Efficient Multi-Task Reinforcement Learning via Task-Specific Action Correction

Jinyuan Feng, Min Chen, Zhiqiang Pu, *Member, IEEE,* Tenghai Qiu, Jianqiang Yi, *Senior, IEEE,* Jie Zhang

***Abstract*—Multi-task reinforcement learning (MTRL) holds potential for building general-purpose agents, enabling them to generalize across a variety of tasks. However, MTRL may still be susceptible to conflicts between tasks. A primary reason for this problem is that a universal policy struggles to balance short-term and dense learning signals across various tasks, e.g. , distinct reward functions in reinforcement learning. In social cognitive theory, internalized future goals, as a form of cognitive representations, can effectively mitigate potential short-term conflicts in multitask settings. Considering the benefits of future goals, we propose a novel and general framework called Task-Specific Action Correction (TSAC) from the goal perspective as an orthogonal research to previous MTRL methods. Specifically, to avoid myopia, TSAC introduces goal-oriented sparse rewards and decomposes policy learning into two separate policies: a shared policy (SP) and an action correction policy (ACP). The SP outputs a short-term perspective action based on guiding dense rewards. To alleviate conflicts resulting from excessive focus on specific tasks' details in SP, the ACP incorporates goal-oriented sparse rewards, enabling an agent to adopt a long-term perspective to output a correction action and achieve generalization across tasks. Finally, the actions output by SP and ACP are combined based on the action correction function to form a final action that interact with the environment. Extensive experiments conducted on Meta-World and multi-task StarCraft II multi-agent scenarios demonstrate that TSAC outperforms existing state-of-the-art methods, achieving significant improvements in sample efficiency, generalization and effective action execution across tasks.**

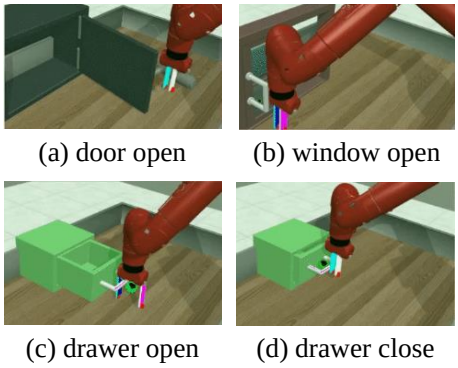
***Index Terms*—Multi-task reinforcement learning, generalization, Lagrangian method, Future goals**

I. Introduction

REINFORCEMENT Learning (RL) has witnessed remarkable advancements in a wide array of decision-making problems with the assistance of neural networks [1]–[3] and has become a crucial methodology in various domains, such as gaming [4]–[6], large language models [7] and real-world applications including robotics [8]. However, the majority of research in RL predominantly focuses on specific problem scenarios, prioritizing mastery of individual tasks through the learning of single policies, often at the expense of generalization. To empower general-purpose agents, Multi-Task Reinforcement Learning (MTRL) is introduced to scale up the RL framework, holding the promise of learning a universal policy capable of accomplishing a diverse range of tasks.

This work was supported in part by the National Natural Science Foundation of China under Grant 62322316, and the Beijing Nova Program under Grant 20220484077 and 20230484435. *(Corresponding author: Min Chen.)*

The authors are with the Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with Department of Computer Science, University of Bath.



(a) door open (b) window open

(c) drawer open (d) drawer close

Fig. 1: a variety of related manipulation tasks. Several actions are similar across these tasks: getting closer to the objects and interacting with them.

Compared to RL, a fundamental challenge of MTRL is that different tasks may conflict with each other when learning simultaneously. Although MTRL naturally incorporates a curriculum-like teaching approach, wherein easier tasks are learned to facilitate the learning of more challenging tasks [9]–[12], MTRL is prone to negative transfer [13]. Negative transfer refers to the phenomenon where the task-specific knowledge acquired from one task may impede the overall learning process of other tasks. This phenomenon is also referred to as task conflict, which becomes more acute as the number of tasks increases. From the optimization standpoint, task conflict arises from conflicting gradients [14] between tasks, where the gradients move in opposite directions. Negative transfer between tasks can result in an excessive focus on the immediate rewards of individual tasks during the multi-task learning process, resulting in myopia and hindering the ability to learn a universal policy that can effectively accomplish multiple tasks.

The key to resolving short-term conflicts between tasks lies in enabling agents to learn to consider long-term goals. In recent researches, incorporating long-term goals or future representations has emerged as one of the primary solutions for addressing myopic decision-making. QFuture [15] adjusts agents' actions to avoid myopia by predicting potential future outcomes. LS-OII [16] infers the opponent's long-term tactical intentions from a macro perspective to learn superior adversarial policies. SAC-X [17] is capable of learning complex behaviors from scratch in the presence of multiple goal-oriented sparse reward signals. The main efforts on this direction have been paid on single-task settings, making MTRL lags thus far.

In current MTRL, guiding dense rewards are designed for

each task, providing feedback to the agent at each state. However, from a multi-task perspective, avoiding conflicts between tasks and designing a globally optimal reward function for each task is challenging. Additionally, the use of guiding dense rewards enables the agent to focus solely on task-specific information and details within each task, which may result in myopia and hinder generalization across tasks. In contrast to guiding dense rewards, goal-oriented sparse rewards use binary rewards to indicate goal achievement [18], [19]. Using goal-oriented sparse rewards to guide training involves relinquishing the design of potentially unreliable rewards, removing assumptions regarding the intermediate processes of tasks as much as possible, and fully restoring the autonomy of learning to the reinforcement learning algorithm. If there are sufficient high-quality samples available, this can enable the agent to coordinate multiple tasks from a global perspective [20]. The advantages and disadvantages of these two reward functions are complementary. Guiding dense rewards are efficient in early exploration but may lead to potential conflicts between tasks. Goal-oriented sparse rewards are less efficient in early exploration, but after sufficient exploration, the agent can alleviate conflicts between tasks from a global perspective. Building upon the existing dense rewards in MTRL, our work introduces goal-oriented sparse learning signals to achieve efficient multi-task learning and cross-task generalization.

Drawing inspiration from goal orientation theory in cognitive science [21], Humans adjust their behavior in handling different tasks based on set future goals to avoid short-term conflicts between tasks. Imagine that humans are simultaneously learning a variety of related manipulation tasks, for instance, as depicted in Fig. 1. Within these tasks, several actions exhibit similarity, such as approaching objects like doors or drawers and interacting with them. Humans have the ability to leverage previously learned behaviors when encountering a specific task, making slight adjustments based on the task's goal. Furthermore, when humans are confronted with numerous tasks simultaneously, having detailed instructions and requirements for each task can lead to confusion and a sense of being overwhelmed, even though they can be helpful in completing individual tasks. In contrast, when each task has a clearly defined goal, humans naturally adopt a long-term perspective regarding conflicts and priorities among tasks. The above human behaviors imply their recognition that current conflicts may not be necessary for achieving the overall objectives of the tasks. Our idea is inspired by this recognition.

We introduce goal-oriented sparse rewards to investigate MTRL in order to address short-term conflicts between tasks and to learn a universal policy from a goal-oriented perspective. In this paper, we propose a novel framework called **T**ask-**S**pecific **A**ction **C**orrection (TSAC) as a general and complementary framework for MTRL. TSAC decomposes policy learning into two policies: a shared policy (SP) and an action correction policy (ACP). SP maximizes well-shaped dense rewards, which focus on task-specific information and accelerate the learning process. Its output actions, referred to as shared actions, are potentially short-sighted. On the other hand, ACP utilizes goal-oriented sparse rewards to generate far-sighted edited actions. The goal-oriented sparse rewards which are sparse and strongly correlated with the completion of the objective. SP and ACP collaborate with each other, where SP provides a suboptimal policy that facilitates the training of ACP in the sparse rewards setting. ACP, in return, improves the overall performance. To balance the training of these two policies and avoid potential conflicts, we assign a virtual expected budget to the sparse rewards and use the Lagrangian method to adjust the weights of the loss in ACP dynamically. Our two-policy paradigm draws inspiration from works in safe reinforcement learning [22]–[24]. However, our framework differs significantly in terms of motivation and interpretation.

To implement our framework, we employ the Soft Actor-Critic algorithm [25] as the underlying reinforcement learning policy. It is worth noting that our approach is algorithm-agnostic and can be integrated with existing MTRL methods, it can even be extended to multi-agent settings. Our experimental results demonstrate the efficiency and significance of the cooperation between the two policies, and simply combining the two rewards do not yield comparable results. Moreover, our experiments conducted on the Meta-world [26] and multi-task StarCraft II multi-agent scenarios showcase significant improvements in both sample efficiency and final performance compared to previous state-of-the-art multi-task policies. In summary, our contributions are as follows:

(1) We propose Task-**S**pecific **A**ction **C**orrection (TSAC), a general and complementary framework for MTRL, which decomposes policy learning into two policies, facilitating efficient MTRL. TSAC can be combined with any existing MTRL methods and extend to multi-agent settings.

(2) We introduce goal-oriented sparse rewards to provide agents with a long-term perspective for handling task conflicts that arise from excessive focus on specific tasks' details. Furthermore, the action correction function we designed integrates the actions of the two policies to facilitate efficient interaction with the environment.

(3) We assign a virtual expected budget to the sparse rewards and utilize the Lagrangian method to simplify the complex optimization process and avoid conflicts between SP and ACP. The Lagrangian multiplier dynamically adjusts the loss weights.

(4) We conduct comprehensive experiments on Meta-world and multi-task StarCraft II multi-agent scenarios. The superior performance of TSAC demonstrate the efficiency and significance of the cooperation between the two policies and show that TSAC achieves significant improvements in both sample efficiency and effective action execution.

II. Related work

A. Multi-Task Reinforcement Learning

A multitude of methods have been developed to alleviate negative transfer and achieve efficient MTRL. Classical approaches include policy distillation [27]–[29], explicit measurement of task relatedness [30], [31], and information sharing [32]–[34], among others. Policy distillation involves training a smaller network structure to bring previous tasks to an expert level, thereby integrating multiple policies into a

single policy. However, these methods increase the number of network parameters as they require separate networks for different tasks and an additional distillation step. Some researchers utilize validation loss on tasks [30] or causal influence [31] to determine better task groupings. One drawback of the aforementioned methods is the need for substantial computational resources. Calculating task correlations and adjusting learning methods can often only be done through trial and error, leading to high costs.

In MTRL, information sharing can be achieved through data sharing, parameters sharing, representations sharing, or behavior sharing. For instance, CDS [33] routes data based on task-specific data to improve information sharing. Similarly, a simple method proposed in [32] applies a zero reward to unlabeled data, facilitating data sharing in theory and practice. Parameter sharing through learning shared representations can effectively transfer knowledge between policies. For instance, a soft modularization method presented in [35], [36] shares parameters by generating soft combinations of different modules. Similarly, AdaShare [37] and an automated multi-task learning algorithm in [38] adaptively determine the feature-sharing mode across multiple tasks. However, these methods suffer from high computational complexity as they require dynamic exploration of network connectivity. Attention mechanisms can be leveraged to share representations, as proposed in [39]–[41], which group task knowledge into sub-networks without requiring prior assumptions. CARE [42] leverages metadata to capture the shared structure among tasks. There is another significant MTRL work that focuses on the challenge of multi-objective optimization from a gradient perspective, aiming to reduce the impact of conflicting gradients by manipulating the gradients based on various criteria [14], [43]. However, these methods impose an additional computational burden. The aforementioned methods reduce or coordinate conflicts between tasks from the perspectives of representation, gradient, task grouping, etc., effectively achieving MTRL. While they neglect the potential causes of task conflict.

Due to the exceptional multi-task parallelism and generalization capabilities of MTRL, researchers have applied it to unlock more general-purpose robotics, achieving significant advancements in autonomous cognition and behavioral development [44]–[46]. BIMTRL [47], which mimics the learning processes in the human brain, enhances parallelism and scalability in multi-task reinforcement learning (MTRL), thereby contributing to improved cognitive capabilities and adaptability in robots. Ding et al. [12] proposed a meta-based task offloading scheduling strategy to solve the adaptive problems in dynamic environments. NuEMT [48], based on the mechanism of multi-task learning, transfers information from a set of auxiliary tasks to the current task. This allows the robot to master complex behaviors more quickly. SURRL [49] and TINet [50] leverage unsupervised learning techniques to assist in multi-task learning, enabling robots to perform subsequent policy learning based on learned structures in the absence of labeled data.

III. Preliminaries

A. Multi-Task Reinforcement Learning

We extend the single-task Markov Decision Process (MDP) problem to multi-task MDP for an agent under the framework of Contextual MDP (CMDP) [51]. CMDP is defined by a tuple $\langle \mathcal{C}, \mathcal{S}, \mathcal{A}, \gamma, \mathcal{M} \rangle$. Here, $\mathcal{C}$ can be viewd as a task set $\mathcal{C} = \{\mathcal{T}_i\}_{i=1}^{N}$, where $\mathcal{T}_i$ denotes the task i and N is the number of tasks. $\mathcal{S}$ represents the shared state space , $\mathcal{A}$ denotes the shared action space, and γ is the discount factor. State $s \in \mathcal{S}$ and action $a \in \mathcal{A}$. $\mathcal{M}$ is a fuction that maps a task $\mathcal{T}_i \in \mathcal{C}$ to MDP parameters, such that $\mathcal{M}(\mathcal{T}_i) = \{P_i, R_i, \rho_i\}$. The transition probability P_i, reward function R_i and initial state distribution ρ_i vary by each task. During training, tasks are sampled from a uniform distribution $p(\mathcal{T})$. The agent's policy takes state s as input and outputs action a. The objective of the agent's policy π is to maximize the expected return $\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathbb{E}_{\pi}[\sum_t \gamma^t R_i(s_t, a_t)]]$, where s_t and a_t represent the state and the action at timestep t.

B. Soft Actor-Critic

In this paper, we adopt Soft Actor-Critic (SAC) [25] with disentangled alphas as the fundamental policy. As observed in [26], SAC, being an off-policy actor-critic algorithm with a strong exploration ability based on maximum entropy, exhibits superior performance.

The concept of SAC goes beyond merely maximizing cumulative rewards. It also introduces additional stochasticity to the policy. In practice, the critic learns a soft Q-function Q_θ parameterized by θ, and the actor learns a policy π_ϕ parameterized by ϕ. A regularization term that incorporates entropy is included in the reinforcement learning objective. The correction term added with entropy can be defined as follows:

$$\pi^* = \arg\max_{\pi_\phi} \mathbb{E}_{\pi_\phi}\left[\sum_t r\left(s_t, a_t\right) + \alpha H\left(\pi_\phi\left(\cdot \mid s_t\right)\right)\right] \tag{1}$$

In this context, α represents a learnable regularization coefficient that controls the significance of the entropy term. H denotes the entropy. The training objective of the critic is to minimize the soft Bellman residual:

$$\begin{aligned} J_Q(\theta) =& \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(\mathbf{s}_t, \mathbf{a}_t\right) - \left(r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right.\right.\right. \\ &\left.\left.\left. + \gamma \mathbb{E}_{\mathbf{s}_{t+1}\sim p}\left[V_{\bar{\theta}}\left(\mathbf{s}_{t+1}\right)\right]\right)^2\right] \end{aligned} \tag{2}$$

where

$$\begin{aligned} V_{\bar{\theta}}(\mathbf{s}_{t+1}) =& \mathbb{E}_{\mathbf{a}_{t+1}\sim\pi}[Q_{\bar{\theta}}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) \\ &- \alpha \log \pi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})] \end{aligned} \tag{3}$$

is the soft state value function. $\mathcal{D}$ is a replay buffer that stores past (s_t, a_t). $\bar{\theta}$ represents the target network parameters, which are obtained as an exponentially moving average of θ. The policy π_ϕ is learned by minimizing:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}[\mathbb{E}_{a_t\sim\pi_\phi}[\alpha \log(\pi_\phi(a_t|s_t)) - Q_\theta(s_t, a_t)]]. \tag{4}$$

The learnable regularization coefficient α is updated by the gradient:

$$J(\alpha) = \mathbb{E}_{a_t\sim\pi_t}[-\alpha \log \pi_t(a_t|s_t) - \alpha\bar{H}], \tag{5}$$

where $\bar{H}$ denotes the target entropy.

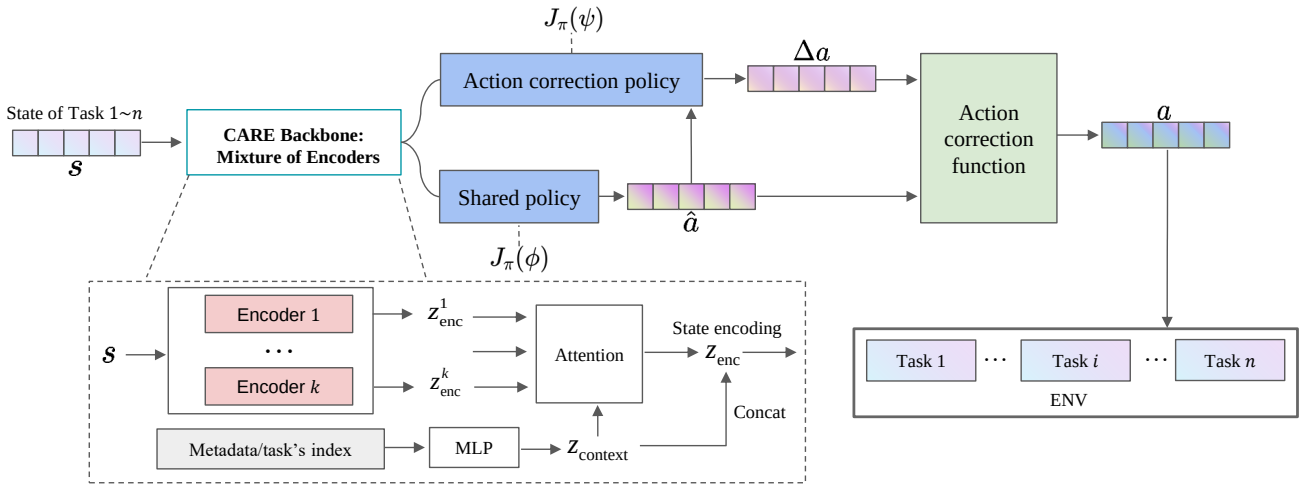


Fig. 2: The structure of TSAC has two policies: a shared policy (SP) and an action correction policy (ACP). Given the states of tasks, the CARE backbone network leverages a mixture of encoders to primarily extract information. Then, the SP and ACP, respectively output actions based on guided dense rewards and goal-oriented sparse rewards. According to the action correction function, the final action is obtained and then interacts with the environment.

IV. Method

In this section, we present a general and complementary framework named **T**ask-**S**pecific **A**ction **C**orrection (TSAC) to decompose policy learning into two separate policies: a shared policy (SP) and an action correction policy (ACP).

A. Overall Structure of TSAC

As illustrated in Fig. 2, TSAC consists of a CARE backbone and a pair of cooperative policies. The CARE backbone [42] is utilized to extract information across all tasks by incorporating a mixture of encoders. Then, the representation is passed to the pair of cooperative policies. The first policy, called the shared policy (SP) and denoted as π_ϕ, optimizes the guiding dense rewards by proposing a preliminary action $\hat{a} \sim \pi_\phi(\cdot|s)$. This preliminary action may be shortsighted. The second policy, referred to as the action correction policy (ACP) and denoted as π_ψ, corrects the preliminary action by providing an action correction $\Delta a \sim \pi_\psi(\cdot|s, \hat{a})$ to execute an effective action. The final action denoted as $a = h(\hat{a}, \Delta a)$, is then output to the environment, where h represents an editing function. ACP is conditioned on SP's output $\hat{a}$. Together, these two policies cooperate to improve sample efficiency and performance. We denote the overall composed policy as $\pi_{\psi\circ\phi}(a|s)$ for simplicity.

Motivation: TSAC decomposes policy learning into two subtasks that focus on guiding dense rewards or goal-oriented sparse rewards. This decomposition is motivated by the following considerations:

1) Different Conflict Horizons: To achieve effective MTRL, most applications incorporate guiding dense rewards into each task. Therefore, in order to balance the dense reward signals between various tasks, SP's actual decision horizon is myopic. SP focuses solely on the details of specific tasks to coordinate conflicts between multiple tasks, while neglecting the overall completion of multiple tasks. In contrast, ACP aims to maximize goal-oriented sparse rewards, adopting a longer decision horizon that focuses on the overall completion of multiple tasks. However, the learning of this strategy necessitates high-quality trajectory data. Due to its ease of optimization, SP can quickly train a suboptimal policy that provides relatively high-quality trajectory data, thereby facilitating the policy learning of ACP under sparse reward settings.

2) Efficient Exploration: From the perspective of SP, its action is altered by ACP. Instead of discouraging SP from taking suboptimal actions, ACP offers suggestions to improve the preliminary action, enabling SP to continue exploration in an effective and far-sighted manner. This guarded exploration leads to a better overall exploration policy because ACP is less likely to hinder SP's exploration. From the perspective of ACP, it determines the action based on SP, enhancing its exploration ability. From an entropy perspective, the decomposition of policy learning into two policy networks introduces additional entropy.

B. Feature extraction and CARE backbone

We utilize CARE [42] backbone for extracting shared representations across all tasks. CARE backbone leverages a mixture of encoders to encode an input state into representations, which enables the incorporation of metadata and priori knowledge. Specifically, CARE uses a mixture of k encoders to learn k state-representations z^k_{enc}. By utilizing prior information, such as metadata or task identifiers, we can apply it to the combination of representations from k encoders. Contextual representation Z_{context} is obtained by passing the metadata or task identifiers through a MLP. Then, the representations of k encoders are combined into a composite representation z_{enc} by performing a weight sum using attention

mechanism:

$$z_{\text{enc}} = \sum_{i}^{k} \alpha_i \times z_{\text{enc}}^i \tag{6}$$

where

$$\alpha_j = \text{softmax}(z_{\text{enc}}^j \cdot z_{\text{context}}^i) \quad \forall j \in \{1, ..., k\} \tag{7}$$

We concatenate the composite representation z_{enc} with the contextual representation Z_{context} as the shared representations across all tasks, which is used as an input to the SP and ACP. Note that the CARE backbone is updated using both the SP policy loss and ACP policy loss.

C. Goal-oriented sparse rewards

Manually designed dense rewards incorporate prior knowledge and effectively guide policy learning. However, the magnitude of reward values does not directly indicate the ability of a policy to accomplish tasks. Therefore, we introduce goal-oriented sparse rewards, which are solely related to the completion of the final objective and are not influenced by biases from prior knowledge. With sufficient exploration and high-quality data, the agent can learn a globally optimal policy. The goal-oriented sparse rewards of a task $\mathcal{T}_i$ is characterized by an "ϵ-region" in state space, represented by:

$$R_i^s(s, a) = \begin{cases} \delta_{s_g}(s) & \text{if } f(s, s_g) \leq \epsilon \\ 0 & \text{else,} \end{cases} \tag{8}$$

In this equation, R_i^s denotes the goal-oriented sparse rewards of task $\mathcal{T}_i$, s is the current state, s_g denotes the goal state, $f(s, s_g)$ is a function that maps the goal state and current state to a latent space, computing the distance between them, i.e. we could set $f(s, s_g) = \|s - s_g\|_2$. δ_{s_g} defines the reward surface within the epsilon region and set $\delta_{s_g} = 1$. ϵ represents a small distance threshold.

D. Problem formulation

Following the description of CMDP (section II. A), the initial state distribution ρ_i determines the probability density of an episode starting at state s_0. For each transition (s_t, a_t, s_{t+1}) from task $\mathcal{T}_i$ at timestep t, the environment produces a scalar $R_i(s_t, a_t)$. It is worth mentioning that the reward function R_i is manually designed and intensive, which we refer to as guiding dense rewards. Similarly, the environment produces a scalar $R_i^s(s_t, a_t)$ as a goal-oriented sparse reward. Higher values for both the guiding dense rewards and the goal-oriented sparse rewards indicate better performance.

For each state s_t, the guiding dense rewards state value of policy π is denoted as:

$$V^{\pi}(s_t) = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathbb{E}_{\pi} \sum_{t'=t}^{\infty} \gamma^{t'-t} R_i(s_{t'}, a_{t'})]. \tag{9}$$

The guiding dense rewards state-action value is denoted as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[R_i(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P_i} V^{\pi}(s_{t+1})]. \tag{10}$$

Similarly, We can define V_s^{π} and Q_s^{π} for the goal-oriented sparse rewards.

After introducing goal-oriented sparse rewards, we reconsider the learning objectives of MTRL:

$$\{\max_{\pi} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathop{\mathbb{E}}_{s_0 \sim \rho_i} V^{\pi}(s_0), \max_{\pi} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathop{\mathbb{E}}_{s_0 \sim \rho_i} V_s^{\pi}(s_0)\}. \tag{11}$$

MTRL itself can be viewed as a multi-objective optimization problem. In our settings, the introduction of sparse rewards adds an additional objective to the MTRL framework, which increases the problem's complexity. Furthermore, the coupling relationship between the outputs of SP and ACP further complicates the optimization process. The dynamic nature of the two training objectives makes training difficult and hinders the collaboration of the two policies. To address this dilemma, we consider transforming multi-objective optimization into single-objective optimization. To convert multi-objective MTRL into single-objective MTRL, we assign a virtual expected budget C to the sparse rewards. This allows us to transform the MTRL objective as follows:

$$\begin{aligned} &\max_{\pi} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathop{\mathbb{E}}_{s_0 \sim \rho_i} V^{\pi}(s_0), \\ s.t. \quad &\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathop{\mathbb{E}}_{s_0 \sim \rho_i} V_s^{\pi}(s_0) + C \geq 0, \end{aligned} \tag{12}$$

In this equation, C represents a virtual expected budget. Specifically for each time step, the constraint in Eq.12 can be rewritten as:

$$\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})}[\mathbb{E}_{\pi}[\sum_{t} \gamma^t (R_i^s(s_t, a_t) + c)]] \geq 0, \tag{13}$$

Here, c denotes the expected budget specific to each step, and it relates to C through the equation $\sum_{t=0}^{\infty} \gamma^t c = \frac{c}{1-\gamma} = C$. Notably, the expected budget represents an average target to be achieved rather than a strict enforcement.

To simplify the problem, we transform the constrained single-objective MTRL into an unconstrained single-objective MTRL using the Lagrangian method. This method converts the constrained optimization problem Eq.(12) into an unconstrained one by introducing a multiplier λ:

$$\begin{aligned} &\min_{\lambda \geq 0} \max_{\pi} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathop{\mathbb{E}}_{s_0 \sim \rho_i} V^{\pi}(s_0) \\ &+ \lambda \left(\mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \mathop{\mathbb{E}}_{s_0 \sim \rho_i} V_s^{\pi}(s_0) + C \right). \end{aligned} \tag{14}$$

In this formulation, the weight of the goal-oriented sparse rewards combined with the guiding dense rewards is represented by λ. We solve this objective by using model-free MTRL algorithms.

E. Objective function

We utilize an off-policy actor-critic approach to train the two policies. Given the overall policy $\pi_{\psi \circ \phi}(a|s)$, we use the typical Temporal Difference (TD) [52] backup to learn $Q^{\pi_{\psi \circ \phi}}(s, a; \theta)$ and $Q_s^{\pi_{\psi \circ \phi}}(s, a; \theta_s)$, which are parameterized as $Q(s, a; \theta)$ and $Q_s(s, a; \theta_s)$ respectively. Here, θ and $\theta_s$represents the network parameters separately for the two state-action values. When provided with s_{t+1} and a_{t+1}, the Bellman backup operator for the guiding dense rewards state-action value is expressed as:

$$\mathcal{T}^{\pi_{\psi \circ \phi}} Q(s_t, a_t; \theta) = \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} R_i(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1}; \theta), \tag{15}$$

where R_i represents the guiding dense rewards obtained from task $\mathcal{T}_i$. The backup operator for the goal-oriented sparse rewards state-action value Q_s is defined similarly with R^s. Both $Q(s,a;\theta)$ and $Q_s(s,a;\theta_s)$ can be learned from transitions (s_t, a_t, s_{t+1}) sampled from a replay buffer.

To achieve off-policy training of ψ and ϕ, we convert Eq.(14) into a bi-level optimization surrogate. The formulation is presented as follows:

$$\begin{aligned} &\text{(a)} \quad \max_{\phi,\psi} \mathop{\mathbb{E}}_{s\sim\mathcal{D}} \left[Q(s,a;\theta) + \lambda Q_s(s,a;\theta_s))\right], \\ &\text{(b)} \quad \min_{\lambda\geq 0} \lambda\Lambda_{\pi_{\psi\circ\phi}}. \quad \Lambda_{\pi_{\psi\circ\phi}} \triangleq \mathop{\mathbb{E}}_{s_0\sim\rho_i} V_s^{\pi_{\psi\circ\phi}}(s_0) + C \end{aligned} \tag{16}$$

Here, $\mathcal{D}$ represents a replay buffer containing a historical marginal state distribution used to train the policies. The initial state distribution ρ is employed to train λ. This distinction arises from the idea that when fine-tuning λ, we should primarily consider how well the policy satisfies the virtual expected budget starting with ρ, rather than with some historical state distribution. Subsequently, We further transform the off-policy objective(Eq.(16),a) into two parts:

$$\begin{aligned} &\text{(a)} \quad \max_{\phi} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D},\hat{a}\sim\pi_\phi(\cdot|s),\\ \Delta a\sim\pi_\psi(\cdot|s,\hat{a})\\ a=h(\hat{a},\Delta a)}} [Q(s,a);\theta], \\ &\text{(b)} \quad \max_{\psi} \mathop{\mathbb{E}}_{\substack{s\sim\mathcal{D},\hat{a}\sim\pi_\phi(\cdot|s),\\ \Delta a\sim\pi_\psi(\cdot|s,\hat{a})\\ a=h(\hat{a},\Delta a)}} [-d(a,\hat{a}) + \lambda Q_s(s,a;\theta_s)], \end{aligned} \tag{17}$$

In this modified formulation, the distance function $d(a,\hat{a})$ quantifies the change from $\hat{a}$ to a. h represents the action correction function, which is used to combine the actions output by the two policy networks to obtain the final action. ACP π_ψ aims to maximize the goal-oriented sparse rewards while minimizing the distance between the actions before and after the correction. On the other hand, SP π_ϕ solely focuses on maximizing the guiding dense rewards. Notably, ACP modifies SP's action, which aligns with the discussed motivation for efficient exploration. The training objective (Eq.(17),b) for ACP relies on a critic Q_s, which learns the expected future goal-oriented sparse rewards. Consequently, guided by Q_s, ACP explores actions with more significant potential in long-term sequences.

F. Action correction function and Distance function

In this section, we present the design for the action correction function $h(\hat{a},\Delta a)$ and the distance function $d(a,\hat{a})$.

1) Action correction function: We opt for the correction function h to be primarily additive and non-parametric, which helps to reduce training difficulty. Without loss of generality, we assume a bounded action space $[-A, A]$, and that both $\hat{a}$ and Δa are already within this space. Consequently, we define:

$$a = h(\hat{a},\Delta a) = \min(\max(2\hat{a} + \Delta a, -A), A), \tag{18}$$

where min and max are element-wise. The multiplication by 2 and the clipping ensure that $a \in [-A, A]$, which means that SP retains full control over the final action and can overwrite ACP's action if necessary. This is crucial because ACP faces challenges in learning an effective correction policy in the short horizon when SP still dominates the learning process. Although the additive operation is simple, the overall editing process is sufficiently general to encompass any modification.

This additive action correction function is driven by the objective of achieving sparsity. Policies are generally assessed using metrics such as success, which are only triggered for specific states. To explicitly incorporate this inductive bias, we adopt the additive action correction function, which guarantees that ACP learns a policy that optimizes metrics like success in relation to SP. This simplifies the optimization landscape of ACP.

2) Distance function: we utilize the hinge loss to compare the guiding dense rewards state-action values of $\hat{a}$ and a:

$$d(a,\hat{a}) \triangleq \max(0, Q(s,\hat{a};\theta) - Q(s,a;\theta)) \tag{19}$$

Here, Q represents the critic and θ denotes the parameters. This loss yields zero if the edited action a already attains a higher state-action value than the preliminary action $\hat{a}$. In such cases, only Q_s is optimized by π_ψ. Otherwise, the inner part of (Eq.(17),b) is recovered as $Q(s,a) + \lambda Q_s(s,a)$. Our distance function in the critic Q is more appropriate than L_2 distance in the action space because we ultimately care about how the Q changes after the action is edited.

G. Training process

To practically train the objectives, we employ stochastic gradient descent (SGD) simultaneously to Eq.(16)(b) and Eq.(17). The re-parameterization trick is utilized for both π_ψ and π_ϕ to enable the application of SGD. To assess $\Lambda_{\pi_{\psi\circ\phi}}$, a batch of rollout experiences $\{(s_n, a_n)\}_{n=1}^N$ following $\pi_{\psi\circ\phi}$ is provided. The gradient of λ (Eq.(16)(b)) is approximated as:

$$\Lambda_{\pi_{\psi\circ\phi}} \approx \frac{1}{N}\sum_{n=1}^{N} R_s(s_n, a_n) + c \tag{20}$$

Here, c represents the virtual expected budget defined in Eq.(13). Subsequent to each rollout, a batch of goal-oriented sparse rewards is collected, each reward is compared to $-c$, and the mean of the differences is used to adjust λ. This approximation enables the updating of λ using mini-batches of data, rather than waiting for complete episodes to conclude or relying on the often inaccurate estimated $V_s^{\pi\psi\circ\phi}$. Furthermore, the policy of SP and ACP are learned by minimizing:

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim D}[\mathbb{E}_{\hat{a}\sim\pi_\phi}[\alpha\log(\pi_\phi(\hat{a}|s_t)) - Q(s_t,\hat{a};\theta)]]. \tag{21}$$

$$\begin{aligned} J_\pi(\psi) =& \lambda\mathbb{E}_{s_t\sim D}[\mathbb{E}_{\Delta a\sim\pi_\psi}[\alpha\log(\pi_\psi(\Delta a|s_t)) \\ &- Q_s(s_t,\Delta a;\theta_s)]] - d(a,\hat{a}). \end{aligned} \tag{22}$$

where α represents a learnable regularization coefficient that controls the significance of the entropy term. Multiple parallel environments are employed to mitigate temporal correlation within the rollout batch data, which constitutes the data to be placed into the replay buffer.

The computational graph presenting Eq.(17) is illustrated in Fig. 3. Our approach is applicable to various goal-oriented sparse rewards and action distance functions. To encourage exploration, we integrate SAC [25], which incorporates the

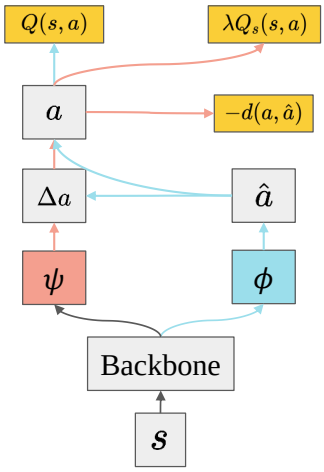


Fig. 3: The computation graph of Eq.(17). Nodes denote variables or networks, and edges denote operations. The orange blocks are negative losses, the blue paths are the gradient paths of ϕ, and the red paths are the gradient paths of ψ.

entropy terms of π_ψ and π_ϕ in Eq.(17), dynamically adjusting their weights based on two entropy targets as described in [25]. In our experiments, both SP and ACP are trained from scratch. The pseudocode for the overall TSAC is shown in the algorithm 1.

Algorithm 1: TSAC: Task-Specific Action Correction

Input: N tasks; virtual expected budget C

Initialize θ,ϕ and ψ; reset the replay buffer $\mathcal{D} \leftarrow \emptyset$
for *each training iteration* **do**
  Reset the rollout batch $\mathcal{B} \leftarrow \emptyset$;
  for *each rollout step* **do**
    for *task* $i \cdots N$ **do**
      Action proposal by SP: $\hat{a} \sim \pi_\phi(\cdot|s)$;
      Action correcting by ACP: $\Delta a \sim \pi_\psi(\cdot|s,\hat{a})$;
      Output action $h(\hat{a}, \Delta a)$;
      Task i's transition $s' \sim \mathcal{P}_i(s'|s,a)$;
      Add the transition to the rollout batch $\mathcal{B} \leftarrow \mathcal{B} \bigcup \{(s,a,s',R_i(s,a),R_i^s(s,a))\}$;
    end
  end
  Store the rollout batch in the buffer $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{B}$;
  Sample a training mini-batch $\mathcal{B}_t$ from the replay buffer $\mathcal{D}$ for computing gradient;
  Perform one gradient step on the critic parameters θ by TD backup (Eq.15) on Q and Q_s;
  Estimate the gradient of the Lagrangian multiplier λ by evaluating Eq.20 on $\mathcal{B}_t$;
  Optimize the multiplier by $\lambda \leftarrow \lambda - \alpha\Lambda_{\pi_{\psi\circ\phi}}$;
  Use SGD to optimize SP: $\phi \leftarrow \phi + \alpha\Delta\phi$ (gradient of Eq.17, a);
  Use SGD to optimize ACP: $\psi \leftarrow \psi + \alpha\Delta\psi$ (gradient of Eq.17, b);
  Update other parameters such as entropy weight, target critic network, etc.
end

V. Experiments

A. Experimental Setup

In this section, we evaluate the performance of TSAC on the Meta-World multi-task RL environment [26] and use Meta-World's MT10 and MT50 benchmarks, which is shown in Fig.4. To further validate that TSAC is algorithm-agnostic, highly generalizable, and can be extended to multi-agent scenarios, we apply it to more challenging multi-task StarCraft II multi-agent scenarios (SMAC) [53], which is shown in Fig.5. SMAC contains several tasks that similar but different, such as the *marine*-series tasks (3m, 8m, 8m_vs_9m...) and the *stalker_zealot*-series tasks (2s3z, 3s5z, 3s5z_vs_3s6z...), which naturally form multi-task settings.

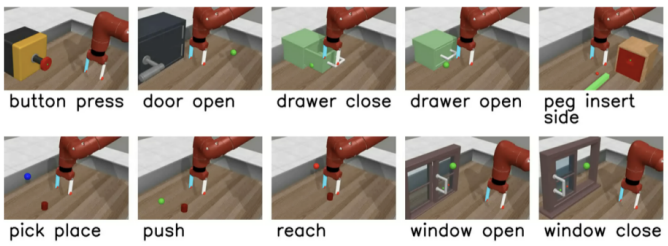


Fig. 4: The MT10 benchmark from Meta-World contains 10 tasks: reach, push, pick, open window and so on.

We compare TSAC against several baseline methods and conduct ablation studies to verify the effectiveness of our approach.

The first goal of our experimental evaluation is to assess Whether TSAC improves the performance of a multi-task agent. In Meta-World, e compare the performance of TSAC in two different settings: a short horizon to evaluate its sample efficiency and a long horizon to measure its overall performance. For comparison, We select CARE [42], MT–SAC, Soft Modularization [35] and PCGrad [14] as our baselines.

(a) SMAC 3s_vs_5z (b) SMAC 2c_vs_64zg (c) SMAC corridor

Fig. 5: Snapshoots of SMAC.

Furthermore, we evaluate different action correction functions to identify which yields the best performance. The action correction functions considered are SP-dominated, ACP-dominated, equal, and Softclip (See Section IV-C for the definitions) .

B. Baselines

In this paper, we will compare our method against the following baselines:

CARE: As a representations sharing method, representations are shared to learn how to compose them by leveraging

additional information about each task.

MT–SAC: This approach directly applies the SAC algorithm to the multi-task setting. It utilizes a shared backbone with disentangled alphas.

Soft Modularization: As a parameters sharing method, Soft Modularization shares parameters and uses a routing network to combine all possible routes for each task softly.

PCGrad: It is a gradient manipulation method that projects a task's gradient onto the normal plane of any other conflicting task. However, it has high time complexity and is not suitable for MT50 in the long horizon.

TSAC(ours): Our proposed method builds upon the CARE backbone, leveraging its representation-sharing module. In contrast, our approach is based on behavior sharing and utilizes goal-oriented sparse rewards. In multi-task StarCraft II multi-agent scenarios (SMAC), We compare TSAC with DT2GS [54], UPDeT [55] and ASN [56], which are mainly dedicated to generalization in multi-agent settings. DT2GS is chosen because it is the SOTA method and possesses sound zero-shot generalization capability across tasks by maintaining consistent yet scalable semantics. It is also used in the attentive action decoder in MA2RL. UPDeT develops a multi-agent framework based on the transformer block to adapt to tasks with varying observation/state/action spaces. ASN considers the semantic difference of actions and forms a foundation of generalizable models in MARL. Similar to the settings in DT2GS, our experiments use "ASN_G" to denote the generalizable ASN. The criteria for choosing baselines depend on whether they can be applied across tasks in MARL or state-of-the-art methods.

In TSAC, we introduce an important hyperparameter:c as in Eq.13, correlated to the virtual expected budget. For MT10, we use $c = 0.6$ and $c = 0.4$ for MT50. c are used to adjust the learning weights between SP and ACP. The hyperparameters of TSAC are shown in Table. II and Table. I.

TABLE I: Hyerparameter values that are common across all the methods

Hyperparameter	Value
batch size	1280
actor/critic size	three fully connected layers with 400 units
activation function	ReLU
policy initialization	standard Gaussian
policy learning rate	3e-4
Q funtion learning rate	3e-4
optimizer	Adam
beta for Adam optimizer	(0.9,0.999)
discount	0.99
episode length	150
reward scale	1.0

TABLE II: Hyperparameter values for TSAC and CARE

Hyperparameter	Value
task encoder size	two layer network. Hidden units=50
number of encoders/experts	4 for MT10 and MT50
temperature	learned and disentangled with tasks
expected budget	0.6 for MT10 and 0.4 for MT50

TABLE III: Success rate of baselines on the short horizon (150k steps per task) for MT10 and MT50.

Agent	MT10	MT50
TSAC(ours)	**0.390 ± 0.115**	**0.362 ± 0.051**
CARE	0.260 ± 0.062	0.277 ± 0.028
MT-SAC	0.198 ± 0.068	0.220 ± 0.035
Soft-Mod	0.180 ± 0.108	0.151 ± 0.040
PCGrad	0.276 ± 0.107	0.238 ± 0.035

TABLE IV: Success rate of baselines on MT10 on the long horizon (1M steps per task). Results are reported at the end of the 0.8M,1M steps and at the best average value.

Agent	0.8M	1M	Best
TSAC(ours)	**0.762 ± 0.109**	**0.722 ± 0.052**	**0.827 ± 0.038**
CARE	0.667 ± 0.082	0.642 ± 0.076	0.708 ± 0.114
MT-SAC	0.574 ± 0.097	0.555 ± 0.166	0.635 ± 0.120
Soft Mod	0.549 ± 0.087	0.560 ± 0.107	0.596 ± 0.102
PCGrad	0.655 ± 0.150	0.644 ± 0.090	0.708 ± 0.114

TABLE V: Success rate of baselines on MT50 on the long horizon (1M steps per task). Results are reported at the end of the 0.8M, 1M steps and at the best average value.

Success——MT50	0.8M	1M	Best
TSAC(ours)	**0.450 ± 0.046**	**0.445 ± 0.045**	**0.524 ± 0.030**
CARE	0.418 ± 0.057	0.395 ± 0.031	0.497 ± 0.035
MT-SAC	0.381 ± 0.044	0.390 ± 0.045	0.431 ± 0.046
Soft Mod	0.155 ± 0.033	0.162 ± 0.034	0.207 ± 0.051

C. Comparative evaluation

Fig. 6a shows the average success rate on the 10 tasks of the MT10 benchmark form Meta-world for TSAC, CARE, MT–SAC, Soft Modularization, and PCGrad. Since the success rate is a binary variable, it is noisy; therefore, the results were averaged across multiple seeds, and the curves were smoothed. Mean and standard error are reported for each value.

We consider 1 million steps as a long horizon and 150 thousand steps as a short horizon. The short horizon is utilized to observe the exploration ability of different methods, while the long horizon is used to visualize the performance of the method at various time points. It is worth noting that all methods are trained using SAC with disentangled alphas.

Table III and Fig. 6a demonstrate that our method outperforms all the baselines on the short horizon in MT10. Additionally, Fig. 6c illustrates that even in MT50 our method still outperforms all baselines. For comparison, it takes around 1 million steps for the Multi-task SAC agent to reach the accuracy that our TSAC agent achieves around 300 thousand steps, suggesting that our method is highly sample-efficient and exploration-efficient.

Table IV and V as well as Fig. 6b and Fig. 6d depict that TSAC is able to learn a good policy on the long horizon. For MT10, TSAC performs best and reaches a top success rate of 0.827 during the training. Furthermore, sampling the success rate around 0.8 million and 1 million steps reveals that TSAC has the best performance. For MT50, conflicts

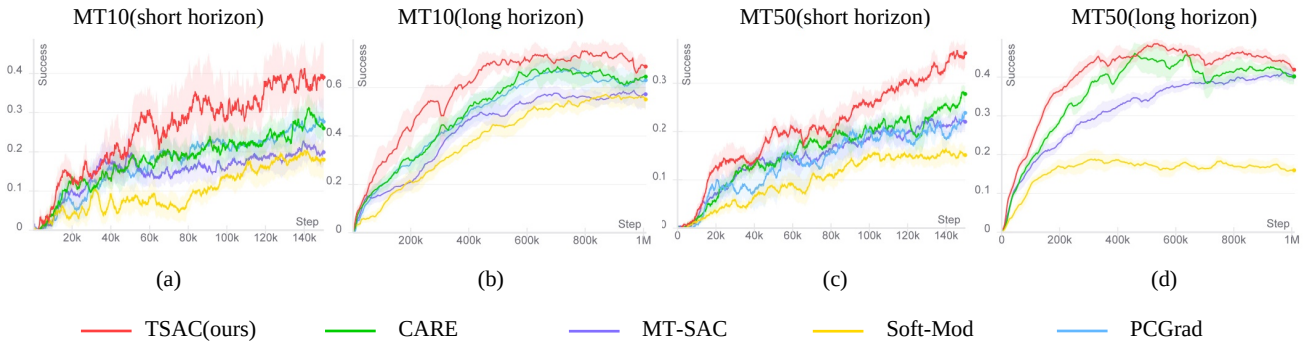


Fig. 6: The performance of TSAC and baselines, including CARE, MT–SAC, Soft Modularization, PCGrad, are compared in MT10 and MT50. The evaluation is conducted on two horizons: short horizon(150K) and long horizon(1M). The bolded lines represents the mean over 4 runs for both the short horizon and long horizon. The shaded area represents the standard error.

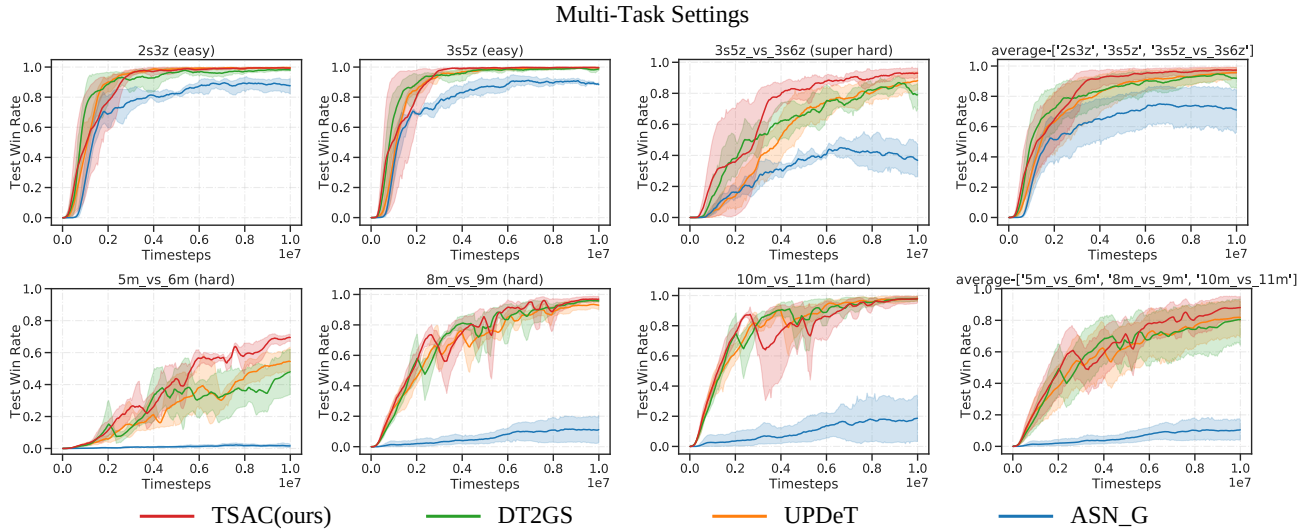


Fig. 7: The performance of TSAC and baselines, including DT2GS, UPDeT, ASN_G, are compared in multi-task settings. The evaluation is conducted on 2 multi-task problems with different distributions of difficulty: (2s3z, 3s5z, 3s5z_vs_3s6z),(5m_vs_6m, 8m_vs_9m, 10m_vs_11m).

between tasks become more acute. CARE stops learning around 600 thousand steps, and performance begins to decline in the following training steps. Despite suffering from the conflicts between tasks, TSAC's performance declines, but it still performs better than CARE. MT-SAC achieves smooth learning and performs similarly to CARE and TSAC at 0.8 million and 1 million steps. However, in terms of the best performance throughout training, MT–SAC is far inferior to TSAC, which outperforms all methods. Since the success metric is a noisy binary variable, the best performance is obtained by smoothing over the curve. Importantly, TSAC achieves average success rates of 0.450 and 0.445 on MT50 at 0.8 million and 1 million steps, surpassing the reported results from CARE, MT–SAC and Soft Modularization.

To further evaluate the proposed method in multi-task multi-agent scenarios, We conduct two multi-task settings with different distributions of difficulty: *stalker_zealot*-series tasks (easy, easy, superhard) and *marine*-series tasks (hard, hard, hard). In each multi-task setting, the policy interacts synchronously with multiple tasks and updates the policy using a mixed experience replay buffer. As shown in Fig. 7, TSAC exhibits the fastest convergence and highest win rate among the compared methods (DT2GS, UPDeT, ASN_G). DT2GS and UPDeT closely follow MA2RL in asymptotic performance. ASN_G fails to win in the hard multi-task setting.

D. Zero-shot generalization

We want to evaluate whether the two policies of TSAC can cooperate and contribute to zero-shot generalization in unseen environments. In the experiment, we trained the agents on eight environments from MT10 and evaluate on two held-out environments. As illustrated in table VI, TSAC surprisingly exhibits promising performance, outperforming all the baselines by a large margin within 1M steps.

To further evaluate TSAC, we evaluate the generalization capability in six different settings in SMAC, each of which includes a target task that is either more difficult or equally difficult compared to the source task. Fig. 8 shows that TSAC outperforms all baselines in terms of zero-shot

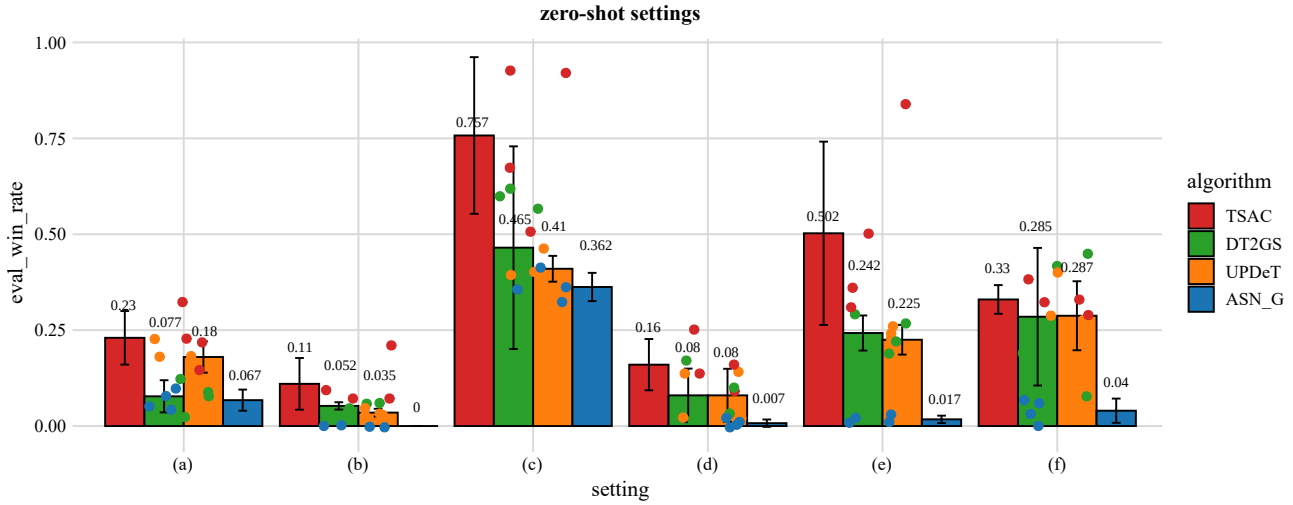


Fig. 8: The zero-shot generalization capability of TSAC and its baselines, including DT2GS, UPDeT, ASN_G, was compared across various source and target tasks. The evaluation was conducted on 6 zero-shot settings and the horizontal axis represent the source task → the target task, where (a) 2s3z→3s5z, (b) 3s_vs_4z→3s_vs_5z, (c) 3s5z→3s5z_vs_3s6z , (d) 8m_vs_9m→5m_vs_6m, (e) 10m_vs_11m→8m_vs_9m, (f) 5m_vs_6m→10m_vs_11m

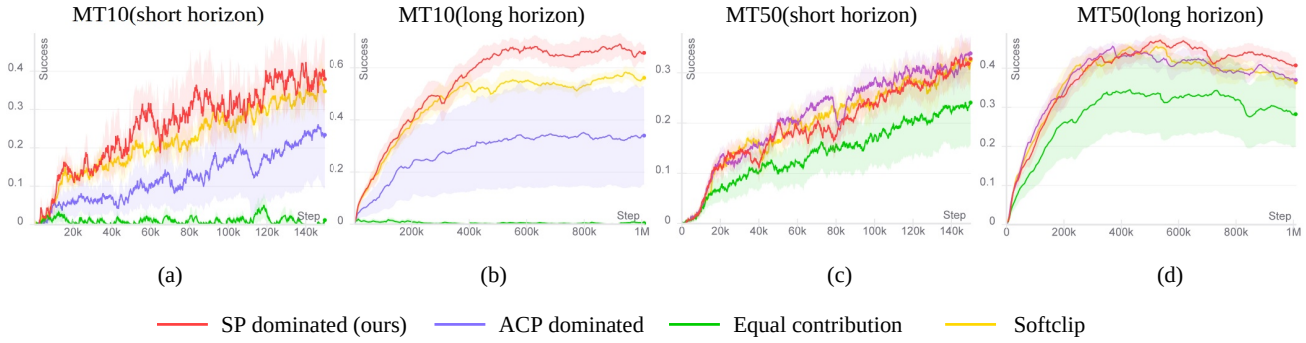


Fig. 9: The performance of TSAC with different action correction functions, including SP dominated, ACP dominated, equal contribution, Softclip, are compared in MT10 and MT50. The evaluation is conducted on two horizons: short horizon (150K) and long horizon (1M). The bolded lines represents the mean over 4 runs for both the short horizon and long horizon. The shaded area represents the standard error.

TABLE VI: Evaluate zero-shot performance on held-out environments from MT10, after training on remaining 8 environments for 1 million steps. Results averaged over 4 seeds and the success rate of each seed is calculated as the average success rate at time steps [0.81M, 0.84M, 0.87M, 0.9M, 0.93M, 0.96M, 0.99M].

Agent	success(mean ± stderr)
TSAC(ours)	**0.1607 ± 0.0592**
CARE	0.0357 ± 0.0357
MT-SAC	0.0535 ± 0.0309
Soft-Mod	0.0357 ± 0.0618
PCGrad	0.0178 ± 0.0309

capability, especially in challenging settings (e.g. 2s3z→3s5z, 8m_vs_9m→5m_vs_6m). We aim to highlight the potential benefits of leveraging goal-oriented sparse rewards for zero-shot generalization. Furthermore, compared to baselines, the structure of two policies in TSAC is more advantageous for generalization.

E. Investigation on action correction functions

The result in the previous section suggests that TSAC is both sample efficient and exploration efficient. Furthermore, TSAC yields a good policy on long horizons. In this section, we further investigate the impact of the various action correction functions employed by TSAC and discuss the potential for improved performance.

In this experiment, We will consider four different functions h:

SP dominated (ours): $h = \min(\max(2\hat{a} + \Delta a, -A), A)$, SP's action dominates the final action.

ACP dominated: $h = \min(\max(\hat{a} + 2\Delta a, -A), A)$, ACP's action dominates the final action.

equal contribution: $h = \min(\max(\hat{a} + \Delta a, -A), A)$, SP and ACP contribute equally to the final action.

Softclip: $h = Softclip(2\hat{a} + \Delta a)$, h use softclip to smooth out the output action and bring in nonlinearity.

Fig. 9 illustrates that SP dominated action correction function outperforms other functions in producing the final action. However, when SP and ACP contribute equally to the final action, the agent encounters difficulties learning a policy. This conflict relationship between SP and ACP presents challenges in optimizing each respective goal. Conversely, when either SP or ACP dominates the action correction function, the agent can successfully learn a well-performing policy. Comparatively, SP outperforms ACP due to ACP's focus on optimizing the goal-oriented sparse rewards, which is challenging to train due to its sparsity. In Fig. 9(b)(d), We found that the SP dominated significantly outperforms the ACP dominated in terms of performance, which can be attributed to the guiding dense rewards. Although ACP is trained under sparse rewards, ACP dominated exhibits similar performance to SP dominated, indicating that excessive focus on specific task details can lead to conflicts between tasks. The Softclip function exhibits a slight decrease in performance compared to our action correction function. This decline can be attributed to the introduction of nonlinearity, which further complicates and hampers the learning process. Consequently, we opt for a simple and linear operation instead of training a learnable network.

F. Investigation on virtual expected budget c and λ

To investigate the effect of virtual expected budget c and λ on the model's performance, we conduct experiments on different values of c and λ as constants ($\lambda = 1$). When $\lambda = 1$, both SP and ACP optimize their objectives during the training process with equal weights.

To further analyze the influence of the virtual expected budget c on the performance of TSAC, experiments were conducted on the MT10 and MT50. The performance is shown in Fig. 10. In MT10, Fig. 10(a) indicates that both too small and too large values of c will affect the performance of TSAC. This is because, during the training process, ACP serves as an auxiliary learner when the performance of SP reaches a bottleneck. When c is too large, the entire training process degenerates into a multi-objective optimization, and the coupling relationship between SP and ACP greatly increases the difficulty of training. Its impact on performance is equivalent to lambda being a constant.

As claimed in Section IV-C, the coupling relationship between the outputs of SP and ACP further complicates the optimization process. Fig. 11 provides strong support for this argument. As shown in the figure, TSAC_$\mathbf{lambda}$ $= 1$ exhibits a performance degradation compared to TSAC and CARE, indicating that SP and ACP interfere with each other, hindering their cooperation. Thus, One key step to ensure the extraordinary performance of TSAC is to transform multi-objective optimization into single-objective optimization.

G. Ablations

We carry out ablation studies to investigate the individual contributions of SP and ACP. We compare TSAC against

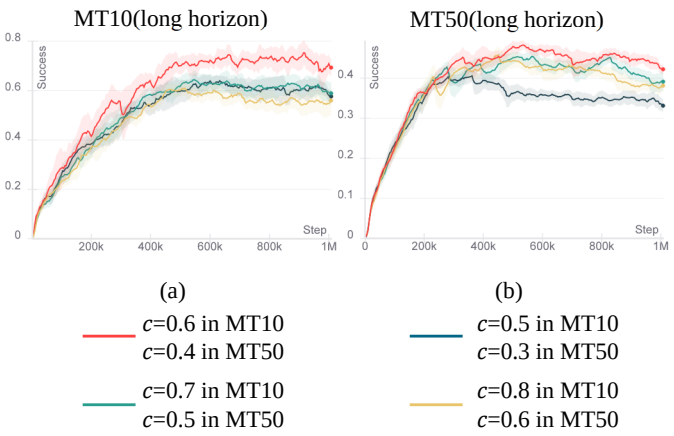


Fig. 10: The performance of TSAC with different c in MT10 and MT50.

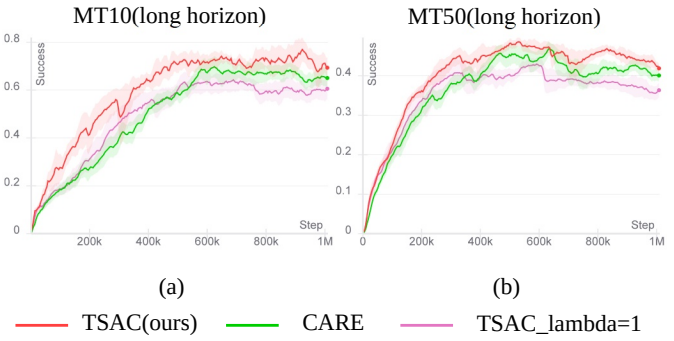


Fig. 11: The performance of TSAC with λ as constants ($\lambda = 1$), including TSAC, CARE(without ACP), TSAC_lambda $= 1$, are compared in MT10 and MT50.

two ablations: (1)TSAC_w/o_ACP(CARE), which excludes ACP. (2)TSAC_w/o_SP, which excludes SP. In these ablations, all other structural models are kept consistent strictly with the full TSAC. As shown in Fig. 12, the removal of ACP significantly drops the performance of TSAC on MT10 and MT50. These results indicate that ACP is crucial in enabling the agent to adopt a long-term perspective regarding conflicts. Furthermore, the comparison of TSAC and TSAC_w/o_SP means that ACP struggles with sparse rewards, while SP can provide high-quality samples for ACP to assist in learning under sparse rewards.

To further elucidate the general applicability of TSAC, we conducted supplementary experiments across environments characterized by varying degrees of reward sparsity. In the context of multitask SMAC, the dense rewards are comprised of two components: the damage reward, which is awarded for inflicting damage on enemy units, and the kill reward, which is granted for successfully eliminating enemy units. As presented in Table VII, we assess the performance of TSAC under different levels of reward sparsity and compare it with UPDeT. The experimental findings indicate that TSAC demonstrates robust performance across a spectrum of sparse reward conditions and exhibits adaptability to diverse task reward configurations.

TABLE VII: The win rate of TSAC in multi-task SMAC with varying degrees of reward sparsity and compare with UPDeT. Results are reported at the end of the 6M steps on *marine*-series tasks and *stalker_zealot*-series tasks.

algorithm in SMAC tasks	complete reward	only damage reward	only death reward
UPDeT in *stalker_zealot*	99.29±0.42; 98.89±0.95; 70.96±13.42	96.33±3.43; 98.06±2.40; 62.71±31.89	98.13±1.36; 99.02±0.80; 86.54±6.90
TSAC in *stalker_zealot*	98.34±1.64; **99.29±0.42**; **88.96±2.90**	**97.75±1.67**; 97.93±1.11; **66.92±10.05**	**98.28±1.24**; **99.03±0.73**; 85.48±2.23
UPDeT in *marine*	35.88±13.54; 84.42±5.80; 94.65±3.24	24.77±15.16; 65.32±21.49; 69.63±18.23	61.63±26.73; 63.44±28.14; 58.10±26.13
TSAC in *marine*	**54.71±4.43**; **91.88±6.68**; 86.28±16.87	**45.26±10.21**; **92.48±3.69**; **94.89±3.46**	54.08±36.46; **69.25±19.85**; **78.35±13.90**

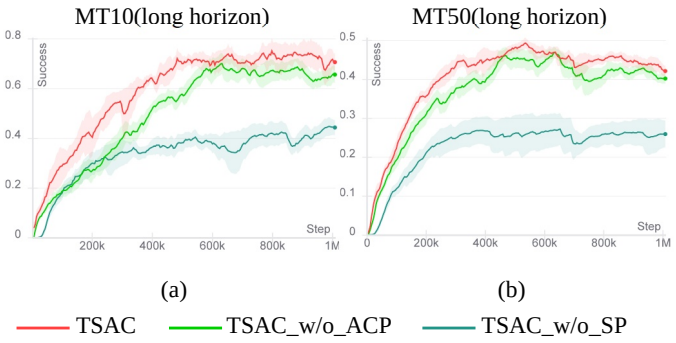


Fig. 12: Ablation studies. TSAC_w/o_ACP(CARE), it removes the usage of ACP and TSAC_w/o_SP, it removes the usage of SP.

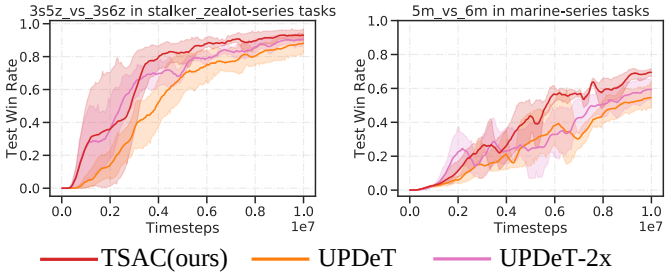


Fig. 13: Experiments on doubling the parameters. UPDeT-2x, it simply doubles the parameters of the policy network.

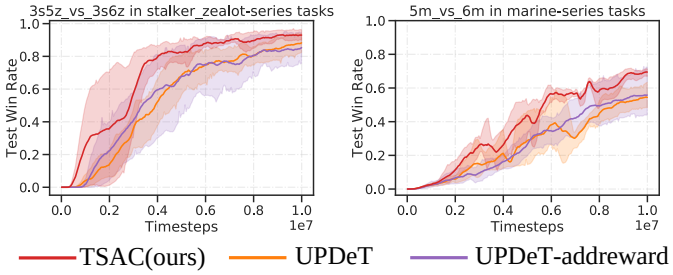


Fig. 14: Experiments on combine goal-oriented sparse rewards. UPDeT-addreward, it directly combines goal-oriented sparse rewards and guiding dense rewards in an additive manner.

H. Analysis of two strategies and sparse rewards

In Section V-G, we discussed the individual contributions of SP and ACP. To further validate the necessity of decomposing policy learning into two policies in TSAC and the performance enhancement in multi-task scenarios brought about by the cooperation of these two strategies, this subsection designs two additional experiments to address the following two questions:
Q1: Why does TSAC decompose the policy into two policy networks rather than directly doubling the parameters of a single policy network?
Q2: Why not directly combine goal-oriented sparse rewards and guiding dense rewards?

As shown in Fig. 13, doubling the parameters of a single policy network enhances the learning capacity of the network, resulting in a slight improvement in overall policy learning. However, it does not effectively address the short-term conflicts between tasks in multi-task settings. TSAC, on the other hand, addresses this challenge through policy decomposition and the introduction of goal-oriented sparse rewards, providing a longer-term perspective for handling short-term conflicts. This framework better leverages the advantages brought by the increase in parameter count, leading to improved performance in multi-task reinforcement learning.

To answer **Q2**, Fig. 14 demonstrates that merely combining sparse rewards with goal-oriented sparse rewards does not lead to performance improvements. This is because, during the learning process of the Q-network, most updates are guided by guiding dense rewards. Dense rewards can mislead the agent into deceptive local optima, rendering goal-oriented sparse rewards nearly ineffective during the policy learning process. In contrast, TSAC employs two policy networks that update based on guiding dense rewards and goal-oriented sparse rewards, respectively. This paradigm effectively leverages the long-term perspective offered by sparse rewards, thereby addressing short-term task conflicts in multi-task settings.

I. Visualization of the interaction between the SP and ACP.

In order to conduct a more detailed analysis of the interaction between two actions, we have plotted the curves of the original actions (a_hat) output by the SP and the final actions (a) corrected by the ACP. In the Metaworld environment, the action space consists of four dimensions, and we have created separate plots for each dimension. As shown in Fig. 15, it is observed that the action output by the SP, denoted as a_{hat} , is more conservative when handling multiple tasks. This conservativeness is reflected in the relatively small fluctuations of the curve in the figure. The underlying reason for this behavior is the trade-off between the dense guiding rewards associated with the multiple tasks, which necessitates that the SP considers all tasks to the greatest extent possible in order to generate its action. Conversely, the action a that has been corrected by the ACP appears to be more aggressive. This shift in behavior can be attributed to the introduction of goal-oriented sparse rewards, which allows the policy to adopt a long-term perspective and disregard short-term conflicts

between tasks. The primary objective here is to facilitate the successful completion of the final task. In the figure, this is manifested as greater fluctuations in the action output.

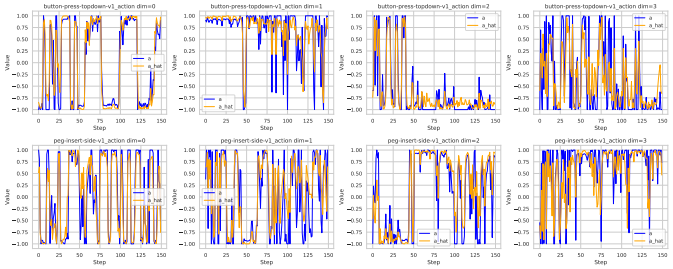


Fig. 15: The interactive of SP and ACP. we have plotted the curves of the original actions (a_hat) output by the SP and the final actions (a) corrected by the ACP.

VI. Conclusions and Future Work

In this paper, we present the Task-Specific Action Correction (TSAC), a general and complementary MTRL framework inspired by Safe Reinforcement Learning, that surpasses the several well-performed baselines on Meta-World and multi-task StarCraft II multi-agent scenarios. We show that TSAC is able to learn a high-performing policy and achieve significant improvements in both sample efficiency and final performance compared to previous state-of-the-art multi-task policies. Furthermore, TSAC is general and complementary, allowing it to be integrated with existing methods such as CARE and MT-SAC, and it can even be combined with multi-agent reinforcement learning approaches. Finally, we show the benefits of decomposing policy learning into two policies and the validity of introducing goal-oriented sparse rewards.

Although the paper presents a novel framework for MTRL, there are still some limitations. TSAC decomposes policy learning into two strategies, which introduces an additional burden in terms of parameter count. Furthermore, the hyperparameter c requires fine-tuning to achieve optimal performance, although its value range can be approximately determined based on empirical observations.

TSAC can be well extended to other domains or types of MTRL problems, particularly in cases where prior knowledge is lacking and guiding dense rewards are not well-designed. We believe that research in fields such as real-world robotics can fully utilize goal-oriented reward signals to enhance learning performance. In future work, We will introduce a pre-trained paradigm which policy is pretrained to maximize guiding dense rewards, and this policy is used as the initialization for SP. In this case, SP cannot be frozen because ACP continuously modifies its MDP. Instead, SP needs to be fine-tuned to adapt to the evolving actions of ACP. This pre-trained SP has the potential to accelerate the convergence of TSAC.

References

[1] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision making in multiagent systems: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 514–529, 2018.

[2] F. Khemakhem, H. Ellouzi, H. Ltifi, and M. B. Ayed, "Agent-based intelligent decision support systems: A systematic review," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 14, no. 1, pp. 20–34, 2022.

[3] 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.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[5] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[6] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[7] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] L. Pinto and A. Gupta, "Learning to push by grasping: Using multiple tasks for effective learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2161–2168.

[10] K. Jothimurugan, S. Hsu, O. Bastani, and R. Alur, "Robust subtask learning for compositional generalization," in *International Conference on Machine Learning*. PMLR, 2023, pp. 15 371–15 387.

[11] A. Silva, N. Moorman, W. Silva, Z. Zaidi, N. Gopalan, and M. Gombolay, "Lancon-learn: Learning with language to enable generalization in multi-task manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1635–1642, 2021.

[12] W. Ding, F. Luo, C. Gu, Z. Dai, and H. Lu, "A multiagent meta-based task offloading strategy for mobile-edge computing," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 1, pp. 100–114, 2024.

[13] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[14] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[15] B. Liu, Z. Pu, Y. Pan, J. Yi, M. Chen, and S. Wang, "Qfuture: Learning future expectation cognition in multiagent reinforcement learning," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 4, pp. 1302–1314, 2024.

[16] S. Wang, Z. Pu, Y. Pan, B. Liu, H. Ma, and J. Yi, "Long-term and short-term opponent intention inference for football multi-player policy learning," *IEEE Transactions on Cognitive and Developmental Systems*, pp. 1–15, 2024.

[17] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *International conference on machine learning*. PMLR, 2018, pp. 4344–4353.

[18] B. Manela and A. Biess, "Curriculum learning with hindsight experience replay for sequential object manipulation tasks," *Neural Networks*, vol. 145, pp. 260–270, 2022.

[19] Y. Luo, Y. Wang, K. Dong, Y. Liu, Z. Sun, Q. Zhang, and B. Song, "D2sr: Transferring dense reward function to sparse by network resetting," in *2023 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2023, pp. 906–911.

[20] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[21] D. H. Schunk and M. K. DiBenedetto, "Motivation and social cognitive theory," *Contemporary educational psychology*, vol. 60, p. 101832, 2020.

[22] H. Yu, W. Xu, and H. Zhang, "Towards safe reinforcement learning with a safety editor policy," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2608–2621, 2022.

[23] Y. Zhang, Q. Vuong, and K. Ross, "First order constrained optimization in policy space," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15 338–15 349, 2020.

[24] Z. Qin, Y. Chen, and C. Fan, "Density constrained reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8682–8692.

[25] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[26] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Proceedings of the Conference on Robot Learning*. PMLR, 2020, pp. 1094–1100.

[27] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[28] Y. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu, "Distral: Robust multitask reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[29] H. Huang, D. Ye, L. Shen, and W. Liu, "Curriculum-based asymmetric multi-task reinforcement learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[30] S. Liu, S. James, A. J. Davison, and E. Johns, "Auto-lambda: Disentangling dynamic task relationships," *arXiv preprint arXiv:2202.03091*, 2022.

[31] C. Fifty, E. Amid, Z. Zhao, T. Yu, R. Anil, and C. Finn, "Efficiently identifying task groupings for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 503–27 516, 2021.

[32] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, S. Levine, and C. Finn, "Conservative data sharing for multi-task offline reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 501–11 516, 2021.

[33] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, C. Finn, and S. Levine, "How to leverage unlabeled data in offline reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 25 611–25 635.

[34] L. Sun, H. Zhang, W. Xu, and M. Tomizuka, "Paco: Parameter-compositional multi-task reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 21 495–21 507, 2022.

[35] R. Yang, H. Xu, Y. Wu, and X. Wang, "Multi-task reinforcement learning with soft modularization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4767–4777, 2020.

[36] J. He, K. Li, Y. Zang, H. Fu, Q. Fu, J. Xing, and J. Cheng, "Not all tasks are equally difficult: Multi-task deep reinforcement learning with dynamic depth routing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 12 376–12 384.

[37] X. Sun, R. Panda, R. Feris, and K. Saenko, "Adashare: Learning what to share for efficient deep multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8728–8740, 2020.

[38] P. Guo, C.-Y. Lee, and D. Ulbricht, "Learning to branch for multi-task learning," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, pp. 3854–3863.

[39] T. Bram, G. Brunner, O. Richter, and R. Wattenhofer, "Attentive multi-task deep reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part III*. Springer, 2020, pp. 134–149.

[40] G. Cheng, L. Dong, W. Cai, and C. Sun, "Multi-task reinforcement learning with attention-based mixture of experts," *IEEE Robotics and Automation Letters*, 2023.

[41] A. Hendawy, J. Peters, and C. D'Eramo, "Multi-task reinforcement learning with mixture of orthogonal experts," *arXiv preprint arXiv:2311.11385*, 2023.

[42] S. Sodhani, A. Zhang, and J. Pineau, "Multi-task reinforcement learning with context-based representations," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9767–9779.

[43] B. Liu, X. Liu, X. Jin, P. Stone, and Q. Liu, "Conflict-averse gradient descent for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 878–18 890, 2021.

[44] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Mt-opt: Continuous multi-task robotic reinforcement learning at scale," *arXiv preprint arXiv:2104.08212*, 2021.

[45] ——, "Scaling up multi-task robotic reinforcement learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 557–575.

[46] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 18 081–18 090.

[47] C. Jin, X. Feng, and H. Yu, "A brain-inspired incremental multitask reinforcement learning approach," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 3, pp. 1147–1160, 2024.

[48] N. Zhang, A. Gupta, Z. Chen, and Y.-S. Ong, "Multitask neuroevolution for reinforcement learning with long and short episodes," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 3, pp. 1474–1486, 2023.

[49] F. Zhang, Y. Chen, H. Qiao, and Z. Liu, "Surrl: Structural unsupervised representations for robot learning," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, no. 2, pp. 819–831, 2023.

[50] M. B. Hafez and S. Wermter, "Continual robot learning using self-supervised task inference," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 16, no. 3, pp. 947–960, 2024.

[51] A. Hallak, D. Di Castro, and S. Mannor, "Contextual markov decision processes," *arXiv preprint arXiv:1502.02259*, 2015.

[52] G. Tesauro *et al.*, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[53] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043*, 2019.

[54] Z. Tian, R. Chen, X. Hu, L. Li, R. Zhang, F. Wu, S. Peng, J. Guo, Z. Du, Q. Guo, and Y. Chen, "Decompose a task into generalizable subtasks in multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 78 514–78 532.

[55] S. Hu, F. Zhu, X. Chang, and X. Liang, "Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers," *ArXiv preprint*, vol. abs/2101.08001, 2021. [Online]. Available: https://arxiv.org/abs/2101.08001

[56] W. Wang, T. Yang, Y. Liu, J. Hao, X. Hao, Y. Hu, Y. Chen, C. Fan, and Y. Gao, "Action semantics network: Considering the effects of actions in multiagent systems," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. [Online]. Available: https://openreview.net/forum?id=ryg48p4tPH

Jinyuan Feng received the B.Eng. degree in computer science and technology from the China University of Geosciences, Beijing, China, in 2022. He is currently pursuing the Ph.D. degree in control theory and control engineering with the Institute of Automation, Chinese Academy of Sciences, Beijing. His current research interests include multi-agent deep reinforcement learning, multi-task learning, and generalization in reinforcement learning.

Min Chen received the B.Sc. degree in physics from University of Chinese Academy of Sciences, Beijing, China, in 2020, and the M.Eng. degree in control theory and control engineering from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2023. He is currently an Assistant Engineer in the Institute of Automation, Chinese Academy of Sciences.

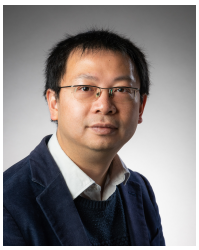
Zhiqiang Pu received the B.Eng. degree in automation from Wuhan University, Wuhan, China, in 2009, and the Ph.D. degree in control theory and control engineering from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014. He is currently a Full Professor in the Institute of Automation, Chinese Academy of Sciences. His research interests include decision intelligence, collective intelligence, the intersection of sports science and artificial intelligence, and also applications of collective intelligence in unmanned systems.

Tenghai Qiu received the B.Eng. degree in automation from Xiamen University, Xiamen, China, in 2013, and the M.Eng. degree in control theory and control engineering from Beihang University, Beijing, China, in 2016. He is currently pursuing the Ph.D. degree in electronic and information engineering with the School of Automation, Central South University, Changsha, China.

He is also a research assistant with the Integrated Information System Research Center, Institute of Automation, Chinese Academy of Sciences. He has been supported by the Innovation Guidance Found, Chinese Academy of Sciences, since 2020. His current research interests mainly include intelligence decision making, multi agent reinforcement learning, swarm intelligence, and the applications of unmanned autonomous systems.

Jianqiang Yi received the B.Eng. degree in mechanical engineering from Beijing Institute of Technology, Beijing, China, in 1985, and the M.Eng. and Ph.D. degrees in automatic control from the Kyushu Institute of Technology, Kitakyushu, Japan, in 1989 and 1992, respectively.

From 1992 to 1994, he worked as a research fellow at the Computer Software Development Company, Tokyo, Japan. From 1994 to 2001, He was a chief engineer at MYCOM, Inc., Kyoto, Japan. Since 2001, he has been a Full Professor in Institute of Automation, Chinese Academy of Sciences, Beijing, China. He has authored or co-authored over 100 international journal papers and 240 international conference papers. He holds more than 40 issued domestic patents. His research interests include theories and engineering applications of intelligent control, adaptive control, aerospace systems, and intelligent robotics. Prof. Yi is an Associate Editor for the Journal of Advanced Computational Intelligence and Intelligent Informatics, and Journal of Innovative Computing, Information and Control.



Jie Zhang earned his Ph.D. from City University of Hong Kong in 2011. He completed postdoctoral research at both Aarhus University and Oxford University. Currently, he serves as an associate professor in Computer Science at the University of Bath, U.K. His research focuses on the intersection of computer science and economics, particularly in algorithmic game theory, mechanism design, and multi-agent systems.