



Universiteit
Leiden
The Netherlands

Meta generative flow networks with personalization for task-specific adaptation

Ji, X.; Zhang, X.; Xi, W.; Wang, H.; Gadyatskaya, O.; Li, Y.

Citation

Ji, X., Zhang, X., Xi, W., Wang, H., Gadyatskaya, O., & Li, Y. (2024). Meta generative flow networks with personalization for task-specific adaptation. *Information Sciences*, 672. doi:10.1016/j.ins.2024.120569

Version: Publisher's Version

License: [Licensed under Article 25fa Copyright Act/Law \(Amendment Taverne\)](#)

Downloaded from: <https://hdl.handle.net/1887/4303808>

Note: To cite this publication please use the final published version (if applicable).



Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Meta Generative Flow Networks with personalization for task-specific adaptation [☆]

Xinyuan Ji ^{a,b,1}, Xu Zhang ^{c,1}, Wei Xi ^a, Haozhi Wang ^d, Olga Gadyatskaya ^b, Yinchuan Li ^{e,*}

^a Faculty of Electronic and Information Engineering, Xi'an Jiaotong University, China

^b Leiden Institute of Advanced Computer Science (LIACS), Leiden University, the Netherlands

^c School of Artificial Intelligence, Xidian University, China

^d School of Electrical and Information Engineering, Tianjin University, China

^e Noah's Ark Lab, Huawei Technologies, China

ARTICLE INFO

Keywords:

Generative Flow Networks
Personalization for task-specific
Multi-task reinforcement learning
Meta reinforcement learning

ABSTRACT

Multi-task reinforcement learning and meta-reinforcement learning have been developed to enhance the learning capabilities on new tasks, but they tend to focus on maximizing rewards, leading to poor performance in exploration. To address this issue, we propose Meta Generative Flow Networks (GFlowMeta): an integration of Generative Flow Networks (GFlowNets) into meta-learning algorithms. By leveraging the unique capabilities of GFlowNets to generate diverse candidate solutions, GFlowMeta exhibits enhanced exploration of tasks. However, GFlowMeta's performance deteriorates when confronted with heterogeneous transitions from distinct tasks, leading to a decrease in its effectiveness. Therefore, we subsequently introduce a personalization approach called personalized Meta Generative Flow Networks (pGFlowMeta), which combines task-specific personalized policies with a meta policy. Each personalized policy balances the loss on its personalized task and the difference from the meta policy, while the meta policy aims to minimize the average loss of all tasks. The theoretical analysis shows that the proposed pGFlowMeta converges at a sublinear rate. Furthermore, extensive experiments demonstrate the superiority of pGFlowMeta over state-of-the-art reinforcement learning algorithms.

1. Introduction

Reinforcement learning (RL) techniques strive to discover optimal policies through engagement with the environment to maximize expected rewards [38]. Although reinforcement learning has been successfully applied in various fields [34,4,19], it often has difficulty adapting to new tasks, resulting in suboptimal performance on unseen tasks. This is because the optimal policy for a new task may be significantly different from the optimal policy for a previously learned task. Furthermore, traditional reinforcement

[☆] This work is supported by the Postdoctoral Fellowship Program of CPSF under Grant Number GZC20232038.

* Corresponding author.

E-mail addresses: jixinyuan1996@stu.xjtu.edu.cn (X. Ji), zhang.xu@xidian.edu.cn (X. Zhang), xiwei@xjtu.edu.cn (W. Xi), wanghaozhi@tju.edu.cn (H. Wang), o.gadyatskaya@liacs.leidenuniv.nl (O. Gadyatskaya), liyinchuan@huawei.com (Y. Li).

¹ Equal contribution.

<https://doi.org/10.1016/j.ins.2024.120569>

Received 28 June 2023; Received in revised form 1 April 2024; Accepted 5 April 2024

Available online 10 April 2024

0020-0255/© 2024 Published by Elsevier Inc.

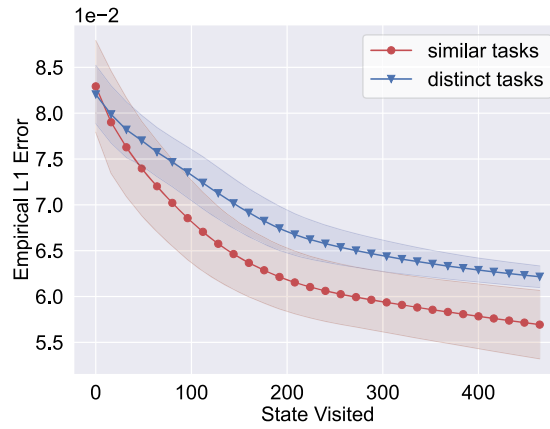


Fig. 1. Averaged Empirical L_1 Error of GFlowMeta on similar and distinct tasks in the 'Frozen lake' environment. This highlights the performance gap of GFlowMeta on distinct and similar tasks.

learning methods that maximize expected rewards also face challenges in exploring diverse high-reward candidate sets. Therefore, these methods perform poorly in new tasks when given sparse rewards and the goal of discovering interesting candidates.

To quickly adapt to new tasks with minimal additional training, researchers have introduced multi-task reinforcement learning (multi-task RL) and meta-reinforcement learning (meta-RL) techniques. Preliminary studies on multi-task reinforcement learning mainly focused on transfer learning. It involves transferring knowledge from relevant training tasks to test tasks in order to improve the performance of the policy employed in the target task. Subsequent studies have introduced alternative approaches, including the utilization of common representation [31,8,40,42], gradients' similarity [43,7], and policy distillation [41,26,37]. Additionally, there are also Meta-RL methods proposed, such as Model-Agnostic Meta-Learning (MAML) [11], E-MAML [36], and Probabilistic Embeddings for Actor-critic meta-RL (PEARL) [30] aiming to enable the policy model to leverage the knowledge learned from prior training tasks to facilitate rapid adaptation to new tasks. However, these methods usually prioritize maximizing the expected reward while ignoring the diversity of the candidate set. As a result, policy models may struggle to explore a wider range of high-reward candidates, leading to suboptimal performance.

To facilitate rapid adaptation to new tasks and enhance exploration performance on tasks, a solution that integrates Generative Flow Networks (GFlowNets) [5,17,6] into meta-learning is proposed, leading to a variant called Meta Generative Flow Networks (GFlowMeta). GFlowMeta combines the advantages of GFlowNet and meta-reinforcement learning. It exploits the power of GFlowNet to generate diverse candidate solutions through approximate sampling techniques guided by a given reward function. Additionally, GFlowMeta incorporates meta-learning techniques that enable it to efficiently adapt to new tasks by leveraging knowledge learned from previous tasks. However, it should be noted that GFlowMeta achieves significant performance improvements mainly when the trained tasks exhibit similarities to each other. Unfortunately, in cases where tasks are distinct, i.e., there are significantly different characteristics between tasks, the performance improvement achieved by GFlowMeta is relatively limited. This can be observed in Fig. 3, which depicts the distribution of sampling transitions between tasks, highlighting clear differences. To further emphasize this distinction, a performance comparison is provided in Fig. 1 showing the impact of training GFlowMeta in the 'Frozen Lake' environment on similar and distinct tasks. The results clearly show significant differences in performance between these two scenarios, increasing the challenge of adapting GFlowMeta to distinct tasks.

In this paper, a personalized Meta Generative Flow Networks (pGFlowMeta) is proposed, as depicted in Fig. 2, to address the limited performance improvements exhibited by GFlowMeta when dealing with distinct tasks in the discrete environments. The framework utilizes a combination of a *meta policy*, which is trained on all tasks, and *personalized policies*, which are trained specifically for each individual task. The meta policy aims to minimize the cumulative loss across all tasks by aggregating the auxiliary meta parameters on each task. On the other hand, each personalized policy aims to strike a balance between minimizing the loss on its respective task and minimizing the discrepancy from the meta policy. To accomplish this framework, an alternating minimization algorithm specifically designed for the pGFlowMeta framework is introduced to enable effective personalization on distinct tasks. The algorithm formulates a bi-level optimization problem and iteratively updates the personalized policies through an approximate solution of their corresponding lower-level problems, thereby penalizing the discrepancy between the meta policy and the personalized policies. Simultaneously, auxiliary meta parameters used for aggregating the meta policy are updated to ensure their alignment with the desired objectives. A convergence analysis is provided for the proposed algorithm, which proves that the algorithm converges sublinearly as the outer loop increases. Besides, a convergence analysis of the personalized policies to the meta policy is provided: the personalized policies on average converge to a neighborhood of the meta policy with a sublinear rate.

The paper makes the following primary contributions:

- GFlowMeta is proposed to facilitate rapid adaptation to new tasks and enhance exploration performance on tasks. Compared with state-of-art methods, GFlowMeta utilizes the capabilities of GFlowNets to generate diverse candidate solutions through approximate sampling techniques guided by a given reward function.

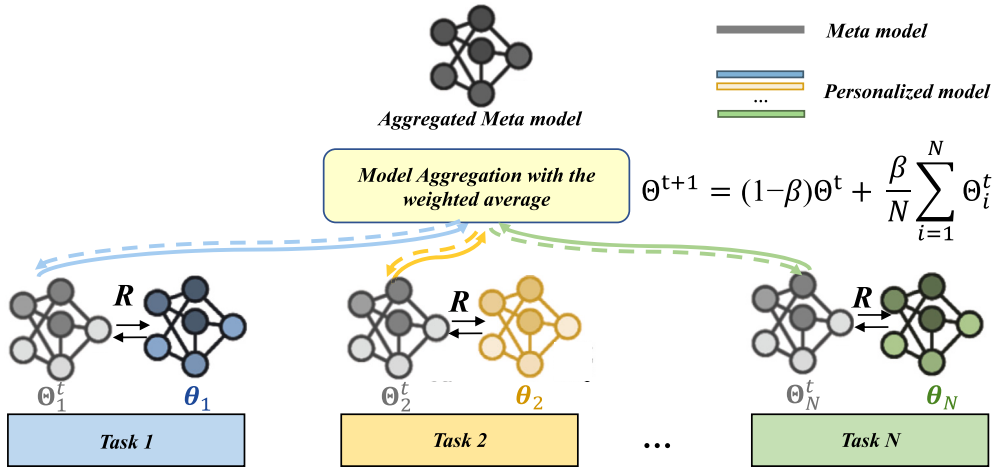


Fig. 2. System diagram for personalized Meta Generative Flow Networks (pGFlowMeta). Consider a meta-learning problem in which there are N tasks. In each task, the agent learns two policy models: the personalized policy θ_i and the meta policy Θ . The training process involves alternating updates between the personalized policy and meta policy models for a specified number of iterations (R). Following the updates, the meta policy models are aggregated using the formula $\Theta^{t+1} = (1-\beta)\Theta^t + \frac{\beta}{N} \sum_{i=1}^N \Theta_i^t$, where β is a weighting parameter and N represents the number of tasks. This framework enables the agent to learn both personalized policies for individual tasks and a meta-level policy that captures common knowledge across tasks.

- pGFlowMeta is proposed to address the challenge of GFlowMeta posed by distinct tasks. pGFlowMeta simultaneously optimizes the meta policy (MP) and the personalized policies (PP) to facilitate rapid learning on new tasks, and to improve the performance on each distinct task.
- An alternating minimization algorithm is introduced to accomplish the pGFlowMeta framework. Specifically, a bi-level optimization problem is formulated to optimize two policies simultaneously, and a proximal operator into the lower-level objective function to discourage divergence between MP and PP.

The remainder of the paper is organized as follows. In Section 2 presents a literature review. In Section 3 the preliminaries are introduced. Section 4 presents the theoretical frameworks of GFlowMeta and pGFlowMeta. Section 5 delves into the theoretical analysis of pGFlowMeta. The details of the experimental design and the analysis of the experimental results are presented in Section 6. Following that, Section 7 discusses insights into real-world applications and the limitations of the work. Finally, Section 8 concludes the paper and outlines potential future research directions.

2. Related work

This section provides a comprehensive review of research closely related to pGFlowMeta, including meta-reinforcement learning, multi-task learning, and GFlowNets.

Meta-reinforcement learning. Meta-reinforcement learning (meta-RL) focuses on learning a policy that can make an agent efficiently learn on test tasks [18,9]. Specifically, meta-RL can develop policies by leveraging the knowledge and experience acquired from previous tasks [11,30]. The meta-reinforcement learning literature covers a range of methods, which can be divided into two main categories: model-free methods and model-based methods. Model-free methods such as, for example, MAML [11], ProMP [32], E-MAML [36], MAESN [13], and ES-MAML [35] directly learn a policy in training tasks without explicitly modeling the environment of the tasks. In contrast, model-based methods take a different approach by learning a model to construct the task's environment information. This learned model is then utilized to simulate various scenarios and generate training transition data for the meta policy. For instance, PEARL [30], and CaDM [20] learn a task inference network to extract task-specific information from the environment, which is represented as task latent variable and a meta policy network with task latent variables capturing the shared patterns and dynamics of state transitions. They plan agent policies by inferring task information and adapting the state transition model accordingly. The aforementioned methods achieve fast adaptation to new tasks by training on multiple similar tasks. However, they often encounter model divergence when applied to tasks with low similarity due to environmental heterogeneity.

Multi-task learning & multi-task reinforcement learning. Multi-task learning (MTL) aims to improve a model's ability to adapt to new tasks by simultaneously learning multiple related tasks [24,27]. In the realm of reinforcement learning, employing MTL is an avenue to boost the effectiveness of RL agents. By simultaneously training the agent to perform multiple related tasks, it can potentially learn to generalize better and more efficiently than a single-task agent. Several approaches and techniques based on multi-task RL have been proposed to further improve the effectiveness of RL. Much of the research in this area, such as that by [29] has centered around transfer learning, in which knowledge learned from the related tasks is re-used on new tasks. Subsequent studies have introduced alternative approaches, including the utilization of common representation [31,40,42,28], gradients similarity [43,7], and policy distillation [41,26,37]. For example, Policy Distillation [37,41] and Actor-Mimic [26] leverage the technology of distillation

to achieve multi-task RL. The central concept here is that the features learned by a complex model can be compressed into a smaller and more efficient model.

Generative Flow Networks. Generative Flow Networks (GFlowNets) were initially introduced by Bengio et al. [5] as a generative modeling framework for sampling from a distribution. Since then, GFlowNets have been studied extensively [25,21,16,23,17]. Madan et al. [23] propose SubTB(λ), a training objective for GFlowNets, which leverages the bias-variance trade-off to improve convergence and enable training in environments where action sequences are lengthier and rewards are sparsely distributed. Jain et al. [16] introduced an active learning algorithm that combines epistemic uncertainty estimation with GFlowNets. This integration enables the generation of diverse and valuable candidate solutions specifically for the design of biological sequences. Zhang et al. [44] introduced a framework that establishes connections between deep generative models and GFlowNets. They provide a unifying perspective by examining the learning process with Markovian trajectories. Li et al. [21] addressed the challenge of handling continuous control tasks in generative modeling by proposing generative continuous flow networks (CFlowNets). Their work introduced a theoretical formulation and a training framework, and demonstrated the superior exploration ability of CFlowNets compared to reinforcement learning methods through experimental results.

3. Preliminaries

This section introduces some important foundational concepts in reinforcement learning and GFlowNets.

3.1. Reinforcement learning

As one of the artificial intelligence approaches [2,3], reinforcement learning (RL) [18] algorithms are specifically developed to tackle problems that involve making sequential decisions in a complex and uncertain environment. In RL, an agent continuously interacts with an environment. The *Agent* takes *Actions* based on *Observations* obtained from perceiving the *Environment*. These *Actions* are then fed back into the *Environment*, which responds by providing the next *Observations* and a *Reward* based on the agent's chosen *Actions*. The goal of the RL is to learn the optimal *Policy* to maximize the *Rewards* obtained from the environment as much as possible.

RL methods exhibit the Markovian property, which means that taking action a_t in the current state s_t and transitioning to the next state s_{t+1} does not depend on previous states $s_1, \dots, s_{t-1}$. Such processes can be mathematically formulated as a tuple $\mathcal{T} = \langle S, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. Here, S is a set of states, $\mathcal{A}$ is a set of actions in the decision-making process, $r_t = \mathcal{R}(s_t, a_t, s_{t+1})$ is the reward which depicts the immediate reward of performing a certain action a_t in the state s_t to the next state s_{t+1} in the t -th time step. The function $\mathcal{P}(s_{t+1} | s_t, a_t)$ represents the state transition probability, indicating the probability of transitioning from the current state s_t to the next state s_{t+1} . Additionally, the discount factor $\gamma \in [0, 1)$ is used to emphasize delayed rewards. Based on these definitions, the agent can aim to learn the optimal policy $\pi^* = \arg \max_{\pi} \sum_{\tau} P(\tau | \pi) R(\tau)$ by maximizing the expected rewards, where τ represents a trajectory, $\tau = (s_1, a_1, s_2, a_2, s_3, a_3, \dots, s_T, a_T)$, and T represents the number of time steps in the trajectory.

By continuous trial-and-error process, RL can acquire the optimal policy without a large amount of direct instruction information. The agent must exploit the existing knowledge to gain rewards while also exploring to allow for better future action selection.

3.2. Generative Flow Network

GFlowNets are primarily designed for environments where direct instruction information is limited, resulting in sparse rewards and requiring increased exploration. Let $X \in S$ represent a set of terminal states, referring to all the states where multiple episodes terminate in an environment. The primary objective of GFlowNets is to transform a given return function into a generative policy $\pi(a|s)$ to satisfy $\pi(x) \propto R(x)$, where $R(x) > 0$ is a reward for a terminal state x . GFlowNets sees the Markov Decision Process (MDP) as a flow network, that is, they leverage the Directed Acyclic Graph (DAG) structure of the MDP. Let $T(s, a)$ be the next state of state-action pair (s, a) . Let $\mathcal{F}(s') = \{s | \exists a \in \mathcal{A}, T(s, a) = s' \in S\}$ to be a set of all states flowing to state s' , $\mathcal{D}(s') = \{s | \exists a \in \mathcal{A}, T(s', a) = s \in S\}$ as the set of all states that diverge from state s' . For any state s' , it can have multiple parent states, i.e., $|\mathcal{F}(s')| \geq 1$, except for the root node represented as s_0 , which has no parent states. Similarly, for any state s' , it can have multiple actions, i.e., $|\mathcal{D}(s')| \geq 1$, except for the sink node denoted as x , which has no actions. Let us define the action flow $F(s \rightarrow s')$ as the flow passing through an edge from state s to state s' . The flow consistency equations are employed, where $R(s') = 0$ for interior nodes and $T(s', a) = \emptyset$ for leaf (sink/terminal) nodes:

$$\sum_{s \in \mathcal{F}(s')} F(s \rightarrow s') = R(s') + \sum_{s \in \mathcal{D}(s')} F(s' \rightarrow s), \quad (1)$$

This implies that for any node s' in the network, the incoming flow of information is equal to the outgoing flow of information, which is denoted as the total flow $F(s')$. With F being a flow, $F(s \rightarrow s') > 0, \forall s$. The unknown function F with parameters Θ is parameterized and denoted as F_{Θ} . This yields the following objective for a trajectory τ :

$$\mathcal{L}(\Theta) = \sum_{s' \in \tau \neq s_0} \left(\sum_{s \in \mathcal{F}(s')} F_{\Theta}(s \rightarrow s') - R(s') - \sum_{s \in \mathcal{D}(s')} F_{\Theta}(s' \rightarrow s) \right)^2. \quad (2)$$

Algorithm 1 GFlowMeta: A Meta Generative Flow Network.

```

1: Input: training tasks  $\{\mathcal{T}_i\}_{i=1}^N$  sampled from  $p(\mathcal{T})$ ,  $\Theta^0$ ,  $\eta$ 
2: for  $t = 1$  to  $T$  do
3:   for  $i = 1$  to  $N$  parallel do
4:      $\Theta_{i,1}^t = \Theta^t$ 
5:     for  $r = 1, \dots, R$  do
6:       Sample trajectories  $\{\tau_i^k\}_{k=1}^K$  from Task  $\mathcal{T}_i$ 
7:       Evaluate  $\nabla \mathcal{L}_i(\Theta_{i,r}^t)$  using  $\{\tau_i^k\}_{k=1}^K$ 
8:       Compute updated parameters by solving Eq. (6) with gradient descent:  $\Theta_{i,r+1}^t = \Theta_{i,r}^t - \eta \nabla \mathcal{L}_i(\Theta_{i,r}^t)$ 
9:     end for
10:   end for
11:   Update meta parameters  $\Theta^{t+1} = \frac{1}{N} \sum_{i=1}^N \Theta_{i,R}^t$ 
12: end for
13: Output: meta parameters  $\Theta^T$ 

```

4. Personalized Meta Generative Flow Networks

This section introduces a personalized Meta Generative Flow Network model (pGFlowMeta) for distinct tasks in reinforcement learning. To make this easier to follow, a Meta Generative Flow Network model (GFlowMeta) is first presented for similar tasks. The algorithms for GFlowMeta and pGFlowMeta are shown in Algorithm 1 and Algorithm 2, respectively.

4.1. Meta Generative Flow Networks

The Meta Generative Flow Network (GFlowMeta) is a framework that focuses on learning a meta stochastic policy capable of adapting to new tasks with limited samples in reinforcement learning. The objective is to ensure that the generated objects from the policy have probabilities that are proportional to the corresponding given rewards associated with the new tasks. Let the tasks $\{\mathcal{T}_i\}$ follow the distribution $p(\mathcal{T})$, i.e., $\mathcal{T}_i \sim p(\mathcal{T})$. Assume that N tasks are sampled from the task distribution $p(\mathcal{T})$. A parameterized flow function F_Θ can be learned by the following objective function:

$$\min_{\Theta \in \mathbb{R}^d} \left\{ \mathcal{L}(\Theta) := \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(\Theta) \right\}, \quad (3)$$

$$\text{where } \mathcal{L}_i(\Theta) = \mathbb{E}_{\tau \sim p_i(\tau)} \left[\sum_{s' \in \tau \neq s_0} \left(\sum_{s \in \mathcal{F}(s')} F_\Theta(s \rightarrow s') - R(s') - \sum_{s \in \mathcal{D}(s')} F_\Theta(s' \rightarrow s) \right)^2 \right]. \quad (4)$$

Here, $\Theta \in \mathbb{R}^d$ denotes the parameter of the flow function F_Θ to be learned, and $p_i(\tau)$ denotes the trajectory distribution of the task $\mathcal{T}_i$.

Assuming K trajectories are sampled from each trajectory distribution $p_i(\tau)$, then the objective function becomes

$$\min_{\Theta \in \mathbb{R}^d} \left\{ \mathcal{L}(\Theta) := \frac{1}{N} \sum_{i=1}^N \tilde{\mathcal{L}}_i(\Theta) \right\}, \quad (5)$$

$$\text{where } \tilde{\mathcal{L}}_i(\Theta) = \frac{1}{K} \sum_{k=1}^K \left[\sum_{s'_i \in \tau_i^k \neq s_{i,0}} \left(\sum_{s_i \in \mathcal{F}(s'_i)} F_\Theta(s_i \rightarrow s'_i) - R(s'_i) - \sum_{s_i \in \mathcal{D}(s'_i)} F_\Theta(s'_i \rightarrow s_i) \right)^2 \right]. \quad (6)$$

Here, τ_i^j denotes the j -th trajectory of the task $\mathcal{T}_i$.

For non-convex objectives (3) and (4), GFlowMeta has the potential to learn a shared prior over model parameters, resulting in a global model by averaging models in the parameter space based on the tasks. The complete pseudo-code for GFlowMeta can be found in Algorithm 1. The training process of GFlowMeta primarily consists of two loops: an outer loop for the aggregation of the meta policy model and an inner loop for the updates of the meta policy model on each task. In the t -th round of the outer loop, every task $\mathcal{T}_i$ samples trajectories $\{\tau_i^k\}_{k=1}^K$ and the meta policy model is updated in the inner loop based on $\Theta_{i,r}^t$ to optimize objective (6). The updated meta policy model after the r -th update in the t -th round is denoted as $\Theta_{i,r+1}^t$. Once all tasks have completed the inner loop training procedure, the new meta policy model Θ^{t+1} is obtained by aggregating the inner updates of the tasks through averaging.

While GFlowMeta can be effective for similar tasks, it may encounter significant performance degradation when applied to tasks that are not closely related. For instance, if the tasks involve disparate datasets or exhibit different input features, GFlowMeta may struggle to capture the underlying data structure and yield subpar performance. Similarly, if the tasks involve distinct control dynamics, GFlowMeta may struggle to generalize to new tasks with different dynamics. As a result, GFlowMeta might not be suitable for different tasks. It is crucial to carefully consider the task characteristics and assess the compatibility between GFlowMeta and the specific task requirements before applying this approach.

Algorithm 2 pGFlowMeta: a personalized Meta Generative Flow Network.

```

1: Input: training tasks  $\{\mathcal{T}_i\}_{i=1}^N$  sampled from  $p(\mathcal{T})$ ,  $\Theta^0$ ,  $\eta$ ,  $\beta$ 
2: for  $t = 1$  to  $T$  do
3:   for  $i = 1$  to  $N$  parallel do
4:      $\Theta_{i,1}^t = \Theta^t$ 
5:     for  $r = 1$  to  $R$  do
6:       Sample trajectories  $\{\tau_i^k\}_{k=1}^K$  from Task  $\mathcal{T}_i$ 
7:       Evaluate  $\nabla \mathcal{L}_i(\theta_i)$  using  $\{\tau_i^k\}_{k=1}^K$ 
8:       Update personalized parameters by solving Eq. (12) inexactly to obtain a  $\zeta$ -approximate  $\hat{\theta}_i(\Theta_{i,r}^t)$ 
9:       Update auxiliary meta parameters
          $\Theta_{i,r+1}^t = \Theta_{i,r}^t - \eta \lambda (\Theta_{i,r}^t - \hat{\theta}_i(\Theta_{i,r}^t))$ 
10:    end for
11:  end for
12:  Update meta parameters  $\Theta^{t+1} = (1 - \beta)\Theta^t + \frac{\beta}{N} \sum_{i=1}^N \Theta_{i,R}^t$ 
13: end for
14: Output: meta parameters  $\Theta^T$  and personalized parameters  $\{\theta_{i,R}^T\}_{i=1}^N$ 

```

4.2. Personalized Meta Generative Flow Networks

To deal with distinct tasks, a personalized meta Generative Flow Network (pGFlowMeta) is proposed, where, similarly with [15], it embodies the ‘divide and conquer’ strategy, but it learns the same network on multiple different yet related tasks and has an innovative design of the personalized policy. In the pGFlowMeta, a meta policy is learned that can adapt to multiple distinct tasks, while also learning a personalized policy specific to each individual task. Assume that N distinct tasks are sampled from the task distribution $p(\mathcal{T})$. The goal of pGFlowMeta is to optimize the following objective function

$$\min_{\Theta \in \mathbb{R}^d} \left\{ \mathcal{L}(\Theta) := \frac{1}{N} \sum_{i=1}^N \mathcal{L}_i(\Theta) \right\}, \text{ where } \mathcal{L}_i(\Theta) = \min_{\theta_i} \mathcal{L}_i(\theta_i) + \lambda f(\Theta, \theta_i) \quad (7)$$

with

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{\tau_i \sim p_i(\tau)} \left[\sum_{s'_i \in \tau_i \neq s_0} \left(\sum_{s_i \in F(s'_i)} F_{\theta_i}(s_i \rightarrow s'_i) - R(s'_i) - \sum_{s_i \in D(s'_i)} F_{\theta_i}(s'_i \rightarrow s_i) \right)^2 \right]. \quad (8)$$

Here, $f(\Theta, \theta_i) = \frac{1}{2} \|\Theta - \theta_i\|^2$ is a regularization term that penalizes the difference between Θ and θ_i , $\Theta \in \mathbb{R}^d$ is the parameter of the meta flow function F_Θ and θ_i is the parameter of the personalized flow function F_{θ_i} . λ is the hyperparameter that balances the loss on its personalized policies θ_i and the difference from the meta policy Θ . Let θ_i^* be the solution of (7), i.e.,

$$\theta_i^*(\Theta) = \arg \min_{\theta_i \in \mathbb{R}^d} \mathcal{L}_i(\theta_i) + \lambda f(\Theta, \theta_i). \quad (9)$$

Assuming K trajectories are sampled from each trajectory distribution $p_i(\tau)$, then the objective function in (7) becomes

$$\tilde{\mathcal{L}}_i(\Theta) = \min_{\theta_i} \tilde{\mathcal{L}}_i(\theta_i) + \lambda f(\Theta, \theta_i) \quad (10)$$

with

$$\tilde{\mathcal{L}}_i(\theta_i) = \frac{1}{K} \sum_{k=1}^K \left[\sum_{s'_i \in \tau_i^k \neq s_0} \left(\sum_{s_i \in F(s'_i)} F_{\theta_i}(s_i \rightarrow s'_i) - R(s'_i) - \sum_{s_i \in D(s'_i)} F_{\theta_i}(s'_i \rightarrow s_i) \right)^2 \right]. \quad (11)$$

Define $\rho_{\Theta}^i(\theta_i) = \mathcal{L}_i(\theta_i) + \lambda f(\Theta, \theta_i)$ and $\tilde{\rho}_{\Theta}^i(\theta_i) = \tilde{\mathcal{L}}_i(\theta_i) + \lambda f(\Theta, \theta_i)$. Let $\hat{\theta}_i(\Theta)$ be the inexact solution of Eq. (10)

$$\hat{\theta}_i(\Theta) \approx \arg \min_{\theta_i \in \mathbb{R}^d} \tilde{\rho}_{\Theta}^i(\theta_i), \quad (12)$$

such that $\hat{\theta}_i(\Theta)$ is the ζ -approximation of $\theta_i^*(\Theta)$, i.e., $\mathbb{E}[\|\hat{\theta}_i(\Theta) - \theta_i^*(\Theta)\|^2] \leq \zeta^2$. The gradient of $\tilde{\mathcal{L}}_i(\Theta)$ with respect to Θ used for the updates of meta policy model is

$$\nabla \tilde{\mathcal{L}}_i(\Theta) = \lambda(\Theta - \hat{\theta}_i(\Theta)). \quad (13)$$

As illustrated in Algorithm 2, the training process of pGFlowMeta consists of two main steps. In the outer loop, the meta policy is aggregated, while in the inner loop both the auxiliary meta policy model and the personalized policy are updated for each task. In the r -th inner update of the t -th round, every task $\mathcal{T}_i$ samples trajectories $\{\tau_i^k\}_{k=1}^K$ and firstly updates the personalized policy by solving (12) inexactly. Then the auxiliary meta policy model is updated by using gradient descent, i.e.,

$$\Theta_{i,r+1}^t = \Theta_{i,r}^t - \eta \nabla \mathcal{L}_i(\Theta_{i,r}^t) = \Theta_{i,r}^t - \eta \lambda (\Theta_{i,r}^t - \hat{\theta}_i(\Theta_{i,r}^t)). \quad (14)$$

After R rounds of inner updates, the new meta policy model Θ^{t+1} is aggregated by $\Theta^{t+1} = (1 - \beta)\Theta^t + \frac{\beta}{N} \sum_{i=1}^N \Theta_{i,R}^t$, where $\beta > 0$ is introduced to accelerate the convergence of the algorithm. These steps are executed repeatedly until reaching the T th round.

For ease of analysis, the updates with concise symbols are rewritten as follows

$$\Theta_{i,r+1}^t = \Theta_{i,r}^t - \eta \lambda (\Theta_{i,r}^t - \hat{\theta}_i(\Theta_{i,r}^t)) = \Theta_{i,r}^t - \eta \lambda g_{i,r}^t \quad (15)$$

$$\begin{aligned} \Theta^{t+1} &= (1 - \beta)\Theta^t + \frac{\beta}{N} \sum_{i=1}^N \Theta_{i,R}^t \\ &= \Theta^t - \eta \beta R \frac{1}{NR} \sum_{i=1}^N \sum_{r=1}^R (\Theta_{i,r}^t - \hat{\theta}_i(\Theta_{i,r}^t)) \\ &\triangleq \Theta^t - \tilde{\eta} g^t, \end{aligned} \quad (16)$$

where $g_{i,r}^t = \Theta_{i,r}^t - \hat{\theta}_i(\Theta_{i,r}^t)$, $\tilde{\eta} \triangleq \eta \beta R$ and $g^t \triangleq \frac{1}{NR} \sum_{i=1}^N \sum_{r=1}^R (\Theta_{i,r}^t - \hat{\theta}_i(\Theta_{i,r}^t)) \triangleq \frac{1}{NR} \sum_{i,r=1}^{N,R} g_{i,r}^t$. Here, $\tilde{\eta}$ represents the step size or the learning rate used in the optimization process of the outer loop. Similarly, g^t refers to the approximate gradient utilized during the outer loop.

5. Theoretical analysis

This section establishes the convergence guarantees for pGFlowMeta. The proofs are presented in Appendix A. To facilitate the following analysis, some reasonable assumptions are first made.

Assumption 1. The variance of stochastic gradients across various tasks during the meta-training process is bounded

$$\mathbb{E} \left[\left\| \nabla \ell_i(\theta_i) - \nabla \tilde{\ell}_i(\theta_i) \right\|^2 \right] \leq \kappa_1^2. \quad (17)$$

The variance of each local gradient to the average of all local gradients is bounded

$$\left\| \nabla \ell_i(\Theta) - \frac{1}{N} \sum_{j=1}^N \nabla \ell_j(\Theta) \right\|^2 \leq \kappa_2^2. \quad (18)$$

Remark 1. Since $\mathbb{E}[\tilde{\ell}_i(\theta_i)] = \ell_i(\theta_i)$, and the variance $\mathbb{E}[\|\nabla \ell_i(\theta_i) - \nabla \tilde{\ell}_i(\theta_i)\|^2]$ decreases with the increase of K , the assumption (17) can be easily satisfied for arbitrary κ_1 with a sufficiently large K . The assumption (18) is standard to bound the diversity of all tasks.

Assumption 2. The functions $\{F_{\theta_i}\}_{i=1}^N$ are B -bounded, L_0 -Lipschitz continuous and L_1 -Lipschitz gradients, i.e.,

$$\|F_{\theta_i}\| \leq B, \quad (19)$$

$$\|F_{\theta_i} - F_{\theta'_i}\| \leq L_0 \|\theta_i - \theta'_i\|, \quad (20)$$

$$\|\nabla F_{\theta_i} - \nabla F_{\theta'_i}\| \leq L_1 \|\theta_i - \theta'_i\|. \quad (21)$$

Remark 2. The assumption (19) is due to the flow consistency equations (1) and the boundness of the rewards. The assumptions (20) and (21) are considered to guarantee the function F_{θ_i} is Lipschitz-continuous and smooth, limiting the function to not fluctuate too much.

In the reinforcement learning environments with discrete actions, it is reasonable to assume that the sets of possible states and the rewards in the environment are finite or bounded.

Assumption 3. The states in the GFlowNets are finite. The maximum length of the trajectories sampled from $\{\mathcal{T}_i\}_{i=1}^N$ is H_1 , and the maximum number of parent nodes and child nodes for all states is H_2 , where $0 < H_1, H_2 < \infty$ are positive integers.

Assumption 4. The rewards are finite with a maximum H_0 .

Next, important properties for the lower-level and upper-level objective functions are presented, demonstrating their respective smoothness.

Lemma 1. Assuming the Assumptions 2, 3 and 4 are true, then $\ell_i(\cdot)$ is L_ℓ -smooth, where

$$L_\ell = 4H_1 \left[(H_0 + 2H_2B)H_2L_1 + 2H_2^2L_0^2 \right]. \quad (22)$$



Fig. 3. Two examples illustrating the generation of distinct tasks. (a) Distinct tasks in the 'Frozen Lake' environment. These tasks were characterized by variations in the location of holes. Different tasks may have holes placed at different positions, resulting in varying levels of difficulty. (b) Distinct tasks in the 'Cliff Walking' environment. These tasks were characterized by variations in the length of the cliff. Each task may have a different cliff length, leading to varying levels of risk and difficulty. On distinct tasks, the agent's transitions would differ across tasks despite using the same policy.

Lemma 2. If $\lambda > 2L_\epsilon$, then the functions $\mathcal{L}$ and $\mathcal{L}_i$ are L_ϵ -smooth with $L_\epsilon \triangleq \lambda$.

This following theorem shows that a ζ -approximation of $\theta_i^*(\Theta)$ can be obtained when $\hat{\theta}_i(\Theta)$ satisfies $\|\nabla \tilde{p}_\Theta^i(\hat{\theta}_i(\Theta))\| \leq \delta$, which can serve as a stopping criterion of solving (12) inexactly.

Theorem 1. Let $\theta_i^*(\Theta)$ be the solution of (9), and $\hat{\theta}_i(\Theta)$ satisfy $\|\nabla \tilde{p}_\Theta^i(\hat{\theta}_i(\Theta))\| \leq \delta$, if $\lambda > L_\epsilon$, then

$$\mathbb{E}[\|\hat{\theta}_i(\Theta) - \theta_i^*(\Theta)\|^2] \leq \zeta^2 \triangleq \frac{2(\kappa_1^2 + \delta^2)}{(\lambda - L_\epsilon)^2}. \quad (23)$$

Remark 3. Theorem 1 presents that the mean squared error decreases as κ_1 and δ decrease. This is reasonable since the variance of stochastic gradients across various tasks and the stopping tolerance for the inexact solution get reduced with the decrease of κ_1 and δ . Theorem 1 also demonstrates that the mean squared error decreases as λ increases. This is because $\rho_\Theta^i(\theta_i)$ and $\tilde{p}_\Theta^i(\theta_i)$ get more similar with the increase of λ .

Finally, the convergence guarantee of the pGFlowMeta algorithm is provided.

Theorem 2. Suppose that Assumptions 1-4 are true. Let the step size satisfy $\eta \leq \min\left\{\frac{\tilde{\eta}_0}{\beta R}, \frac{1}{2L_\epsilon \sqrt{(1+R)R}}\right\}$, where $\tilde{\eta}_0 \triangleq \frac{1}{70L_\epsilon \lambda^2}$ and $\lambda^2 - 8L_\epsilon^2 \geq 1$. Let $\tilde{t}$ be sampled uniformly from the index set $\{1, \dots, T\}$. Then the following inequalities hold

$$\mathbb{E}[\|\nabla \mathcal{L}(\Theta^{\tilde{t}})\|^2] \leq \mathcal{O}\left(\frac{\Delta}{\tilde{\eta}_0 T} + \sqrt[3]{\frac{\Delta^2 L_\epsilon^2 (\lambda^2 \zeta^2 + \kappa_\epsilon^2)}{\beta^2 T^2}} + \sqrt{\frac{\Delta L_\epsilon \kappa_\epsilon^2}{T} + \lambda^2 \zeta^2}\right), \quad (24)$$

and

$$\frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\hat{\theta}_i(\Theta^{\tilde{t}}) - \Theta^{\tilde{t}}\|^2] \leq \mathcal{O}\left(\zeta^2 + \frac{\kappa_\epsilon^2}{\lambda^2}\right) + \mathcal{O}\left(\mathbb{E}[\|\nabla \mathcal{L}(\Theta^{\tilde{t}})\|^2]\right), \quad (25)$$

where $\kappa_\epsilon^2 = \frac{\lambda^2 \kappa_2^2}{\lambda^2 - 8L_\epsilon^2}$, $\Delta \triangleq \mathcal{L}(\Theta^1) - \mathcal{L}(\Theta^*)$ and Θ^* is the optimal solution of $\mathcal{L}(\Theta)$.

Remark 4. Theorem 2 demonstrates that the algorithm converges with a sublinear convergence rate $1/\sqrt{T}$. Besides, the personalized policies converge on average to a neighborhood of the meta policy with a sublinear rate. The radius of the neighborhood is $\mathcal{O}((\lambda^2 + 1)\zeta^2 + \kappa_\epsilon^2/\lambda^2)$, which is determined by the regularization parameter λ , the approximation error ζ , and the parameter κ_ϵ . In particular, the radius gets smaller with the decrease of ζ and κ_ϵ . Besides, a suitable choice of λ should be made to balance the two terms of the radius.

6. Experiments

In this section, the performance of pGFlowMeta is comprehensively verified with three goals in mind: 1) Show that pGFlowMeta finds higher rewards and more diverse end states faster than state-of-art baselines. 2) Demonstrate that pGFlowMeta can effectively solve the problem of performance degradation when encountering distinct tasks. 3) Explore the effects of β and λ .

6.1. Environments setting

Four environments are considered: the environment ‘Grid World’ defined in [5] and three Gym environments, namely ‘Frozen Lake’, ‘Cliff Walking’, and ‘Mountain Car’. The settings of ‘Frozen Lake’ and ‘Cliff Walking’ are modified to have sparse positive rewards and trajectories forming Directed Acyclic Graphs (DAGs), as discussed in the following paragraphs. Table 1 displays the task variables assigned for each environment, leading to generating diverse transitions and distinct tasks. For the ‘Grid World’ environment, different R_0 are set for distinct tasks. For ‘Frozen Lake’, different positions of holes are set for distinct tasks. For ‘Cliff Walking’, different lengths of cliff walking are set for distinct tasks. Fig. 3 provides two examples that illustrate the generation of distinct tasks.

Frozen Lake The ‘Frozen Lake’ environment is structured as a grid, with each grid cell categorized into four types: Start (S), Frozen (F), Hole (H), and Goal (G). The agent’s objective is to navigate from the start cell to the goal cell while avoiding falling into any of the holes. The agent is restricted to four possible movement directions: up, down, left, or right, and each move has a chance of slipping and moving the agent in a direction different from the intended direction. An episode in the ‘Frozen Lake’ environment concludes when the agent successfully reaches the goal cell or falls into a hole, resulting in a termination of the episode. The rewards are set as follow: reaching a hole (H) or a frozen cell (F) yields a reward of 0, while reaching the goal (G) yields a reward of 1.

The original ‘Frozen Lake’ environment is modified to make the rewards positive and the trajectories form DAGs. In this modified version, the agent is restricted to taking actions that only move downwards or to the right. An episode terminates when the agent achieves one of the following conditions: reaching the goal, falling into a hole, or reaching the boundary of the lake. The rewards are set for reaching a hole (H) or a frozen cell (F) as 0.1, and the reward for reaching the goal (G) as 1. It should be noted that different valid maps are created to generate distinct tasks, ensuring that each task has a valid path from the start to the goal and defined valid parents for each state.

Cliff Walking In the original ‘Cliff Walking’ environment, an agent is placed in a grid-world environment represented as a grid of cells. The agent’s goal is to move from the starting point located at the lower-left corner to the goal point at the lower-right corner while ensuring that it avoids the hazardous cliff area in the center-bottom. There is a 4×12 grid matrix in the environment, where the starting position is located at the coordinates $[3, 0]$, the goal position at the coordinates $[3, 11]$, and the cliff positions span from $[3, 1]$ to $[3, 10]$. The agent has the capability to perform actions that allow it to move in four different directions: upward, downward, toward the left, or toward the right. If the agent steps on a cliff position, it is immediately returned to the start position. An episode concludes when the agent successfully reaches the desired goal position. The agent receives a reward of -1 at each time step, and if it mistakenly enters the hazardous cliff area, a more significant penalty of -100 is incurred. Note that there are $3 \times 12 + 2$ possible states in the original environment. The agent cannot step on the cliff; it can occupy all positions of the first 3 rows, as well as the bottom-left cell and the goal position.

The original Cliff Walking environment is modified to make the rewards positive and the trajectories form DAGs. In the modified environment, the starting position is $[0, 0]$ at the top-left, and the agent is restricted to taking actions that only move it downwards or to the right. The episode concludes when the agent accomplishes one of the following: reaching the goal position, falling into the cliff area, or reaching the boundary of the grid world. The rewards are set to reach the cliff as 0.01, the goal as 1, and other positions as 0.1. The length of the cliff varies from 5 to 11 to generate distinct tasks.

Mountain Car The goal of ‘Mountain Car’ is to strategically accelerate the car to reach the goal position on the right of a sinusoid. The states are the position p , velocity v , and step s of the car, and the actions consist of accelerating to the left, and right, and no acceleration. The episode concludes when the position of the car is greater than or equal to the goal position or the car has taken a maximum of 200 steps. The transition dynamics for the mountain car problem can be described as follows:

$$\begin{aligned} v_{t+1} &= v_t + (a - 1)F - G \cos 3 \cdot p_t \\ p_{t+1} &= p_t + v_{t+1} \\ s_{t+1} &= s_t \end{aligned} \quad (26)$$

where $a = 0, 1, 2$ represent accelerating to the left, right, and no acceleration, respectively. The force $F = 0.01$ and gravity $G = 0.025$. Sparse rewards for concluding the trajectory at x are defined as follows:

$$R(x) = 5 - 0.1 * s + 0.1 \quad (27)$$

Distinct tasks with different sine waves are generated by setting various offsets ξ , where the relative position of the starting point and the goal are not changed. In addition, the observation states are discrete and form a finite state space ($64 * 64$) according to the Eq. (28) to satisfy the assumption that the sequences must be of finite length for GFlowNets.

Table 1

Goal positions and generating distinct tasks in diverse environments. Distinct tasks are generated by the 'Distinct Task Variable'. The range of this variable is recorded in the 'Range' row. Furthermore, each environment is associated with a specified 'Goal Position'. In cases where no specific goal position exists for a particular environment, it is denoted by the symbol '-'.

Environment	Grid World	Frozen Lake	Mountain Car	Cliff Walking
Distinct task variable	R_0	Hole position	offsets ξ	Cliff length
Range	[0, 0.1)	[0, 7] X [0, 7]	(-0.3, 0.3)	[8, 11]
Goal position 1	[1, 1]	[0, 7]	0.5	[3, 7]
Goal position 2	[1, 7]	[7, 4]	-	-
Goal position 3	[7, 1]	[7, 7]	-	-
Goal position 4	[7, 7]	-	-	-

Table 2

The hyperparameter settings of pGFlowMeta on different environments.

Hyperparameters	Value			
	Grid World	Frozen Lake	Cliff Walking	MountainCar
Personalized learning rate	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$
Global update round T	30	30	30	30
Batch size	16	16	16	16
Policy network	[256, 256]	[256, 256]	[256, 256]	[256, 256]
Personalized update round R	20	20	20	20
Parameter λ	15	15	15	15
Learning rate η	0.005	0.005	0.005	0.005
Discount factor γ	0.99	0.99	0.99	0.99

$$(p_d, v_d) = \left(\frac{n*(p - \min(p))}{\max(p) - \min(p)}, \frac{n*(v - \min(v))}{\max(v) - \min(v)} \right) \quad (28)$$

6.2. Experiment details

We conducted the experiments on the NVIDIA Tesla K80 environment. The code was implemented using the Python programming language. All the code and the environments in this paper can be referred to at <https://github.com/jixinyuan1996/pGFlowMeta>. The fixed random seed is used to mitigate the potential impact of random factors and allow replication. Table 1 and Table 2 present detailed information regarding the distinct tasks and parameters used in pGFlowMeta. A neural network model is employed as the policy model, featuring an input layer dimension equal to the observation space dimension, 2 hidden layers with 256 neurons each, and an output layer dimension equal to the action space dimension. 'Batch size' refers to the size of trajectories rather than individual transitions. Additionally, the critical hyperparameter settings are provided for pGFlowMeta in the four environments: 'Grid World', 'Frozen Lake', 'Cliff Walking', and 'Mountain Car'. The other parameters of pGFlowMeta are set to be the same as in [5]. Other compared algorithms were trained for the same number of training steps as pGFlowMeta. As baselines, the native *GFlowNets* [5], non meta-RL methods *PPO* [33], *GFlowMeta*, and meta-RL methods like *PEARL* [30], *MAML* [11] were considered to compare their performance with pGFlowMeta, which includes both the meta policy (MP) and personalized policy (PP).

The Empirical L_1 Error [5] is used to test whether the algorithm generates transitions proportional to the given positive returns. The Empirical L_1 Error is calculated by Empirical L_1 Error = $\mathbb{E}[|p(x) - \pi(x)|]$, where $p(x) = \frac{R(x)}{\sum_{x \in \mathcal{X}} R(x)}$ represents the known distribution of positive rewards in the environment, and $\pi(x)$ is estimated by repeated sampling and counting frequencies for each possible end state x . Modes Found [5] is used to evaluate the exploration performance. It represents the number of modes with at least one visited state as a function of the total number of visited states. The Averaged Rewards metric is used to measure the performance of an agent in a specific task or environment. It is calculated by taking the mean of the batch end-state rewards from sampled trajectories of test tasks. These evaluation metrics provide valuable insights into the algorithm's performance and its ability to generate appropriate transitions, explore the environment effectively, and achieve high rewards in distinct tasks.

6.3. Comparison to the state-of-the-art methods

The experiments are designed to compare the performance of pGFlowMeta with state-of-the-art methods in four environments. These environments have been described in detail in the section 6.1.

Empirical L_1 error Fig. 4 shows the evaluation results of Empirical L_1 Error on four environments. Fig. 4 demonstrates that GFlowNets, GFlowMeta, pGFlowMeta achieve lower Empirical L_1 Error compared to the PEARL and PPO methods. This suggests that the methods within the GFlowNets class have the capability to generate transitions proportional to positive rewards, which is valuable in exploration tasks. Traditional RL methods like PPO and PEARL show higher Empirical L_1 Error because they typically generate transitions associated with the highest reward, while ignoring other high-reward transitions. One of the reasons for PPO's

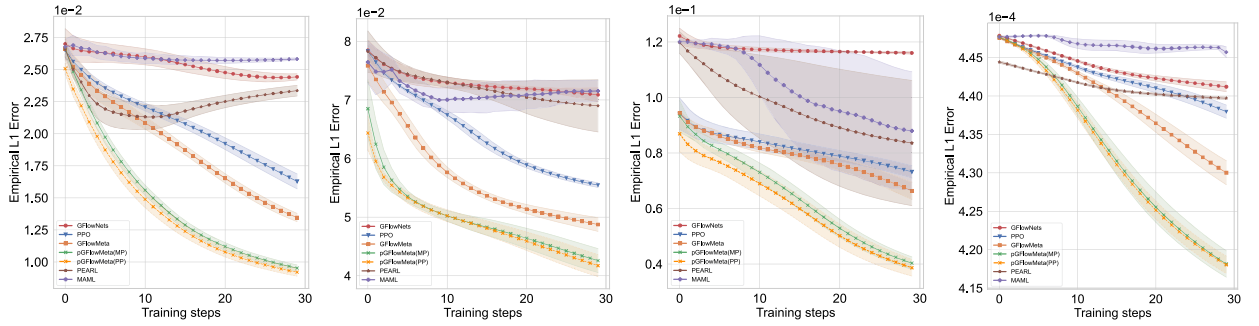


Fig. 4. Empirical L1 Error of various algorithms in the environment 'Grid World', 'Frozen Lake', 'Cliff Walking', and 'Mountain Car'.

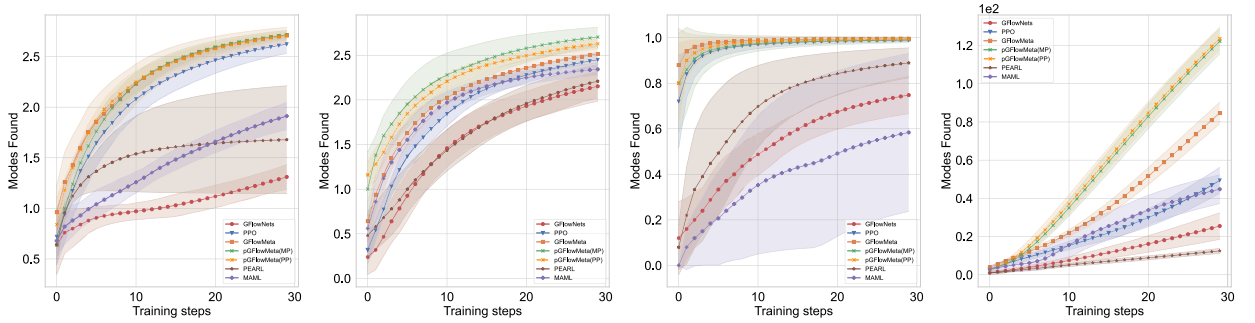


Fig. 5. Modes Found of various algorithms in the environment 'Grid World' (left), 'Frozen Lake' (middle), and 'Mountain Car'.

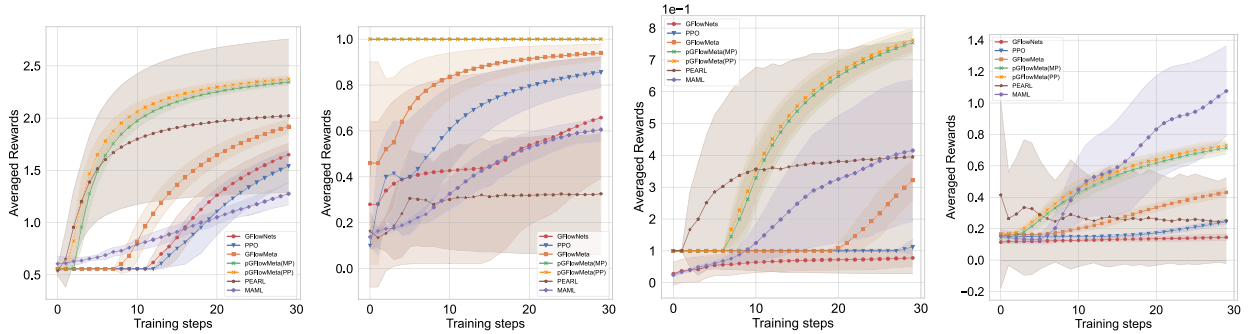


Fig. 6. Averaged rewards of various algorithms in the environment 'Grid World', 'Frozen Lake', 'Cliff Walking', and 'Mountain Car'.

stable L1 Error performance across different cases is that PPO is primarily designed for single tasks, and it may not adapt well to different tasks, often focusing on one or very few candidate end states in various tasks. PEARL shows lower Empirical L1 Error compared to PPO, because PEARL is designed for multi-tasks, and it can end up with different candidate states for different tasks. Among the methods compared, pGFlowMeta shows the lowest Empirical L_1 Error, which aligns with the experimental expectations. This shows that pGFlowMeta effectively adapts to distinct tasks compared to GFlowMeta.

Modes Found Fig. 5 illustrates the Modes Found in four different environments. It depicts the number of modes visited at least once for various methods, as the number of visited states increases. The total number of Modes Found is primarily influenced by the specific design of the reward function. If there are n end states with positive rewards, the total number of Modes Found equals n . From Fig. 5, it can be observed that pGFlowMeta can frequently achieve faster visits to all the end states in different environments. That is, pGFlowMeta has the best exploration in general situations, which helps to visit all the interesting modes faster.

Rewards The average rewards of different methods are shown in Fig. 6, which demonstrates that pGFlowMeta achieves the best average rewards compared to other methods in 'Grid World', 'Frozen Lake', and 'Cliff Walking'. In contrast, pGFlowMeta does not perform well in the 'Mountain Car' environment. Since the rewards in 'Grid World', 'Frozen Lake', and 'Cliff Walking' are more evenly distributed, these three tasks are more inclined to exploration. pGFlowMeta has a better exploration ability and hence can converge stably. As for the 'Mountain Car' environment, its reward distribution is relatively steep, and sampling near the maximum reward

Table 3
Ranking comparison of algorithms on the Friedman test.

	Algorithms	Grid World	Frozen Lake	Cliff Walking	Mountain Car	Averaged ranking
Average Rewards	PPO	5	5	7	7	6
	GFlowNets	6	4	6	6	5.5
	GFlowMeta	4	3	5	4	4
	PEARL	3	7	4	5	4.75
	MAML	7	6	3	1	4.25
	pGFlowMeta (MP)	2	1	2	3	2
	pGFlowMeta (PP)	1	2	1	2	1.5
	Algorithms	Grid World	Frozen Lake	Cliff Walking	Mountain Car	Averaged ranking
Empirical L1 Error	PPO	6	6	7	6	6.25
	GFlowNets	4	4	4	4	4
	GFlowMeta	3	3	3	3	3
	PEARL	5	5	5	5	5
	MAML	7	7	6	7	6.75
	pGFlowMeta (MP)	2	2	2	2	2
	pGFlowMeta (PP)	1	1	1	1	1
	Algorithms	Grid World	Frozen Lake	Cliff Walking	Mountain Car	Averaged ranking
Modes Found	PPO	7	7	6	6	6.5
	GFlowNets	4	4	4	4	4
	GFlowMeta	2	3	1	3	2.25
	PEARL	6	6	5	7	6
	MAML	5	5	7	5	5.5
	pGFlowMeta (MP)	1	1	3	2	1.75
	pGFlowMeta (PP)	3	2	2	1	2

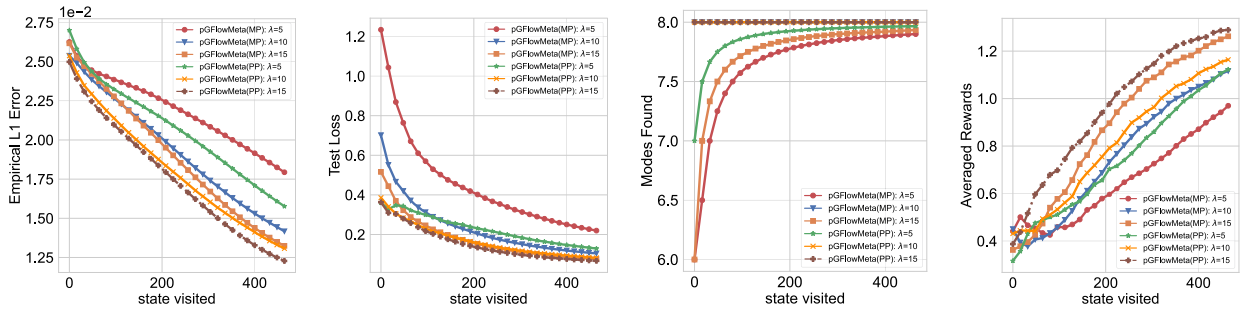


Fig. 7. Effect of λ on the convergence of pGFlowMeta in 'Grid World' environment. $R = 20$, $\beta = 1$.

can achieve faster convergence. It is reasonable for pGFlowMeta to perform worse than other baselines in this environment in terms of reward.

Friedman test This section reports the results of the Friedman test applied to assess the existence of significant differences among multiple algorithms [12]. First, the rankings of different algorithms are listed on four environments as shown in Table 3, including Average Rewards, Empirical L_1 Error, and Modes Found. This table shows that pGFlowMeta consistently ranks higher than other considered algorithms. The test statistic is computed according to Eq. (29)

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \text{ and } F_F = \frac{(N-1)\chi_F^2}{N(k-1) - \chi_F^2}, \quad (29)$$

where R_j is the averaged ranking of algorithm j , k is the number of algorithms and N is the number of datasets [12]. The value $Q_1 = 4.72$ (Averaged Rewards), the value $Q_2 = 221$ (Empirical L_1 Error), and the value $Q_3 = 21.89$ (Modes Found).

According to the scipy package [39], the critical value for an F distribution with 6 (i.e., k) and 18 (i.e., $(k-1)(N-1)$) degrees of freedom for 5% of the significance level is 2.661. All the calculated Q values are greater than the critical value of Q for the 0.05 significance level. As $Q_{\text{calculated}} > Q_{\text{critical}}$, hence the null hypotheses H_0 can be rejected that the evaluated results of different algorithms are all the same. Thus, the Friedman test demonstrates that pGFlowMeta reliably outperforms the other considered algorithms.

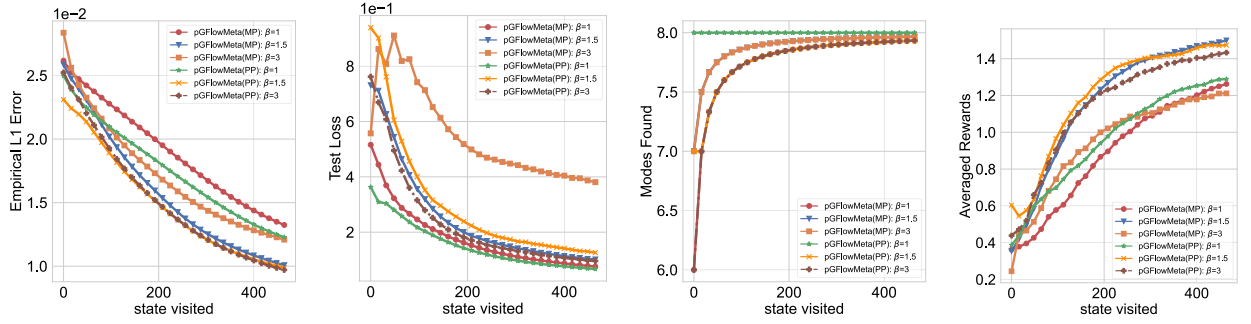


Fig. 8. Effect of β on the convergence of pGFlowMeta in 'Grid World' environment. $R = 20$, $\lambda = 15$.

Table 4

The averaged reward comparison results for each algorithm.

Algorithms	Grid World	Frozen Lake	Cliff Walking
GFlowNets*	2.42 ± 0.03	1.00 ± 0.00	0.77 ± 0.04
GFlowNets	1.42 ± 0.07	0.86 ± 0.07	0.11 ± 0.02
GFlowMeta	1.92 ± 0.07	0.94 ± 0.04	0.32 ± 0.06
pGFlowMeta (MP)	2.34 ± 0.03	1.00 ± 0.00	0.75 ± 0.03
pGFlowMeta (PP)	2.38 ± 0.03	1.00 ± 0.00	0.76 ± 0.04

6.4. Effect of β and λ values

Another important aspect that requires investigation is the impact of the β and λ values on pGFlowMeta. These experiments are run in the 'Grid World' environment. Fig. 7 shows the performance of pGFlowMeta with different values of λ , while keeping $B = 16$, $R = 20$, $\eta = 0.005$, $\beta = 1$. λ is set to control the distance between the meta policy and the personalized policy. A larger value of λ discourages personalization, while a smaller value of λ encourages it. Fig. 7 indicates that $\lambda = 15$ can make pGFlowMeta to converge faster. Therefore, the parameter $\lambda = 15$ is recommended in all the environments in this paper. However, it is worth noting that larger λ may also cause pGFlowMeta to diverge. Hence, carefully selecting λ for different environments can further improve performance.

In Fig. 8, the influence of β ($\beta \geq 1$) on both the personalized and meta policies in pGFlowMeta is explored, using $B = 16$, $R = 20$, $\eta = 0.005$, $\lambda = 15$. β is set to control the convergence speed of the aggregated meta policy. It is recommended to set β to a larger value in order to achieve faster convergence. However, turning β carefully is also important to prevent the divergence and instability of pGFlowMeta. Here, $\beta = 1$ implies that the aggregation of the meta policy in pGFlowMeta is the same as that in GFlowMeta. As depicted in Fig. 8, different evaluation indices exhibit distinct optimal β values. Notably, a β value of 1.5 yields the best performance in terms of Empirical L1 Error, while $\beta = 1$ demonstrates the optimal performance for test loss. Hence, the selection of β should be done thoughtfully, taking into account the specific use case or the objective at hand. In this paper, the parameter $\beta = 1$ is recommended for all the considered environments.

Furthermore, it is important to highlight that the personalized policy consistently outperforms the meta policy in pGFlowMeta, regardless of the chosen metric. This can be attributed to the fact that the personalized policy is capable of adapting to the unique characteristics and requirements of individual tasks, whereas the meta policy serves as a general policy applied to all tasks.

6.5. Ablation experiments

The comparison of the performance of pGFlowMeta is conducted with GFlowNets, GFlowNets*, and GFlowMeta, and the results are presented in Table 4, which shows the averaged rewards achieved by each algorithm. GFlowNets* represents the averaged rewards obtained by the optimal GFlowNets policy for each task, while GFlowNets represents the averaged rewards achieved by the GFlowNets policy across multiple tasks.

From Table 4, our personalized policy (PP) is considerably closer to the optimal policy compared to the meta policy (MP), indicating that it performs well in adapting to the specific requirements of individual tasks. Additionally, our meta policy outperforms the other algorithms, suggesting that it exhibits superior overall performance. Figs. 4, 5, 6 provide further insights into the comparison, indicating that the personalized policy outperforms the meta policy in terms of suitability for its specific task. This discovery highlights the generalization capacity of the personalized policy within the constraints imposed by the meta policy. Furthermore, the performance of meta policy in pGFlowMeta is better than that in GFlowMeta, indicating that personalized policy can improve the performance of meta policy in pGFlowMeta.

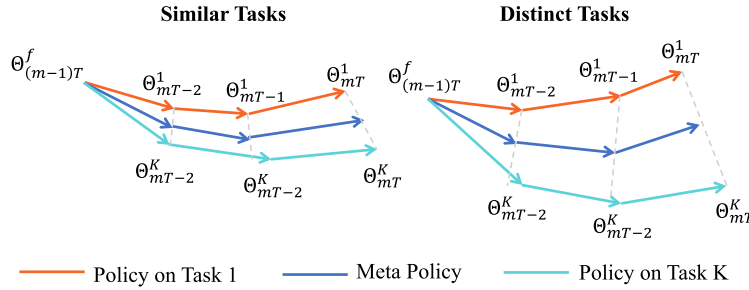
6.6. Performance on similar / distinct tasks

In this experiment, the comparison between pGFlowMeta and GFlowMeta methods on two types of tasks is conducted: distinct tasks and similar tasks. The purpose is to assess the performance difference between GFlowMeta and pGFlowMeta in these task

Table 5

The averaged reward comparison results on similar and distinct tasks for each algorithm.

Environments	Algorithms	Similar Tasks	Distinct Tasks
Grid World	GFlowMeta	1.96 ± 0.08	1.92 ± 0.07
	pGFlowMeta (MP)	2.36 ± 0.03	2.34 ± 0.03
	pGFlowMeta (PP)	2.38 ± 0.04	2.38 ± 0.03
Frozen Lake	GFlowMeta	0.96 ± 0.05	0.94 ± 0.04
	pGFlowMeta (MP)	1.00 ± 0.00	1.00 ± 0.00
	pGFlowMeta (PP)	$0.99(8) \pm 0.00(2)$	1.00 ± 0.00
Cliff Walking	GFlowMeta	0.36 ± 0.13	0.32 ± 0.06
	pGFlowMeta (MP)	0.78 ± 0.01	0.75 ± 0.03
	pGFlowMeta (PP)	0.76 ± 0.04	0.76 ± 0.04

**Fig. 9.** Illustration of the weight divergence for the policy with similar tasks and distinct tasks.

scenarios. The experiments are conducted five times, and the results are then averaged to obtain the final outcome. For each run, distinct tasks are generated according to the instructions provided in Section 6.1 and Table 1. To be fair, the different parameters of tasks are used to create similar tasks for each experiment run.

Table 5 presents the averaged rewards for GFlowMeta and pGFlowMeta, with the best results on each environment highlighted in bold. As expected, both pGFlowMeta and GFlowMeta experience a decrease in performance when dealing with distinct tasks. This decrease can be attributed to the differences between the task distributions. Fig. 9 provides an illustration to help understand this phenomenon. When tasks are similar, the divergence between the policy on task k and the meta policy is small. Even after the m -th synchronization, the policy on task k remains still close to the meta policy. However, when tasks are distinct, the divergence between the policy on task k and the meta policy becomes much larger and accumulates rapidly due to the differences in the data distribution. This results in a significant increase in divergence. Furthermore, pGFlowMeta demonstrates lower performance degradation compared to GFlowMeta, indicating its higher robustness to distinct tasks. This showcases the advantage of pGFlowMeta in adapting and maintaining performance even when faced with varying and diverse task scenarios.

6.7. Performance on unseen tasks

In the evaluation of GFlowMeta and pGFlowMeta (MP) algorithms on unseen tasks in various environments, reported in Table 6, significant performance differences are evident. Notably, the averaged rewards of pGFlowMeta(MP) consistently outperform GFlowMeta in the 'Grid World', 'Cliff Walking', and 'Mountain Car' environments, as seen in Table 6. This indicates that the meta policy of pGFlowMeta can adapt and excel in complex scenarios. This trend is also reflected in the Empirical L_1 Error and the Modes Found, where pGFlowMeta (MP) demonstrates greater flexibility and exploration capability, finding more candidates with high rewards.

Overall, these results suggest that pGFlowMeta (MP) is particularly well-suited for intricate and dynamic environments, where it consistently outperforms GFlowMeta. Its adaptability, high rewards, and accuracy in value predictions make pGFlowMeta (MP) a promising choice for a variety of applications, especially those with complex and evolving task scenarios.

7. Discussion

In summary, our experimental results demonstrate that pGFlowMeta significantly outperforms the state-of-the-art methods on the Modes Found in the discrete environments. Moreover, pGFlowMeta can further adapt well to distinct tasks. To enhance the practical deployment of pGFlowMeta, this section provides insights into real-world applications and limitations of pGFlowMeta.

Table 6

The performance on unseen tasks for GFlowMeta, pGFlowMeta (MP) algorithm.

Averaged Rewards												
Environments	Algorithms	task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10	Average
Grid World	GFlowMeta	1.94	1.96	1.95	1.94	1.93	1.95	1.93	1.98	1.98	1.93	1.95 $\pm$ 0.02
	pGFlowMeta(MP)	2.34	2.36	2.35	2.34	2.33	2.35	2.33	2.38	2.38	2.33	2.35 $\pm$ 0.02
Frozen Lake	GFlowMeta	1.00	0.10	1.00	1.00	1.00	1.00	1.00	0.10	1.00	1.00	0.82 $\pm$ 0.36
	pGFlowMeta(MP)	1.00	0.10	1.00	1.00	1.00	1.00	1.00	0.10	1.00	1.00	0.82 $\pm$ 0.36
Cliff Walking	GFlowMeta	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30 $\pm$ 0.00
	pGFlowMeta(MP)	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77	0.77 $\pm$ 0.00
Mountain Car	GFlowMeta	0.45	0.34	0.39	0.43	0.52	0.39	0.53	0.26	0.23	0.54	0.41 $\pm$ 0.10
	pGFlowMeta(MP)	0.73	0.59	0.67	0.73	0.82	0.66	0.83	0.46	0.45	0.86	0.68 $\pm$ 0.14
Mode Finds												
Environments	Algorithms	task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10	Average
Grid World	GFlowMeta	2.64	2.54	2.63	2.49	2.65	2.64	2.65	2.79	2.67	2.62	2.63 $\pm$ 0.08
	pGFlowMeta(MP)	2.76	2.71	2.58	2.71	2.65	2.72	2.74	2.76	2.75	2.85	2.72 $\pm$ 0.07
Frozen Lake	GFlowMeta	2.88	1.71	2.90	2.91	2.67	2.86	2.89	1.96	2.95	2.89	2.66 $\pm$ 0.42
	pGFlowMeta(MP)	2.94	1.65	2.84	2.92	2.78	2.89	2.95	1.94	2.93	2.92	2.68 $\pm$ 0.45
Cliff Walking	GFlowMeta	0.98	0.95	0.97	0.97	0.97	0.99	0.98	0.97	0.93	0.99	0.97 $\pm$ 0.02
	pGFlowMeta(MP)	0.99	0.97	0.94	0.99	0.98	0.97	0.95	0.99	0.99	0.99	0.98 $\pm$ 0.02
Mountain Car	GFlowMeta	82.32	67.57	79.84	88.01	94.28	73.30	92.11	44.58	42.91	94.83	75.97 $\pm$ 18.19
	pGFlowMeta(MP)	127.01	105.53	111.77	126.57	136.85	113.17	132.39	81.70	74.40	138.69	114.81 $\pm$ 21.16
Empirical L_1 Error												
Environments	Algorithms	task 1	task 2	task 3	task 4	task 5	task 6	task 7	task 8	task 9	task 10	Average
Grid World	GFlowMeta	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01 $\pm$ 0.00
	pGFlowMeta(MP)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01 $\pm$ 0.00
Frozen Lake	GFlowMeta	0.05	0.08	0.06	0.06	0.06	0.06	0.06	0.08	0.05	0.06	0.06 $\pm$ 0.01
	pGFlowMeta(MP)	0.05	0.08	0.05	0.05	0.06	0.05	0.06	0.09	0.05	0.06	0.06 $\pm$ 0.01
Cliff Walking	GFlowMeta	0.07	0.07	0.07	0.07	0.07	0.08	0.07	0.08	0.07	0.07	0.07 $\pm$ 0.00
	pGFlowMeta(MP)	0.04	0.04	0.04	0.04	0.04	0.05	0.04	0.05	0.04	0.04	0.04 $\pm$ 0.01
Mountain Car	GFlowMeta	4.29e-04	4.39e-04	4.31e-04	4.28e-04	4.25e-04	4.35e-04	4.24e-04	4.48e-04	4.48e-04	4.25e-04	4.33e-04 $\pm$ 8.60e-06
	pGFlowMeta(MP)	4.16e-04	4.28e-04	4.21e-04	4.17e-04	4.12e-04	4.23e-04	4.12e-04	4.39e-04	4.40e-04	4.11e-04	4.22e-04 $\pm$ 1.02e-05

Applications Similar to real-world applications of GFlowNets, pGFlowMeta can also be utilized for causal inference, molecule design, candidate generation, etc. pGFlowMeta can distinguish itself by its ability to rapidly learn and adapt to different tasks within these specific domains.

- (1) Causal inference: pGFlowMeta can be used to learn causal structures and mechanisms from data, enabling causal reasoning and interventions. For instance, the posterior distribution of causal graphs can be inferred from data, and new causal graphs can be sampled from it.
- (2) Molecule design: pGFlowMeta can be employed in designing molecules with specific properties, facilitating drug discovery and materials science. For example, a probability distribution can be learned from protein, DNA, or RNA sequences, and new sequences can be sampled from it.
- (3) Candidate generation: pGFlowMeta can be used to generate diverse and high-quality candidates, thereby improving the efficiency of tasks such as search, recommendation, and decision-making. A probability distribution can be learned from user queries or contexts, and relevant and interesting candidates can be sampled from it.

Limitations Nevertheless, it's important to note that pGFlowMeta does have some limitations, including:

- (1) Although pGFlowMeta achieves excellent performance in the discrete environments when compared to other baselines, it cannot be applied to continuous environments due to its dependency on the limitations of the naive GFlowNets. This is because GFlowNets need to compute the flow-matching loss by traversing the inflows and outflows of each node in the trajectory.
- (2) pGFlowMeta involves combining personalized policies with a meta policy, which could add extra complexity to the learning process, requiring additional computational and storage resources to implement and scale. However, it is a necessary trade-off to enhance its performance on distinct tasks. In cases where tasks are similar, using GFlowMeta is recommended, offering a less complex alternative.
- (3) Although pGFlowMeta can achieve excellent performance in sparse and discrete environments, it still suffers from problems of brittleness. Understanding and incorporating uncertainty into pGFlowMeta is critical to success in real-world situations and robustness to new environments [1,10,22].

8. Conclusions and future work

Although existing multi-task and meta-reinforcement learning methods can adapt to new tasks quickly with less additional training, they ignore exploration to find a diverse candidate set. This paper proposes GFlowMeta that builds on the capabilities of GFlowNets to generate diverse candidate solutions. Based on the limited performance of GFlowMeta on distinct tasks, a personalized Meta Generative Flow Network called pGFlowMeta is proposed. pGFlowMeta achieves dual learning: acquiring a meta policy that applies to all tasks, while also acquiring personalized policies tailored to each individual task. This is facilitated by incorporating a personalization constraint into the objective function.

Numerical experiments are conducted to compare the proposed GFlowMeta and pGFlowMeta with the state of art methods. Experimental results showcase the effectiveness of GFlowMeta and pGFlowMeta, as it outperforms several advanced algorithms on Modes Found in sparse and discrete control tasks. Moreover, experimental results highlight the capability of pGFlowMeta to achieve superior performance on distinct tasks compared to GFlowMeta, demonstrating the effectiveness of personalizing policies for individual tasks within the meta policy framework.

However, at present, pGFlowMeta is only suitable for discrete environments. In the future, efforts will be directed towards extending pGFlowMeta to continuous environments, broadening its applicability to a wider range of scenarios.

CRedit authorship contribution statement

Xinyuan Ji: Writing – review & editing, Writing – original draft. **Xu Zhang:** Writing – review & editing, Formal analysis. **Wei Xi:** Writing – review & editing. **Haozhi Wang:** Writing – review & editing, Supervision. **Olga Gadyatskaya:** Writing – review & editing. **Yinchuan Li:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

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

Data availability

Data will be made available on request.

Appendix A. Proof of Main Theorems

Firstly, some useful existing inequalities are provided in Appendix A.1, including Jensen's inequality and Young's inequality. Next, important lemmas are proven in Appendix A.2 and the proofs of Theorems 1 and 2 are given in Appendix A.3 and Appendix A.4, respectively.

A.1. Some useful results

Fact 1 (Jensen's inequality).

$$\left\| \frac{1}{M} \sum_{i=1}^M x_i \right\|^2 \leq \frac{1}{M} \sum_{i=1}^M \|x_i\|^2 \quad \text{and} \quad \left\| \sum_{i=1}^M x_i \right\|^2 \leq M \sum_{i=1}^M \|x_i\|^2 \quad (\text{A.1})$$

Fact 2 (Young's inequality). For any vector $x_1, x_2 \in \mathbb{R}^d$ and any $\epsilon > 0$, then

$$2\langle x_1, x_2 \rangle \leq \frac{1}{\epsilon} \|x_1\|_2^2 + \epsilon \|x_2\|_2^2, \quad \text{and} \quad \|x_1 - x_2\|_2^2 \leq \left(1 + \frac{1}{\epsilon}\right) \|x_1\|_2^2 + (1 + \epsilon) \|x_2\|_2^2, \quad (\text{A.2})$$

A.2. Useful lemmas

Lemma 3. If $\tilde{\eta}^2 \leq \frac{\beta^2 R}{4(1+R)L_c^2}$, the task drift error is upper-bounded

$$\mathbb{E} \left[\|s_{i,r}^t - \nabla \mathcal{L}_i(\Theta^t)\|^2 \right] \leq 2\lambda^2 \zeta^2 + \frac{32\tilde{\eta}^2 L_c^2}{\beta^2} \left(\mathbb{E} [\|\nabla \mathcal{L}_i(\Theta^t)\|^2] + 2\lambda^2 \zeta^2 \right). \quad (\text{A.3})$$

Lemma 4. By considering the fulfillment of Assumption 1, $\lambda > 2\sqrt{2}L_c$, then

$$\frac{1}{N} \sum_{i=1}^N \left\| \nabla \mathcal{L}_i(\Theta) - \nabla \mathcal{L}(\Theta) \right\|^2 \leq 2\kappa_c^2 + \frac{8L_c^2}{\lambda^2 - 8L_c^2} \left\| \nabla \mathcal{L}(\Theta) \right\|^2, \quad (\text{A.4})$$

where $\kappa_{\mathcal{L}}^2 = \frac{\lambda^2 \kappa_2^2}{\lambda^2 - 8L_{\mathcal{L}}^2}$.

A.3. Proof of Theorem 1

Since $\ell_i(\cdot)$ is $L_{\mathcal{L}}$ -smooth and $f_i(\cdot, \Theta)$ is λ -strongly-convex, the $\rho_{\Theta}(\cdot)$ is $(\lambda - L_{\mathcal{L}})$ -strongly convex. Besides, $\theta_i^*(\Theta)$ is the unique solution of $\rho_{\Theta}(\theta)$ and $\nabla \rho_{\Theta}(\theta_i^*(\Theta)) = 0$. Together with the strongly-convexity of $\rho_{\Theta}(\theta)$, one obtains

$$\begin{aligned} \|\hat{\theta}_i(\Theta) - \theta_i^*(\Theta)\|^2 &\leq \frac{1}{(\lambda - L_{\mathcal{L}})^2} \|\nabla \rho_{\Theta}(\hat{\theta}_i(\Theta)) - \nabla \rho_{\Theta}(\theta_i^*(\Theta))\|^2 \\ &= \frac{1}{(\lambda - L_{\mathcal{L}})^2} \|\nabla \rho_{\Theta}(\hat{\theta}_i(\Theta))\|^2 \\ &\leq \frac{2}{(\lambda - L_{\mathcal{L}})^2} \left(\|\nabla \rho_{\Theta}(\hat{\theta}_i(\Theta)) - \nabla \tilde{\rho}_{\Theta}(\hat{\theta}_i(\Theta))\|^2 + \|\nabla \tilde{\rho}_{\Theta}(\hat{\theta}_i(\Theta))\|^2 \right) \\ &\leq \frac{2}{(\lambda - L_{\mathcal{L}})^2} \left(\|\nabla \ell_i(\hat{\theta}_i(\Theta)) - \nabla \tilde{\ell}(\hat{\theta}_i(\Theta))\|^2 + \delta^2 \right), \end{aligned} \quad (\text{A.5})$$

In the 2nd inequality, Jensen's inequality is applied, and in the final inequality, the assumption $|\nabla \tilde{\rho}_{\Theta}^i(\hat{\theta}_i(\Theta))| \leq \delta$ is utilized. By taking the expectation over the trajectories, the following implication can be deduced

$$\mathbb{E} \left[\|\hat{\theta}_i(\Theta) - \theta_i^*(\Theta)\|^2 \right] \leq \frac{2}{(\lambda - L_{\mathcal{L}})^2} (\kappa_1^2 + \delta^2), \quad (\text{A.6})$$

Serving as the concluding step in the proof, (A.6) demonstrates the desired result.

A.4. Proof of Theorem 2

Since $\mathcal{L}(\cdot)$ is $L_{\mathcal{L}}$ -smooth, then

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] &\leq \mathbb{E}[\langle \nabla \mathcal{L}(\Theta^t), \Theta^{t+1} - \Theta^t \rangle] + \frac{L_{\mathcal{L}}}{2} \mathbb{E}[\|\Theta^{t+1} - \Theta^t\|^2] \\ &= -\tilde{\eta} \mathbb{E}[\langle \nabla \mathcal{L}(\Theta^t), g^t \rangle] + \frac{\tilde{\eta}^2 L_{\mathcal{L}}}{2} \mathbb{E}[\|g^t\|^2] \\ &= -\tilde{\eta} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] - \tilde{\eta} \mathbb{E}[\langle \nabla \mathcal{L}(\Theta^t), g^t - \nabla \mathcal{L}(\Theta^t) \rangle] + \frac{\tilde{\eta}^2 L_{\mathcal{L}}}{2} \mathbb{E}[\|g^t\|^2]. \end{aligned} \quad (\text{A.7})$$

By using the fact that $2|\langle x, y \rangle| \leq \|x\|^2 + \|y\|^2$ for any vectors $x, y \in \mathbb{R}^d$, one obtains

$$\mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] \leq -\frac{\tilde{\eta}}{2} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] + \frac{\tilde{\eta}}{2} \mathbb{E}[\|g^t - \nabla \mathcal{L}(\Theta^t)\|^2] + \frac{\tilde{\eta}^2 L_{\mathcal{L}}}{2} \mathbb{E}[\|g^t\|^2]. \quad (\text{A.8})$$

By using AM-GM inequality and Jensen's inequality, the obtained result is

$$\begin{aligned} \|g^t\|^2 &\leq 3 \left[\left\| \frac{1}{NR} \sum_{i,r=1}^{N,R} g_{i,r}^t - \frac{1}{N} \sum_{i=1}^N \nabla \mathcal{L}_i(\Theta^t) \right\|^2 + \left\| \frac{1}{N} \sum_{i=1}^N \nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t) \right\|^2 + \left\| \nabla \mathcal{L}(\Theta^t) \right\|^2 \right] \\ &\leq \frac{3}{NR} \sum_{i,r=1}^{N,R} \|g_{i,r}^t - \nabla \mathcal{L}_i(\Theta^t)\|^2 + \frac{3}{N} \sum_{i=1}^N \left\| \nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t) \right\|^2 + 3 \left\| \nabla \mathcal{L}(\Theta^t) \right\|^2. \end{aligned} \quad (\text{A.9})$$

Incorporating (A.9) into (A.8) and using Jensen's inequality, one obtains

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] &\leq -\frac{\tilde{\eta}(1 - 3L_{\mathcal{L}}\tilde{\eta})}{2} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] + \frac{3\tilde{\eta}^2 L_{\mathcal{L}}}{2} \frac{1}{N} \sum_{i=1}^N \mathbb{E}[\left\| \nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t) \right\|^2] \\ &\quad + \frac{\tilde{\eta}(1 + 3\tilde{\eta}L_{\mathcal{L}})}{2} \frac{1}{NR} \sum_{i,r=1}^{N,R} \mathbb{E}[\|g_{i,r}^t - \nabla \mathcal{L}_i(\Theta^t)\|^2]. \end{aligned} \quad (\text{A.10})$$

By using Lemma 3, one obtains

$$\begin{aligned} \mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] &\leq -\frac{\tilde{\eta}(1 - 3L_{\mathcal{L}}\tilde{\eta})}{2} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] + \frac{3\tilde{\eta}^2 L_{\mathcal{L}}}{2} \frac{1}{N} \sum_{i=1}^N \mathbb{E}[\left\| \nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t) \right\|^2] \\ &\quad + \frac{\tilde{\eta}(1 + 3\tilde{\eta}L_{\mathcal{L}})}{2} \left(2\lambda^2 \zeta^2 + \frac{32\tilde{\eta}^2 L_{\mathcal{L}}^2}{\beta^2} \left(\frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\nabla \mathcal{L}_i(\Theta^t)\|^2] + 2\lambda^2 \zeta^2 \right) \right). \end{aligned} \quad (\text{A.11})$$

Using the fact that $\frac{1}{N} \sum_{i=1}^N \|x_i\|^2 = \|\bar{x}\|^2 + \frac{1}{N} \sum_{i=1}^N \|x_i - \bar{x}\|^2$, where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$, expression (A.12) can be obtained, shown in the following.

$$\begin{aligned}
& \mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] \\
& \leq -\frac{\tilde{\eta}(1-3L_{\mathcal{L}}\tilde{\eta})}{2} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] + \frac{3\tilde{\eta}^2 L_{\mathcal{L}}}{2} \frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t)\|^2] \\
& \quad + \frac{\tilde{\eta}(1+3\tilde{\eta}L_{\mathcal{L}})}{2} \left(2\lambda^2 \zeta^2 + \frac{32\tilde{\eta}^2 L_{\mathcal{L}}^2}{\beta^2} \left(\frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t)\|^2] + \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] + 2\lambda^2 \zeta^2 \right) \right) \\
& = \tilde{\eta}(1+3\tilde{\eta}L_{\mathcal{L}})\lambda^2 \zeta^2 \left(1 + \frac{32\tilde{\eta}^2 L_{\mathcal{L}}^2}{\beta^2} \right) + \left(\frac{16\tilde{\eta}^3(1+3\tilde{\eta}L_{\mathcal{L}})L_{\mathcal{L}}^2}{\beta^2} - \frac{\tilde{\eta}(1-3L_{\mathcal{L}}\tilde{\eta})}{2} \right) \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] \\
& \quad + \left(\frac{16\tilde{\eta}^3(1+3\tilde{\eta}L_{\mathcal{L}})L_{\mathcal{L}}^2}{\beta^2} + \frac{3\tilde{\eta}^2 L_{\mathcal{L}}}{2} \right) \left(\frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\nabla \mathcal{L}_i(\Theta^t) - \nabla \mathcal{L}(\Theta^t)\|^2] \right). \tag{A.12}
\end{aligned}$$

By using Lemma 4, one can then obtain

$$\begin{aligned}
& \mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] \\
& \leq \tilde{\eta}(1+3\tilde{\eta}L_{\mathcal{L}})\lambda^2 \zeta^2 \left(1 + \frac{32\tilde{\eta}^2 L_{\mathcal{L}}^2}{\beta^2} \right) + \left(\frac{32\tilde{\eta}^3(1+3\tilde{\eta}L_{\mathcal{L}})L_{\mathcal{L}}^2}{\beta^2} + 3\tilde{\eta}^2 L_{\mathcal{L}} \right) \kappa_{\mathcal{L}}^2 \\
& \quad + \tilde{\eta} \left[\frac{16\tilde{\eta}^2(1+3\tilde{\eta}L_{\mathcal{L}})L_{\mathcal{L}}^2}{\beta^2} \frac{\lambda^2}{\lambda^2 - 8L_{\mathcal{L}}^2} - \frac{(1-3L_{\mathcal{L}}\tilde{\eta})}{2} + \frac{12\tilde{\eta}L_{\mathcal{L}}L_{\mathcal{L}}^2}{\lambda^2 - 8L_{\mathcal{L}}^2} \right] \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] \\
& \leq 2\tilde{\eta}\lambda^2 \zeta^2 + 3\tilde{\eta}^2 L_{\mathcal{L}} \kappa_{\mathcal{L}}^2 + \frac{64\tilde{\eta}^3 L_{\mathcal{L}}^2 (\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}{\beta^2} \\
& \quad - \tilde{\eta} \left[\frac{1}{2} - \tilde{\eta}L_{\mathcal{L}} \left(\frac{3}{2} + \frac{16\lambda^2}{\lambda^2 - 8L_{\mathcal{L}}^2} + \frac{12L_{\mathcal{L}}^2}{\lambda^2 - 8L_{\mathcal{L}}^2} \right) \right] \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2], \tag{A.13}
\end{aligned}$$

where 2nd inequality uses $\tilde{\eta} \leq \frac{\beta}{2L_{\mathcal{L}}}$, $\beta \geq 1$ and $1+3\tilde{\eta}L_{\mathcal{L}} \leq 1 + \frac{3}{70\lambda^2} < 2$ since $\tilde{\eta} \leq \frac{1}{70L_{\mathcal{L}}\lambda^2}$ and $\lambda^2 - 8L_{\mathcal{L}}^2 \geq 1$. Using $\lambda^2 - 8L_{\mathcal{L}}^2 \geq 1$ again implies

$$L_{\mathcal{L}} \left(\frac{3}{2} + \frac{16\lambda^2}{\lambda^2 - 8L_{\mathcal{L}}^2} + \frac{12L_{\mathcal{L}}^2}{\lambda^2 - 8L_{\mathcal{L}}^2} \right) \leq L_{\mathcal{L}} \left(\frac{3}{2} + 16\lambda^2 + 12L_{\mathcal{L}}^2 \right) \leq \frac{35}{2} L_{\mathcal{L}} \lambda^2. \tag{A.14}$$

Using the assumption that $\tilde{\eta} \leq \frac{1}{70L_{\mathcal{L}}\lambda^2} \triangleq \tilde{\eta}_o$ yields

$$\frac{1}{2} - \tilde{\eta}L_{\mathcal{L}} \left(\frac{3}{2} + \frac{16\lambda^2}{\lambda^2 - 8L_{\mathcal{L}}^2} + \frac{12L_{\mathcal{L}}^2}{\lambda^2 - 8L_{\mathcal{L}}^2} \right) \geq \frac{1}{4}, \tag{A.15}$$

then one obtains

$$\mathbb{E}[\mathcal{L}(\Theta^{t+1}) - \mathcal{L}(\Theta^t)] \leq 2\tilde{\eta}\lambda^2 \zeta^2 + 3\tilde{\eta}^2 L_{\mathcal{L}} \kappa_{\mathcal{L}}^2 + \frac{64\tilde{\eta}^3 L_{\mathcal{L}}^2 (\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}{\beta^2} - \frac{\tilde{\eta}}{4} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2], \tag{A.16}$$

By reorganizing the above inequality and telescoping, one obtains

$$\frac{1}{4T} \sum_{t=1}^T \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] \leq \frac{\mathbb{E}[\mathcal{L}(\Theta^1) - \mathcal{L}(\Theta^{T+1})]}{\tilde{\eta}T} + \frac{64\tilde{\eta}^2 L_{\mathcal{L}}^2 (\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}{\beta^2} + 3\tilde{\eta}L_{\mathcal{L}}\kappa_{\mathcal{L}}^2 + 2\lambda^2 \zeta^2, \tag{A.17}$$

Let Θ^* represent the optimal solution for the objective function $\mathcal{L}(\Theta)$. Define $\Delta \triangleq \mathcal{L}(\Theta^1) - \mathcal{L}(\Theta^*)$. It's obvious that

$$\Delta \triangleq \mathcal{L}(\Theta^1) - \mathcal{L}(\Theta^*) \geq \mathbb{E}[\mathcal{L}(\Theta^1) - \mathcal{L}(\Theta^{T+1})]. \tag{A.18}$$

The upper bound can be discussed in two cases: 1) if $\tilde{\eta}_o \geq \min \left\{ \sqrt{\frac{\Delta}{3TL_{\mathcal{L}}\kappa_{\mathcal{L}}^2}}, \sqrt[3]{\frac{\beta^2 \Delta}{64TL_{\mathcal{L}}^2(\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}} \right\} \triangleq c_o$, choosing $\tilde{\eta} = c_o$ implies

$$\frac{1}{4T} \sum_{t=1}^T \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] \leq 2\sqrt[3]{\frac{64\Delta^2 L_{\mathcal{L}}^2 (\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}{\beta^2 T^2}} + 2\sqrt{\frac{3\Delta L_{\mathcal{L}} \kappa_{\mathcal{L}}^2}{T}} + 2\lambda^2 \zeta^2. \tag{A.19}$$

2) if $\tilde{\eta}_o \leq c_o$, choosing $\tilde{\eta} = \tilde{\eta}_o$ implies

$$\frac{1}{4T} \sum_{t=1}^T \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] \leq \frac{\Delta}{\tilde{\eta}_0 T} + \sqrt[3]{\frac{64\Delta^2 L_{\mathcal{L}}^2 (\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}{\beta^2 T^2}} + \sqrt{\frac{3\Delta L_{\mathcal{L}} \kappa_{\mathcal{L}}^2}{T}} + 2\lambda^2 \zeta^2. \quad (\text{A.20})$$

Therefore, combining (A.19) and (A.20) implies that

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2] = \mathbb{E}[\|\nabla \mathcal{L}(\Theta^{\tilde{t}})\|^2] \leq \mathcal{O}\left(\frac{\Delta}{\tilde{\eta}_0 T} + \sqrt[3]{\frac{\Delta^2 L_{\mathcal{L}}^2 (\lambda^2 \zeta^2 + \kappa_{\mathcal{L}}^2)}{\beta^2 T^2}} + \sqrt{\frac{\Delta L_{\mathcal{L}} \kappa_{\mathcal{L}}^2}{T}} + \lambda^2 \zeta^2\right), \quad (\text{A.21})$$

where $\tilde{t}$ is sampled uniformly from the index set $\{1, \dots, T\}$.

Next, one obtains

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\hat{\theta}_i(\Theta_t) - \Theta_t\|^2] &\leq \frac{1}{N} \sum_{i=1}^N 2\left(\mathbb{E}[\|\hat{\theta}_i(\Theta_t) - \theta^*(\Theta_t)\|^2] + \mathbb{E}[\|\Theta_t - \theta^*(\Theta_t)\|^2]\right) \\ &\leq 2\zeta^2 + \frac{2}{N} \sum_{i=1}^N \frac{\mathbb{E}[\|\nabla \mathcal{L}_i(\Theta_t)\|^2]}{\lambda^2} \\ &= 2\zeta^2 + \frac{2}{N} \sum_{i=1}^N \frac{\mathbb{E}[\|\nabla \mathcal{L}_i(\Theta_t) - \nabla \mathcal{L}(\Theta_t)\|^2 + \|\nabla \mathcal{L}(\Theta_t)\|^2]}{\lambda^2} \\ &\leq 2\zeta^2 + \frac{2\kappa_{\mathcal{L}}^2}{\lambda^2} + \frac{2}{\lambda^2 - 8L_{\mathcal{L}}^2} \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2], \end{aligned} \quad (\text{A.22})$$

In the 1st inequality, Jensen's inequality is employed, while in the 2nd inequality, the ζ -approximation and Eq. (13) is utilized. The third line utilizes the fact that $\frac{1}{N} \sum_{i=1}^N \|x_i\|^2 = \|\bar{x}\|^2 + \frac{1}{N} \sum_{i=1}^N \|x_i - \bar{x}\|^2$, and the final inequality is derived from the application of Lemma 4.

By taking the average over t from 1 to T for Eq. (A.22), one obtains:

$$\frac{1}{TN} \sum_{t=1}^T \sum_{i=1}^N \mathbb{E}[\|\hat{\theta}_i(\Theta_t) - \Theta_t\|^2] \leq 2\zeta^2 + \frac{2\kappa_{\mathcal{L}}^2}{\lambda^2} + \frac{2}{\lambda^2 - 8L_{\mathcal{L}}^2} \frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\nabla \mathcal{L}(\Theta^t)\|^2]. \quad (\text{A.23})$$

Together with Eq. (A.20), the proof is finished with

$$\frac{1}{N} \sum_{i=1}^N \mathbb{E}[\|\hat{\theta}_i(\Theta_{\tilde{t}}) - \Theta_{\tilde{t}}\|^2] \leq \mathcal{O}\left(\zeta^2 + \frac{\kappa_{\mathcal{L}}^2}{\lambda^2}\right) + \mathcal{O}\left(\mathbb{E}[\|\nabla \mathcal{L}(\Theta^{\tilde{t}})\|^2]\right). \quad (\text{A.24})$$

A.5. Proof of important lemmas

A.5.1. Proof of Lemma 1

Define

$$\begin{aligned} \delta_F(\theta_i, s') &= \sum_{s_i \in \mathcal{F}(s'_i)} F_{\theta_i}(s_i \rightarrow s'_i) - R(s'_i) - \sum_{s_i \in \mathcal{D}(s'_i)} F_{\theta_i}(s'_i \rightarrow s_i), \\ \delta_{\nabla F}(\theta_i, s') &= \sum_{s_i \in \mathcal{F}(s'_i)} \nabla_{\theta_i} F_{\theta_i}(s_i \rightarrow s'_i) - \sum_{s_i \in \mathcal{D}(s'_i)} \nabla_{\theta_i} F_{\theta_i}(s'_i \rightarrow s_i). \end{aligned} \quad (\text{A.25})$$

For each $\theta_i, \theta'_i \in \mathbb{R}^d$, using triangle inequality yields

$$\begin{aligned} &\|\nabla \mathcal{L}_i(\theta_i) - \nabla \mathcal{L}_i(\theta'_i)\| \\ &= \mathbb{E}_{\tau_i \sim p_i(\tau)} \left[\sum_{s'_i \in \tau_i \neq s_0} \left(\delta_F(\theta_i, s') \delta_{\nabla F}(\theta_i, s') - \delta_F(\theta'_i, s') \delta_{\nabla F}(\theta'_i, s') \right) \right] \\ &= \mathbb{E}_{\tau_i \sim p_i(\tau)} \left[\sum_{s'_i \in \tau_i \neq s_0} \left(\delta_F(\theta_i, s') \delta_{\nabla F}(\theta_i, s') - \delta_F(\theta_i, s') \delta_{\nabla F}(\theta'_i, s') + \delta_F(\theta_i, s') \delta_{\nabla F}(\theta'_i, s') \right. \right. \\ &\quad \left. \left. - \delta_F(\theta'_i, s') \delta_{\nabla F}(\theta'_i, s') \right) \right] \\ &\leq \mathbb{E}_{\tau_i \sim p_i(\tau)} \left[\sum_{s'_i \in \tau_i \neq s_0} \left(\|\delta_F(\theta_i, s')\| \cdot \|\delta_{\nabla F}(\theta_i, s') - \delta_{\nabla F}(\theta'_i, s')\| \right) \right] \end{aligned}$$

$$+ \left\| \left[\delta_F(\theta_i, s') - \delta_F(\theta'_i, s') \right] \delta_{\nabla F}(\theta'_i, s') \right\|. \quad (\text{A.26})$$

Under Assumptions 2, 3 and 4, then

$$\|\delta_F(\theta_i, s')\| \leq H_0 + 2H_2B, \quad (\text{A.27})$$

$$\|\delta_{\nabla F}(\theta_i, s') - \delta_{\nabla F}(\theta'_i, s')\| \leq 2H_2L_1\|\theta_i - \theta'_i\|, \quad (\text{A.28})$$

$$\|\delta_F(\theta_i, s') - \delta_F(\theta'_i, s')\| \leq 2H_2L_0\|\theta_i - \theta'_i\|, \quad (\text{A.29})$$

$$\|\delta_{\nabla F}(\theta'_i, s')\| \leq 2H_2L_0. \quad (\text{A.30})$$

Combining (A.27)-(A.30) with (A.26), one obtains

$$\begin{aligned} \|\nabla \mathcal{L}_i(\theta_i) - \nabla \mathcal{L}_i(\theta'_i)\| &\leq H_1 \left[(H_0 + 2H_2B)2H_2L_1\|\theta_i - \theta'_i\| + 2H_2L_0\|\theta_i - \theta'_i\|2H_2L_0 \right] \\ &= H_1 \left[(H_0 + 2H_2B)2H_2L_1 + 4H_2^2L_0^2 \right] \|\theta_i - \theta'_i\|. \end{aligned} \quad (\text{A.31})$$

A.5.2. Proof of Lemma 2

See the proof in Corollary 3.4 b) of [14].

A.5.3. Proof of Lemma 3

By using Jensen's inequality, the smoothness of $\nabla \mathcal{L}_i$ and ζ -approximation between $\hat{\theta}_i(\Theta_{i,r}^t)$ and $\theta_i^*(\Theta_{i,r}^t)$, one obtains

$$\begin{aligned} \mathbb{E} \left[\|\mathbf{g}_{i,r}^t - \nabla \mathcal{L}_i(\Theta^t)\|^2 \right] &\leq 2\mathbb{E} \left[\|\mathbf{g}_{i,r}^t - \nabla \mathcal{L}_i(\Theta_{i,r}^t)\|^2 + \|\nabla \mathcal{L}_i(\Theta_{i,r}^t) - \nabla \mathcal{L}_i(\Theta^t)\|^2 \right] \\ &\leq 2\lambda^2 \mathbb{E} \left[\|\hat{\theta}_i(\Theta_{i,r}^t) - \theta_i^*(\Theta_{i,r}^t)\|^2 \right] + 2L_{\mathcal{L}}^2 \mathbb{E} \left[\|\Theta_{i,r}^t - \Theta^t\|^2 \right] \\ &\leq 2\lambda^2 \zeta^2 + 2L_{\mathcal{L}}^2 \mathbb{E} \left[\|\Theta_{i,r}^t - \Theta^t\|^2 \right]. \end{aligned} \quad (\text{A.32})$$

Next, the upper bound of $\mathbb{E} \left[\|\Theta_{i,r}^t - \Theta^t\|^2 \right]$ is given. By replacing $\Theta_{i,r}^t$ with its update rule (15), and applying Young's inequality and Jensen's inequality, the following inequality can be established for any q :

$$\begin{aligned} \mathbb{E} \left[\|\Theta_{i,r}^t - \Theta^t\|^2 \right] &= \mathbb{E} \left[\|\Theta_{i,r-1}^t - \eta \mathbf{g}_{i,r-1}^t - \Theta^t\|^2 \right] \\ &\leq \left(1 + \frac{1}{q} \right) \mathbb{E} \left[\|\Theta_{i,r-1}^t - \Theta^t\|^2 \right] + (1+q)\eta^2 \mathbb{E} \left[\|\mathbf{g}_{i,r-1}^t\|^2 \right] \\ &\leq \left(1 + \frac{1}{q} \right) \mathbb{E} \left[\|\Theta_{i,r-1}^t - \Theta^t\|^2 \right] + 2(1+q)\eta^2 \mathbb{E} \left[\|\nabla \mathcal{L}_i(\Theta^t)\|^2 \right] \\ &\quad + 2(1+q)\eta^2 \mathbb{E} \left[\|\mathbf{g}_{i,r-1}^t - \nabla \mathcal{L}_i(\Theta^t)\|^2 \right]. \end{aligned} \quad (\text{A.33})$$

Incorporating Eq. (A.32) into (A.33), one obtains

$$\begin{aligned} &\mathbb{E} \left[\|\Theta_{i,r}^t - \Theta^t\|^2 \right] \\ &\leq \left(1 + \frac{1}{q} \right) \mathbb{E} \left[\|\Theta_{i,r-1}^t - \Theta^t\|^2 \right] + 2(1+q)\eta^2 \mathbb{E} \left[\|\nabla \mathcal{L}_i(\Theta^t)\|^2 \right] + 4(1+q)\eta^2 (\lambda^2 \zeta^2 + L_{\mathcal{L}}^2 \mathbb{E} \left[\|\Theta_{i,r-1}^t - \Theta^t\|^2 \right]) \\ &= \left(1 + \frac{1}{q} + 4(1+q)\eta^2 L_{\mathcal{L}}^2 \right) \mathbb{E} \left[\|\Theta_{i,r-1}^t - \Theta^t\|^2 \right] + 2(1+q)\eta^2 \mathbb{E} \left[\|\nabla \mathcal{L}_i(\Theta^t)\|^2 \right] + 4(1+q)\eta^2 \lambda^2 \zeta^2. \end{aligned} \quad (\text{A.34})$$

Let $q = R$. Under the condition that $\tilde{\eta}^2 \leq \frac{\beta^2 R}{4(1+R)L_{\mathcal{L}}^2}$, it follows that $4(1+R)\eta^2 L_{\mathcal{L}}^2 = 4(1+R)L_{\mathcal{L}}^2 \frac{\tilde{\eta}^2}{\beta^2 R^2} \leq \frac{1}{R}$. Subsequently

$$\begin{aligned} \mathbb{E} \left[\|\Theta_{i,r}^t - \Theta^t\|^2 \right] &\leq \left(1 + \frac{2}{R} \right) \mathbb{E} \left[\|\Theta_{i,r-1}^t - \Theta^t\|^2 \right] + \frac{4\tilde{\eta}^2}{\beta^2 R} \mathbb{E} \left[\|\nabla \mathcal{L}_i(\Theta^t)\|^2 \right] + \frac{8\tilde{\eta}^2 \lambda^2 \zeta^2}{\beta^2 R} \\ &\leq \frac{4\tilde{\eta}^2}{\beta^2 R} \left(\mathbb{E} \left[\|\nabla \mathcal{L}_i(\Theta^t)\|^2 \right] + 2\lambda^2 \zeta^2 \right) \sum_{r=0}^{R-1} \left(1 + \frac{2}{R} \right)^r \\ &\leq \frac{16\tilde{\eta}^2}{\beta^2} \left(\mathbb{E} \left[\|\nabla \mathcal{L}_i(\Theta^t)\|^2 \right] + 2\lambda^2 \zeta^2 \right), \end{aligned} \quad (\text{A.35})$$

where the last inequality applies that $(1 + a/n)^n \leq e^a$ for all $a \in \mathbb{R}$ and $n \in \mathbb{N}$ and $\sum_{r=0}^{R-1} \left(1 + \frac{2}{R} \right)^r = \frac{(1+2/R)^R - 1}{2/R} \leq \frac{e^2 - 1}{2/R} < 4R$. By combining equations (A.32) and (A.35), the proof is completed.

A.5.4. Proof of Lemma 4

Since the gradient of $\mathcal{L}_i(\Theta)$ is $\nabla \mathcal{L}_i(\Theta) = \lambda(\Theta - \theta_i^*(\Theta))$, then

$$\left\| \nabla \mathcal{L}_i(\Theta) - \nabla \mathcal{L}(\Theta) \right\|^2 = \left\| \lambda(\Theta - \theta_i^*(\Theta)) - \frac{1}{N} \sum_{k=1}^N \lambda(\Theta - \theta_k^*(\Theta)) \right\|^2 \quad (\text{A.36})$$

The first-order optimal condition of Eq. (10) implies that $\nabla \mathcal{L}_i(\theta_i^*(\Theta)) - \lambda(\Theta - \theta_i^*(\Theta)) = 0$. Together with Jensen's inequality, it follows that

$$\left\| \nabla \mathcal{L}_i(\Theta) - \nabla \mathcal{L}(\Theta) \right\|^2 = \left\| \nabla \mathcal{L}_i(\theta_i^*(\Theta)) - \frac{1}{N} \sum_{j=1}^N \nabla \mathcal{L}_j(\theta_j^*(\Theta)) \right\|^2 \quad (\text{A.37})$$

$$\leq 2 \left\| \nabla \mathcal{L}_i(\theta_i^*(\Theta)) - \frac{1}{N} \sum_{j=1}^N \nabla \mathcal{L}_j(\theta_i^*(\Theta)) \right\|^2 \quad (\text{A.38})$$

$$+ 2 \left\| \frac{1}{N} \sum_{j=1}^N \nabla \mathcal{L}_j(\theta_i^*(\Theta)) - \frac{1}{N} \sum_{j=1}^N \nabla \mathcal{L}_j(\theta_j^*(\Theta)) \right\|^2.$$

By taking the average over N tasks, one obtains

$$\frac{1}{N} \sum_{i=1}^N \left\| \nabla \mathcal{L}_i(\Theta) - \nabla \mathcal{L}(\Theta) \right\|^2 \leq 2\kappa_2^2 + \frac{2}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left\| \nabla \mathcal{L}_j(\theta_i^*(\Theta)) - \nabla \mathcal{L}_j(\theta_j^*(\Theta)) \right\|^2 \quad (\text{A.39})$$

$$\leq 2\kappa_2^2 + \frac{2L_\ell^2}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left\| \theta_i^*(\Theta) - \theta_j^*(\Theta) \right\|^2 \quad (\text{A.40})$$

$$\leq 2\kappa_2^2 + \frac{2L_\ell^2}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left[2 \left\| \theta_i^*(\Theta) - \Theta \right\|^2 + 2 \left\| \theta_j^*(\Theta) - \Theta \right\|^2 \right] \quad (\text{A.41})$$

$$= 2\kappa_2^2 + \frac{4L_\ell^2}{N^2 \lambda^2} \sum_{i=1}^N \sum_{j=1}^N \left[\left\| \nabla \mathcal{L}_i(\Theta) \right\|^2 + \left\| \nabla \mathcal{L}_j(\Theta) \right\|^2 \right] \quad (\text{A.42})$$

$$= 2\kappa_2^2 + \frac{8L_\ell^2}{N \lambda^2} \sum_{i=1}^N \left\| \nabla \mathcal{L}_i(\Theta) \right\|^2 \quad (\text{A.43})$$

$$= 2\kappa_2^2 + \frac{8L_\ell^2}{\lambda^2} \left[\frac{1}{N} \sum_{i=1}^N \left\| \nabla \mathcal{L}_i(\Theta) - \nabla \mathcal{L}(\Theta) \right\|^2 + \left\| \nabla \mathcal{L}(\Theta) \right\|^2 \right] \quad (\text{A.44})$$

In the inequality (A.40), assumption (17) is employed, along with Jensen's inequality. The inequality (A.41) is derived from Lemma 1. The (A.42) inequality is obtained by applying Jensen's inequality again. The equality (A.40) is based on the relationship $\nabla \mathcal{L}_i(\Theta) = \lambda(\Theta - \theta_i^*(\Theta))$, while the equality (A.41) follows from the fact that $\frac{1}{N} \sum_{i=1}^N \|x_i\|^2 = \|\bar{x}\|^2 + \frac{1}{N} \sum_{i=1}^N \|x_i - \bar{x}\|^2$, where $\bar{x} = \frac{1}{N} \sum_{i=1}^N x_i$. When $\lambda > 2\sqrt{2}L_\ell$, reorganizing the above inequality yields

$$\frac{1}{N} \sum_{i=1}^N \left\| \nabla \mathcal{L}_i(\Theta) - \nabla \mathcal{L}(\Theta) \right\|^2 \leq \frac{2\lambda^2 \kappa_2^2}{\lambda^2 - 8L_\ell^2} + \frac{8L_\ell^2}{\lambda^2 - 8L_\ell^2} \left\| \nabla \mathcal{L}(\Theta) \right\|^2. \quad (\text{A.45})$$

References

- [1] A. Abbaszadeh Shahri, C. Shan, S. Larsson, A novel approach to uncertainty quantification in groundwater table modeling by automated predictive deep learning, *Nat. Resour. Res.* 31 (2022) 1351–1373, <https://doi.org/10.1007/s11053-022-10051-w>.
- [2] Z. Abo-Hammour, O. Abu Arqub, S. Momani, N. Shawagfeh, et al., Optimization solution of Troesch's and Bratu's problems of ordinary type using novel continuous genetic algorithm, *Discrete Dyn. Nat. Soc.* 2014 (2014), <https://doi.org/10.1155/2014/401696>.
- [3] O.A. Arqub, Z. Abo-Hammour, Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm, *Inf. Sci.* 279 (2014) 396–415, <https://doi.org/10.1016/j.ins.2014.03.128>.
- [4] M. Baucum, A. Khojandi, C. Myers, L. Kessler, Optimizing substance use treatment selection using reinforcement learning, *ACM Trans. Manag. Inf. Syst.* 14 (2023) 1–30, <https://doi.org/10.1145/3563778>.
- [5] E. Bengio, M. Jain, M. Korablyov, D. Precup, Y. Bengio, Flow network based generative models for non-iterative diverse candidate generation, in: *Advances in Neural Information Processing Systems*, 2021.
- [6] Y. Bengio, S. Lahlou, T. Deleu, E.J. Hu, M. Tiwari, E. Bengio, Gflownet foundations, *J. Mach. Learn. Res.* 24 (2023) 1–55, <http://jmlr.org/papers/v24/22-0364.html>.
- [7] Z. Chen, V. Badrinarayanan, C.Y. Lee, A. Rabinovich, Gradnorm: gradient normalization for adaptive loss balancing in deep multitask networks, in: *International Conference on Machine Learning*, PMLR, 2018, pp. 794–803.
- [8] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, J. Peters, et al., Sharing knowledge in multi-task deep reinforcement learning, in: *International Conference on Learning Representations*, 2020, pp. 1–11.

- [9] Y. Duan, X. Chen, R. Houthoofd, J. Schulman, P. Abbeel, Benchmarking deep reinforcement learning for continuous control, in: *International Conference on Machine Learning*, PMLR, 2016, pp. 1329–1338.
- [10] A. Ez-Zizi, S. Farrell, D. Leslie, G. Malhotra, C.J. Ludwig, Reinforcement learning under uncertainty: expected versus unexpected uncertainty and state versus reward uncertainty, *Comput. Brain Behav.* (2023) 1–25, <https://doi.org/10.1007/s42113-022-00165-y>.
- [11] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: *International Conference on Machine Learning*, PMLR, 2017, pp. 1126–1135.
- [12] M. Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, *J. Am. Stat. Assoc.* 32 (1937) 675–701, <https://api.semanticscholar.org/CorpusID:120581754>.
- [13] A. Gupta, R. Mendonca, Y. Liu, P. Abbeel, S. Levine, Meta-reinforcement learning of structured exploration strategies, in: *Advances in Neural Information Processing Systems*, 2018.
- [14] T. Hoheisel, M. Laborde, A. Oberman, A regularization interpretation of the proximal point method for weakly convex functions, *J. Dyn. Games* 7 (2020) 79, <https://doi.org/10.3934/jdg.2020005>.
- [15] S.A. Hosseini, A. Abbaszadeh Shahri, R. Asheghi, Prediction of bedload transport rate using a block combined network structure, *Hydrol. Sci. J.* 67 (2022) 117–128, <https://doi.org/10.1080/02626667.2021.2003367>.
- [16] M. Jain, E. Bengio, A. Hernandez-Garcia, J. Rector-Brooks, B.F. Dossou, C.A. Ekbote, J. Fu, T. Zhang, M. Kilgour, D. Zhang, et al., Biological sequence design with GFlowNets, in: *International Conference on Machine Learning*, PMLR, 2022, pp. 9786–9801.
- [17] M. Jain, T. Deleu, J. Hartford, C.H. Liu, A. Hernandez-Garcia, Y. Bengio, Gflownets for ai-driven scientific discovery, *Digit. Discov.* (2023), <https://doi.org/10.1039/d3dd00002h>.
- [18] L.P. Kaelbling, M.L. Littman, A.W. Moore, Reinforcement learning: a survey, *J. Artif. Intell. Res.* 4 (1996) 237–285.
- [19] A. Kumari, S. Tanwar, A reinforcement-learning-based secure demand response scheme for smart grid system, *IEEE Int. Things J.* 9 (2021) 2180–2191, <https://doi.org/10.1109/JIOT.2021.3090305>.
- [20] K. Lee, Y. Seo, S. Lee, H. Lee, J. Shin, Context-aware dynamics model for generalization in model-based reinforcement learning, in: *International Conference on Machine Learning*, PMLR, 2020, pp. 5757–5766.
- [21] Y. Li, S. Luo, H. Wang, J. Hao, Cflownets: continuous control with generative flow networks, in: *International Conference on Learning Representations*, 2023.
- [22] O. Lockwood, M. Si, A review of uncertainty for deep reinforcement learning, in: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2022, pp. 155–162.
- [23] K. Madan, J. Rector-Brooks, M. Korablyov, E. Bengio, M. Jain, A.C. Nica, T. Bosc, Y. Bengio, N. Malkin, Learning gflownets from partial episodes for improved convergence and stability, in: *International Conference on Machine Learning*, PMLR, 2023, pp. 23467–23483.
- [24] S.P. Mirfallah Lialestani, D. Parcerisa, M. Himi, A. Abbaszadeh Shahri, Generating 3d geothermal maps in Catalonia, Spain using a hybrid adaptive multitask deep learning procedure, *Energies* 15 (2022) 4602, <https://doi.org/10.3390/en15134602>.
- [25] A.C. Nica, M. Jain, E. Bengio, C.H. Liu, M. Korablyov, M.M. Bronstein, Y. Bengio, Evaluating generalization in GFlowNets for molecule design, in: *ICLR2022 Machine Learning for Drug Discovery*, 2022.
- [26] E. Parisotto, L.J. Ba, R. Salakhutdinov, Actor-mimic: deep multitask and transfer reinforcement learning, in: *International Conference on Learning Representations*, 2016.
- [27] N. Pu, Z. Zhong, X. Ji, N. Sebe, Federated generalized category discovery, *arXiv preprint*, arXiv:2305.14107, 2023.
- [28] N. Pu, Z. Zhong, N. Sebe, Dynamic conceptional contrastive learning for generalized category discovery, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7579–7588.
- [29] N. Pu, Z. Zhong, N. Sebe, M.S. Lew, A memorizing and generalizing framework for lifelong person re-identification, *IEEE Trans. Pattern Anal. Mach. Intell.* (2023), <https://doi.org/10.1109/TPAMI.2023.3297058>.
- [30] K. Rakelly, A. Zhou, C. Finn, S. Levine, D. Quillen, Efficient off-policy meta-reinforcement learning via probabilistic context variables, in: *International Conference on Machine Learning*, PMLR, 2019, pp. 5331–5340.
- [31] C. Rosenbaum, T. Klinger, M. Riemer, Routing networks: adaptive selection of non-linear functions for multi-task learning, in: *International Conference on Learning Representations*, ICLR, 2018.
- [32] J. Rothfuss, D. Lee, I. Clavera, T. Asfour, P. Abbeel, Promp: proximal meta-policy search, in: *International Conference on Learning Representations*, 2018, OpenReview.net.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, preprint, arXiv:1707.06347, 2017.
- [34] 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* 529 (2016) 484–489, <https://doi.org/10.1038/nature16961>.
- [35] X. Song, W. Gao, Y. Yang, K. Choromanski, A. Pacchiano, Y. Tang, Es-maml: simple Hessian-free meta learning, in: *International Conference on Learning Representations*, 2020.
- [36] B.C. Stadie, G. Yang, R. Houthoofd, X. Chen, Y. Duan, Y. Wu, P. Abbeel, I. Sutskever, Some considerations on learning to explore via meta-reinforcement learning, in: *Advances in Neural Information Processing Systems*, 2018, pp. 9280–9290.
- [37] Y. Teh, V. Bapst, W.M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, R. Pascanu, Distral: robust multitask reinforcement learning, in: *Advances in Neural Information Processing Systems*, 2017.
- [38] C. Vignon, J. Rabault, R. Vinuesa, Recent advances in applying deep reinforcement learning for flow control: perspectives and future directions, *Phys. Fluids* 35 (2023).
- [39] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., Scipy 1.0: fundamental algorithms for scientific computing in python, *Nat. Methods* 17 (2020) 261–272, <https://doi.org/10.1038/s41592-019-0686-2>.
- [40] T.L. Vuong, D.V. Nguyen, C.M. Bui, H.D. Kieu, V.C. Ta, Q.L. Tran, T.H. Le, Sharing experience in multitask reinforcement learning, in: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3642–3648.
- [41] Z. Xu, K. Wu, Z. Che, J. Tang, J. Ye, Knowledge transfer in multi-task deep reinforcement learning for continuous control, in: *Advances in Neural Information Processing Systems*, 2020, pp. 15146–15155.
- [42] R. Yang, H. Xu, Y. Wu, X. Wang, Multi-task reinforcement learning with soft modularization, in: *Advances in Neural Information Processing Systems*, 2020, pp. 4767–4777.
- [43] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, C. Finn, Gradient surgery for multi-task learning, in: *Advances in Neural Information Processing Systems*, 2020, pp. 5824–5836.
- [44] D. Zhang, R.T. Chen, N. Malkin, Y. Bengio, Unifying generative models with gflownets, *arXiv preprint*, arXiv:2209.02606, 2022.



Xinyuan Ji received B.S. and M.S. degrees in computer science and technology from Shanxi University and Shaanxi Normal University, China, in 2016 and 2019, respectively. She is currently pursuing both Ph.D. degrees at Xi'an Jiaotong University and Leiden University in computer science and technology. Her research interests include distributed computing, data mining, machine learning, and privacy & security.



Xu Zhang is an assistant professor at the School of Artificial Intelligence, Xidian University. Previously, He was a Postdoc at the State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences from 2021 to 2023. He received his B.S. degree and Ph.D. degree in Electronics Engineering from the School of Information and Electronics, Beijing Institute of Technology, Beijing, China, in 2015 and 2021. He was a visiting student in the Ming Hsieh Department of Electrical Engineering at the University of Southern California from 2018 to 2019. His research interests include federated learning, distributed optimization, compressed sensing, and inverse problems.



Wei Xi is currently an Associate Professor at the School of Computer Science and Technology, Xi'an Jiaotong University. He received his Ph.D. degree in Computer Science from Xi'an Jiaotong University in 2014. His main research interests include the Internet of Things, artificial intelligence, and network security. He is a member of CCF, ACM, and IEEE.



Haozhi Wang received his B.Eng. from Tianjin University, Tianjin, China, in 2019. He is currently working toward a Ph.D. degree in information and communication engineering with the School of Electrical and Information Engineering, Tianjin University, Tianjin, China. His research interests include reinforcement learning, distributed learning, and intelligent signal processing.



Olga Gadyatskaya is an associate professor at Leiden Institute of Advanced Computer Science, Leiden University. Prior to joining Leiden University, she has been research associate at the University of Luxembourg, Luxembourg, and a postdoc at the University of Trento, Italy. She holds a Ph.D. in Mathematics from Novosibirsk State University, Russia. Her research focuses on cybersecurity, and her current research interests include security risk management, mobile security, secure AI systems, and cybersecurity decision-making in organizations.



Yinchuan Li (Member, IEEE) received the B.S. and Ph.D. degrees in electronic engineering from the Beijing Institute of Technology (BIT), Beijing, China, in 2015 and 2020, respectively. From November 2017 to November 2019, he was a joint training Ph.D. with the Department of Electrical Engineering, Columbia University, New York, NY, USA. From December 2019 to August 2020, he was a senior technical consultant in Santé Ventures, Austin, TX, USA. He now works at Noah's Ark Lab, Huawei Technologies, Beijing, China, as a principal researcher. His current research interests include machine learning, deep learning, reinforcement learning, and sparse signal processing.

Dr. Li was the recipient of the Best Ph.D. Thesis Award of Chinese Institute of Electronics of 2022, and the IEEE International Conference on Signal, Information, and Data Processing Excellent Paper Award of 2019.