2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Imitation Learning of Diverse Expert Behaviors for Advanced Machining System Optimizations

Qinge Xiao, Zhile Yang*,Chengke Wu, Yuanjun Guo
Shenzhen Institute of Advanced Technology
Chinese Academy of Sciences
Shenzhen, China
Email: qg.xiao@siat.ac.cn; zl.yang@siat.ac.cn; ck.wu@siat.ac.cn; yj.guo@siat.ac.cn

***Abstract*—The potential intelligence behind advanced machining systems (AMSs) offers positive contributions toward process improvement. Compared with conventional meta-heuristics, imitation learning (IL) appears to provide a more powerful tool to exploit such intelligence by observing demonstrations from technologists. This paper proposes a novel IL-based policy search algorithm that equips the agent with the optimization knowledge by executing upper-level policy learning to generate an imitation policy distribution with diverse decision behaviors. The experimental results of heavy cutting scenarios show that the proposed method rather than meta-heuristics is more viable for solving AMS optimization problems.**

Keywords—Imitation learning; Policy search; Advanced machining system

I. INTRODUCTION

The advanced machining system (AMS) that relies on computer numerical control machines, was developed for fast, stable, and accurate processing in contemporary manufacturing sectors [1]. However, with the improvement of machine intelligence, AMS requires more energy consumption to support complex electrical control structures, which in return gives rise to environmental impacts and emissions [2]. Therefore, energy-efficient measures and technologies for process optimization are particularly important to maintain enterprises' business competitiveness without violating the green missions.

Even though meta-heuristic algorithms have achieved some success in the parameters optimization for AMS, obstacles still exist owing to the following reasons: (i) Meta-heuristics may suffer from the curse of dimensionality with too many conflicting objectives, while AMS often involves a large number of stakeholders and functionalities. Compared with ordinary machining systems, AMS needs to pursue the balance of various requirements more rigorously and comprehensively [3]. (ii) The performance of meta-heuristic algorithms depends strongly on the accuracy of mathematic models. However, AMS is hard to specify as it exhibits real-time and probabilistic behaviors in processing, which poses a significant challenge to formulate an appropriate theoretical model. Compounding errors will occur when learning on unreliable models and even cause serious mistakes with devastating consequences [4].

The AMS, as a typical human-computer interaction system. If intelligence can sufficiently be included with AMS optimization, the optimization processes would require almost no ad hoc mathematical modeling; instead, we can directly learn optimization from technologists' demonstrations because these data are powerful replacements for manually coding domain intelligence [5]. This specialty is likely to address the limitations of the meta-heuristics mentioned above. In this paradigm, an alternative and often more practical approach, termed imitation learning (IL), is adopted to help us reveal intelligence (knowledge) hidden in demonstration and turn it into a crucial competitive advantage [6].

IL is characteristically inclined towards solutions that approximate experts' thinking and satisfy realistic preferences. However, for most policy-based IL algorithms, demonstrations from one or multiple experts are assumed to obey the identical deciding principles and only a single behavioral policy is used to learn from these samples [7]. A recent breakthrough in IL, named generative adversarial imitation learning (GAIL), encourages the imitator to match the state-action distribution of the demonstrator [8]. Unfortunately, it retains a single policy for each task. Yet, it cannot be guaranteed that all consistently follow the same policy because human attention is limited and varies. These methods are inherently local and will suffer from mode-averaging when an agent has to learn a policy across trajectory samples with diverse decision behaviors. Motivated by this, this paper attempts to learn optimization principles from these demonstrations and help to resolve the negative impacts regarding objective-dimension failure and strong model dependence suffered by conventional meta-heuristic algorithms. The rest of the paper is organized as follows. Section II introduces the optimization problem setups. The proposed method is presented in Section III. Section IV shows the experimental results and analysis. Section V gives the conclusions.

II. PROCESS OPTIMIZATION OF ADVANCED MACHINING SYSTEM

A. Problem setups

Similar to conventional machining systems, AMS seeks to determine economically and environmentally beneficial

process parameters for multi-pass operations. Without loss of generality, multi-pass process optimization can be transformed into a static, constrained, and multi-objective optimization problem described by Eq. 1 [9].

$$\begin{cases} \underset{X}{argmin} f^q(X) = [f_1^q(X), f_2^q(X), \ldots, f_p^q(X)], X \in \Omega \\ s.t.\ h_i^q(X) = 0; g_i^q(X) < 0, i = 1,2,\ldots \end{cases} \quad (1)$$

where $X = \{(v_c^u, f_r^u, a_p^u)\}_{u \le U}$ are the decision variables: cutting speed v_c, feed rate f_r, and cutting depth a_p of U passes. $\Omega = \prod_{s=1}^{n}[a_s, b_s]$ denotes the decision space and n is the dimensionality of the decision space. The mapping $f^q : \prod_{s=1}^{n}[a_s, b_s] \to \mathbb{R}^p$ consists of p continuous conflicting objective functions $f_1^q, f_2^q, \ldots, f_p^q$ under the machining configuration q=<workpiece property w_t, machine tool m_t, cutting tool c_t>. h_i^q and g_i^q are the i-th constraints that the system must meet.

B. MDP setups

A general issue encountered in AMS that concerns significant machining dynamics due to variations of q [10]. To solve it, the problem is placed in a Markov decision process (MDP) framework which is standard for describing dynamic systems. Initially, basic notions from reinforcement learning (RL) are defined and the MDP model is represented by a tuple M_A=(S,A,P,R,γ). Action A=$\{a_t\}$ in which a_t =< v_c^t, f_r^t, a_p^t > specifies the process parameters of each pass. State S=$\{s_t\}$ where s_t=<D_t,κ_t,u_t,a_{t-1}> describes the observations, namely workpiece geometry D_t and machining allowance κ_t, the notion of the remaining time u_h, and the action selected in the previous step a_{t-1} when the agent enters an AMS. $P \in \Delta_S^{S \times A}$ is the Markovian transition kernel that gives the probability of an outcome, e.g., state s' by an action executed in state s. $R \in \mathbb{R}^{S \times A}$ is the reward function that defines the gain at each time-step and contains the information that guides the agent towards the objectives. $\gamma \in [0,1]$ is the discount factor representing how fast the contribution of the present reward decays.

In this formalism, the agent follows a stochastic policy $\pi(a|s)$: $S \times A \to [0,1]$ that represents the action choice probabilities for each state and satisfies $P(s_{t+1}|s_t) = \sum_{a_t} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)$. To justify the quality of a policy π regarding the reward R, the state-action value function is defined as $Q^\pi = \mathbb{E}_{s,a}^{\pi}[\sum_{t=0}^{+\infty} \gamma R(s_t, a_t)]$ mapping state-action pairs (s,a) to the expected discounted cumulative reward for starting in state s, taking the action a, and following the policy π afterward.

III. Proposed Method

A. Generative adversarial imitation learning (GAIL)

This study relied on GAIL, a popular model-free imitation learning framework that, similarly to the Generative Adversarial Network (GAN), produces an imitation policy. GAIL is derived from inverse reinforcement learning (IRL), which aims at imitating a policy $\pi \in \prod$ approximating the unknown expert policy π_E with an occupancy measure condition $\rho_\pi := \rho_{\pi_E}$ [11] The learning process can be formulated as an optimization problem with the following objective function:

$$\min_{\pi \in \prod} \{\psi(\rho_\pi(s,a) - \rho_{\pi_E}(s,a)) - H(\pi)\} \quad (2)$$

where $\rho_\pi(s,a) = \sum_t \gamma^t p(s_t = s, a_t = a)$ is the state-action visitation distribution defining the one-to-one correspondence between the policy and its occupancy measure. $\psi(\cdot)$ denotes distribution discrepancy of the occupancy measures. $H(\pi)$ is the regularisation of the policy.

B. Motivations for generating imitation policy

Intelligence in AMS is often vague and multi-intentioned. We then consider diverse behaviors in each expert domain and solve the problem differently, where demonstrations are assumed to be a trajectory mixture distribution, in the sense that applying GAIL is equivalent to learning a segment of the mixture distribution. To fit the trajectory distribution, we introduce the concept of an upper-level policy $\pi_\varphi(\theta)$ [12] – by defining a parameterized distribution over θ – that selects the parameters of the actual control (low-level) policy $\pi_\theta(a|s)$. A generator $G(z;\varphi): Z \times \Phi \to \Theta$ is used to transfer the distribution $z \sim \mathcal{N}(0,1)$ into an upper-level policy $\pi_\varphi(\theta)$, where Z, Φ, and Θ denote the sampled input domain, G's parameter space, and the low-level policy parameter space, respectively. The overview of GAIL and the proposed method are shown in Figure 1, in which the difference is depicted intuitively.

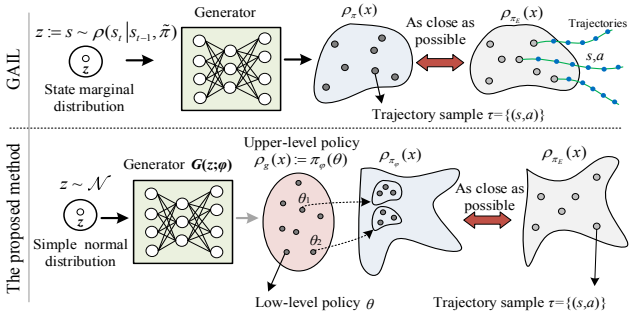

Fig. 1 Graphic representation of GAIL and proposed method

C. Learning from near-optimal demonstrations

The agent learns behaviors to advance economic interests by employing adversarial imitation. A demonstration dataset $D_{\pi_E} = \{(s_0, a_0, s_1, a_1, \ldots)|\pi_{E_i}\}$ is first constructed by aggregating all trajectories in the policy context π_{E_i}. These demonstrations are assumed to be near-optimal and are used for preliminary training of generator $G(z;\varphi)$ and discriminator $D(s,a;\omega)$. The generator takes samples $z \sim \mathcal{N}(0,1)$ from a simple Gaussian distribution and outputs an upper-level policy $\pi_\varphi(\theta)$, from which the low-level policies $\pi_\theta(a|s)$ can be generated. By executing these policies in the MDP environment, we then obtain sets of trajectories that imitate the expert behaviors. A discriminator is concurrently trained to assign each state-action pair with log$D(s,a)$, evaluating the immediate cost at the time.

1)Gradient estimator for generator

Instead of directly finding the parameters θ of the low-level policy, we attempt to find the optimal parameter vector of $G(z;\varphi)$. The benefit is that the policy distribution can directly be used to explore the parameter space. Hereby, learning an extension of GAIL with upper-level policy can be formulated as a replacement expression of Eq. 2:

$$\begin{aligned}\min_{\varphi}\max_{\omega}\mathcal{L}(\varphi,\omega)=\min_{\varphi}\max_{\omega}\mathbb{E}_{\pi_\theta\sim G(z;\varphi)}[\mathbb{E}_{s,a\sim\rho_{\pi_\theta}}\log(D(s,a;\omega))]\\+\mathbb{E}_{\rho_{\pi_E}}[\log(1-D(s,a;\omega))]-\lambda\mathbb{E}_{\pi_\theta\sim G(z;\varphi)}[H(\pi_\theta)]\end{aligned}\tag{3}$$

where $D(s,a;\omega)$ denotes the discriminator network with weights ω. The first step of Eq. 3 concludes the fitness function $J_\omega(\pi_\varphi)$ for learning the upper-level policy expanded as Eq. 4. Then, the above problem is solved by the policy-gradient method.

$$\begin{aligned}J_\omega(\pi_\varphi)&=\mathbb{E}_{\pi_\theta\sim G(z;\varphi)}[\mathbb{E}_{s,a\sim\rho_{\pi_\theta}}\log(D(s,a;\omega))]\\&=\int_\Theta\pi_\varphi(\theta)\int_S P(s|\theta)\int_A\pi_\theta(a|s)\log(D(s,a;\omega))dadsd\theta\end{aligned}\tag{4}$$

We can derive a gradient estimator for the imitation policy generator, which helps to learn the optimal distribution parameters φ. The gradient of the policy optimization has the form

$$\nabla_\varphi\mathcal{L}(\varphi)=\nabla_\varphi J(\pi_\varphi)-\lambda\nabla_\varphi\mathbb{E}_{\pi_\theta\sim G(z;\varphi)}[H(\pi_\theta)].\tag{5}$$

The optimization problem for learning upper-level policies can be reformulated as Eq. 6, where $\hat{J}(\pi_\theta)$ is updated using traditional gradient estimation with N state-action pairs in a trajectory.

$$\nabla_\varphi J(\pi_\varphi)\approx\frac{1}{M}\sum_{m=1}^{M}\nabla_{\theta^{(m)}}\hat{J}(\pi_{\theta^{(m)}})\nabla_\varphi G(\hat{z}^{(m)};\varphi);\ \hat{z}^{(m)}\sim\rho(z)\tag{6}$$

As seen from Eq. 6, the problem lies in finding a means of computing the term $\nabla_\varphi\theta=\nabla_\varphi G(z;\varphi)$. Typically, this term is calculated by using the inverse of the path $g^{-1}(\alpha,\theta)$ [13]. However, since an irreversible Multi-Layer Perceptron (MLP) was used to construct the imitation policy generator, another way of writing $\nabla_\varphi G(z;\varphi)$ is by analyzing the backpropagation of each layer.

2) Gradient estimator for discriminator

A discriminator is developed to infer the cost function underlying the demonstrated behaviors. To recover cost functions that are more robust to changes in a dynamic AMS, a discriminator that relies on energy-based models is specially developed.

The discriminator shares the same objective function as the policy generator (see Eq. 3). The negative loss is used to invert the maximization problem into one of minimization. To reduce the variance, Finn et al. [14] used a mixture distribution between the expert dataset and the policy samples (denoted as μ) to reduce the variation when the policy $\pi(a|s)$ has poor coverage over the demonstrations in the early stages of training. Then, the loss function is

$$\begin{aligned}-\mathcal{L}(\omega)&=\mathbb{E}_{\pi_\varphi}\left[\mathbb{E}_{\pi_\theta}[\log(D(s,a;\omega))]\right]+\mathbb{E}_{\pi_E}[\log(1-D(s,a;\omega))]\\&=\mathbb{E}_{\pi_\varphi}\left[\mathbb{E}_{\pi_\theta}[\log\frac{\exp\{\lambda_\omega(s,a)\}}{\exp\{\lambda_\omega(s,a)\}+\pi(a|s)}]\right]+\mathbb{E}_{\pi_E}[\log\frac{\pi(a|s)}{\exp\{\lambda_\omega(s,a)\}+\pi(a|s)}]\\&=\mathbb{E}_{\pi_\varphi}\left[\mathbb{E}_{\pi_\theta}\{\lambda_\omega(s,a)\}\right]+\mathbb{E}_{\pi_E}[\log\pi(a|s)]-2\mathbb{E}_\mu[\log(\exp\{\lambda_\omega(s,a)\}+\pi(a|s))]\end{aligned}\tag{7}$$

To find the optimal weighting ω, the policy gradient method is applied to minimize the objective $-\mathcal{L}(\omega)$. Taking the derivative with respect to ω, we have:

$$\begin{aligned}\nabla_\omega\mathcal{L}(\omega)&=-\mathbb{E}_{\pi_\varphi}\left[\mathbb{E}_{\pi_\theta}\{\nabla_\omega\lambda_\omega(s,a)\}\right]+\mathbb{E}_\mu[\frac{\exp\{\lambda_\omega(s,a)\}\nabla_\omega\lambda_\omega(s,a)}{\frac{1}{2}\exp\{\lambda_\omega(s,a)\}+\frac{1}{2}\pi(a|s)}]\\&=\frac{1}{M\times N}[\sum_{m=1}^{M}\sum_{n=1}^{N}-\nabla_\omega\lambda_\omega(s_{mn},a_{mn})+\sum_{o=1}^{M\times N}\frac{\exp\{\lambda_\omega(s_o,a_o)\}\nabla_\omega\lambda_\omega(s_o,a_o)}{\frac{1}{2}\exp\{\lambda_\omega(s_o,a_o)\}+\frac{1}{2}\pi(a_o|s_o)}]\end{aligned}\tag{8}$$

where $\nabla_\omega\lambda_\omega(s,a)$ is easily calculated through automatic derivation by the neural network.

IV. Experimental Results and Discussions

The proposed is tested by two kinds of experiments. First, the application experiments in a practical AMS environment are conducted to illustrate the effectiveness of the proposed method. Second, computational experiments focusing on the performance of policy imitation.

A. Experimental configurations

For the application experiments, different machine tools, cutting tools, and workpieces were used to design two AMS scenarios. Table 1 lists the specifications of the primary components, and all of the data were obtained in a real workshop [15].

The maximum mean discrepancy (MMD) was used to distinguish the distance between the learned trajectory (distributions) and the demonstration distributions. The closer the MMD approached zero, the higher the similarity with the expert samples. Additionally, three noise factors, namely the tool wear, cutting temperature, and machine tool chatter, were investigated. These factors reflect the system's stability. The heavy cutting processes associated with the measuring devices are depicted in Figure 2.

Fig. 2: Machining processes and data measurement setup

The computational experiments were conducted in Python 3.6.3 using an Intel 3.40 GHz PC with 16 GB of RAM. The algorithm parameters and neural networks were rigorously set by the trial-and-error method. The upper-level policy was parameterized in a feed-forward MLP with an 8-64(Relu)-128(Relu)-256(Relu)-1312(Tanh) structure. All of the low-level policy networks followed the same 6(Tanh)-32(Tanh)-3 structure. The discriminator network was arranged as a 9(Tanh)-64(Tanh)-1 structure. We selected the Adam optimizer to train the gradient-based generator and discriminator. The Adam optimizer was characterized by the $\{\xi_1, \xi_2, l_p, \psi\}$ parameters, where ξ_1 and ξ_2 denote two exponential decay rates, l_p denotes the learning rates, and ψ avoids division by zero. For the generator, we set ξ_1=0.0, ξ_2 = 0.9, l_p=5e-4, and ψ=1e-8. For the discriminator, the three parameters were changed to ξ_1=0.9, ξ_2=0.99, and l_p=3e-4. M=150 low-level policies were sampled each time the generator was trained. The IL agent traversed 25 parallel environments to collect 100 trajectories (solutions) for learning human behaviors from 500 demonstrations. For training the networks, the batch size was set to 32.

Table 1: Machining configurations for AMS

Item	Value	
	Task 1	Task 2
Machine tool	C2-6150HK/1	CHK560
Max cutting force	1.8	1.0
Max power	9500	9000
Standby power	945	860
Cutting tool	CNMG120408	ENMG120408
Rake angle	10	9
Tool clearance	4	11
Cutting edge inclination	9	3
Circle radius	0.3	0.7
Max tool life	30	55
Wokpiece	Shaft	Shaft
Material	40CrNiMnA	HT300
Diameter	80	30
Air-cutting length	15	15
Cutting length	100	80
Coolant condition	Water-based	Water-based
Requirements	-	-
Machining allowance	15	10
Cutting speed	[50,120]	[40,100]
Feed rate	[0.06,0.500]	[0.06,0.400]
Surface roughness	2.5	2.5

B. *Application experiments*

We conducted physical tests in a real AMS environment to demonstrate the efficacy of learning from demonstrations. Four state-of-the-art meta-heuristic approaches, including NSGA-III, GDE3, SMPSO, and MOEA/D-IEpsilon were also investigated as references. The programming code and algorithm descriptions are available online (https://github.com/jMetal/jMetalPy). Two traditional metrics, namely the time Tp and cost Cp, were observed and these metrics exist in the fitness functions of algorithms in the form of the mathematical model reported by Xiao et al. (2021). Additionally, we tested the CPU runtime in Phase III, which represented the computational efficiency of GPoLI-PI during application. Table 2 lists the SEC, Cp, and Tp values and the CPU runtime obtained for the five algorithms when applied to solve two task instances. The best and worst values of each metric are marked in bold.

Table 2 clearly shows that all the solutions generated by the meta-heuristics produced smaller SECs since larger parameters were chosen in the heavy cutting processes. In Task 1, e.g., the SEC of SMPSO was 8.51% lower than that of GPoLI-PI. Although the meta-heuristics improve energy efficiency, the large parameters will cause severe tool wear, extending the tool change time, and thereby increasing the total processing time. Under the optimal schemes, the Tp of GPoLI-PI could be decreased by up to 11.27% (compared to SMPSO in Task 1). Likewise, the tool change cost would increase as sharply as the increase in tool change time. Despite this, the cost increase caused by tool changing does not dominate the total cost, which would thus decrease when a large MRR is selected. From the data, the Cp of GDE3 was 7.65% lower than that of GPoLI-PI. Besides, meta-heuristics must conduct exploration and exploitation online, resulting in a reduced convergence rate. The computational efficiency of GPoLI-PI could be improved by up to 58.91% (compared to NSGAIII in Task 2).

In addition to traditional metrics, we also investigated the noise factors which reflect the robustness of AMS. The test indicators include the vibration signals (Figure 3(a)) representing the machine tool environment, temperature signals (Figure 3(b)) representing the workpiece environment, and tool wear (Figure 3(c)) representing the tool environment. As the figures show, it is difficult to add the noise factors into the models mathematically; therefore, their adverse effects on processing stability cannot be considered during optimization. This makes the algorithm converge to a larger parameter combination and increases the risk of workpiece machining failure. Overall, meta-heuristics demonstrate little preponderance when operating on most test problems due to both noise factors and traditional economic metrics. This adverse effect cannot be alleviated by increasing the optimization time or performance of the meta-heuristic algorithms.

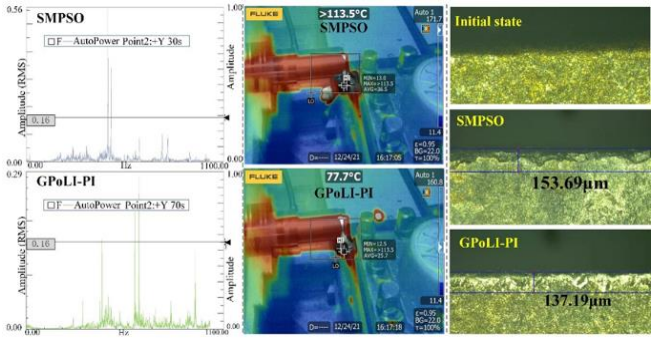

Fig.3 Vibration (a), temperature (b), and tool wear (c) measurements of SMPSO and GPoLI-PI

Table 2 Quantitative metrics of proposed algorithm and meta-heuristics during 50 runs

Task	Metric	GPoLI-PI	NSGAIII	GDE3	SMPSO	MOEA/D-IEpsilon
1	SEC	**21.495±5.126**	20.372±5.754	19.764±4.680	**19.665±4.651**	19.986±5.098
	T_p	**340.866±24.876**	392.422±27.981	387.126±23.775	**383.864±21.932**	390.763±25.289
	C_p	**2.575±0.293**	2.415±0.432	**2.378±0.199**	2.384±0.217	2.392±0.426
	CPU time	**53.010±6.5674**	**91.487±9.0556**	63.933±14.685	71.143±11.212	88.542±6.597
2	SEC	**13.127±4.078**	13.087±4.187	**11.542±3.865**	11.544±3.173	11.765±4.264
	T_p	**287.532±25.876**	319.964±24.085	319.096±18.763	**317.856±20.986**	323.218±24.165
	C_p	**2.094±0.182**	2.043±0.293	1.996±0.132	**1.975±0.129**	1.984±0.115
	CPU time	**20.356±4.0732**	**49.536±10.037**	28.155±15.221	38.162±9.836	47.697±5.917

C. Computational experiments

In this section, we evaluate the performance of GPoLI-PI based on several comparative tests on two continuous AMS tasks of varying scales and divergent expert demonstrations in the training phase. For the given tasks, the imitation policy was optimized using 50, 100, and 150 trajectories. The divergence for the multidimensional trajectory distributions was estimated via maximum mean discrepancy, and the K-nearest neighbor algorithm was used to classify the extent of divergence as small (S), middle (M), or large (L). Given the preliminary level of experiments and the absence of various state-of-the-art works on the recent generative policy distribution-based IL, we adopted three baseline algorithms, as introduced below.

Baseline I (GAIL): This is a framework to extract a single policy from diverse demonstrations, which matches the state-action distribution of the learned policy to that of the expert policy. The test validates using the policy distribution against the single policy to fit the demonstrations.

Baseline II (GPowD): The conventional neural network $D_\omega : S \times A \to (0,1)$ with weights ω is used to construct a discriminator. The remainder, including generator training and policy-manifold learning, follow the same steps as GPoLI-PI. It illustrates the effectiveness of the energy-based discriminator.

Table 3 Algorithm comparisons

Algorithms	Scenarios	Task 1			Task 2		
		50	150	300	50	150	300
GPoLI-PI	S	0.805±0.047	0.681±0.044	0.634±0.015	0.981±0.096	0.837±0.071	0.815±0.028
	M	1.105±0.076	0.729±0.053	0.617±0.017	1.499±0.127	0.983±0.088	0.875±0.034
	L	1.116±0.057	0.735±0.048	0.630±0.031	1.406±0.135	0.986±0.089	0.852±0.072
GAIL	S	0.894±0.180	0.736±0.254	0.758±0.217	2.358±0.141	1.895±0.132	2.061±0.106
	M	1.917±0.184	1.343±0.296	1.482±0.230	2.576±0.208	2.130±0.194	2.386±0.127
	L	2.126±0.441	1.678±0.417	1.703±0.398	2.893±0.217	2.762±0.236	2.774±0.204
GPowD	S	0.781±0.053	0.685±0.046	0.659±0.038	0.994±0.096	1.023±0.077	0.978±0.042
	M	0.994±0.064	0.786±0.058	0.783±0.022	1.498±0.087	1.047±0.086	1.052±0.060
	L	1.086±0.063	0.781±0.076	0.787±0.046	1.087±0.143	1.065±0.105	0.987±0.097

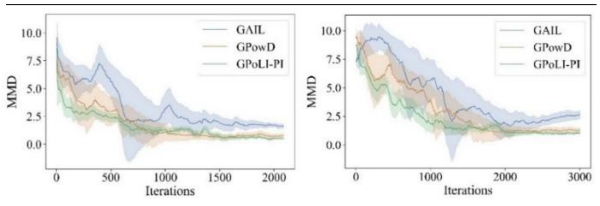

Fig.4 Convergence plots of task 1 and task 2

It follows from Table 3 and Fig.4 that our method with an energy-based discriminator may converge in under 1500 and 2000 interactions with the environment of Tasks 1 and 2, respectively. Compared to GPowD, which uses the same policy network but a conventional discriminator, GPoLI-PI is superior in imitation accuracy and stability even though it takes 10%~15% more learning time. In contrast, GAIL is more likely to converge to the local optimum slowly and may suffer discriminator degeneration in complex tasks. This means that GAIL is limited to a mode-average regression owing to the forced usage of a single policy to map the varied expert trajectories.

D. Visualisation of trajectory distributions

To illustrate how the GPoLI-PI algorithm performs behavior imitation in Phase I and applies it to the Pareto-policy manifold learning processes, the evolutionary trajectory/policy distributions are visualized as iterations (see Figure 5). Given that the above two distributions are high-dimensional, t-distributed stochastic neighbor embedding (t-SNE) is used to reduce them to two dimensions and regulate the value range to [0,1].

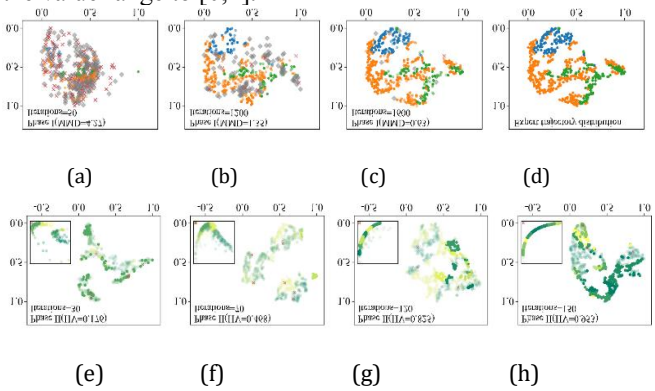

Fig.5 Visualisations of trajectory and policy distributions during Phases I and II

From the figures, it follows that: After initialization, the policy manifold is randomly scattered. After several iterations, the number of red points decreases, indicating that the algorithm discovers useful knowledge when constraints are avoided. Also, the reduction of grey diamonds implies that, increasingly, the generated trajectories approximate expert decisions. Because even Phase I required >1500 iterations to imitate near-optimal demonstrations, training GPoLI-PI models for more robust optimization is advisable and improves the practicality of AMS applications.

V. CONCLUSION

To undertake the process optimization for robust and energy-efficient AMS, we have developed an amalgamation of upper-level policy IL. The summarised results include the following three perspectives. First, the optimization of AMS is newly formulated to imitate the behaviors of process experts who have extensive optimization knowledge. The formulation is demonstrated more robustly and realistically than conventional meta-heuristics because it avoids addressing multi-objective optimization and model inaccuracy caused by noise factors. Then, the generative neural network is reconstructed to serve as an upper-level policy approximator and opens new doors to single-policy IL to alleviate the problem of mode-averaging. Besides, the efficacy of the energy-based discriminator is validated by good performance when overcoming the degraded learning of the cost function.

ACKNOWLEDGMENT

This work was supported in part by the Natural Science Foundation of Guangdong Province (2020A1515110541).

REFERENCES

[1] Mansouri, S.A., Aktas, E., and Besikci, U. (2016) Green scheduling of a two-machine flowshop: Trade-off between makespan and energy consumption. European Journal of Operational Research, 248, 772-788.

[2] Yoon, H. S., Kim, E. S., Kim, M. S., Lee, J. Y., Lee, G. B. and Ahn, S. H. (2015). Towards greener machine tools - a review on energy saving strategies and technologies. Renewable & Sustainable Energy Reviews, 48, 870-891.

[3] Deb, K. and Jain, H. (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: solving problems with box constraints, IEEE Transactions on Evolutionary Computation, 18(4), 577-601.

[4] Hung, J. and Lin, W. (2019) Investigation of the dynamic characteristics and machining stability of a Bi-rotary milling tool. Advances in Science and Technology-Research Journal, 13(1), 14-22.

[5] Diego, R.F., Trindade, P., Lobo, J. and Dias, J. (2014) Knowledge-based reasoning from human grasp demonstrations for robot grasp synthesis, Robotics and Autonomous Systems, 62(6): 794-817.

[6] Finn, C., Christiano, P., Abbeel, P., and Levine, S. (2018) A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. arXiv:1710.11248v2.

[7] Zuo, G., Zhao, Q., Huang, S., Li, J. and Gong, D. (2021) Adversarial imitation learning with mixed demonstrations from multiple demonstrators. Neurocomputing, 457, 365-376.

[8] Ho, J. and Ermon, S. (2016) Generative adversarial imitation learning, 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

[9] Xiao, Q., Li, C., Tang, Y., Li, L. and Li L. (2019) A knowledge-driven method of adaptively optimizing process parameters for energy efficient turning, Energy, 166, 142-156.

[10] Xiao, Q., Niu, B. and Chen Y. (2021) Policy manifold generation for multi-task multi-objective optimization of energy flexible machining systems. IISE Transactions. DOI: 10.1080/24725854.2021.1934756.

[11] Ng, A.Y. and Russell, S. (2000) Algorithms for Inverse Reinforcement Learning, Poc. ICML, 663-670.

[12] Parisi, S., Pirotta, M. and Peters, J. (2017) Manifold-based multi-objective policy search with sample reuse, Neurocomputing, 263, 3-14.

[13] Mohamed, S., Rosca, M., Figurnov, M. and Mnih A. (2020) Monte Carlo gradient estimation in machine learning. Journal of Machine Learning Research, 21(132), 1-62.

[14] Finn, C., Levine, S. and Abbeel, P. (2016) Guided cost learning: Deep inverse optimal control via policy optimization, Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA.

[15] Li, C., Chen, X., Tang, Y. and Li L. (2017) Selection of optimum parameters in multi-pass face milling for maximum energy efficiency and minimum production cost. Journal of Cleaner Production, 140, 1805–1818.