UNIVERSITY OF SOUTH AUSTRALIA

Information Technology

RESEARCH PROPOSAL

PhD Candidate:
YUGU LI

Supervisor:
Jimmy Cao

Co-Supervisor:
Ryszard Kowalcyzk
Dhika Pratama

Explicit and Implicit Knowledge Representation for Reinforcement Learning

Yugu Li

Contents

Abstract

My PhD project studies the coupling of knowledge representations with reinforcement learning to accelerate its learning. RL is a powerful framework for optimizing decision-making in dynamic environments, however, it is often plagued by high sampling costs, low sample efficiency, and limited robustness in complex problems. To tackle these challenges, this study investigates the adoption of explicit vs implicit knowledge and the consequences of explicit knowledge in multi-agent RL (MARL) and few-shot learning settings. My research aims to tackle three fundamental challenges: (1) the definition of structured explicit knowledge that facilitates RL exploration, (2) the capabilities required for agents to achieve autonomous reasoning and implicit knowledge extraction, and (3) the utilization of knowledge-rich samples for few-shot learning. Solutions leverage game theory notions such as the nucleolus(various concepts) for credit assignment in MARL, apply Monte Carlo tree search to enhance reasoning in large language models(classically), and utilize reinforced fine-tuning for making the agent learn optimally with few pieces of data(derivatives based) This work aims to improve sample efficiency, computational costs, agent reasoning and generalization, by developing process for converting prior knowledge into mechanism of integration. Such findings will directly contribute to narrowing the gap between theoretical RL pursuit and practical usage of RL, paving the way towards real world applications such as robotics, autonomous systems, and intelligent decision-making.

***Keywords*—** MARL, LLM, Reasoning, Few-Shot

1 Introduction

Artificial Intelligence has its origins in the 1950s. As early as 1943, McCulloch and Pitts mathematically modeled artificial neural networks [1], providing a theoretical foundation for the development of AI. In 1950 Alan Turing [2] suggested the classical measurement of the intelligence of the machine, the "Turing test", that is: Can the human being to determine that the person he is talking to is a human or a robot. The term “artificial intelligence” was first coined at the Dartmouth Conference in 1956, which is considered the official birth of AI. AI has since development in stages from symbolic to knowledge driven and data driven [3–6]. The symbolist stage was based on logical reasoning and rules systems, while statistical machine learning and data based AI approaches became dominant after the 1990s [7–10]. In the 2010s the rapid rise of deep learning based on neural networks has been made possible by the development of computational images. AlexNet [11] achieved a breakthrough performance based on convolutional neural networks in the 2011 ImageNet Large-scale Visual Recognition Challenge for image classification tasks, with the online Top-5 classification error rate decreased by nearly 10 percentage points to 15.3% over traditional methods (26.2% error). This breakthrough result stunned academia and industry, became one of the watershed events in the history of artificial intelligence, and directly led to heavy attention on deep learning. Then in 2016, The deepmind team defeated Lee Sedol with AlphaGo [12] based on reinforcement learning (RL), was not only a technical milestone, but also gave the public a deeper understanding of the potential of AI.

1.1 Background

RL [13] is a machine learning methodology that simulates how an agent learns to make optimal decisions by interacting with its environment. The core goal is to optimize the agent's decision-making policy through exploratory actions and feedback, ultimately maximizing cumulative rewards. Specifically, at a given state, the agent selects an action based on its current policy, interacts with the environment, and receives a reward signal that reflects the outcome of its action. By continuously repeating this process, the agent iteratively refines its policy to perform tasks more effectively. A key concept in RL is the **trajectory**, which records the complete sequence of states s, actions a, and rewards r generated as the agent interacts with the environment. Trajectories serve as critical data for training models and evaluating policies. Each trajectory begins from an initial state, and as the agent takes actions, the environment transitions between states while producing rewards. A trajectory can be expressed as a sequence: $(s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T)$. By analyzing the distribution of cumulative rewards and decision paths within trajectories, an agent can identify more effective strategies, enabling it to perform better in increasingly complex tasks. RL is recognized for its flexibility and adaptability, making it particularly well-suited for solving high-dimensional, dynamic, and complex optimization problems. In practice, RL has become an indispensable tool across various fields, including Game AI [14], Autonomous Driving [15], Robotic Control [16], Financial Trading [17] and so

on.

As algorithms and computational power continue to evolve, RL is expected to deliver even more outstanding results in addressing complex problems. The future of RL lies in its integration with other advanced techniques. This combination will enable RL to solve decision-making problems with greater creativity and efficiency. Key research directions include incorporating human prior knowledge to improve sample efficiency, reducing computational costs, and enhancing system safety and robustness. These advancements will further broaden RL's potential applications and increase its impact across various domains.

1.2 Research Motivation

RL relies heavily on the agent's interaction with the environment to generate trajectories. This process can be highly exploratory, particularly in environments with large state and action spaces or high complexity. Purely model-free RL methods often depend on random exploration, leading to significant challenges such as high sampling cost and low sample efficiency. High sampling may generate trajectories often requires extensive interaction, which can be computationally and time-intensive. Low sample efficiency is mean many trajectories provide limited actionable insights for policy optimization.

Addressing these gaps requires the generation of higher-quality trajectories and more efficient utilization of collected trajectories. Our motivation is to improve the learning efficiency of agents by introducing prior knowledge into the RL process, which has several advantages as below:

1. Reducing the search space: Prior knowledge helps narrow down the set of states and actions the agent needs to explore, making the search process more targeted and efficient.

2. Guiding the initial policy: By leveraging prior knowledge, we can accelerate the transition of the policy from random and ineffective to structured and efficient, significantly improving early-stage performance.

3. Enhancing sample efficiency: Prior knowledge reduces the number of interactions required with the environment by enabling the agent to learn more effectively from each trajectory.

4. Improving robustness: Incorporating prior knowledge helps prevent catastrophic errors in specific tasks by embedding domain constraints and rules into the agent's decision-making process.

By leveraging these advantages, the integration of prior knowledge into RL has the potential to significantly improve learning efficiency, reduce computational costs, and enhance the agent's performance across diverse and challenging environments.

1.3 Research Problems

My research addresses three critical questions aimed at improving the efficiency and effectiveness of RL by integrating knowledge into the learning process:

1. **Explicit Knowledge Representation for Agents Efficient Learning**:

   How can explicit knowledge representations be effectively incorporated to enable agents to learn more efficiently? This involves defining clear, structured knowledge inputs, such as domain rules, heuristics, or task constraints, that directly guide the agent's exploration and decision-making process, reducing the burden of random exploration.

2. **Implicit Knowledge Representation and Autonomous Reasoning**:

   How can agents autonomously discover and utilize implicit knowledge embedded in their interactions with the environment? This question focuses on enabling agents to extract and reason about latent patterns, relationships, or principles hidden in trajectories and observations. By uncovering such implicit knowledge, agents can enhance their reasoning capabilities and improve learning efficiency.

3. **Few-shot Learning with Knowledge-rich Samples**:

   How can agents effectively extract and leverage intrinsic knowledge from a small set of knowledge-rich samples to achieve few-shot learning? This involves designing methods that allow agents to generalize from limited data, identifying the underlying principles or strategies embedded in these samples to perform efficiently in new or unseen scenarios.

By addressing these research problems, I aim to advance the state-of-the-art in RL with in MAS and LLM, making it more practical, sample-efficient, and capable of reasoning and generalization in complex environments.

1.4 The Route of Research Proposal

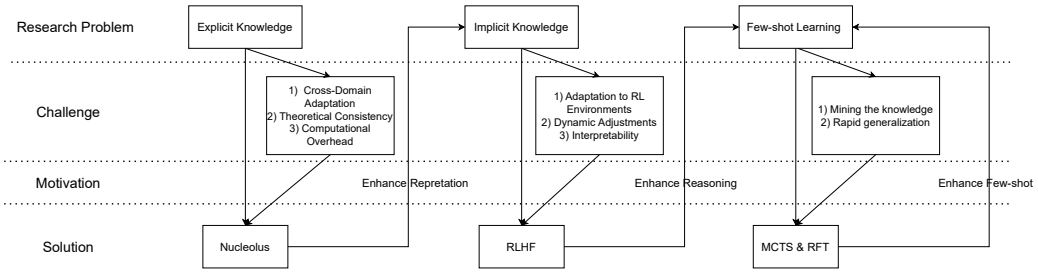

Figure 1: Research Route

2 Literature Reviews

2.1 Reinforcement Learning

2.1.1 Markov Games

The decentralized partially observable Markov decision process (dec-POMDP) [18–20] is a key framework for modeling multi-agent decision making problems in reinforcement learning, being one of the driving forces of the area. The Dec-POMDP for a system with n agents is defined as

$$\langle \mathcal{S}, \{\mathcal{A}i\}i = 1^n, P, R, \{\mathcal{O}i\}i = 1^n, \gamma \rangle.$$

where:

- $\mathcal{S}$: State space of the environment

  $\mathcal{A}_i$: agent i's action space, $i = 1, 2, \ldots, n$
- P: The state transition kernel which describes the transition of a state to the next state $s_{t+1} \sim P(\cdot|s_t, a_t)$, where a_t is joint action, $a_t \in \mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$.
- R: Reward function based on current state s_t and joint action a_t, denoted by $r_t = R(s_t, a_t)$
- $\{\mathcal{O}i\}i = 1^n$**:** The collection of observation spaces for each agent, where agent i receives local observations, $o_i^t \sim o_i(\cdot|s_t)$.
- $\gamma \in [0, 1)$ is the discount factor, which specifies the value of future rewards.

In this framework,agents are partially observables which means each agent can only access its own local observation o_i^t and needs to make decision using its independent policy π^i. Each agent i seeks to optimize its policy π^i by maximizing its expected discounted total reward:

$$J^i(\pi^i) = \mathbb{E}_{\pi^1,\ldots,\pi^n}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i(s_t, a_t^1, \ldots, a_t^n)\right], \quad i = 1, 2, \ldots, n. \tag{1}$$

In a fully cooperative game, all agents share the same reward, which means their objectives are perfectly aligned. This leads to the worst behaviour of one agent perfectly aligns with the best behaviour of all the others. Coordinating learning becomes simpler since all agents are trying to maximize the same global reward [21].

2.1.2 Deep Q-learning

The learning objective is to learn an optimal control policy that maximizes the action-value function $Q(S_t, A_t)$ [22] in single-agent reinforcement learning (RL) problems. Once Q is defined, this becomes an optimization problem. The Bellman equation tells us how to update Q values:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\big]. \tag{2}$$

So the basic form of the Q-learning algorithm uses a lookup table to determine the maximum expected future rewards for each action given each state. However, many real-world reinforcement learning problems involve continuous and high-dimensional state and action spaces. In such cases, maintaining a Q table is impractical or too expensive. This limitation resulted in Q-learning with deep neural networks, which is popularly known as the Deep Q Network (DQN) [14].

DQN iterates over the parameters θ of a deep neural network based on the action-values (feedback) through the loss function to reach its target for optimal control. It uses two networks, namely, a target network and a predictive network. They utilize a target network parameterized by θ^- for approximating the true value and a predictive network (with parameterization θ) for learning. Following are what the update rule for DQN looks like:

$$Q(S_t, A_t; \theta) \leftarrow Q(S_t, A_t; \theta) + \alpha\big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta^-) - Q(S_t, A_t; \theta)\big], \tag{3}$$

and the loss function is defined as:

$$L(\theta) = \big[R_t + \gamma \max_{a_{t+1}} Q(S_{t+1}, a_{t+1}; \theta^-) - Q(S_t, A_t; \theta)\big]^2. \tag{4}$$

By using a deep neural network to approximate the Q values, DQN can return actionable outputs with far less computation than was required by the original Q-learning algorithm, making it effective in real-world applications.

2.2 Mulit-agents Reinforcement Learning

In multi-agent RL, strategies for single agents are generated via RL, so that agents can work together to coordinate their actions. The end result is cooperation and reaching common goals. This learning paradigm has widespread applications, such as robotic co-operations, drone formation control [23] and intelligent transportation systems [24]. The research in MARL mainly follows the four directions as below:

1. **Value Decomposition-Based Methods**: In multi-agent environments, agents often learn from a shared global reward signal. Nonetheless, this signal tends to inadequately reflect each agent's contribution to the final performance. This challenge

demands effective reward allocation methods that disaggregate the global reward to derive specific agent reward.

2. **Coordination and Communication**: In multi-agent environments, coordination and communication between agents are of utmost importance for achieving the successful completion of collaborative tasks. This line of research is all about creating communication protocols and decision-making schemas that facilitate cooperative behavior among agents that are either static or have restricted communication channels.

3. **Scalability and Robustness**: The large scale and complexity involved in multi-agent systems pose significant challenges to algorithms, especially when an environment is dynamic or when information may be only partially lost. A major area of research is to achieve system stability and efficiency in such high-stress scenarios.

4. **Exploration and Exploitation**: In multi-agent systems, agents need to refine their strategies while concurrently learning the behavior of other agents, in both cooperative and adversarial environments. Balancing exploration and exploitation is still a major challenge in MARL. This proposed structured research framework helps pave the way for future research directions for advancing MARL applications to a variety of complex and diverse domains.

In the **Appendix A.1**, I further introduce three common training frameworks [23, 25, 26] in MARL as well as some classical algorithms [27–29].

2.2.1 Game Theory in MARL

Core: In cooperative game theory, the Core [30] is the set of feasible allocations or imputations where no coalition of agents can benefit by breaking away from the grand coalition. For MDP, [31] extended the definition of the core in cooperative game theory, called Markov core. It can assess whether the payoff distribution during the learning process in an MDP meets certain conditions, specifically that the current payoff prevents all agents from having the incentive to leave the current coalition and form new coalitions to obtain greater rewards. Formally, the Markov core (MC) can be modified as follows:

$$MC = \left\{ \left(\max_{a_i} x_i(s, a_i) \right)_{i \in N} \middle| \max_{a_C} x(s, a_C|C) \geq \max_{a_C} v(s, a_C), \forall C \subseteq N, \forall s \in S \right\} \quad (5)$$

where $x_i(s, a_i)$ represent the imputation of agent i by given state s and action a_i, $x(s, a_C|C) = \sum_{i \in C} x_i(s, a_i)$ is the imputation of coalition C by given state s and joint action $a_C \in A_C = \times_{i \in C} A_i$, and $v(s, a_C) \in \mathbb{R}_0^+$ is the assets of coalition C by given state s and coalition joint action a_C.

Shapley Value: Shapley Value [32] is almost the famous method to distrubute payoff in cooperative game. In math, it can be represented as below:

$$Sh_i(\Gamma) = \sum_{C \subset N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N|!} . \mu(C) \quad (6)$$

Where C represented the subcoalition of the grand coalition N, and $\mu(C)$ is the marginal contribution of coalition C. Shapley value has properties as follow: 1) Efficiency: The total value of the coalition is fully distributed among the agents. 2) Symmetry: Agents with identical contributions receive equal payoffs. 3) Dummy Player: Agents that contribute nothing receive no payoff. 4) Additivity: The Shapley Value is additive across independent games or tasks. This makes it a robust tool for ensuring equitable distribution in collaborative settings, such as reward allocation in MARL.

Nucleolus: The nucleolus [33] is a concept of fairness used to solve payoff distribution in cooperative games. It focuses on allocating payoffs among coalitions to minimize the maximum dissatisfaction (excess), thereby achieving a stable and fair distribution [33]. It provides stability and predictability, as it is a unique and definitive solution in every game [34]. The nucleolus is typically located within or near the core of the game, ensuring the stability of the distribution scheme, and it balances the interests of different coalitions to reduce the likelihood of unfair allocation. This approach is widely applied in fields such as economics, politics, and supply chain management to benefit distribution in cooperation settings.

Let (N, u, v) be a cooperative game [35], where $N = \{1, \ldots, n\}$ is the set of agents, $u(C)$ is the unity function to measure the profits earned by a coalition $C \subseteq N$, v represents the total assets that can be distributed. Let $x = (x_1, \ldots, x_n)$ be a payoff distribution over N. For a coalition C, the excess $e(C, x)$ of the coalition at x is defined as

$$e(C, x) = u(C) - \sum_{i \in C} x_i \tag{7}$$

where x_i represented the payoff of agent$_i$ in coalition C. Nucleolus is a distribution method to minimize the maximum degree of dissatisfaction of coalition in cooperative games. Formally, the optimized process of the nucleolus is as follows:

$$\min_{x} \max_{C \subseteq N} e(C, x) \tag{8}$$

Since there are n agents, there are a total of 2^n coalitions (including empty sets), so for a set of imputation x, there are always 2^n excesses. And then sort excesses by non-increment called excesses sequence $\theta(x)$, and it is defined as follows:

$$\theta(x) = [e\left(C_1, x\right), e\left(C_2, x\right), \ldots, e\left(C_{2^n}, x\right)] \tag{9}$$

The final nucleolus x^* is a unique payoff allocation that satisfies:

$$x^* = \{\theta(x) \preceq \theta(y) \mid \forall y \in I(v)\} \tag{10}$$

where $\preceq$ is lexicographical order and $I(v)$ is the set of all possible payoff distributions x, we call $\theta(x)$ as the Nucleolus. Note that the nucleolus always exists and lies in the core (if the core is non-empty), and it is unique [33].

2.3 Large Language Models

2.3.1 Reinforcement Learning with Human Feedback

Reinforcement Learning with human feedback (RLHF) [36] is a flavor of RL in which human feedback is used to improve the performance of AI models and align models with human expectations. In classic RL, the environment usually directly generates reward signals. Nevertheless, creating well-defined reward signals for elaborate tasks, like assessing the consistency or factual correctness of an output, or the morality of a decision, can be difficult or cumbersome. RLHF approaches this issue by training Reward Model construct reward signals based on human evaluation, embedding implicit human preferences and values in RL framework.

RLHF has three stages which form its core process. Specifically, an initial model is trained on labeled data using a supervised learning method to learn a base ability for the task. Candidate outputs are then evaluated by human annotators, who rank them in preference order and use this information to train the reward model. Finally, the feedback is used to train a reward model to predict the quality of outputs to generate reward signals, replacing the environment-derived rewards used in traditional RL.

RLHF is commonly used in various practical domains, including natural language processing, robotic control, and recommendation systems. Particularly for training large language models such as ChatGPT, RLHF helps improve the coherence, human-likeness, and relevance of generated text while minimizing the potential for harmful or biased content. Moreover, RLHF enables the fine-tuning of complex interactive behavior in applications like improving user experience or task carrying capability to make robots smarter and ensure AI applications are safer [37].

2.3.2 Fine Tune

Fine-tuning [38] is a technique that allows large language models (LLMs) to be tailored to specific tasks, domains, or user needs by further training the model on specific datasets. LLMs are typically pretrained on massive general-purpose data to learn the building blocks from the ground up; fine-tuning, however, is responsible for taking the pretrained parameters and tweaking them to tailor the LLM to a specific task, often using a more limited, task-domain data for that purpose. This needs second-order information, which is why we know that this way will work better for specialized tasks that either need dynamic behavior or in-depth exploration of domain-specific information, like generating domain-specific text or the specialized classification.

Fine-tuning: The Process Fine-tuning consists of a few key steps. It begins with data preparation that involves collecting, labeling, and prepossessing task-relevant datasets. We load the pretrained model and fine-tune its parameters using the new data. Full fine-tuning [39, 40]: Updating all model parameters for adapting the model to downstream tasks can be done when need to do such adaptation with better accuracy or even with slowly changing downstream tasks while the domain knowledge is not static knowledge. The fine-tuned model is tested against test datasets after training to make sure its expected

performance metrics are achieved. And Fine-tuning also has a branch named Parameter-Efficient Fine-Tuning, a representative method named LoRA [41]: LoRA is a parameter-efficient fine-tuning method designed for adapting large pre-trained models to specific downstream tasks with minimal computational and storage overhead

Fine-tuning is useful across industries. In the domain of natural language processing, it facilitates advancements in various tasks such as sentiment analysis, summarization, and question answering. Domain adaptation, where LLMs are fine-tuned in specialized fields such as medicine, law, or finance, is also required. Moreover, fine-tuning enables the development of AI models for specific use cases, such as customer support, creative writing, and education, resulting in highly personalized and effective solutions. .

3 Research Problems

3.1 Explicit Knowledge Representation for Agents Efficient Learning

How can explicit knowledge [42,43] representations be effectively incorporated to enable agents to learn more efficiently ? This means providing the agent with well-organized knowledge, like domain rules, helpful strategies, or task-specific guidelines [44], to directly steer its exploration and decision-making, making the process more efficient and less reliant on random trial and error. The depth of neural networks increases and the number of neurons grows, the computational power and time required for network training also rise significantly. Introducing prior knowledge to accelerate network training has been validated as a feasible approach. However, the environments faced by deep RL (DRL), which integrates deep learning, possess a much broader search space [45]. Thus, explicitly incorporating prior knowledge into the training process to accelerate the iteration of agents' policies and the training itself becomes a key factor in making DRL applicable to real-world scenarios.

When introducing prior knowledge into RL (RL), several gaps arise:

1. **Cross-Domain Adaptation**: How can prior knowledge from other domains be effectively converted for use in RL, given the gap between different domains?

2. **Theoretical Consistency**: How can we ensure that the converted prior knowledge retains the same theoretical properties it possessed in its original domain when applied to RL?

3. **Computational Overhead**: How can the additional computational cost introduced by prior knowledge be kept within an acceptable range?

Compared to single-agent RL, where prior knowledge primarily focuses on learning theory, MARL involves a wider range of domain knowledge [46–49] and can be more closely linked to fields such as game theory and social psychology. Therefore, I further refine our research question 1 to specifically address how to incorporate prior knowledge

into MARL to enhance training efficiency. My research question is outlined as follows: In MARL, credit assignment lacks interpretability [50], making it difficult to evaluate the contributions of individual agents. This leads to inaccuracies in value decomposition, preventing the discovery of optimal strategies. Consequently, introducing prior knowledge into MARL requires addressing the following issues: 1) What types of prior knowledge can enhance the interpretability of credit assignment? 2) How can the introduced prior knowledge accurately evaluate the contributions of individual agents?

3.2 Implicit Knowledge Representation and Autonomous Reasoning

How can agents discover and exploit the implicit knowledge present in their interaction with the environment in an unsupervised manner? This question is concerned with allowing agents to mine and reason about latent patterns, relationships, or principles that are embedded in trajectories and observations [51–53]. Identifying such tacit knowledge allows agents to augment their reasoning power as well as conduct more sampling-efficient learning.

Multi-domain models that simply encode knowledge from single domains explicitly are less effective as this approach ignores the synergies that come from cross-domain knowledge [54, 55] and is prone to catastrophic forgetting. In particular, there are some gaps in incorporating implicit knowledge into RL training:

1. **Adaptation to RL Environments**: How can we ensure that the introduced knowledge aligns with the current RL environment? Knowledge from different domains may conflict with the objectives of the current task.

2. **Dynamic Adjustments**: RL environments are typically dynamic, whereas implicit knowledge may originate from static settings or different assumptions. How can implicit knowledge be dynamically adjusted to remain relevant in new environments? If the knowledge becomes outdated or irrelevant, it may mislead the agent's learning.

3. **Interpretability**: Implicit knowledge is embedded within neural networks, making its reasoning and decision-making processes difficult to explain. This lack of transparency can lead to trust issues, especially in safety-critical applications such as autonomous driving.

LLMs utilizing RLHF exhibit powerful reasoning abilities [56, 57], helping to address the above gaps. By incorporating prior knowledge through pretrained datasets, LLMs acquire knowledge across multiple domains. However, the quality of these datasets can differ significantly, and much of the information is expressed in natural language instead of being organized into clear, structured formats. Consequently, extracting potential knowledge from these datasets is an inevitable gaps for scaling RLHF algorithms. These challenges can be summarized into two primary issues in RLHF: **Alignment** [58] and **Reasoning** [59].

The integration of implicit knowledge to enhance RLHF training efficiency presents the following problems: 1) Alignment problem is how to convert natural language to machine-readable language. More specifically, how can natural language be effectively translated into machine-understandable language, enabling AI to comprehend human preferences, values, and other information? 2) Reasoning Capability is mean how can models infer the relationships or logical structures hidden within data to improve reasoning performance?

3.3 Few-shot Learning with Knowledge-rich Samples

Achieving few-shot learning [60, 61]: how can agents efficiently extract and leverage intrinsic knowledge from a few knowledge-rich samples? This requires developing techniques that enable agents to learn generalized behavior from scarce data points, inferring the principles or strategies represented through these samples to act effectively in novel or unseen environments.

The Few-Shot problem in RL involves teaching agents how to efficiently learn and perform good policies given limited interaction experiences. The main challenge lies in how to identify the knowledge contained in a small amount of data and how to generalize this knowledge to other tasks through reasoning ability. The specific gaps are as follows:

1. **Selectione and Representation of Knowledge**: How to use proper forms of knowledge (or implicit) for the current RL task and how to encode them efficiently in the agent's representation space? Learnable knowledge must correlate tightly with environmental dynamics (e.g., limited by physical law constraints) but implicit knowledge is encoded in the weights of a neural network and lacks clear interpretability.

2. **Loading Training Data And Transfer Learning**: What are the methods to transfer knowledge from source tasks to target tasks so that it can be used? The lack of knowledge matching will introduce biases in strategy, or will hinder exploration to the best efficiency.

3. **Enhancing Reasoning Ability**: How to developing a better and faster understanding of the existing knowledge that provides grounding to boost the reasoning ability of the algorithm Since Few-Shot performance relies heavily on models to capture implicit knowledge from samples, correctly inferring the implicit knowledge in sample is crucial to boost the sample efficiency and strengthen Few-Shot performance.

4. **Enhancing the Efficiency of Exploration**: A more profound understanding of how introduced knowledge can speed up the exploration processes while safeguarding the exploration ability of the agent in the whole state space.

5. **Alignment**: How to structure knowledge transfer mechanisms to generalize to other tasks?

Integrating knowledge in Few-Shot RLHF solving not only improves agents' efficiency of learning, but can also offer inspiration on how to make effective decisions in practice. Future studies need to explore the combination between explicit and implicit knowledge, multi-task transfer mechanisms optimization, and knowledge adaptation algorithms refinement to improve this area of research.

4 Methods

In this chapter, we address the research problems introduced in this section by incorporating game theory, Monte Carlo tree search, and fine-tuned knowledge.

4.1 Game Theory Enhance MARL Efficiency

4.1.1 Nucleolus in Cooperative Game Theory

The Nucleolus [33] is a classic concept in cooperative game theory, serving as a payoff distribution rule to allocate the benefits generated by cooperation. It achieves stability by minimizing the excess values, which represent the difference between the total payoff a coalition receives and the proposed distribution, reflecting the coalition's dissatisfaction. The primary goal of the Nucleolus is to minimize the dissatisfaction of the most discontented coalition, thereby optimizing fairness in the allocation. Theoretically elegant, it is particularly suited for cooperative scenarios requiring a balance of diverse interests and is effective in evaluating the contributions of agents within a coalition. The calculation of the Nucleolus typically involves linear programming, progressively reducing dissatisfaction from the highest to the lowest excess value. While the Nucleolus solution is unique and stable in most cooperative games, its computational complexity can be high, especially in scenarios with a large number of players. Widely applied in resource allocation, market distribution, and dispute resolution, the Nucleolus emphasizes fairness and stability, effectively preventing excessive dissatisfaction among coalitions and fostering cohesive cooperation. Therefore, the Nucleolus can be utilized as prior knowledge to evaluate the contributions of different agents in MARL.

4.1.2 Nucleolus in Multi-agents RL

I extend the traditional concept of the Nucleolus from cooperative game theory to dynamic environments and propose the Markov Nucleolus, which addresses the credit assignment problem in dynamic tasks within MARL. The Markov Nucleolus leverages the characteristics of MDP to dynamically optimize reward distribution. Building upon the Nucleolus theory, the Markov Nucleolus incorporates state transitions and joint action rewards, de-

scribing coalition dissatisfaction and its allocation. Its formal definition is as follows:

$$\begin{aligned} e(C, x(s,a)) &= u(s,a|C) - x(s,a|C) \\ &= \max_{a_C} v(s,a_C) - \sum_{i \in C} x_i(s,a_i) \end{aligned} \tag{11}$$

Based on the above definition, the Markov Nucleolus identifies the allocation with the smallest lexicographical order.

Furthermore, we use the Markov Nucleolus to assign the global Q-value to individual Q-values, defined as follows:

corollary 1. *To assign the global Q-value $Q_{CS,global}(s,a)$ by given state s and joint action a under coalition structure CS, we modify the payoff distribution as:*

$$Q_{CS}(s,a) = \left\{ Q_{CS,1}(s,a_1), \ldots, Q_{CS,n}(s,a_n) \right\} \tag{12}$$

where $Q_{CS,i}(s,a_i)$ is the individual Q-value of agent i and $\sum_{i \in N} Q_{CS,i}(s,a_i) = Q_{CS,global}(s,a)$. Then, we model the Eq 11 to define the excess in coalition C in Q-learning.

$$e(C, Q_{CS}(s,a)) = \max_{a_C} V(s,a_C) - \sum_{i \in C} Q_{CS,i}(s,a_i) \tag{13}$$

where $V(s,a_C)$ is the Q-value of coalition C with given state s and joint action $a_C \in A = \times_{i \in C} A_i$ and $V(s,a_C) \in \mathbb{R}_0^+$. We use a similar definition of the excess sequence in Markov nucleolus as:

$$\theta(Q_{CS}(s,a)) = [e(C_1, Q_{CS}(s,a)), e(C_{2^n}, Q_{CS}(s,a))] \tag{14}$$

Then the Nucleolus Q-value $Q^*_{CS}(s,a)$ can be formally defined as:

$$Q^*_{CS}(s,a) = \{\theta(Q^*_{CS}(s,a)) \preceq \theta(Q_{CS}(s,a)) | \forall Q_{CS}(s,a) \in I(Q_{CS}(s,a))\} \tag{15}$$

where $I(Q_{CS.global}(s,a))$ is all possible payoff distribution coalition structure CS.

We theoretically guarantee that the Nucleolus Q-value satisfies the following properties:

Theorem 1. *The Nucleolus Q-value in EC-POMDP can guarantee*

1. *Each agent's individual Q-value in the optimal coalition structure is greater than it is in other coalition structures;*
2. *The actions and the coalition structure exhibit consistency, meaning that the coalition formed under the optimal actions is also optimal.*

To solve the problem of computational overhead, I used an approximate solution to approach the Nucleolus Q value. Due to space limitation, I put this part in the **Appendix A.2**.

4.2 LLM Enhance Reinforcement Learning Agents Reasoning Capacity

While there are many ways to introduce unstructured knowledge into the learning capabilities of agents - including natural language texts or other unstructured datasets — we need to ensure that we can first improve alignment capacities of agents with respect to human intentions. This enables agents to be more in sync with the types of human thinking, leading to the improvement of their new understanding of knowledge. Then, it emphasizes improving the reasoning capacity of agents, making them able to generalize and effectively utilize implicit knowledge to accelerate their learning.

4.2.1 Alignment

To enhance the alignment capabilities of agents, we plan to conduct research in the following four directions.

1. **Data Preparation and Augmentation**: Building high-quality datasets will be the first step towards improving alignment capabilities, whether that be multi-domain and multilingual natural language texts, task descriptions, expert-labeled dataset, etc. Implicit knowledge and clear aptitude for the agent are jointly used by these datasets. And generative models (like GPT) can be used to increase the size of the training samples, whereas data augmentation techniques (like word substitution and semantic expansion) improve the agent's ability to align with more different human intents.

2. **Model Optimization and RL**: Model optimization is achieved through supervised fine-tuning, multi-task learning, and transfer learning to strengthen alignment capabilities. Reward modeling introduces human preference-based reward functions to guide optimization. RL from human feedback (RLHF) is a key method for enhancing alignment. By training a reward model and optimizing policies, the agent dynamically adjusts its behavior to produce outputs that are more consistent with human intent.

3. **Semantic Understanding and Prompt Optimization**: Agent's Implicit Knowledge Decomposition. While it is said that implicit orders of semantic embedding methods (notably BERT, GPT embedding layer) and longer context modeling reserves help to capture semantic dependence in implicit knowledge. Moreover, nuanced prompts (Prompt Engineering) help direct the agent towards understanding task goals. By widening context windows or integrating external knowledge bases, we also increase the agent's understanding of sophisticated language patterns of language and hidden logic.

4. **Continuous Learning and Multimodal Integration**: By integrating text, images, and audio, multimodal data enriches the agent's understanding of human intent.

Continuous learning systems further optimize alignment capabilities by dynamically updating the model based on real-time user feedback. This ensures that the agent remains consistently aligned with human intentions, improving its performance in complex tasks.

4.2.2 Reasoning

To enhance the reasoning capabilities of agents, we plan to conduct research in the following four directions.

1. **Knowledge-Driven Reasoning Enhancement**: Through Logical and symbolic reasoning Agents can infer logical information using logical inference in explicit rule-based frameworks, or they can deal with complex data in various environments using neural networks, which facilitates interpretable symbolic reasoning. Moreover, embedding external knowledge bases (like ConceptNet) and using knowledge graphs enable agents to create semantic associations and causal relationships and reasoning with implicit knowledge. Probabilistic models (e.g., Bayesian networks) and causal reasoning can help agents model uncertainty and predict the effects of interventions.

2. **Strengthening Chain-of-Thought and Recursive Reasoning**: Large language models can progressively decompose complex problems and induce conclusions step-by-step via chain-of-thought (CoT) reasoning. Recursive reasoning aids agents in breaking down tasks into many subproblems, enabling the efficient completion of complex tasks. Explicit reasoning paths offer transparency in the underlying reasoning process, enabling users to verify and refine the outputs of models. When combined with inductive and analogical reasoning, this would allow the agents to generalize patterns observed across instances and transfer knowledge over to similar tasks.

3. **Integration of Reasoning with Large Models**: By leveraging pre-trained large models (e.g., GPT, BERT) and fine-tuning techniques, agents can demonstrate enhanced reasoning capabilities in language, vision, and multimodal tasks. Multimodal integration enables agents to combine text, image, and audio data for comprehensive reasoning. Additionally, the inclusion of visualization tools and interpretability modules provides transparency and credibility to the reasoning process. These advancements collectively enhance agents' understanding and generalization of implicit knowledge, enabling superior performance in complex environments.

4.3 Reforced Fine Tuning to Enhance Reinforcement Learning Agents based on Few-shot Learning

The few-shot essentially requiring the agent to have the ability to easily reason about a specific class of data, using only a small number of samples. We will work on two

computation methods to improve the few-shot abilities of the agent, one through Monte Carlo Tree Search (MCTS) [62] and the other through the Reinforced fine Tuning (RFT) [63].

4.3.1 Monte Carlo Tree Search

There are two steps use MCTS to improve the reasoning capability in LLM.

1. **Framework Design and Core Techniques**: By combining MCTS with large language models (LLMs), the reasoning capabilities of agents can be significantly enhanced. MCTS constructs a search tree where each node represents a decision or reasoning point, and LLMs provide node evaluation values and candidate paths. In this framework, LLMs generate initial reasoning paths and action suggestions, while MCTS explores and selects the optimal path, dynamically expands the reasoning depth, and updates node values through backpropagation to optimize the overall reasoning process.

2. **Optimization for Few-Shot Scenarios** In Few-shot learning scenarios, the integration of MCTS and LLMs mitigates the constraints of small sample sizes. The scoop is that limited samples of LLMs are used to generate list of candidate reasoning paths, while MCTS explores the various hypotheses and validation of the plausibility of the paths, while promoting the generalization of the model through either simulation or feedback. Also, MCTS employs dynamic calls to LLMs for reasoning step refinement, one key point is the adaptively adjust the search depth, this characteristic allows agents learning new tasks rapidly even in a few sample scenario.

The combination of MCTS with LLMs not only improves reasoning capability but also increases the adaptability and transparency of tasks. MCTS offers clear multi-path exploration and enhanced feedback when needed, while LLMs provide high-quality generation capabilities, resulting in increased interpretability of the reasoning process. Being great for complex logical and arithmetic reasoning, mathematical problem-solving, complex question answering, and code generation and optimization, We evaluate LangChain's support for agents which can help perform complex multi-step tasks.

4.3.2 Reinforced Fine-Tuning

I will study how to enhance the reasoning capabilities of agents through RFT by focusing on four key aspects.

1. **Framework Design**: RFT marries RL and fine-tuning to improve large-language models (LLMs) reasoning. RFT leverages a reward model (RM) trained on human feedback to grade reasoning paths and outputs produced by the LLM. Rewards depend on correctness, efficiency, and diversity. RL algorithms such as Proximal Policy Optimization (PPO) are used to enhance the LLMs' policies so as to optimize increasingly logical and efficient chains of reasoning.

2. **Core Technical Components**: At its core, the method starts from Few-shot Prompts prepared from a few annotated example(s) and pre-trained LLMs as a backbone. Next, reward functions are designed to match task-specific goals in terms of logic consistency, task completion, and process transparency. This process of policy update tailored for the tasks enables RL to guarantee a steady optimization of the model's reasoning capabilities via updating the policy of a neural network according to the task-specific reward.

3. **Enhancing Reasoning Processes**: RFT strengthens the generation of reasoning paths by promoting step-by-step reasoning (e.g., Chain-of-Thought) and optimizing paths for efficiency and logic through reward signals. The model is also encouraged to provide clear and interpretable reasoning processes alongside the final output, improving transparency and enabling users to validate the results.

4. **Optimization for Few-shot Scenarios**: In few-shot learning, RFT leverages the scarce annotated data through reward-based mechanism to direct reasoning paths generation towards a reward maximizing paths whose annotated paths are limited. LLMs use very few shot prompts and RL to create patterns learned from few examples that apply to unseen problem domains, enabling them to perform well despite the scarcity of data.

RFT notably improves LLM's reasoning quality including the degree of logical consistency, enhances sample efficiency and interpretability. This is especially useful for complex tasks such as question-answering, mathematical reasoning, or professional domain-specific tasks like medical diagnosis or legal analysis, where multi-step reasoning and strong generalization are required.

5 Research Schedule and Expected Outcomes

My research plan and expected outputs are as follows:

1. July 2024 - January 2025: Conduct research on research question 1 and produce one paper.

2. February 2025 - June 2025: Conduct research on research question 2 and expect to produce 1 papers based on the research results.

3. July 2025 - June 2026: Conduct research on research question 3 and expect to produce 2 papers based on the research results.

4. July 2026 - July 2027: Complete the writing of the doctoral dissertation.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[2] A. M. Turing, *Computing Machinery and Intelligence*. Dordrecht: Springer Netherlands, 2009, pp. 23–65. [Online]. Available: https://doi.org/10.1007/978-1-4020-6710-5_3

[3] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green, and S. N. Cohen, "Computer-based consultations in clinical therapeutics: explanation and rule acquisition capabilities of the mycin system," *Computers and biomedical research*, vol. 8, no. 4, pp. 303–320, 1975.

[4] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI magazine*, vol. 27, no. 4, pp. 12–12, 2006.

[5] J. R. Anderson, M. Matessa, and C. Lebiere, "Act-r: A theory of higher level cognition and its relation to visual attention," *Human–Computer Interaction*, vol. 12, no. 4, pp. 439–462, 1997.

[6] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

[7] A. Newell, J. C. Shaw, and H. A. Simon, "Report on a general problem solving program," in *IFIP congress*, vol. 256. Pittsburgh, PA, 1959, p. 64.

[8] M. L. Minsky, *Semantic information processing*. The MIT Press, 1969.

[9] F. M. Brown, *The frame problem in artificial intelligence: Proceedings of the 1987 workshop*. Morgan Kaufmann, 2014.

[10] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," *arXiv preprint arXiv:1706.10059*, 2017.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[16] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*. Pmlr, 2014, pp. 387–395.

[17] J. Moody, M. Saffell, Y. Liao, and L. Wu, "Reinforcement learning for trading systems and portfolios: Immediate vs future rewards," in *Decision Technologies for Computational Finance: Proceedings of the fifth International Conference Computational Finance*. Springer, 1998, pp. 129–140.

[18] S. Iqbal, C. A. S. De Witt, B. Peng, W. Böhmer, S. Whiteson, and F. Sha, "Randomized entity-wise factorization for multi-agent reinforcement learning," in *International Conference on Machine Learning*, 2021, pp. 4596–4606.

[19] B. Liu, Q. Liu, P. Stone, A. Garg, Y. Zhu, and A. Anandkumar, "Coach-player multi-agent reinforcement learning for dynamic team composition," in *International Conference on Machine Learning*, 2021, pp. 6860–6870.

[20] F. A. Oliehoek, "Decentralized pomdps," *Reinforcement learning: state-of-the-art*, pp. 471–503, 2012.

[21] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, no. 1, pp. 1–31, 2012.

[22] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[23] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[24] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE transactions on intelligent transportation systems*, vol. 21, no. 3, pp. 1086–1095, 2019.

[25] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

[26] F. A. Oliehoek, M. T. Spaan, and N. Vlassis, "Optimal and approximate q-value functions for decentralized pomdps," *Journal of Artificial Intelligence Research*, vol. 32, pp. 289–353, 2008.

[27] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, 2018, p. 2085–2087.

[28] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[29] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 10 199–10 210.

[30] L. S. Shapley and M. Shubik, "The assignment game i: The core," *International journal of game theory*, vol. 1, no. 1, pp. 111–130, 1971.

[31] J. Wang, Y. Zhang, Y. Gu, and T.-K. Kim, "Shaq: Incorporating shapley value theory into multi-agent q-learning," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 5941–5954.

[32] F. Ruggeri, W. Emanuelsson, A. Terra, R. Inam, and K. H. Johansson, "Rollout-based shapley values for explainable cooperative multi-agent reinforcement learning," in *2024 IEEE International Conference on Machine Learning for Communication and Networking*, 2024, pp. 227–233.

[33] D. Schmeidler, "The nucleolus of a characteristic function game," *SIAM journal on applied mathematics*, vol. 17, no. 6, pp. 1163–1170, 1969.

[34] A. Rapoport, *Game theory as a theory of conflict resolution*, 2012, vol. 2.

[35] R. Branzei, D. Dimitrov, and S. Tijs, *Models in cooperative game theory*, 2008, vol. 556.

[36] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[37] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. Henighan *et al.*, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[38] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," *arXiv preprint arXiv:2403.14608*, 2024.

[39] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[40] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[41] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[42] R. Maclin and J. W. Shavlik, "Creating advice-taking reinforcement learners," *Machine Learning*, vol. 22, no. 1, pp. 251–281, 1996.

[43] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[44] L. Von Rueden, S. Mayer, K. Beckh, B. Georgiev, S. Giesselbach, R. Heese, B. Kirsch, J. Pfrommer, A. Pick, R. Ramamurthy *et al.*, "Informed machine learning–a taxonomy and survey of integrating prior knowledge into learning systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 614–633, 2021.

[45] S. S. Mousavi, M. Schukat, and E. Howley, "Deep reinforcement learning: an overview," in *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 2*. Springer, 2018, pp. 426–440.

[46] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of reinforcement learning and control*, pp. 321–384, 2021.

[47] P. Zheng, L. Xia, C. Li, X. Li, and B. Liu, "Towards self-x cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach," *Journal of Manufacturing Systems*, vol. 61, pp. 16–26, 2021.

[48] C. Schroeder de Witt, J. Foerster, G. Farquhar, P. Torr, W. Boehmer, and S. Whiteson, "Multi-agent common knowledge reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[49] P. Mannion, S. Devlin, J. Duggan, and E. Howley, "Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning," *The Knowledge Engineering Review*, vol. 33, p. e23, 2018.

[50] A. Alharin, T.-N. Doan, and M. Sartipi, "Reinforcement learning interpretation methods: A survey," *IEEE Access*, vol. 8, pp. 171 058–171 077, 2020.

[51] V. Tshitoyan, J. Dagdelen, L. Weston, A. Dunn, Z. Rong, O. Kononova, K. A. Persson, G. Ceder, and A. Jain, "Unsupervised word embeddings capture latent knowledge from materials science literature," *Nature*, vol. 571, no. 7763, pp. 95–98, 2019.

[52] G. Bonaccorso, *Machine Learning Algorithms: Popular algorithms for data science and machine learning*. Packt Publishing Ltd, 2018.

[53] X. Shu and Y. Ye, "Knowledge discovery: Methods from data mining and machine learning," *Social Science Research*, vol. 110, p. 102817, 2023.

[54] D. Li, X. Yu, C. Xu, L. Petersson, and H. Li, "Transferring cross-domain knowledge for video sign language recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6205–6214.

[55] H. B. Ammar, E. Eaton, J. M. Luna, and P. Ruvolo, "Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[56] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[57] H. Lee, S. Phatale, H. Mansoor, K. R. Lu, T. Mesnard, J. Ferret, C. Bishop, E. Hall, V. Carbune, and A. Rastogi, "Rlaif: Scaling reinforcement learning from human feedback with ai feedback," 2023.

[58] J. Ji, T. Qiu, B. Chen, B. Zhang, H. Lou, K. Wang, Y. Duan, Z. He, J. Zhou, Z. Zhang *et al.*, "Ai alignment: A comprehensive survey," *arXiv preprint arXiv:2310.19852*, 2023.

[59] P. Lu, L. Qiu, W. Yu, S. Welleck, and K.-W. Chang, "A survey of deep learning for mathematical reasoning," *arXiv preprint arXiv:2212.10535*, 2022.

[60] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[61] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[62] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[63] L. Trung, X. Zhang, Z. Jie, P. Sun, X. Jin, and H. Li, "Reft: Reasoning with reinforced fine-tuning," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 7601–7614.