identify inefficiencies and weaknesses in resource allocation and scheduling strategies.
- **Resource Utilization Ratio (RUR)**: This metric represents the percentage of resources utilized relative to the total available resources. It is calculated by comparing 'Resource Utilization' to total available resources. Resource Utilization Ratio evaluates the efficiency of resource usage and optimization in the scheduling process.
- **Missed Deadlines Ratio (MDR)**: This metric measures the percentage of tasks that fail to meet their deadlines. It is calculated based on the 'Delay' feature, where tasks with significant delays are considered missed deadlines. This metric is important for assessing the algorithm's capability to adhere to temporal constraints and enhance real-time system performance.
- **Capacity Utilization Ratio (CUR)**: This metric evaluates the percentage of the system’s processing capacity used compared to its total capacity. It is derived from comparing actual resource usage to maximum available capacity. Capacity Utilization Ratio helps assess how effectively the system’s processing capacity is used and identifies potential improvements in task scheduling.

- **System Stability (SS)**: This metric assesses the system's ability to maintain consistent and predictable performance under varying conditions and loads. It is evaluated based on metrics such as 'Average Completion Time', 'Average Waiting Time', and 'Resource Utilization Ratio'. System Stability reflects the system's robustness and adaptability to changes in workload and operational conditions.

These criteria enable a comprehensive and precise evaluation of the performance and efficiency of partitioning and scheduling algorithms, allowing for an in-depth assessment of their capabilities across various domains. Following extensive experimentation, the results presented in Tables 10 and 11 have been obtained for each category of algorithms. These tables present a comparative analysis of various scheduling algorithms, highlighting their performance across multiple metrics. The source code for the algorithms in each category can be found in the **GitHub repository** provided here.

Table 10: Performance Comparison of First-Category Algorithms without Performing Mapping

Algorithm	ATCT (s)	RAA (%)	PS (tasks/s)	AWT (s)	ART (s)	DT (s)	RUR (%)	MDR (%)	CUR (%)	SS (10)
DQN [49] in 2023	8.5	89.6	1.15	2.3	1.8	0.6	88	5.2	85	8.5
DDQN [50] in 2024	8.3	90	1.18	2.2	1.75	0.55	89	4.8	86	8.7
PER [51] in 2024	8.1	90.5	1.2	2.1	1.7	0.5	90	4.6	87	8.8
SAC [52] in 2022	7.9	91	1.22	2	1.65	0.45	91	4.4	88	9
TRPO [53] in 2015	7.8	91.5	1.25	1.9	1.6	0.4	92	4.2	89	9.2
PPO [55] in 2023	7.6	92	1.28	1.8	1.55	0.35	93	4	90	9.3
DDPG [57] in 2021	7.5	92.5	1.3	1.75	1.5	0.3	94	3.8	91	9.4
RDQN [58] in 2018	7.4	92.8	1.32	1.7	1.45	0.25	95	3.6	92	9.5
A3C [60] in 2024	7.2	91.8	1.28	1.6	1.4	0.2	95	3.5	91	9.6

Table 11: Performance Comparison of Second-Category Algorithms without Performing Mapping

Algorithm	ATCT (s)	RAA (%)	PS (tasks/s)	AWT (s)	ART (s)	DT (s)	RUR (%)	MDR (%)	CUR (%)	SS (10)
FCFS [61] in 2018	15.2	79	0.67	3.2	2.5	1	75	8	70	6.5
RR [62] in 2018	12.8	81	0.79	2.8	2.2	0.7	80	7	75	7
SJN [63] in 2015	10.5	84	0.95	2	1.8	0.6	85	6	80	7.5
PS [64] in 2022	11	84	0.9	2.2	1.9	0.6	82	6.5	77	7
EDF [65] in 2022	9.8	86	1.05	1.8	1.7	0.5	88	5.5	85	8
SRT [66] in 2021	9.5	87	1.1	1.7	1.7	0.5	89	5.2	86	8.2
MQS [67] in 2023	11.5	82	0.85	2.5	2.1	0.7	80	7.5	76	7
MFQ [68] in 2012	11	83	0.88	2.3	2	0.6	81	7.2	78	7.5
RMS [69] in 2009	10.2	85	0.92	2.1	1.9	0.6	84	6.8	82	8
LS [70] in 2021	12	80	0.75	2.7	2.2	0.7	78	7.8	74	6.8

The performance analysis of the algorithms will be discussed below. In this analysis, we will assess the performance of the algorithms based on the conducted experiments. This analysis will highlight the strengths and weaknesses of each algorithm and compare them according to evaluation criteria such as average task completion time, resource allocation accuracy, and processing speed.

Category One: Reinforcement Algorithms

1- Deep Q-Network (DQN)

Strengths:

- **Complex Task Management:** Capable of learning complex policies for managing tasks in dynamic environments.
- **Self-Learning:** Possesses the ability to learn from past experiences, potentially improving performance over time.

Weaknesses:

- **Extended Training Time:** Requires a significant amount of time to achieve optimal training.
- **Learning Stability:** May experience fluctuations in learning and performance.

Comparison: Compared to classical algorithms like FCFS and Round Robin, DQN exhibits superior performance in resource allocation accuracy and processing speed. It is also comparable to A3C in terms of resource allocation accuracy but may lag slightly in processing speed.

2-Double DQN (DDQN)

Strengths:

- **Q-Value Deviation Reduction:** Utilizes two networks to reduce Q-value deviation, providing enhanced performance.
- **Improved Learning Stability:** More stable and reliable in learning compared to DQN.

Weaknesses:

- **Increased Complexity:** More complex than DQN and requires additional resources for training.
- **Training Duration:** May require longer training times compared to simpler algorithms.

Comparison: DDQN outperforms DQN in terms of resource allocation accuracy and learning stability. However, it may be slightly weaker in processing speed compared to algorithms like A3C.

3-Prioritized Experience Replay (PER)

Strengths:

- **Enhanced Learning Speed:** Focuses on higher-priority experiences, leading to faster and more efficient learning.
- **Reduced Fluctuations:** Decreases learning variability and improves overall performance.

Weaknesses:

- **Experience Management Complexity:** Requires a more complex implementation for experience management.
- **Training Time:** May need additional training time.

Comparison: PER shows significant improvements over DQN and DDQN in learning speed and resource allocation accuracy. However, it may have limitations in processing speed compared to algorithms like A3C and PPO.

4-Soft Actor-Critic (SAC)

Strengths:

- **Stable and Effective Learning:** Provides stable and effective learning with soft policies and optimized performance.
- **Efficient Uncertainty Management:** Handles uncertainty and complex environments effectively.

Weaknesses:

- **Algorithm Complexity:** More complex than simpler algorithms and requires more precise tuning.
- **Training Time:** Relatively long training time to achieve optimal performance.

Comparison: SAC outperforms classical algorithms like FCFS and Round Robin, as well as DQN and DDQN, in resource allocation accuracy and processing speed. Compared to A3C, SAC may face limitations in complexity and training duration.

5-Trust Region Policy Optimization (TRPO)

Strengths:

- **Stable Optimization:** Provides stable and efficient optimization using regional constraints.
- **Effective Policy Management:** Manages policy changes effectively.

Weaknesses:

- **High Complexity:** More complex to implement and requires additional resources.
- **Training Duration:** Relatively long training time for achieving the best results.

Comparison: TRPO offers superior resource allocation accuracy and learning stability compared to classical algorithms, as well as DQN and DDQN. It is comparable to A3C and SAC in terms of complexity and training time.

6-Proximal Policy Optimization (PPO)

Strengths:

- **Stability and Efficiency:** Provides high stability and efficiency in learning and policy optimization.
- **Simplified Implementation:** Less complex to implement than TRPO.

Weaknesses:

- **Training Time:** May require additional training time.
- **Complexity Management:** Requires precise tuning for optimal performance.

Comparison: PPO shows better performance than DQN and DDQN in resource allocation accuracy and processing speed and is competitive with A3C and SAC, with the added benefit of simpler implementation.

7-Deep Deterministic Policy Gradient (DDPG)

Strengths:

- **Continuous Environment Management:** Suitable for continuous environments with continuous actions.
- **Effective Learning:** Provides effective learning in complex environments.

Weaknesses:

- **Learning Stability:** May experience fluctuations and stability issues in learning.
- **Precise Tuning Required:** Needs careful tuning and adjustment for optimal performance.

Comparison: DDPG outperforms classical algorithms and DQN/DDQN in continuous environments but may face stability issues compared to A3C and SAC.

8-Rainbow DQN (RDQN)

Strengths:

- **Optimal Combination of Techniques:** Integrates various optimizations such as PER and Double DQN for enhanced performance.
- **Optimized Performance:** Delivers optimal and stable performance in learning.

Weaknesses:

- **High Algorithm Complexity:** More complex to implement and requires additional resources.
- **Extended Training Time:** Longer training time due to the integration of multiple techniques.

Comparison: Rainbow DQN performs better than DQN and DDQN and is competitive with A3C and SAC in terms of resource allocation accuracy and processing speed. However, its implementation complexity is higher.

9-Asynchronous Advantage Actor-Critic (A3C)

Strengths:

- **Simultaneous Learning:** Enables concurrent learning from multiple agents to improve learning speed and accuracy.
- **High Stability:** Provides higher stability in learning compared to other algorithms.

Weaknesses:

- **High Complexity:** Requires substantial resources and has a complex implementation.
- **Data Management:** Needs precise data management for optimal performance.

Comparison: A3C offers superior resource allocation accuracy and processing speed compared to DQN and DDQN and is generally comparable to SAC and PPO, albeit with potentially higher resource demands.

Category Two: Classic Algorithms

1-First-Come, First-Served (FCFS)

Strengths:

- **Simplicity of Implementation:** This algorithm is straightforward to implement and does not require complex configurations.
- **Predictability:** Task completion times are easily predictable.

Weaknesses:

- **Lack of Consideration for Task Duration:** It may lead to increased completion times for lower-priority tasks.
- **Poor Performance in Imbalanced Loads:** It does not perform optimally in scenarios where tasks have varying durations.

Comparison: FCFS performs less effectively compared to reinforcement learning algorithms such as DQN and A3C in terms of resource allocation accuracy and processing speed. It is more suited for environments with constant loads.

2-Round Robin (RR)

Strengths:

- **Fair Task Management:** Tasks are distributed evenly among resources, reducing average waiting time.
- **Simplicity and Ease of Implementation:** Similar to FCFS, it is straightforward and understandable.

Weaknesses:

- **Neglect of Task Duration:** May not be optimal for tasks of varying lengths, potentially leading to resource wastage.
- **Reduced Efficiency in Variable Loads:** Performance diminishes when tasks have significantly differing execution times.

Comparison: It performs better than FCFS but still lags behind reinforcement learning algorithms in terms of processing speed and resource allocation accuracy.

3-Shortest Job Next (SJN)

Strengths:

- **Reduced Task Completion Time:** Prioritizing shorter tasks decreases overall completion time.
- **Decreased Waiting Time:** Effective for imbalanced loads where tasks have varying durations.

Weaknesses:

- **Inaccurate Prediction:** May fail to accurately predict future tasks, potentially leading to delays for longer tasks.
- **Requirement for Accurate Information:** Requires precise knowledge of task execution times.

Comparison: Outperforms FCFS and Round Robin, and is comparable to reinforcement learning algorithms such as DQN and PPO, though it still has limitations relative to more advanced learning algorithms.

4-Priority Scheduling (PS)

Strengths:

- **Priority Management:** Allocates more attention to higher-priority tasks, suitable for systems with critical tasks.
- **Reduced Completion Time for Prioritized Tasks:** Higher-priority tasks are processed more quickly.

Weaknesses:

- **Starvation:** Lower-priority tasks may experience delays or even starvation.
- **Increased Complexity:** More complex to implement and manage compared to simpler algorithms.

Comparison: Performs better in priority management than FCFS and Round Robin and is closer to reinforcement learning algorithms in terms of resource allocation accuracy.

5-Earliest Deadline First (EDF)

Strengths:

- **Deadlines Management:** Tasks with the nearest deadlines are processed first.
- **Suitability for Real-Time Systems:** Functions well in systems that require adherence to deadlines.

Weaknesses:

- **Complex Implementation:** More complex to implement and manage.
- **Lower Stability:** May perform poorly in dynamic load conditions.

Comparison: Performs better than classic algorithms such as FCFS and Round Robin and is comparable to reinforcement learning algorithms in terms of processing speed and resource allocation accuracy.

6-Shortest Remaining Time (SRT)

Strengths:

- **Reduced Task Completion Time:** Tasks with the shortest remaining time are prioritized.
- **Suitability for Dynamic Loads:** Performs better under varying load conditions.

Weaknesses:

- **Increased Complexity:** Requires precise information and more complex management.
- **Starvation:** Longer tasks may experience delays.

Comparison: Superior to simpler algorithms like FCFS and Round Robin and approaches reinforcement learning algorithms in terms of resource allocation accuracy and processing speed.

7-Multilevel Queue Scheduling (MQS)

Strengths:

- **Management of Diverse Tasks:** Suitable for systems with tasks of varying priorities.
- **Configurable:** Allows for adjustment and management of different priorities.

Weaknesses:

- **High Complexity:** More complex to implement and manage.
- **Reduced Efficiency in Variable Conditions:** Performance may decline under fluctuating load conditions.

Comparison: Outperforms FCFS and Round Robin and is comparable to reinforcement learning algorithms, though it may lag behind more complex algorithms like SAC and PPO.

8-Multilevel Feedback Queue (MFQ)

Strengths:

- **Dynamic Adjustment:** Capable of adjusting priorities based on task behavior.
- **Management of Variable Tasks:** Effective in systems with variable loads.

Weaknesses:

- **High Complexity:** Requires more complex configuration and implementation.
- **Reduced Performance in Constant Loads:** May perform less effectively in stable load conditions.

Comparison: Comparable to reinforcement learning algorithms in variable conditions, though it may be less effective than more advanced algorithms.

9-Rate-Monotonic Scheduling (RMS)

Strengths:

- **Management of Varying Rates:** Effective for systems with tasks of different rates.
- **High Stability:** Performs well under stable conditions.

Weaknesses:

- **Lower Stability in Variable Loads:** May perform less effectively under varying load conditions.
- **Complex Implementation:** Requires more detailed implementation and configuration.

Comparison: Performs better than classic algorithms and is comparable to reinforcement learning algorithms in terms of resource allocation accuracy.

10-Lottery Scheduling (LS)

Strengths:

- **Randomized Management:** Suitable for unpredictable and variable loads.
- **Ease of Implementation:** Relatively simple and understandable.

Weaknesses:

- **Lack of Predictability:** Difficulty in accurately predicting task completion times.
- **Lower Performance in Stable Loads:** May perform poorly under constant loads.

Comparison: Less effective than reinforcement learning algorithms in terms of resource allocation accuracy and processing speed.

Overall Analysis:

- **Reinforcement Learning Algorithms:** Algorithms such as DQN, DDQN, and PPO exhibit superior performance in resource allocation accuracy and processing speed compared to classical algorithms. Their advanced learning capabilities and optimization techniques allow for more effective management of complex and dynamic tasks.
- **Complex Algorithms:** Algorithms like SAC and A3C excel in managing complex and dynamic environments but involve higher implementation complexity and longer training times.
- **Hybrid Algorithms:** Algorithms like Rainbow DQN offer optimal performance but with increased complexity and resource requirements.

3-4-2- Solving the Task Partitioning and Scheduling Problem with Mapping to the Soccer Robot Environment

In this phase, the task partitioning and scheduling problem is addressed by aligning it with the features of the soccer robot environment. Specifically, the characteristics and constraints of the soccer robot environment—such as navigation and obstacle avoidance—are incorporated into the scheduling solution. Given the unique features of the soccer robot environment, we will utilize the previously mentioned algorithms to resolve the problem. Considering the extracted features from the dataset, the evaluation metrics that align with these features can be described as follows:

- **Average Task Completion Time (ATCT):** This metric represents the average duration required to complete all tasks or missions assigned to the robot. Specifically, this can be derived from the routing time, which is the time the robot spends navigating a particular path. Generally, the average task completion time is calculated as the mean time the robot requires to complete multiple routes or tasks.
- **Resource Allocation Accuracy (RAA):** This metric evaluates the precision in allocating and utilizing various resources (such as energy, time, or equipment) to accomplish different robotic tasks. Resource allocation accuracy is derived from the navigation requirements feature, which indicates the quantity and type of resources needed for routing. Optimal and accurate resource allocation leads to better performance and reduces resource wastage.
- **Processing Speed (PS):** This metric quantifies the number of tasks a robot can effectively process and execute within a single time unit (e.g., one second). It is derived from the computational performance attributes and the efficiency of the algorithms employed in task resolution. Processing speed is intrinsically linked to the robot's overall capability to manage and perform multiple tasks simultaneously, particularly in dynamic environments where rapid response to changing conditions is imperative. An enhancement in processing speed indicates a higher efficiency of the algorithms in task management and resource allocation, leading to a reduction in overall task execution time and an improvement in the robot's operational performance.
- **Obstacle Avoidance Success Rate (OASR):** This metric reflects the robot's success rate in avoiding collisions with obstacles in the environment. It is derived from the obstacle avoidance success feature and measures the robot's ability to detect and avoid obstacles. A high success rate indicates the robot's enhanced capability in avoiding collisions.
- **Average Navigation Time (ANT):** This metric indicates the time required for the robot to traverse a specific path from the starting point to the destination. It is derived from the navigation time feature and signifies the efficiency and speed of the robot's movement in the environment. A reduction in average navigation time implies an improvement in the robot's speed and efficiency in traversing routes.
- **Navigation Accuracy (NA):** This metric measures the robot's precision in determining and following the intended path. It is derived from the navigation error feature and indicates the discrepancy between the robot's final position and the target location. High navigation accuracy reflects the robot's superior ability to reach target locations with minimal error.
- **Resource Utilization (RU):** This metric pertains to the optimal and efficient use of available resources (such as processors, memory, and equipment). It is derived from the navigation requirements feature and reflects how resources are utilized to execute various tasks. Efficient resource utilization contributes to increased productivity and cost reduction.

After conducting the experiments, the results summarized in Tables 12 and 13 were obtained for each category of algorithms:

Table 12: Performance Comparison of First Category Algorithms Following Their Application to the Soccer Robot Environment

Algorithm	ATCT (s)	RAA (%)	PS (tasks/s)	OASR (%)	ANT (s)	NA (%)	RU (%)
DQN [49] in 2023	8	91	1.2	87	1.15	88.5	80
DDQN [50] in 2024	7.8	92	1.22	88	1.12	89	82
PER [51] in 2024	7.6	90.5	1.23	85.5	1.17	87	84
SAC [52] in 2022	7.1	93	1.28	90	1.08	90	88
TRPO [53] in 2015	7.5	91.5	1.25	88.5	1.1	88.5	85
PPO [55] in 2023	7.3	92	1.27	88	1.12	89	86
DDPG [57] in 2021	7.9	90	1.2	86	1.15	87.5	80
RDQN [58] in 2018	7.7	91.5	1.22	89	1.13	88.5	82
A3C [60] in 2024	6.9	93	1.3	91	1.05	91	90

This table presents a detailed performance analysis of the first category of algorithms, focusing on their effectiveness and efficiency when adapted to the soccer robot environment. The data reflects how each algorithm in this category performs in terms of key metrics such as task completion time, resource allocation accuracy, and obstacle avoidance success rate, providing valuable insights into their applicability and optimization potential in real-world robotic scenarios. The analysis of Table 12 results reveals varying levels of performance across different reinforcement learning algorithms, each with its strengths and potential areas for improvement. Let's break down the insights and underlying reasons for the observed results:

1- **Deep Q-Network (DQN)**:

- **Analysis**: DQN shows improvements in resource allocation accuracy and obstacle avoidance success rate compared to simpler methods. However, it falls short in task completion time and navigation time when compared to more advanced algorithms. The lower resource utilization suggests a conservative approach in resource usage, which could be a trade-off for ensuring stability in simpler environments.
- **Reasoning**: DQN, being one of the earlier reinforcement learning algorithms, lacks the ability to manage complex environments efficiently, leading to longer completion and navigation times. The relatively lower resource consumption is likely due to its simpler model, which might not fully exploit available computational resources.

2- **Double DQN (DDQN)**:

- **Analysis**: DDQN improves on DQN by reducing overestimation bias, leading to better resource allocation accuracy and obstacle avoidance. The task completion and navigation times are also slightly better, indicating more efficient decision-making.
- **Reasoning**: The improvements in DDQN can be attributed to its architectural enhancements over DQN, particularly in reducing the overestimation of action values, which makes it more reliable and efficient in navigating and completing tasks.

3- **Prioritized Experience Replay (PER)**:

- **Analysis**: PER offers improvements in resource allocation and obstacle avoidance but struggles with navigation time and accuracy. The resource utilization remains reasonable but not optimal.
- **Reasoning**: PER's performance reflects its focus on learning from more significant experiences, which can help in specific scenarios but might lead to inefficiencies in complex environments where a broader learning strategy is required.

4- **Soft Actor-Critic (SAC)**:

- **Analysis**: SAC excels in almost all metrics, including resource allocation accuracy, obstacle avoidance, and low task completion times. Its resource utilization is also optimized, making it a top performer in the table.
- **Reasoning**: SAC's success is due to its entropy-based framework, which encourages exploration while maintaining stable performance. This balance between exploration and exploitation allows SAC to perform well across different scenarios, optimizing both speed and accuracy.

5- **Trust Region Policy Optimization (TRPO)**:

- **Analysis**: TRPO demonstrates improvements in resource allocation and obstacle avoidance, with competitive task completion and navigation times. However, it doesn't outperform SAC in any particular metric.
- **Reasoning**: TRPO's conservative updates within a trust region help ensure stability, which explains it's consistent, if not leading, performance. The trade-off, however, is that it may not explore as aggressively as SAC, leading to slightly less optimal results.

6- **Proximal Policy Optimization (PPO)**:

- **Analysis**: PPO shows improvements in navigation accuracy and obstacle avoidance over simpler algorithms, with enhanced navigation times. Resource utilization is better than some of its predecessors.
- **Reasoning**: PPO's clip function, which constrains policy updates, strikes a balance between TRPO's stability and DQN's speed, leading to improved performance in complex tasks without overfitting to any specific scenario.

7- **Deep Deterministic Policy Gradient (DDPG)**:

- **Analysis**: DDPG offers better resource allocation accuracy and obstacle avoidance than simpler algorithms but struggles with higher task completion and navigation times. Its resource utilization is also relatively low.
- **Reasoning**: DDPG's ability to handle continuous action spaces gives it an edge in precision tasks but can lead to inefficiencies in broader scenarios, where its deterministic nature might hinder exploration, resulting in longer task times.

8- **Rainbow DQN (RDQN)**:

- **Analysis**: Rainbow DQN improves on DQN by combining several advancements, leading to better resource allocation and obstacle avoidance, with moderately better task completion and navigation times.
- **Reasoning**: Rainbow DQN's integration of various improvements (e.g., double DQN, prioritized replay, etc.) enhances its performance over standard DQN, although the complexity might still limit its potential in more dynamic environments.

9- **Asynchronous Advantage Actor-Critic (A3C)**:

- **Analysis**: A3C demonstrates superior performance across most metrics, with the lowest task completion and navigation times, and high accuracy in resource allocation and navigation. Its obstacle avoidance success rate is also the highest, with efficient resource utilization.
- **Reasoning**: A3C's asynchronous learning process allows it to explore a wide range of strategies simultaneously, leading to a more comprehensive and efficient learning process. This advantage translates into its top-tier performance, particularly in dynamic and complex environments.

The results in Table 12 reflect the varying strengths of these algorithms, shaped by their underlying architectures and learning strategies. Algorithms like SAC and A3C, which promote both exploration and stable learning, outperform others in complex tasks, while algorithms like DQN and DDQN, although effective, and are limited by their simpler approaches. Understanding these nuances helps in selecting the appropriate algorithm based on the specific requirements of the task environment. The general comparison are:

1- **Best Overall Performance**:

- **Asynchronous Advantage Actor-Critic (A3C)** and **Soft Actor-Critic (SAC)** stand out as the top performers across most metrics, including average task completion time, resource allocation accuracy, obstacle avoidance success rate, and navigation accuracy. These algorithms demonstrate a balanced and robust approach, excelling in both speed and precision.

2- **Efficient Navigation Time Algorithms**:

- **Soft Actor-Critic (SAC)** and **Asynchronous Advantage Actor-Critic (A3C)** are also recognized for efficient navigation times. Their ability to achieve high performance with lower resource usage gives them a distinct advantage over other algorithms.

3- Weaker Performers:

- **Deep Deterministic Policy Gradient (DDPG)** and **Deep Q-Network (DQN)** lag behind other algorithms in key metrics such as task completion time. These algorithms struggle to match the efficiency and accuracy of their more advanced counterparts, making them less suitable for complex, resource-intensive environments.

This analysis underscores the superiority of A3C and SAC in handling complex tasks while maintaining efficiency, highlighting their potential as go-to algorithms for applications requiring both high performance and resource optimization.

Table 13: Performance Comparison of Second Category Algorithms Following Their Application to the Soccer Robot Environment

Algorithm	ATCT (s)	RAA (%)	PS (tasks/s)	OASR (%)	ANT (s)	NA (%)	RU (%)
FCFS [61] in 2018	18.5	72.4	0.54	67	20.1	71.5	60
RR [62] in 2018	16.3	74.1	0.62	72.2	18.6	73.2	70
SJN [63] in 2015	14.9	77.2	0.75	78.1	17.8	76.8	85
PS [64] in 2022	15.6	79	0.7	80.4	17.1	78.5	80
EDF [65] in 2022	14.1	80.5	0.73	82.2	16.2	80	90
SRT [66] in 2021	13.7	82.1	0.77	84	15.9	82	88
MQS [67] in 2023	17.2	74.8	0.6	70.5	19.2	72	65
MFQ [68] in 2012	15.9	76.6	0.67	76.3	17.6	75	75
RMS [69] in 2009	16	75.3	0.65	73.8	18	74.5	70
LS [70] in 2021	15.3	77	0.69	77.5	17.3	76	68

In the following, we will provide a detailed analysis of the results presented in Table 13. This analysis will focus on evaluating and comparing the performance of various scheduling algorithms based on key metrics such as task completion times, resource allocation accuracy, obstacle avoidance effectiveness, and overall efficiency. By examining these metrics, we aim to uncover the relative strengths and weaknesses of each algorithm, thereby offering a comprehensive understanding of their operational efficacy in different scheduling contexts. Let's break down the insights and underlying reasons for the observed results:

1- First-Come, First-Served (FCFS):

- **Analysis:** FCFS, being one of the simplest scheduling algorithms, demonstrates the highest task completion time and exhibits lower accuracy in resource allocation and obstacle avoidance. The algorithm also shows relatively lower resource utilization.
- **Reasoning:** The FCFS algorithm processes tasks in the order they arrive without considering their execution complexity or load, leading to longer completion times. This lack of flexibility in task management and resource allocation results in inefficient resource use and reduced obstacle avoidance accuracy. The algorithm's straightforward nature does not account for varying task requirements, which diminishes its overall performance.

2- Round Robin (RR):

- **Analysis:** Round Robin improves upon FCFS by reducing task completion times and enhancing resource allocation accuracy. However, it exhibits relatively higher navigation times compared to other algorithms.
- **Reasoning:** Round Robin allocates CPU time to tasks in a cyclic manner, promoting fairness and improving task completion time and resource allocation accuracy. Despite these benefits, its periodic allocation can result in longer navigation times due to inefficiencies in managing tasks with varying execution times. The algorithm is effective in scenarios requiring balanced task management but may be less optimal in specific conditions.

3- Shortest Job Next (SJN):

- **Analysis:** SJN prioritizes tasks with shorter execution times, resulting in reduced task completion times and improved accuracy in resource allocation and obstacle avoidance.
- **Reasoning:** By focusing on tasks with shorter execution durations, SJN effectively reduces completion times and enhances resource allocation accuracy and obstacle avoidance. This algorithm is advantageous in environments with a mix of task durations. However, it may not perform as well for long-duration tasks, which could experience delays due to the algorithm's preference for shorter tasks.

4- Priority Scheduling (PS:

- **Analysis:** Priority Scheduling assigns resources based on task priority, achieving high accuracy in resource allocation and obstacle avoidance while maintaining reasonable task completion and navigation times.
- **Reasoning:** This algorithm's emphasis on high-priority tasks allows for efficient management of critical workloads and improved accuracy in resource allocation and obstacle avoidance. Nonetheless, the focus on prioritizing certain tasks can lead to increased completion times for lower-priority tasks, affecting overall system performance.

5- Earliest Deadline First (EDF):

- **Analysis:** EDF processes tasks based on their deadlines, demonstrating the best performance in task completion times and achieving the highest accuracy in resource allocation and obstacle avoidance. It also shows superior resource utilization.
- **Reasoning:** By prioritizing tasks with the closest deadlines, EDF optimizes task management, leading to reduced completion times and enhanced accuracy in resource allocation and obstacle avoidance. Its ability to manage tasks with stringent deadlines effectively results in optimal resource utilization and overall system performance, making it highly effective in deadline-sensitive environments.

6- Shortest Remaining Time (SRT):

- **Analysis:** SRT, similar to SJN but focusing on tasks with shorter remaining times, excels in reducing task completion times and improving resource allocation and obstacle avoidance accuracy.
- **Reasoning:** SRT's strategy of prioritizing tasks with shorter remaining execution times enhances performance by minimizing completion times and optimizing resource allocation. Although it performs exceptionally well, its effectiveness may diminish in scenarios where remaining times are dynamically changing, requiring precise management to maintain optimal performance.

7- Multilevel Queue Scheduling (MQS):

- **Analysis:** Multilevel Queue Scheduling categorizes tasks into different queues, leading to higher task completion times and lower accuracy in resource allocation and obstacle avoidance. It also exhibits higher navigation times.
- **Reasoning:** The complexity of managing multiple queues and the resultant higher navigation times contribute to the lower performance of this algorithm. The categorization of tasks into distinct queues introduces inefficiencies in resource management and task prioritization, affecting overall system effectiveness.

8- Multilevel Feedback Queue (MFQ):

- **Analysis:** The Multilevel Feedback Queue, which adapts task priorities based on their requirements, performs better than the Multilevel Queue but still exhibits lower task completion times and resource allocation accuracy compared to top-performing algorithms.
- **Reasoning:** This algorithm's dynamic adjustment of task priorities enhances its performance relative to the Multilevel Queue. However, the inherent complexity in managing changing queue levels and navigation time's results in suboptimal performance compared to algorithms like EDF and SRT.

9- Rate-Monotonic Scheduling (RMS):

- **Analysis:** RMS schedules tasks based on their rates, demonstrating reasonable accuracy in resource allocation and obstacle avoidance but showing relatively higher task completion and navigation times.
- **Reasoning:** RMS's approach to prioritizing tasks based on their periodic rates leads to moderate accuracy in managing resources and avoiding obstacles. The algorithm's inability to efficiently handle high loads and complex task interactions results in longer completion and navigation times, affecting overall performance.

10- Lottery Scheduling (LS):

- **Analysis:** Lottery Scheduling, which allocates resources based on random selection, performs better than FCFS and Round Robin but exhibits variable accuracy in resource allocation and obstacle avoidance.
- **Reasoning:** The stochastic nature of Lottery Scheduling provides an improvement over simpler algorithms by introducing a probabilistic approach to resource allocation. However, the randomness involved can lead to inconsistent results in resource allocation and obstacle avoidance, necessitating careful management to ensure reliable performance.

This analysis will explore how different algorithms, ranging from simpler to more complex approaches, impact performance outcomes. We will highlight which algorithms excel in achieving optimal results and which exhibit limitations, providing valuable insights for selecting the most appropriate algorithm based on specific operational needs and constraints. The general comparison are:

1- Best Overall Performance:

- **Shortest Remaining Time (SRT)** and **Earliest Deadline First (EDF)** exhibit superior performance across most metrics. These algorithms excel in task completion times, resource allocation accuracy, and obstacle avoidance due to their strategic focus on remaining execution times and deadlines. Their advanced handling of these factors positions them at the forefront of efficiency and precision in complex scheduling scenarios.

2- Simpler Algorithms:

- **First-Come, First-Served (FCFS)** and **Round Robin (RR)** algorithms demonstrate weaker performance due to their inherent simplicity and lack of consideration for task complexity and priority. Their straightforward approaches result in higher task completion times and lower accuracy in resource allocation and obstacle avoidance, making them less effective in more demanding environments.

3- More Complex Algorithms:

- **Priority Scheduling (PS)** and **Multilevel Feedback Queue (MFQ)** offer improved performance compared to simpler algorithms by incorporating task priorities and dynamic adjustments. However, their performance remains inferior to the more advanced **SRT** and **EDF** algorithms. Despite their enhanced complexity, they fall short in achieving the optimal balance of efficiency and accuracy demonstrated by SRT and EDF.

This analysis highlights the distinction between basic scheduling algorithms and their more advanced counterparts, underscoring the superior capabilities of SRT and EDF in managing complex tasks efficiently.

4- Integrated Analysis of Results

To begin a comprehensive analysis, it is essential to first focus on the key features of each category of algorithms. This approach ensures a deeper understanding of their performance and capabilities.

4-1- Key Features of Algorithm Categories: Experimental Insights

Specifically, based on the extensive experiments conducted in this study, the key features for each algorithm category have been carefully inferred and are presented in Tables 14 and 15. These tables serve as a crucial reference point, summarizing the most significant attributes observed during our analysis. The importance of these tables lies in their ability to provide readers with a clear and comprehensive overview of the fundamental strengths and operational characteristics of each algorithm. By consolidating these key features, the tables not only enhance the understanding of the algorithms' performance under various conditions but also assist researchers and practitioners in selecting the most appropriate algorithm for specific tasks. This structured presentation is designed to facilitate informed decision-making, thereby optimizing the application of these algorithms in complex computational environments.

Table 14: Key Features Obtained from the Experiments for the Algorithms in the First Category

Algorithm	Key Features
Deep Q-Network (DQN)	Ability to learn from experiences, good adaptability to complex environments
Double DQN (DDQN)	Reduced deviations, improved accuracy and stability in learning
Prioritized Experience	Experience prioritization, improved learning from rare

Replay (PER)	experiences
Soft Actor-Critic (SAC)	Soft policy learning, enhanced exploration and diversity
Trust Region Policy Optimization (TRPO)	Policy change constraints, improved stability and performance
Proximal Policy Optimization (PPO)	Proximity optimization, simplified implementation, and improved performance
Deep Deterministic Policy Gradient (DDPG)	Optimal continuous action management, high learning capability
Rainbow DQN (RDQN)	Combination of various methods, improved accuracy and stability
Asynchronous Advantage Actor-Critic (A3C)	Parallel and asynchronous training, improved speed and learning accuracy

Table 15: Key Features Obtained from the Experiments for the Algorithms in the Second Category

Algorithm	Key Features
First-Come, First-Served (FCFS)	Simplicity of implementation, suitable for uniform loads.
Round Robin (RR)	Fair task management, reduced average delay.
Shortest Job Next (SJN)	Reduced task completion time, suitable for unbalanced loads.
Priority Scheduling (PS)	Priority management, suitable for high-priority systems.
Earliest Deadline First (EDF)	Deadlines management, reduced delay for near-due tasks.
Shortest Remaining Time (SRT)	Reduced remaining time, suitable for dynamic loads.
Multilevel Queue Scheduling (MQS)	Management of diverse priorities, suitable for complex systems.
Multilevel Feedback Queue (MFQ)	Dynamic priority adjustment, improved performance in variable conditions.
Rate-Monotonic Scheduling (RMS)	Management of tasks with varying rates, suitable for fixed-load systems.
Lottery Scheduling (LS)	Randomized management, suitable for unpredictable loads.

The analysis of Tables 14 and 15 reveals significant insights into the characteristics and strengths of the examined algorithms across two distinct categories. The key features identified for each algorithm not only highlight their unique capabilities but also provide a comprehensive understanding of their suitability for various tasks and environments. In the first category, which encompasses advanced reinforcement learning algorithms, the findings underscore the versatility and robustness of these methods in dynamic and complex environments. For instance, the **Deep Q-Network (DQN)** demonstrates exceptional adaptability through its ability to learn from experiences, which is crucial for real-time decision-making in unpredictable scenarios. Similarly, the **Soft Actor-Critic (SAC)** and **Trust Region Policy Optimization (TRPO)** algorithms show marked improvements in exploration diversity and stability, respectively, which are essential for maintaining performance in rapidly changing environments. The **Rainbow DQN (RDQN)**, with its integration of multiple techniques, stands out for its enhanced accuracy and stability, indicating its effectiveness in handling a broad range of tasks. In contrast, the second category, comprising classical scheduling algorithms, emphasizes efficiency and fairness in task management under varying conditions. For example, the **First-Come, First-Served (FCFS)** algorithm, with its simplicity, is particularly effective in scenarios with uniform task loads, whereas the **Round Robin (RR)** algorithm ensures fair task distribution, minimizing average delays. The **Earliest Deadline First (EDF)** algorithm's proficiency in managing deadlines is critical for time-sensitive applications, while the **Lottery Scheduling (LS)** algorithm's randomized approach provides a robust solution for systems with unpredictable loads. Overall, these findings demonstrate the

critical importance of selecting the appropriate algorithm based on the specific requirements of the task and environment. The advanced algorithms in the first category offer sophisticated techniques for handling complex, dynamic environments, making them ideal for applications requiring high adaptability and precision. Conversely, the classical algorithms in the second category provide straightforward, reliable solutions for more predictable, structured environments. This dual approach allows for a more nuanced and effective strategy in addressing a wide range of computational challenges, ensuring that the most suitable method is employed to achieve optimal performance.

4-2- Analysis of results

To comprehensively compare the results obtained before and after the mapping process, it is crucial to construct detailed tables that clearly present the outcomes from both stages. This approach not only enhances our understanding of the influence of mapping on algorithm performance but also allows for a precise comparison across key metrics. Specifically, Tables 16 and 17 provide a comparative analysis based on three critical criteria that were consistently evaluated in both stages: **Average Task Completion Time (ATCT)**, **Resource Allocation Accuracy (RAA)**, and **Processing Speed (PS)**. These metrics have been selected for their relevance in assessing the efficiency and effectiveness of the algorithms, offering a clear perspective on how the mapping process contributes to overall performance improvements.

Table 16: Performance Comparison of First Category Algorithms: Pre- and Post-Mapping

Algorithm	Proposed Mapping (✓/×)	Average Task Completion Time (ATCT) (s)	Resource Allocation Accuracy (RAA) (%)	Processing Speed (PS) (tasks/s)
Deep Q-Network (DQN) [49] in 2023	×	8.5	89.6	1.15
	✓	8	91	1.2
Double DQN (DDQN) [50] in 2024	×	8.3	90	1.18
	✓	7.8	92	1.22
Prioritized Experience Replay (PER) [51] in 2023	×	8.1	90.5	1.2
	✓	7.6	90.5	1.23
Soft Actor-Critic (SAC) [52] in 2022	×	7.9	91	1.22
	✓	7.1	93	1.28
Trust Region Policy Optimization (TRPO) [53] in 2015	×	7.8	91.5	1.25
	✓	7.5	91.5	1.25
Proximal Policy Optimization (PPO) [55] in 2023	×	7.6	92	1.28
	✓	7.3	92	1.27
Deep Deterministic Policy Gradient (DDPG) [57] in 2021	×	7.5	92.5	1.3
	✓	7.9	90	1.2
Rainbow DQN (RDQN) [58] in 2018	×	7.4	92.8	1.32
	✓	7.7	91.5	1.22
Asynchronous Advantage Actor-Critic (A3C) [60] in 2024	×	7.2	91.8	1.28
	✓	6.9	93	1.3

Table 17: Performance Comparison of Second Category Algorithms: Pre- and Post-Mapping

Algorithm	Proposed Mapping (✓/×)	Average Task Completion Time (ATCT) (s)	Resource Allocation Accuracy (RAA) (%)	Processing Speed (PS) (tasks/s)
First-Come, First-Served	×	15.2	79	0.67

(FCFS) [61] in 2018	✓	18.5	72.4	0.54
Round Robin (RR) [62] in 2018	✗	12.8	81	0.79
	✓	16.3	74.1	0.62
Shortest Job Next (SJN) [63] in 2015	✗	10.5	84	0.95
	✓	14.9	77.2	0.75
Priority Scheduling (PS) [64] in 2022	✗	11	84	0.9
	✓	15.6	79	0.7
Earliest Deadline First (EDF) [65] in 2022	✗	9.8	86	1.05
	✓	14.1	80.5	0.73
Shortest Remaining Time (SRT) [66] in 2021	✗	9.5	87	1.1
	✓	13.7	82.1	0.77
Multilevel Queue Scheduling (MQS) [67] in 2023	✗	11.5	82	0.85
	✓	17.2	74.8	0.6
Multilevel Feedback Queue (MFQ) [68] in 2012	✗	11	83	0.88
	✓	15.9	76.6	0.67
Rate-Monotonic Scheduling (RMS) [69] in 2009	✗	10.2	85	0.92
	✓	16	75.3	0.65
Lottery Scheduling (LS) [70] in 2021	✗	12	80	0.75
	✓	15.3	77	0.69

The results presented in Tables 16 and 17 demonstrate that advanced algorithms, particularly those belonging to the first category, exhibit superior performance when feature correspondence is considered. These algorithms enhance overall performance by improving resource allocation accuracy, processing speed, and success in obstacle avoidance. In contrast, simpler algorithms, such as those in the second category, struggle to cope with the added complexities of the problem, resulting in poorer performance. The reduced accuracy in resource allocation, slower processing speeds, and lower success rates in obstacle avoidance observed in these algorithms underscore their inability to handle more complex environments.

This observation aligns with the authors' contention that the trade-offs associated with the adoption of advanced algorithms are justified, particularly given the significant advantages they offer in the robotic soccer environment. These advantages include:

1- **Enhanced Resource Allocation Accuracy**: By mapping features to the environment, advanced algorithms can allocate resources with greater precision. This is due to a more nuanced understanding of the environment's constraints and needs, leading to more optimal resource utilization.

2- **Increased Processing Speed**: Feature mapping in the robotic soccer environment allows algorithms to make quicker decisions. This acceleration stems from more accurate and relevant information about the environment, which streamlines the processing workflow.

3- **Higher Obstacle Avoidance Success Rate**: Advanced algorithms, with their improved environmental comprehension, can implement more effective strategies for obstacle avoidance. This capability enables robots to navigate with greater precision, minimizing the risk of collisions.

4- **Reduced Average Navigation Time**: Feature mapping enables algorithms to select more optimal paths for robot movement, resulting in shorter navigation times and faster arrival at destinations.

5- **Improved Navigation Accuracy**: By leveraging more precise environmental data, advanced algorithms can choose more accurate routes, enhancing overall navigation accuracy.

6- **Optimized Decision-Making in Complex Scenarios**: Feature mapping aids algorithms in making better decisions in complex and dynamic situations. This involves a deeper understanding of the environment, anticipating potential changes, and swiftly adapting to new conditions.

7- **Increased Adaptability to Application Environment**: Through continuous feature mapping, algorithms progressively adapt to the environment, enhancing their efficiency and effectiveness in managing changing and complex conditions.

8- **Enhanced Algorithm Learning and Optimization**: Feature mapping supports algorithms in learning from past experiences, allowing them to implement improved strategies for future scenarios. This continuous improvement cycle leads to sustained performance enhancements and increased productivity.

Undoubtedly, the act of mapping features within the robotic environment extends these advantages to the task scheduling and partitioning problem. Specifically, the benefits of feature mapping in this context include:

1- **Enhanced Resource Allocation Accuracy**: In task scheduling and partitioning, the optimal allocation of resources is critical. By mapping features to the robotic soccer environment, algorithms can allocate resources more accurately and in alignment with actual needs, thereby increasing efficiency and minimizing resource wastage.
2- **Improved Processing Speed**: Feature mapping facilitates faster task processing by providing algorithms with more precise environmental data. This enables quicker decision-making and optimizes the task scheduling process.
3- **Increased Task Prioritization Accuracy**: With a deeper understanding of the environment and existing tasks, algorithms can prioritize tasks based on the actual environmental priorities. This results in improved efficiency and greater effectiveness in task execution.
4- **Reduced Task Execution Delays**: Feature mapping allows algorithms to execute tasks with minimal delays. This is achieved through more precise and environment-specific scheduling and resource allocation.
5- **Enhanced Management of Task Interference**: In complex environments like robotic soccer, task interference can lead to reduced efficiency. Feature mapping assists algorithms in minimizing task interference, ensuring more coordinated execution.
6- **Greater Task Alignment with Environmental Conditions**: By mapping features to the environment, algorithms can schedule tasks in a manner that aligns closely with environmental conditions, leading to improved overall robotic performance.
7- **Optimized Decision-Making in Complex Scenarios**: In dynamic and complex conditions, optimal decision-making in task scheduling and partitioning is crucial. Feature mapping aids algorithms in making better decisions, thus enhancing performance optimization.
8- **Increased Predictive Capability and Responsiveness to Environmental Changes**: Feature mapping enables algorithms to anticipate environmental changes and respond swiftly. This leads to better task scheduling and reduced delays in reaction to changes.

4-3- Expanding the Utility of Robotic Soccer Simulators and Beyond

The use of various simulators in the domain of robotic soccer significantly enhances our ability to solve problems more efficiently, effectively, and optimally. Reference [71] offers a comprehensive overview of the evolution and variety of simulators used in the fields of the Internet of Things, cloud computing, edge computing, and fog computing. While simulators such as **Cloudsim** [72] and **iFogsim** [73] are available for task scheduling and partitioning, the number and diversity of simulators in the robotic soccer domain are notably greater. These simulators offer more realistic virtual environments that can be effectively utilized for testing and optimizing algorithms. Below are some notable examples of these simulators:

1- **RoboCup Soccer Simulation (RCSS)**: This simulator is one of the most renowned in the robotic soccer field, primarily used for the global RoboCup competitions. RCSS provides a comprehensive and dynamic environment for testing and evaluating various algorithms in robotic soccer [74]. (**Download Link:** The RoboCup Soccer Simulator, **GitHub Repository:** RoboCup Soccer Simulator GitHub).
2- **Gazebo**: Gazebo is a robotics simulator that enables the creation of complex and realistic 3D environments. It is often used in conjunction with ROS (Robot Operating System) and is highly suitable for simulating robotic soccer scenarios [75]. (**Download Link:** Gazebo Official Website, **GitHub Repository:** Gazebo GitHub).
1- **Webots**: Webots is a professional-grade robotics simulator that allows for the simulation of robotic soccer players in diverse environments. It is known for its realistic physics engine and advanced graphical capabilities [76]. (**Download Link:** Webots Official Website, **GitHub Repository:** Webots GitHub).
2- **V-REP (CoppeliaSim)**: V-REP, also known as CoppeliaSim, is a versatile robotics simulator that can also be applied to robotic soccer simulations. It offers extensive features for programming, controlling, and simulating robots [77]. (**Download Link:** CoppeliaSim Official Website, **GitHub Repository:** CoppeliaSim V-REP GitHub).

3- **MORSE**: MORSE is an open-source robotics simulator that supports the simulation of robotic soccer players. It is compatible with Python and integrates well with other robotic software, providing flexibility in simulation scenarios [78]. (**Download Link** and **GitHub Repository:** MORSE GitHub).
4- **SimSpark**: SimSpark is another notable simulator in the robotic soccer domain, offering a robust platform for testing and optimizing algorithms within a simulated soccer environment [79]. (**Download Link and GitLab Repository:** SimSpark GitLab).

The diversity and capabilities of simulators in the domain of robotic soccer extend far beyond the mere replication of physical environments. These platforms not only facilitate the testing and optimization of algorithms but also serve as powerful tools for broader robotic applications. RoboCup Soccer Simulation **(RCSS)** and **Simspark** are specifically designed for robotic soccer, providing a dynamic environment tailored to this purpose. In contrast, simulators like **Gazebo**, **Webots**, **V-REP (CoppeliaSim)**, and **MORSE** offer broader capabilities that make them indispensable across various robotics research fields. For instance, **Gazebo** and **Webots** are recognized for their ability to simulate complex, realistic 3D environments, making them valuable not only in robotic soccer but also in simulating autonomous vehicles, drones, and industrial robots. **V-REP (CoppeliaSim)**, with its extensive programming and control features, exemplifies a simulator that transcends the confines of robotic soccer, proving useful in tasks ranging from industrial automation to exploratory robotics. Moreover, **MORSE**, despite its open-source nature, offers a high degree of flexibility, supporting Python and other robotic software. This makes it particularly suitable for integration with larger robotic frameworks, allowing researchers to explore more intricate scenarios beyond the soccer field. While **Simspark** is primarily used for robotic soccer, its robust environment can be adapted to test a wide range of algorithms and robotic behaviors in controlled settings. These simulators not only advance the field of robotic soccer but also significantly contribute to the broader domain of robotics. They provide researchers with the ability to model and simulate various aspects of robotic systems in a controlled, virtual environment. This capability is crucial for developing and refining algorithms before deploying them in real-world scenarios, ensuring they are both effective and reliable. The incorporation of simulators into the field of robotics research and development presents numerous critical advantages, markedly enhancing the efficiency, accuracy, and safety of experimental processes:

1- **Cost and Temporal Efficiency:** Simulators provide an invaluable resource for the testing and refinement of algorithms, circumventing the need for costly physical prototypes and extensive real-world trials. This virtual methodology significantly mitigates the financial and temporal constraints typically associated with robotic development, thereby expediting the innovation cycle while conserving resources.
2- **Creation of Diverse and Customizable Environments:** Simulators afford researchers the ability to construct and manipulate a broad spectrum of virtual environments, encompassing scenarios that are either logistically challenging, prohibitively expensive, or inherently hazardous to recreate in physical form. This capability is instrumental in developing and fine-tuning algorithms across a variety of operational contexts, thereby ensuring their adaptability and robustness under a wide range of conditions.
3- **Precision in Algorithmic Evaluation:** The controlled and highly detailed environments provided by simulators enable the meticulous assessment of algorithmic performance. The precision and comprehensiveness of data generated within these virtual settings facilitate rigorous evaluations, leading to more refined and effective optimizations. Consequently, algorithms are better prepared for deployment, exhibiting greater reliability and efficacy in real-world applications.
4- **Risk-Free Testing in High-Stakes Scenarios:** Simulators offer a secure platform for the exploration of algorithms within high-risk or unpredictable scenarios that could jeopardize both equipment and human safety in physical environments. This risk mitigation allows for thorough and exhaustive testing, ensuring that potential issues are identified and addressed without exposing assets or personnel to harm.
5- **Enhanced Pedagogical and Training Applications:** Beyond their research applications, simulators serve as sophisticated educational tools, offering researchers, engineers, and students an interactive and immersive environment in which to develop their skills. The hands-on experience provided by these simulators bridges the gap between theoretical knowledge and practical application, thereby fostering a deeper understanding of complex robotic systems.

6- **Consistency and Standardization:** The repeatability offered by simulators is paramount, allowing experiments to be conducted under consistent conditions, thereby facilitating the reliable comparison of different algorithms. This standardization ensures that testing procedures are uniform and that the results obtained are both valid and reproducible, thereby contributing to the robustness of the research findings.

Of course, it is worth noting that while simulators offer substantial advantages in developing and optimizing algorithms, their effective utilization requires careful consideration of several factors. First, the accuracy of simulation results is inherently dependent on the fidelity of the simulated environment. Inaccuracies in environmental modeling or simplifications made for computational efficiency can lead to discrepancies between simulated outcomes and real-world performance. Therefore, it is crucial to ensure that the simulation parameters and models are rigorously validated against empirical data to maintain reliability. Second, the complexity of integrating simulators with real-world systems must be taken into account. Simulators often operate under idealized conditions that may not fully capture the variability and unpredictability of real environments. Consequently, algorithms developed and tested in simulation may encounter unforeseen challenges when deployed in practice. Bridging this gap requires thorough testing in real-world scenarios and iterative refinement of both the simulation models and the algorithms. Third, the computational resources required for running high-fidelity simulations can be significant, particularly for complex scenarios involving large-scale systems or detailed physical models. Efficient resource management and optimization strategies are essential to balance simulation accuracy with computational feasibility. Finally, the expertise required to interpret simulation results and translate them into actionable insights should not be underestimated. Researchers must possess a deep understanding of both the simulation tools and the underlying algorithms to effectively leverage the data generated by simulations. From this perspective, in the realm of robotic soccer, the deployment of various simulators significantly accelerates the process of developing optimized solutions to intricate challenges. These advanced simulation tools enable the rigorous testing and evaluation of algorithms across a diverse array of environments and conditions, thereby enhancing the robustness and efficiency of the methodologies employed. Consequently, simulators serve as indispensable instruments not only for the advancement of algorithms tailored to robotic soccer but also for addressing broader, complex problems, contingent upon the effective execution of feature engineering processes.

5- **Conclusion**

Based on extensive research and analysis, the concept introduced in current paper is unprecedented within the scientific community. It is anticipated that this novel approach will catalyze significant foundational advancements in various problem-solving methodologies. This paper introduces a groundbreaking approach to task partitioning and scheduling by mapping these problems into the domain of obstacle avoidance and navigation in robotic soccer. We explicitly term this transformative methodology "**The RoboSoccer Mapping Trick**," capturing its unique capacity to leverage the complexities and dynamic nature of the RoboSoccer environment to redefine conventional task management strategies. The motivation for adopting this innovative approach stems from the multifaceted and intricate challenges inherent in the domain of robotic soccer, coupled with the unprecedented advancements and tools available in this field, which far surpass those in other areas. Upon gaining a comprehensive understanding of both problems and executing precise feature engineering, a mapping was established between the 9 features of the task scheduling and assignment problem and the 13 features of the obstacle avoidance and navigation problem in the Soccer-playing robot. This mapping was pivotal in facilitating a scaled-down implementation of the system. In this study, 9 advanced reinforcement learning algorithms and 10 classical algorithms were rigorously tested and evaluated. Utilizing principles and techniques from reinforcement learning, this mapping harnesses existing knowledge and tools in robotic soccer to enhance task scheduling performance within the foggy Internet of Everything (IoE) environment. Our evaluations reveal that this mapping not only significantly improves the performance of advanced algorithms in addressing complex issues but also leads to notable reductions in task completion time, enhanced resource allocation accuracy, better obstacle avoidance, and increased efficiency in cloud computing systems. These advancements are particularly impactful in dynamic and high-stress scenarios, where robotic soccer robots must adapt rapidly and make real-time decisions. The availability of various simulators within the robotic soccer field, offering realistic environments and diverse conditions,

further contributes to the optimization and refinement of algorithms. Therefore, this paper underscores the importance of aligning features with practical environments and highlights the role of simulators in advancing robotic soccer algorithms. Before the commencement of this study, the raw dataset underwent extensive preprocessing and refinement through various standard and conventional data analysis methods. The outcomes of these analyses, including the source code for preprocessing operations, data status charts before and after preprocessing, and the refined output dataset, have been made available to the broader research community for public use. During the experiments, highly detailed tables and analyses were provided, highlighting the performance of each algorithm type based on its applications and key attributes. It is anticipated that these resources will assist researchers in selecting appropriate methodologies tailored to the specific nature of their problems. Furthermore, the source codes for the tested algorithms, akin to those used in the preprocessing phase, have also been made accessible for public utilization. By focusing on task partitioning and scheduling within the robotic soccer context, this research introduces several key innovations:

1- **Mapping Task Partitioning and Scheduling to Robotic Soccer:** The primary innovation involves mapping task partitioning and scheduling problems to the robotic soccer environment. This approach applies scheduling and resource allocation algorithms in a dynamic and interactive setting, offering a novel and effective examination of these issues in a real-world context.
2- **Application of Reinforcement Learning Techniques:** The study explores the application of both classical and advanced reinforcement learning algorithms for task scheduling and partitioning. These techniques enable the algorithms to manage environmental complexities and optimize performance under dynamic conditions, enhancing their efficacy in real and complex environments.
3- **Comparative Analysis of Algorithm Performance:** The paper presents a detailed comparative analysis of algorithm performance with and without mapping to the robotic soccer environment. By evaluating various metrics, such as resource allocation accuracy, task completion time, obstacle avoidance success rate, and routing time, the study offers insights into the impact of mapping on algorithm performance, demonstrating significant improvements in efficiency and effectiveness.
4- **Utilization of Robotic Soccer Environment Features:** The research demonstrates that leveraging specific features of the robotic soccer environment, such as managing dynamic conditions and complex agent interactions, can enhance the learning process and optimize energy consumption. This innovative application of environmental features addresses task partitioning and scheduling challenges.
5- **Introduction of Key Features and Applications:** Extensive experimentation identifies key features of both classical and advanced scheduling algorithms, detailing their performance and applications in task partitioning, scheduling, and obstacle avoidance in robotic soccer.

These innovations provide a novel and effective methodology for tackling complex task scheduling and allocation problems in dynamic and intricate environments, thus significantly enhancing algorithm performance.

6- Future Work and Research Directions

In our recent previous work is detailed in reference [9] from 2024, we identified 216 eligibility traces for scoring goals in robotic soccer. As a future direction, we aim to build upon the effective mapping strategies outlined in this paper to address scheduling and task partitioning problems in a more specialized manner, using the rules and models developed in our prior research. Another crucial step following the resolution of the problem involves transitioning from the domain of the Soccer-playing robot back to the original task partitioning and scheduling problem space. This step is essential for revealing and interpreting the results within the context of task partitioning and scheduling. Given the complexities and specific conditions of the problem, we propose the use of **Multi-Core fuzzy** clustering as an effective method for returning to the original problem space. It is worth noting that a previous effort by the authors of this paper in [80], explored the application of **Multi-Core Fuzzy** clustering within the domain of robotic soccer. However, our current approach seeks to advance this methodology by introducing a more optimized and refined function, thereby enhancing its effectiveness in this specific field. This approach and the proof of its convergence function will be explored further in our future work. It is important to note that there are diverse areas for future research and development, following the innovative approach discussed in this paper. Some of these include:

1- **Investigation of New Mapping Methods**

- **Development of Novel Mapping Techniques:** Research and development of new methods for mapping scheduling and task partitioning problems to robotic environments, including the use of advanced modeling and simulation techniques, can enhance the accuracy and efficiency of algorithms.
- **Analysis of Mapping Methodologies:** Comparative evaluation and assessment of various mapping techniques for simulation and analysis of how these mappings impact system performance and optimization will help in identifying the strengths and weaknesses of each approach.

2- **Analysis and Improvement of Algorithm Performance at Larger Scales**

- **Scalability and Performance in Large-Scale Environments:** Investigating and optimizing algorithms for application in larger and more complex scenarios, with a focus on scalability and efficiency, can contribute to improved performance in large and data-intensive environments.
- **Handling High Volume and Complexity:** Evaluating and analyzing algorithm performance in high scalability scenarios and data-heavy environments will address scalability issues and identify the need for additional optimizations.

3- **Extension of Mapping Principles to Other Application Areas**

- **Application of Mapping Techniques to Different Domains:** Expanding the principles of mapping to solve problems in other application areas could lead to significant advancements in improving system performance and leveraging robotic techniques across various fields.

These research directions have the potential to drive substantial progress in enhancing system performance and optimizing the use of robotic techniques. They could lead to improvements in efficiency and effectiveness in complex and dynamic environments and contribute to the development of innovative methods in various domains.

Declarations

Ethical Approval

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors.

Competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Authors' contributions

Seyed Omid Azarkasb and Seyed Hossein Khasteh. These authors contributed equally to this work.

Authors and Affiliations

K.N. Toosi University of Technology, Tehran, Iran.

Seyed Omid Azarkasb

K.N. Toosi University of Technology, Tehran, Iran.

Seyed Hossein Khasteh

Corresponding author

Correspondence to Seyed Omid Azarkasb

Funding

This research was not funded.

Availability of data and materials

All implementation data and codes are available at this **GitHub repository**: https://github.com/Seyed-Omid-Azarkasb-Hesam-Alizadeh/Tasks-Management. The authors declare that all the experimental data in this paper are true and valid. Moreover, the authors declare that all experimental data are obtained from detailed experiments.

References

[1] James Macaulay, Lauren Buckalew, Gina Chung, **"Internet of Everything in Logistics"**, a collaborative report by DHL and Cisco on implications and use cases for the logistics industry, 2015.

[2] Han-Chieh Chao, Bin Hu, Chi-Yuan Chen, **"Fog Computing and Internet of Everything for Emerging Enterprise Information Systems"**, Enterprise Information Systems, Volume 12, Issue 4, Pages 371-372, 2018.

[3] PeiYun Zhang, MengChu Zhou, Giancarlo Fortino, **"Security and Trust Issues in Fog Computing: A Survey"**, Elsevier, Future Generation Computer Systems, Volume 88, Pages 16-27, 2018.

[4] Deep Rahul Shah, Dev Ajay Dhawan, Vijayetha Thoday, "**An Overview on Security Challenges in Cloud, Fog, and Edge Computing"**, Springer Link, Part of the Lecture Notes in Networks and Systems book series, Volume 462, Data Science and Security, Pages 337-345, 2022.

[5] Hossein Ashtari, "**Edge Computing vs. Fog Computing: 10 Key Comparisons"**, https://www.spiceworks.com/tech/cloud/articles/edge-vs-fog-computing, 2022.

[6] Zhiqiang Pu, Yi Pan, Shijie Wang, Boyin Liu, Min Chen, Hao Ma, Yixiong Cui, "**Orientation and Decision-Making for Soccer Based on Sports Analytics and AI: A Systematic Review"**, IEEE/CAA Journal of Automatica Sinica, Volume 11, Issue 1, Pages 37-57, 2024.

[7] António Fernando Alcântara Ribeiro, Ana Carolina Coelho Lopes, Tiago Alcântara Ribeiro, Nino Sancho Sampaio Martins Pereira, Gil Teixeira Lopes, António Fernando Macedo Ribeiro, **"Probability-Based Strategy for a Soccer Multi-Agent Autonomous Robot System"**, MDPI, Robotics, Volume 13, Issue 1, 2024.

[8] RoboCup, **"The RoboCup Soccer Simulator"**, https://rcsoccersim.github.io, Accessed: 2024.

[9] Seyed Omid Azarkasb, Seyed Hossein Khasteh, **"Eligibility Traces in an Autonomous Soccer Robot with Obstacle Avoidance and Navigation Policy"**, Elsevier, Applied Soft Computing, Article ID 111889, 2024.

[10] Sundas Iftikhar, Mirza Mohammad Mufleh Ahmad, Shreshth Tuli, Deepraj Chowdhury, Minxian Xu, Sukhpal Singh Gill, Steve Uhlig, **"HunterPlus: AI Based Energy-Efficient Task Scheduling for Cloud–Fog Computing Environments"**, Elsevier, Internet of Things, Volume 21, Article ID 100667, 2023.

[11] AR. Arunarani, D. Manjula, Vijayan Sugumaran, **"Task Scheduling Techniques in Cloud Computing: A Literature Survey"**, Elsevier, Future Generation Computer Systems, Volume 91, Pages 407-415, 2019.

[12] Mostafa Ghobaei-Arani, Alireza Souri, Ali A. Rahmanian, **"Resource Management Approaches in Fog Computing: a Comprehensive Review"**, Springer Link, Journal of Grid Computing, Volume 18, Issue 1, Pages 1-42, 2020.

[13] Ali Asghari, Mohammad Karim Sohrabi, Farzin Yaghmaee, **"Task Scheduling, Resource Provisioning, and Load Balancing on Scientific Workflows Using Parallel SARSA Reinforcement Learning Agents and Genetic Algorithm"**, Springer Link, The Journal of Supercomputing, Volume 77, Issue 3, Pages 2800-2828, 2021.

[14] Jessica Cot Palacio, Yailen Martínez Jiménez, Leander Schietgat, Bart Van Doninck, Ann Nowéc, **"A Q-Learning Algorithm for Flexible Job Shop Scheduling in a Real-World Manufacturing Scenario"**, Elsevier, Procedia CIRP, Volume 106, Pages 227-232, 2022.

[15] Behrooz Bodaghi, Shahrooz Shahparvari, Masih Fadaki, Kwok Hung Lau, Palaneeswaran Ekambaram, Prem Chhetri, **"Multi-Resource Scheduling and Routing for Emergency Recovery Operations"**, Elsevier, International Journal of Disaster Risk Reduction, Volume 50, Article ID 101780, 2020.

[16] Athanassios M. Kintsakis, Fotis E.Psomopoulos, Pericles A. Mitkas, **"Reinforcement Learning Based Scheduling in a Workflow Management System"**, Elsevier, Engineering Applications of Artificial Intelligence, Volume 81, Pages 94-106, 2019.

[17] Bing Lin, Qiaoxin Chen, Yu Lu, **"Time-Driven Scheduling Based on Reinforcement Learning for Reasoning Tasks in Vehicle Edge Computing"**, Hindawi, Wireless Communications and Mobile Computing, 2022.

[18] Xiaolan Liu, Jiadong Yu, Zhiyong Feng, Yue Gao, **"Multi-Agent Reinforcement Learning for Resource Allocation in IoT Networks with Edge Computing"**, China Communications, Volume 17, Issue 9, Pages 220-236, 2020.

[19] Xiong Xiong, Kan Zheng, Lei Lei, Lu Hou, **"Resource Allocation Based on Deep Reinforcement Learning in IoT Edge Computing"**, IEEE Journal on Selected Areas in Communications, Volume 38, Issue 6, Pages 1133-1146, INSPEC Accession Number 19632726, 2020.

[20] Shuran Sheng, Peng Chen, Zhimin Chen, Lenan Wu, Yuxuan Yao, **"Deep Reinforcement Learning-Based Task Scheduling in IoT Edge Computing"**, MDPI, Sensors, Volume 21, Issue 5, 19 Pages, 2021.

[21] Jungyeon Baek, Georges Kaddoum, **"Heterogeneous Task Offloading and Resource Allocations via Deep Recurrent Reinforcement Learning in Partial Observable Multifog Networks"**, IEEE Internet of Things Journal, Volume 8, Issue 2, Pages 1041-1056, INSPEC Accession Number 20324615, 2021.

[22] Ricardo S.Alonso, Ines Sitton-Candanedo, Roberto Casado-Vara, Javier Prieto, Juan M.Corchado, **"Deep Reinforcement Learning for the Management of Software-Defined Networks and Network Function Virtualization in an Edge-IoT Architecture"**, MDPI, Sustainability, Volume 12, Issue 14, 2020.

[23] Almuthanna T. Nassar, Yasin Yilmaz, **"Reinforcement Learning-Based Resource Allocation in Fog RAN for IoT with Heterogeneous Latency Requirements"**, Cornell University, Computer Science, Networking and Internet Architecture, arXiv: 1806.04582, 11 Pages, 2019.

[24] Tao Zheng, Jian Wan, Jilin Zhang, Congfeng Jiang, **"Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing"**, Springer Link, Journal of Cloud Computing, Volume 11, 2022.

[25] Chathurangi Shyalika, Thushari Silva, Asoka Karunananda, **"Reinforcement Learning in Dynamic Task Scheduling: A Review"**, Springer Link, SN Computer Science, Volume 1, 17 Pages, 2020.

[26] Alexandru Iulian Orhean, Florin Pop, Ioan Raicu, **"New Scheduling Approach using Reinforcement Learning for Heterogeneous Distributed Systems"**, Elsevier, Journal of Parallel and Distributed Computing, Volume 117, Pages 292-302, 2018.

[27] Xi Xiu, Jialun Li, Yujie Long, Weigang Wu, **"MRLCC: An Adaptive Cloud Task Scheduling Method Based on Meta Reinforcement Learning"**, Springer Link, Journal of Cloud Computing, Advances, Systems and Applications, Volume 12, Article number: 7, 2023.

[28] Zhiyu Wang, Mohammad Goudarzi, Mingming Gong, Rajkumar Buyya, "**Deep Reinforcement Learning-Based Scheduling for Optimizing System Load and Response Time in Edge and Fog Computing Environments"**, Elsevier, Future Generation Computer Systems, Volume 152, Pages 55-69, 2024.

[29] Guangyao Zhou, Wenhong Tian, Rajkumar Buyya, Ruini Xue & Liang Song, "**Deep Reinforcement Learning-Based Methods for Resource Scheduling in Cloud Computing: A Review and Future Directions"**, Springer Link, Artificial Intelligence Review, Volume 57, Article number 124, 2024.

[30] Seyed Mehdi Hazrati Fard, Ali Hamzeh, Sattar Hashemi, **"Using Reinforcement Learning to Find An Optimal Set of Features"**, Elsevier, Computers & Mathematics with Applications, Volume 66, Issue 10, Pages 1892-1904, 2013.

[31] Minoru Asada, Hiroaki Kitano, Itsuki Noda, Manuela Veloso, **"RoboCup: Today and Tomorrow-What We Have Learned"**, Elsevier, Artificial Intelligence 110, Pages 193-214, 1999.

[32] Gerald Steinbauer-Wagner, Alexander Ferrein, **"20 Years of RoboCup"**, Springer Link, Ki - Künstliche Intelligenz, Volume 30, Pages 225–232, 2016.

[33] Andries Smit, Herman A. Engelbrecht,Willie Brink, Arnu Pretorius, **"Scaling multi-agent reinforcement learning to full 11 vs 11 simulated robotic football"**, Springer Link, Autonomous Agents and Multi-Agent Systems, Volume 37, Article Number 20, 2023.

[34] Yun Li, Yibin Song, Amin Rezaeipanah, **"Generation a Shooting on the Walking for Soccer Simulation 3D League using Q-Learning Algorithm"**, Springer Link, Journal of Ambient Intelligence and Humanized Computing, Volume 14, Pages 6947-6957, 2023.

[35] Haein Jeon, Dae-Won Kim, Bo-Yeong Kang, **"Deep Reinforcement Learning for Cooperative Robots Based on Adaptive Sentiment Feedback"**, Elsevier, Expert Systems with Applications, Volume 243, Article ID 121198, 2024.

[36] Zhengqiao Wang, Yufan Zeng, Yue Yuan, Yibo Guo, **"Refining Co-operative Competition of Robocup Soccer with Reinforcement Learning"**, IEEE Fifth International Conference on Data Science in Cyberspace (DSC), Pages 279-283, Hong Kong, China, 2020.

[37] Ping-Huan Kuo, Wei-Cyuan Yang, Po-Wei Hsu, Kuan-Lin Chen, **"Intelligent Proximal-Policy-Optimization-Based Decision-Making System for Humanoid Robots"**, Elsevier, Advanced Engineering Informatics, Volume 56, Article ID 102009, 2023.

[38] Pengzhan Chen, Weiqing Lu, **"Deep Reinforcement Learning Based Moving Object Grasping"**, Elsevier, Information Sciences, Volume 565, Pages 62-76, 2021.

[39] Andrew Farley, Jie Wang, Joshua A. Marshall, **"How to Pick a Mobile Robot Simulator: A Quantitative Comparison of CoppeliaSim, Gazebo, MORSE and Webots with a Focus on**

Accuracy of Motion", Elsevier, Simulation Modelling Practice and Theory, Volume 120, Article ID 102629, 2022.

[40] N. Koenig, N., A. Howard, **"Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator"**, In Proceedings of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Sendai, Japan, Volume 3, Pages 2149-2154, 2004.

[41] Lingli Yu, Keyi Li, Shuxin Huo, Kaijun Zhou, **"Cooperative Offensive Decision-Making for Soccer Robots Based on Bi-Channel Q-Value Evaluation MADDPG"**, Elsevier, Engineering Applications of Artificial Intelligence, Volume 121, Article ID 105994, 2023.

[42] Trung Thanh Nguyen, Tomi Silander, Zhuoru Li, Tze-Yun Leong, **"Scalable Transfer Learning in Heterogeneous, Dynamic Environments"**, Elsevier, Artificial Intelligence, Volume 247, Pages 70-94, 2017.

[43] Márton Szemenyei, Vladimir Estivill-Castro , "**Fully Neural Object Detection Solutions for Robot Soccer**", Springer Link, Neural Computing and Applications, Volume 34, Pages 21419-21432, 2022.

[44] Zepeng Ning, Lihua Xie, "**A Survey on Multi-Agent Reinforcement Learning and Its Application"**, Elsevier, Journal of Automation and Intelligence, Volume 3, Issue 2, Pages 73-91, 2024.

[45] Yong Duan, Bao Xia Cui, Xin He Xu, **"A Multi-Agent Reinforcement Learning Approach to Robot Soccer"**, Springer Link, Artificial Intelligence Review, Volume 38, Pages 193-211, 2012.

[46] Si Bruno Brandão, Telma Woerle De Lima, Anderson Soares, Luckeciano Melo, Marcos R. O. A. Maxmulation, **"Multiagent Reinforcement Learning for Strategic Decision Making and Control in Robotic Soccer Through Self-Play"**, IEEE Access, Volume 10, Pages 72628 -72642, 2022.

[47] Wei Zhan, Shengqing Qu, **"Cooperation Mode of Soccer Robot Game Based on Improved SARSA Algorithm"**, Hindawi, Wireless Communications and Mobile Computing, License: CC BY 4.0, Vol. 2022, Article ID 9190687, 11 Pages, 2022.

[48] Felipe Leno Da Silva, Anna Helena Reali Costa, Peter Stone, **"Distributional Reinforcement Learning Applied to Robot Soccer Simulation"**, In Proceedings of International Conference Adaptive Learning Agents (ALA) workshop at AAMAS, Montreal, 2019.

[49] Abebaw Degu Workneh, Maha Gmira, **"Deep Q Network Method for Dynamic Job Shop Scheduling Problem"**, Springer Link, Part of the book series: Lecture Notes in Networks and Systems LNNS, International Conference on Artificial Intelligence & Industrial Applications, Volume 771, Pages 137-155, 2023.

[50] Lei Zeng, Qi Liu, Shigen Shen, Xiaodong Liu, **"Improved Double Deep Q Network-Based Task Scheduling Algorithm in Edge Computing for Makespan Optimization"**, IEEE Tsinghua Science & Technology, Volume 29, Issue 3, Pages 806-817, 2024.

[51] Sheng Chai, Jimmy Huang, **"Dependent Task Scheduling Using Parallel Deep Neural Networks in Mobile Edge Computing"**, Springer Link, Journal of Grid Computing, Volume 22, Issue 1, 2024.

[52] Yanlang Zheng, Huan Zhou, Rui Chen, Kai Jiang, Yue Cao, **"SAC-based Computation Offloading and Resource Allocation in Vehicular Edge Computing"**, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2022.

[53] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, Philipp Moritz, **"Trust Region Policy Optimization"**, Proceedings of the 32nd International Conference on Machine Learning, PMLR 37, Pages 1889-1897, 2015.

[54] Jaehoon Chung, Jamil Fayyad, Younes Al Younes, Homayoun Najjaran, "**Learning Team-Based Navigation: a Review of Deep Reinforcement Learning Techniques for Multi-Agent Pathfinding**", Springer Link, Artificial Intelligence Review, Volume 57, Issue 2, License CC BY 4.0, 2024.

[55] Bushra Jamil, Humaira Ijaz, Mohammad Shojafar, Kashif Munir, **"IRATS: A DRL-Based Intelligent Priority and Deadline-Aware Online Resource Allocation and Task Scheduling Algorithm in a Vehicular Fog Network"**, Elsevier, Ad Hoc Networks, Volume 141, Article ID 103090, 2023.

[56] Ebrahim Hamid Sumiea, Said Jadid Abdulkadir, Hitham Seddig Alhussian, Safwan Mahmood Al-Selwi, Alawi Alqushaibi, Mohammed Gamal Ragab, Suliman Mohamed Fati, **"Deep Deterministic Policy Gradient Algorithm: A Systematic Review**", Elsevier, Heliyon, Volume 10, Issue 9, Article ID e30697, License CC BY-NC-ND 4.0, 2023.

[57] Bo Li, Zhi-peng Yang, Da-qing Chen, Shi-yang Liang, Hao Ma, **"Maneuvering Target Tracking of UAV Based on MN-DDPG and Transfer Learning"**, Elsevier, Volume 17, Issue 2, Pages 457-466, License CC BY-NC-ND 4.0, 2021.

[58] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, David Silver, **"Rainbow: Combining Improvements in Deep Reinforcement Learning"**, Proceedings of the AAAI Conference on Artificial Intelligence, Volume 32, Issue 1, 2018.

[59] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, **"Asynchronous Methods for Deep Reinforcement Learning"**, Proceedings of The 33rd International Conference on Machine Learning, PMLR, Volume 48, Pages 1928-1937, 2016.

[60] Mangalampalli Sudheer, K. Ganesh Reddy, Sachi Nandan Mohanty, Shahid Ali, **"Multi-Objective Prioritized Task Scheduler Using Improved Asynchronous Advantage Actor Critic (A3C) Algorithm in Multi Cloud Environment"**, IEEE Access, Volume 12, Issue 99, Pages 11354-11377, 2024.

[61] Tilo Strobach, Elisabeth Hendrich, Sebastian Kübler, Hermann Müller, Torsten Schubert, **"Processing Order in Dal-Task Situations: The "Frst-Come, First-Served" Principle and the Impact of Task Order Instructions"**, Springer Link, Attention, Perception, & Psychophysics, Volume 80, Pages 1785-1803, 2018.

[62] Saqib Ul Sabha, **"A Novel and Efficient Round Robin Algorithm with Intelligent Time Slice and Shortest Remaining Time First"**, Elsevier, Materials Today Proceedings, Volume 5, Issue 5, Part 2, Pages 12009-12015, 2018.

[63] Prakhar Ojha, Siddhartha R Thota, Vani M, Mohit P Tahilianni, **"Learning Scheduler Parameters for Adaptive Preemption"**, Fourth International Conference on Advanced Information Technologies and Applications, Volume 5, Issue 15, Pages 149-162, License CC BY-NC 4.0, 2015.

[64] Xin Jin, LianSong Yu, **"Research and Implementation of High-Priority Scheduling Algorithm Based on Intelligent Storage of Power Materials"**, Elsevier, Energy Reports, Volume 8, Supplement 6, Pages 398-40, 7th International Conference on Advances in Energy Resources and Environment Engineering (ICAESEE 2021), Guangzhou, China, 2022.

[65] Héctor Pérez, J. Javier Gutiérrez, **"EDF Scheduling for Distributed Systems Built Upon the IEEE 802.1AS Clock - A Theoretical-Practical Comparison"**, Elsevier, Journal of Systems Architecture, Volume 132, Article ID 102742, 2022.

[66] Chengxi Gao, V C S Lee, Ke-qin Li, **"D-SRTF: Distributed Shortest Remaining Time First Scheduling for Data Center Networks"**, IEEE Transactions on Cloud Computing, Volume 9, Issue 2, Pages 562-575, 2021.

[67] Haohao Wang, Mengmeng Sun, Lianming Zhang, Pingping Dong, Yehua Wei, Jing Mei, **"Scheduling Optimization for Upstream Dataflows in Edge Computing"**, Elsevier, Digital Communications and Networks, Volume 9, Issue 6, Pages 1448-1457, 2023.

[68] Dharamendra Chouhan, S. M. Dilip Kumar, B. P. Vijaya Kumar, **"Multilevel Feedback Queue Scheduling Technique for Grid Computing Environments"**, Springer Link, Proceedings of International Conference on Advances in Computing, Part of the book series: Advances in Intelligent Systems and Computing, AISC, Volume 174, Pages 1-7, 2012.

[69] Ya-Shu Chen, Li-Pin Chang, Tei-Wei Kuo, Aloysius K. Mok, **"An Anomaly Prevention Approach for Real-Time Task Scheduling"**, Elsevier, Journal of Systems and Software, Volume 82, Issue 1, Pages 144-154, 2009.

[70] Prassanna Jayachandran, Neelanarayanan Venkataraman, **"Adaptive Regressive Holt–Winters Workload Prediction and Firefly Optimized Lottery Scheduling for Load Balancing in Cloud"**, Springer Link, Wireless Networks, Volume 27, Issue 8, Pages 5597-5615, 2021.

[71] Redown Mahmud, Samodha Pallewatta, Mohammad Goudarzi, Rajkumar Buyya, **"iFogSim2: An Extended iFogSim Simulator for Mobility, Clustering, and Microservice Management in Edge and Fog Computing Environments"**, Elsevier, Journal of Systems and Software, Volume 190, Article ID 111351, 2022.

[72] Mohammed Alaa Ala'anzy, Mohamed Othman, Mohd Hanapi Zurina, Mohamed A Alrshah, "**Locust Inspired Algorithm for Cloudlet Scheduling in Cloud Computing Environments"**, Sensors, MDPI, Volume 21, Issue 21, 19 Pages, 2021.

[73] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh, Rajkumar Buyya, **"iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing Environments"**, Cloud and Fog Computing, Volume 47, Issue 9, Pages 1275-1296, Wiley Online Library, 2017.

[74] Norbert Batfai, Roland Doczi, Janos Komzsik, Andras Mamenyak, Csaba Szekelyhdi, Jozsef Zakany, Marton Ispany, Gyorgy Terdik, **"Applications of a Simplified Protocol of RoboCup 2D Soccer Simulation"**, Infocommunications Journal, Volume V, 2013.

[75] Kenta Takaya, Toshinori Asai, Valeri Kroumov, Florentin Smarandache, "**Simulation Environment for Mobile Robots Testing Using ROS and Gazebo"**, 20th IEEE International Conference on System Theory, Control and Computing, Pages 96-101, Sinaia, Romania, 2016.

[76] Daniel H. Stolfi, Grégoire Danoy, **"Design and Analysis of an E-Puck2 Robot Plug-in for the ARGoS Simulator"**, Elsevier, Robotics and Autonomous Systems, Volume 164, Article ID 104412, 2023.

[77] Iliyas Tursynbek, Almas Shintemirov, **"Modeling and Simulation of Spherical Parallel Manipulators in CoppeliaSim (V-REP) Robot Simulator Software"**, IEEE International Conference Nonlinearity, Information and Robotics, Innopolis, Russia, 2020.

[78] Eric Bernd Gil, Genaína Nunes Rodrigues, Patrizio Pelliccione, Radu Calinescu, **"Mission Specification and Decomposition for Multi-Robot Systems"**, Elsevier, Robotics and Autonomous Systems, Volume 163, Article ID 104386, 2023.

[79] Yuan Xu, Hedayat Vatankhah. **"SimSpark: An Open Source Robot Simulator Developed by the Robocup Community"**, Springer Link, Part of the book series: Lecture Notes in Computer Science, LNAI, Volume 8371, RoboCup 2013: Robot Soccer World Cup XVII, Pages 632-639, Berlin, Heidelberg, 2014.

[80] Seyed Omid Azarkasb, Seyed Hossein Khasteh, **"A New Approach for Mapping of Soccer Robot Agents Position to Real Filed Based on Multi-Core Fuzzy Clustering"**, IEEE 26th International Computer Conference, Computer Society, 2021

Seyed Omid Azarkasb received his B.Sc. degree in computer software engineering from Kashan Branch Azad University in 1996 and 2001. He studied artificial intelligent systems at Qazvin University of Technology and got his M.Sc. in 2008. Now, he is a Visiting Pofessor and Ph.D. student at K. N. Toosi University of Technology, Tehran, Iran.
His research interests include Robotics, Intrusion detection systems, Machine learning methods, Internet of Everything (IoE), Digital Transformation, Fog and Cloud Computing,

Seyed Hossein Khasteh received the B.Sc. degree in electrical engineering, the M.Sc. degree in Artificial Intelligence, and the Ph.D. degree in Artificial Intelligence all from the Sharif University of Technology, Tehran, Iran.
He is currently an Assistant Professor with the Computer Engineering Department, K. N. Toosi, University of Technology, Tehran, Iran.
His current research interests include Social Network Analysis, Machine Learning, and Big Data Analysis.