BIFLOW: FIRST-ORDER BILEVEL COARSE-TO-FINE FLOW MATCHING POLICY FOR CONTINUOUS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Classical actor-critic methods for continuous control use Gaussian or deterministic MLP actors that produce actions through a single forward pass. While efficient, this one-step state-to-action mapping amortizes critic-based action optimization and can be restrictive in high-dimensional tasks that require iterative action optimization or fine-grained adjustment along the action-value landscape. Diffusion and flow matching policies offer a more expressive alternative by casting action generation as an iterative transport process, but existing methods either employ a monolithic generative actor with shared parameters across the trajectory or decouple coarse proposal and fine refinement with separate objectives. We propose BiFlow, a first-order bilevel coarse-to-fine flow matching policy. At the population level, the lower problem selects a coarse actor under the exact value function of the composed policy, while the upper problem optimizes the refiner; hence the lower solution depends on the refiner through that value function. Our practical solver approximates it with a learned critic, a short K-step lookahead, and updates the refiner without unrolling this inner loop. This tractable first-order approximation preserves cross-cycle coupling through the critic trained under the deployed composed policy and the lookahead-induced outer data. We derive a critic comparison for the deployed numerical action, separating weighted flow-matching error, training–deployment source mismatch, and numerical execution error. Empirically, BiFlow achieves strong performance against recent diffusion- and flow-based continuous control baselines.

1 INTRODUCTION

Gaussian and deterministic MLP actors are efficient, but they produce actions through a single-step state-to-action mapping. In actor-critic RL, such an actor acts as an amortized optimizer of the learned critic, compressing action selection and action improvement into one forward pass. This can be restrictive in high-dimensional continuous control, where effective policies may require iterative action optimization, or fine-grained adjustment guided by the action-value landscape (Chi et al., 2025; Wang et al., 2024; Psenka et al., 2023). Diffusion and flow matching models provide an alternative parameterization by representing action generation as an iterative or continuous transport process from a source distribution toward target actions (Lipman et al., 2022; Tong et al., 2023; Lv et al., 2025). This perspective has led to strong empirical performance in both offline (Wang et al., 2022; Hansen-Estruch et al., 2023) and online (Psenka et al., 2023; Wang et al., 2024; Lv et al., 2025; McAllister et al., 2025) reinforcement learning.

Existing diffusion- and flow-based RL methods fall into two distinct families. *Monolithic generative actors* (Psenka et al., 2023; Wang et al., 2024; Ding et al., 2024; Chen et al., 2025; Lv et al., 2025) use a single generative network whose denoising or transport trajectory implicitly performs both global action-mode placement and local action refinement. However, the two roles share the same parameter set and are optimized under a single objective, leaving no explicit mechanism to specialize either stage. *Two-stage decoupled actors* (Zhang et al., 2025; Yang et al., 2023; Mark et al., 2024) split action generation into a coarse proposal followed by a learned refiner. Yet the two modules are still trained with separate objectives: the coarse actor optimizes its own value objective, while the refiner is trained to regress onto improved actions generated from a separately optimized source. As a result, the coarse actor is not explicitly encouraged to produce source actions that are compatible with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

downstream refinement, and the structural opportunity offered by the coarse-to-fine decomposition remains underexploited.

In contrast, we formulate coarse-to-fine action generation as a bilevel optimization problem. In the ideal population problem, the exact composed-policy value $Q^{\mu_{\psi,\phi}}$ makes the lower solution $\phi^\star(\psi)$ depend on the refiner. In practice, we replace this value with a learned critic, and train the refiner on the induced Q-guided targets through the lookahead. The critic is refreshed under the deployed composed policy across training cycles, retaining the intended coupling while avoiding the exact response hypergradient. We summarize our main contributions as follows:

- We formulate coarse-to-fine continuous-control policy learning as a bilevel optimization problem in which the policy-induced value couples both levels, and derive a practical first-order solver using a lookahead and a direct outer update.
- We introduce BiFlow, a practical lookahead-and-refine policy that learns Q-guided local transports from the lookahead source to improved actions, coupling coarse proposal and flow refinement. We derive a guarantee for the deployed action that explicitly accounts for FM fitting error, the mismatch between temporary and retained sources, and numerical execution error.
- We conduct experiments on a suite of continuous-control benchmarks. BiFlow achieves the best performance across various continuous control tasks.

2 Preliminaries

2.1 Markov Decision Process

We consider an infinite-horizon discounted Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ is the state space, $\mathcal{A}$ is the action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. A stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ induces a trajectory distribution $\tau \sim \pi$ and is evaluated by the expected discounted return $J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$.

The state-action value function is defined as $Q^\pi(s, a) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a\right]$, and satisfies the Bellman recursion $Q^\pi(s, a) = r(s, a) + \gamma\mathbb{E}_{s'\sim P, a'\sim\pi}[Q^\pi(s', a')]$. We operate in the off-policy setting with a replay buffer $\mathcal{D}$ of past transitions (s, a, r, s'), and learn parameterized approximations Q_ϕ and π_θ to maximize $J(\pi_\theta)$.

2.2 Flow Matching

Flow matching (FM) (Lipman et al., 2022) learns a time-dependent vector field $v_\psi : \mathcal{S} \times \mathcal{A} \times [0, 1] \to \mathbb{R}^{d_a}$ that transports a source distribution to a target distribution by integrating an ordinary differential equation. Given source-target pairs $(a_{\text{src}}, a_{\text{tgt}})$ and the affine interpolation path $a_t = (1 - t)\, a_{\text{src}} + t\, a_{\text{tgt}}, \quad t \in [0, 1]$, the target velocity is the time derivative $\frac{d}{dt}a_t = a_{\text{tgt}} - a_{\text{src}}$. The vector field is trained to match this target via the conditional flow-matching loss

$$\mathcal{L}_{\text{FM}}(\psi) = \mathbb{E}_{s,\, t\sim\mathcal{U}[0,1],\, (a_{\text{src}}, a_{\text{tgt}})}\left[\|v_\psi(s, a_t, t) - (a_{\text{tgt}} - a_{\text{src}})\|^2\right]. \tag{1}$$

At inference, the refined action is obtained by integrating the learned velocity: $a_{\text{final}} = a_{\text{src}} + \int_0^1 v_\psi(s, a_t, t)\, dt$.

2.3 Diffusion and Flow Matching Policies in RL

Diffusion and flow matching models have emerged as expressive policy classes for continuous control (Choi et al., 2026). Existing online RL methods in this family fall into two structurally distinct categories: *monolithic actors*, which use a single generative network and inject Q-value signal into its training objective, and *two-stage decoupled actors*, which separate coarse action proposal from fine refinement.

Monolithic actors. A single generative network $\pi_\theta(\cdot \mid s)$ is trained to sample from $p_1(a \mid s) \propto \exp(Q(s,a)/\alpha)$ via different Q-injection mechanisms: QSM (Psenka et al., 2023) aligns the score $\nabla_a \log p_t$ with $\nabla_a Q$; DACER (Wang et al., 2024) backpropagates through the full reverse chain $x_T \to x_0$ with an entropy regularizer; QVPO (Ding et al., 2024) reweights the FM/diffusion ELBO by a monotone $f(Q)$. DPMD and SDAC (Ma et al., 2025) derive reweighted score-matching losses from policy mirror descent and maximum-entropy policy optimization, respectively. These methods enable efficient Q-guided diffusion-policy training without explicitly decomposing action generation into proposal and refinement stages. Flow-based variants achieve faster inference: FPMD (Chen et al., 2025) derives a one-step velocity field $v_\theta(x_t, s, t)$ from the policy mirror descent closed form; FlowRL (Lv et al., 2025) imposes a W_2 constraint between π_θ and a buffer-derived behavioral reference. While the timeline $t \in [0,1]$ implicitly performs coarse-to-fine generation, the parameters θ are shared across all t and trained against a single objective, preventing specialized optimization at either end of the timeline.

Two decoupled actors. A second line introduces explicit coarse and fine modules with separate parameters (θ_1, θ_2). GoRL (Zhang et al., 2025) parameterizes $\pi_1(\cdot \mid s) = \mathcal{N}(\mu_{\theta_1}(s), \sigma)$ and a conditional FM decoder $\pi_2(\cdot \mid s, z)$ where $z \sim \mathcal{N}(0, I)$, training θ_1 by PPO and θ_2 by supervised regression onto improved actions. DIPO (Yang et al., 2023) sets π_1 as a fixed buffer sampler with $\nabla_a Q$ ascent, and trains π_2 as a DDPM regressing onto the ascended actions; PA-RL (Mark et al., 2024) generalizes this to a propose-optimize-distill recipe with a conservative critic. While these methods structurally separate θ_1 and θ_2, they train them with disjoint losses $\mathcal{L}_1(\theta_1)$ and $\mathcal{L}_2(\theta_2; \theta_1)$ rather than a lower objective defined by the value of their deployed composition. QFlex (Wei et al., 2026) parameterizes an exploratory flow policy with a Gaussian source policy and a state-dependent velocity field, training the velocity by conditional flow matching on Q-gradient-ascent targets. BiFlow instead makes source selection refiner-aware through the continuation value of the composed policy, then trains the refiner on the resulting lookahead source distribution.

3 BiFlow

In this section, we first present an idealized bilevel view of BiFlow, where the coarse proposal and flow refiner are coupled through the value function of the composed policy. Section 3.2 then instantiates this view as a practical first-order lookahead-and-refine update. The Bellman regression update is described in Section 3.3.

3.1 Coarse-to-Fine Policies as Bilevel Optimization

As discussed in Section 2.3, existing diffusion- and flow-based RL policies can be viewed as coarse-to-fine action generators. These methods, however, typically either merge the coarse and fine stages within a single network or separate them without feedback between the stages. In both cases, they do not fully exploit the structure provided by the coarse-to-fine decomposition.

We address this by formulating the coarse and fine stages as a bilevel optimization problem. A lightweight MLP π_1 produces an initial proposal $a_{\text{init}} = \pi_1(s)$, which serves as the input to a flow-matching refiner π_2. For any coarse proposal π_1 and refiner π_2, we denote the composed coarse-to-fine policy by $\mu_{\pi_2,\pi_1} := \pi_2 \circ \pi_1$, i.e., $\mu_{\pi_2,\pi_1}(s) = \pi_2(s, \pi_1(s))$. The key idea is to couple the two stages through the value function induced by this composed policy. Here π_1 is the lower-level variable and π_2 is the upper-level variable: the lower problem identifies optimal coarse sources for a fixed refiner, and the upper problem trains the refiner using those sources.

For a fixed refiner π_2, the lower-level problem identifies coarse policies whose proposals are valuable under the composed coarse-to-fine value function. This defines the set of lower-level optimal coarse sources as:

$$\mathcal{S}^*(\pi_2) := \arg\max_{\tilde{\pi}_1 \in \Pi_1} \mathbb{E}_{s\sim\mathcal{D}} \left[Q^{\mu_{\pi_2,\tilde{\pi}_1}}(s, \tilde{\pi}_1(s)) \right]. \tag{2}$$

Here $Q^{\mu_{\pi_2,\tilde{\pi}_1}}$ is the exact policy-induced value function, not an additional optimization variable. Thus the conceptual problem has two levels: the lower variable π_1 and upper variable π_2. Its dependence on both policies is part of the lower objective and makes $\mathcal{S}^*(\pi_2)$ a genuine solution map of the refiner; learning a parametric critic is only an implementation device introduced later.

The lower objective intentionally evaluates the coarse source action rather than the currently refined action. Its role is to place the source in a high-value region under the continuation value induced by the deployed composed policy, while the upper level is responsible for optimizing the refinement of that source. Therefore, the dependence of the lower solution on the refiner arises through the composed-policy continuation value, preserving distinct roles for the two levels.

Among the lower-level optimal coarse sources in $\mathcal{S}^*(\pi_2)$, the upper level selects the one whose refined action achieves the largest value under the composed policy:

$$\max_{\pi_2 \in \Pi_2} \max_{\pi_1 \in \mathcal{S}^*(\pi_2)} J(\pi_1, \pi_2) := \mathbb{E}_{s\sim\mathcal{D}} \left[Q^{\mu_{\pi_2,\pi_1}} \left(s, \pi_2(s, \pi_1(s))\right)\right]. \tag{3}$$

The lower objective evaluates $\pi_1(s)$ as the current source action, while π_2 affects its value through the composed continuation policy. This is distinct from evaluating the already-refined current action $\pi_2(s, \pi_1(s))$, as the outer objective does; neither choice is an algebraic substitute for the other. Our design asks the lower level to locate valuable sources on the refiner-aware value landscape, then asks the upper level to learn the remaining local transport. Conversely, the selected coarse policy determines the refiner's training distribution.

To instantiate this bilevel formulation, we parameterize π_1 as a deterministic coarse actor $c_\phi : \mathcal{S} \to \mathcal{A}$. We parameterize π_2 as a flow-matching refiner. Given a state s and an input action a_0, the refiner integrates the velocity field $v_\psi(s, a_t, t)$ from $a_{t=0} = a_0$ to $t = 1$:

$$\frac{da_t}{dt} = v_\psi(s, a_t, t), \qquad F_\psi(s, a_0) = a_{t=1}. \tag{4}$$

Thus the deployed coarse-to-fine policy is

$$\mu_{\psi,\phi}(s) = F_\psi(s, c_\phi(s)). \tag{5}$$

Under this parameterization, Eq. (2) defines $\phi^\star(\psi)$ through the chain $\psi \mapsto \mu_{\psi,\phi} \mapsto Q^{\mu_{\psi,\phi}} \mapsto \phi^\star(\psi)$. We next approximate this ideal solution map with a practical first-order solver.

Toy illustration. We further use a 2D synthetic bandit example to visualize the role of the lookahead source in BiFlow. The initial source distribution a_{init} is placed at the top of the action space, while the high-value target actions form three modes at the bottom. A single-stage flow-matching policy has to learn the full long-range transport from a_{init} to the target modes. In contrast, BiFlow first obtains a K-step lookahead source $a_{\text{src}}^{(K)}$ by moving the initial source toward high-value regions, and then trains the flow refiner to transport from $a_{\text{src}}^{(K)}$ to the final target actions.

In this example, the final error decreases from 0.480 (Single Stage FM) to 0.385 (BiFlow).

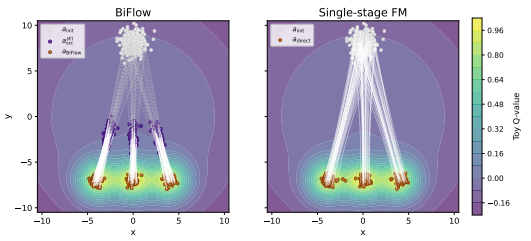

Figure 1: Toy illustration of BiFlow's lookahead and refinement mechanism. BiFlow first moves the initial source a_{init} to a K-step lookahead source $a_{\text{src}}^{(K)}$, then performs local flow refinement to obtain a_{BiFlow}. In contrast, the single-stage FM baseline directly transports a_{init} to a_{direct}.

3.2 First-Order Bilevel Optimizer

Inner step. At each training iteration, let ϕ denote the coarse-source parameters before the update. Starting from $\phi^{(0)} = \phi$, we construct a short inner trajectory $\{\phi^{(k)}\}_{k=0}^{K_{\text{in}}}$ by critic-gradient ascent. From the current coarse parameter $\phi^{(0)} = \phi$, we perform K_{in} ascent steps on the proposal's value under the current composed policy:

$$\phi^{(k+1)} = \phi^{(k)} + \eta_\phi \nabla_{\phi^{(k)}} \mathbb{E}_{s\sim\mathcal{D}} \left[Q^{\mu_{\psi,\phi^{(k)}}} \left(s,\, c_{\phi^{(k)}}(s)\right)\right], \qquad k = 0, \ldots, K_{\text{in}} - 1, \tag{6}$$

The update therefore improves the source action on the composed-policy continuation-value landscape. The temporary inner parameters are not committed as the deployed actor; they define $a_{\text{src}}(s) := \text{sg}[c_{\phi^{(K_{\text{in}})}}(s)]$, where sg is stop-gradient. Thus, no gradient from the flow-matching update is propagated through the inner trajectory.

Outer step. Given the temporary lookahead source $a_{\mathrm{src}}(s)$ produced by the inner trajectory in Eq. (6), we construct Q-guided local targets for the refiner. For each state s, we sample N_{cand} candidates by perturbing $a_{\mathrm{src}}(s)$ and improve each candidate by several action-space ascent steps on the composed-policy value $Q^{\mu_{\psi,\phi^{(K_{\mathrm{in}})}}}$, clipping the action to the valid range after each step.

$$a_j^{(0)} = \mathrm{clip}(a_{\mathrm{src}}+\sigma_c\epsilon_j, -1, 1), \qquad a_j^{(\ell+1)} = \mathrm{clip}\Big(a_j^{(\ell)} + \eta_a \nabla_a Q^{\mu_{\psi,\phi^{(K_{\mathrm{in}})}}}(s, a_j^{(\ell)}), -1, 1\Big), a_j^{\star} = a_j^{(L)}. \tag{7}$$

This local search produces Q-guided targets $\{a_j^*(s)\}_{j=1}^{N_{\text{cand}}}$.

For $t \sim \mathrm{Unif}[0,1]$, the linear interpolant from the lookahead source to each Q-guided target is:

$$a_{j,t} = (1-t)a_{\mathrm{src}}(s) + t a_j^*(s). \tag{8}$$

The refiner is then trained by weighted conditional flow matching:

$$\mathcal{L}_{\mathrm{FM}}(\psi;\phi) = \mathbb{E}_{s,t}\left[\sum_{j=1}^{N_{\text{cand}}} w_j(s)\left\|v_\psi(s,a_{j,t},t) - \big(a_j^*(s) - a_{\mathrm{src}}(s)\big)\right\|_2^2\right]. \tag{9}$$

The weights $w_j(s)$ are the improvement-based weights assigned to targets $a_j^*(s)$, as defined in Eq. (14). Both a_{src} and a_j^* are detached. The semicolon in $\mathcal{L}_{\mathrm{FM}}(\psi;\phi)$ indicates that ϕ determines the source–target pairs used to train the refiner, but the outer gradient is taken only with respect to ψ:

$$\psi \leftarrow \psi - \eta_\psi \nabla_\psi \mathcal{L}_{\mathrm{FM}}(\psi;\phi), \tag{10}$$

Equation (10) is the practical analogue of retaining a direct outer update, but it uses the FM surrogate rather than claiming to evaluate the ideal derivative exactly. Within a cycle, the composed-policy value shapes the inner lookahead, which determines the outer sources, targets, and weights. This coupling remains even though the response gradient is omitted.

We do not use the flow-matching loss to update the coarse parameter ϕ. The reason is that $\mathcal{L}_{\mathrm{FM}}$ is a regression loss for learning the velocity field once the source-target pairs are fixed. It is therefore a suitable surrogate for updating the refiner ψ, but not for improving the source policy.

The retained coarse parameter is instead updated by a first-order meta-gradient of the refined-action objective evaluated at the adapted parameters $\phi^{(K_{\mathrm{in}})}$. Specifically, we compute

$$g_\phi^{\mathrm{FO}} = \nabla_{\phi^{(K_{\mathrm{in}})}} \mathbb{E}_{s\sim\mathcal{D}}\left[Q^{\mu_{\psi,\phi^{(K_{\mathrm{in}})}}}\left(s, F_\psi(s, c_{\phi^{(K_{\mathrm{in}})}}(s))\right)\right], \tag{11}$$

and apply this gradient to the retained coarse parameters:

$$\phi \leftarrow \phi + \eta_\phi g_\phi^{\mathrm{FO}}. \tag{12}$$

This follows the first-order bilevel approximation: the refined-action objective is differentiated with respect to the adapted parameters $\phi^{(K_{\mathrm{in}})}$, but we do not backpropagate through the K_{in} inner updates that produced them.

Improvement-based target weighting. Since the local search is guided by the composed-policy value, not all generated targets provide the same value improvement. The improvement-based weighting therefore selects locally beneficial transports while discarding directions that decrease value. Specifically, for each Q-guided target $a_j^*(s)$, we compute its value improvement over the lookahead source:

$$\Delta_j(s) = Q^{\mu_{\psi,\phi^{(K_{\mathrm{in}})}}}\left(s, a_j^*(s)\right) - Q^{\mu_{\psi,\phi^{(K_{\mathrm{in}})}}}\left(s, a_{\mathrm{src}}(s)\right). \tag{13}$$

Let $\mathcal{I}_+(s) = \{j : \Delta_j(s) > 0\}$ denote the set of improving candidates. When $\mathcal{I}_+(s) \neq \emptyset$, we normalize the positive improvements with a softmax:

$$w_j(s) = \mathbf{1}[j \in \mathcal{I}_+(s)] \frac{\exp(\Delta_j(s)/T_\alpha)}{\sum_{k\in\mathcal{I}_+(s)} \exp(\Delta_k(s)/T_\alpha)}, \qquad \mathcal{I}_+(s) \neq \emptyset. \tag{14}$$

When $\mathcal{I}_+(s) = \emptyset$, we replace the candidate set by a single stay target $a_{\mathrm{stay}}^*(s) = a_{\mathrm{src}}(s)$ with weight 1, yielding a zero-transport pair. Equivalently, the target-weight set used in Eq. (9) is

$$\mathcal{T}(s) = \begin{cases} \left\{(a_j^*(s), w_j(s)) : j \in \mathcal{I}_+(s)\right\}, & \mathcal{I}_+(s) \neq \emptyset, \\ \left\{(a_{\mathrm{src}}(s), 1)\right\}, & \mathcal{I}_+(s) = \emptyset. \end{cases} \tag{15}$$

This rule suppresses candidates that fail to improve over the lookahead source, while preserving multiple locally beneficial transport directions. In the fallback case, the refiner is trained on a zero-transport pair rather than toward any lower-value action.

3.3 Critic Update for the Bilevel Optimizer

Each training cycle begins with Bellman regression. Let ψ and ϕ be the current refiner and retained coarse-source parameters. The learned Q_θ approximates the population quantity $Q^{\mu_{\psi,\phi}}$ in Eqs. (2)–(3); it is not a third variable in the conceptual bilevel problem. Its update belongs to the practical actor–critic solver and uses the deployed policy $\mu_{\psi,\phi}$ from Eq. (5).

We learn two critics Q_{θ_1} and Q_{θ_2} following clipped double Q-learning (Fujimoto et al., 2018). Given a replay transition (s, a, r, s', d), the target action is constructed from the current deployed coarse-to-fine policy with target smoothing noise:

$$\tilde{a}' = \mathrm{clip}(\mu_{\psi,\phi}(s') + \epsilon, -1, 1)\,, \qquad \epsilon \sim \mathrm{clip}\big(\mathcal{N}(0, \sigma_{\mathrm{sm}}^2 I), -\epsilon_{\max}, \epsilon_{\max}\big)\,. \tag{16}$$

The Bellman target is

$$y_Q = r + \gamma(1-d) \min_{i\in\{1,2\}} Q_{\bar{\theta}_i}(s', \tilde{a}'), \tag{17}$$

where $Q_{\bar{\theta}_i}$ are Polyak-averaged target critics. Each critic is fitted by the squared Bellman error

$$\mathcal{L}_Q(\theta_i) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\big(Q_{\theta_i}(s,a) - y_Q\big)^2\right], \qquad i \in \{1,2\}. \tag{18}$$

After the critic update, we define the critic aggregate:

$$\bar{Q}(s,a) = \min_{i\in\{1,2\}} Q_{\theta_i}(s,a), \tag{19}$$

This clipped double-Q aggregate follows prior work on reducing overestimation bias in bootstrapped value learning (van Hasselt, 2010; van Hasselt et al., 2016; Fujimoto et al., 2018). It is the local value oracle for Eq. (6) and the Q-guided targets. In later cycles it is refitted under the updated deployed composition, so changes in ψ change future lower objectives without differentiating through critic training inside the current cycle.

4 Theoretical Analysis

We analyze one training cycle while distinguishing the temporary source used for flow matching from the retained source used at execution. Write $x_K(s) = c_{\phi^{(K)}}(s)$ for the lookahead source and $c_{\mathrm{dep}}(s)$ for the coarse actor retained after the cycle's actual coarse update. To isolate the effect of coarse-to-fine refinement, we abstract away critic estimation error and assume access to an accurate critic $\bar{Q}$.

4.1 Main Results

For each candidate set, Algorithm 1 produces $\{a_j^\star(s), w_j(s)\}_{j=1}^{N_{\mathrm{cand}}}$, including the stay-target fallback, with $w_j(s) \geq 0$ and $\sum_{j=1}^{N_{\mathrm{cand}}} w_j(s) = 1$. Define

$$\begin{aligned} X_{j,t} &= (1-t)x_K(s) + t a_j^\star(s), \\ \mathcal{E}_K(\psi) &= \mathbb{E}\sum_{j=1}^{N_{\mathrm{cand}}} w_j(s) \int_0^1 \big\| v_\psi(s, X_{j,t}, t) - \big(a_j^\star(s) - x_K(s)\big)\big\|_2^2\, dt. \end{aligned} \tag{20}$$

This is the population weighted FM objective at the execution refiner. Let F_ψ denote the exact time-1 ODE flow and let $\widehat{a}_B(s) = \mathrm{ODESolve}(v_\psi, s, c_{\mathrm{dep}}(s))$ be the numerical action returned by Algorithm 1, before external exploration noise. Define

$$\delta_K = \big(\mathbb{E}\|c_{\mathrm{dep}}(s) - x_K(s)\|_2^2\big)^{1/2}, \qquad \epsilon_{\mathrm{num}} = \big(\mathbb{E}\|\widehat{a}_B(s) - F_\psi(s, c_{\mathrm{dep}}(s))\|_2^2\big)^{1/2}. \tag{21}$$

Any execution-time action postprocessing is included in $\widehat{a}_B$ and therefore in ϵ_{num}.

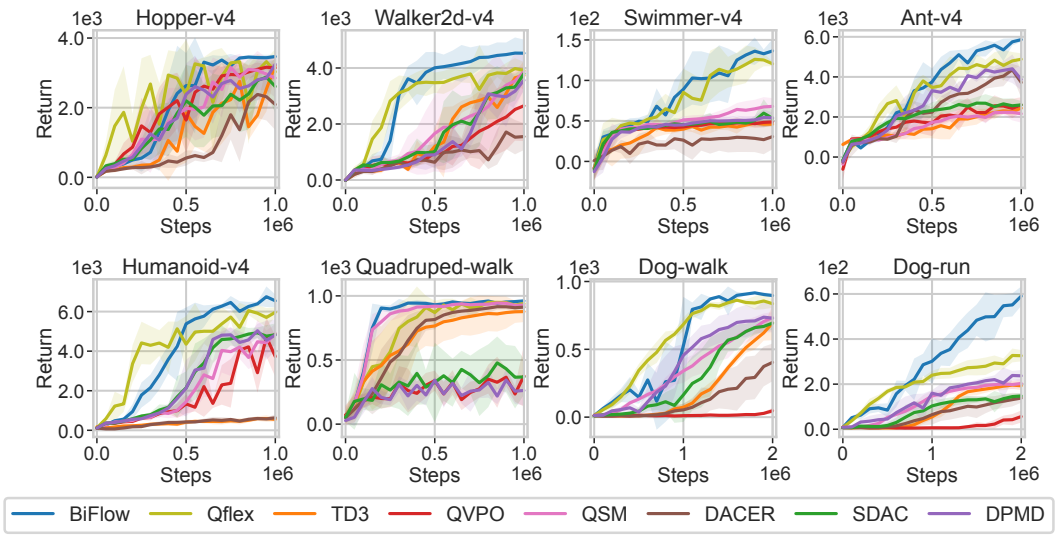

Figure 2: Average evaluation return during training on MuJoCo and DeepMind Control Suite tasks. Curves show the mean return over 5 random seeds, and shaded regions denote one standard deviation.

Theorem 1 (FM fitting with a different deployment source)**.** *Assume the ODE solutions exist on* $[0,1]$*, the vector field is uniformly* L_v*-Lipschitz in action on the relevant trajectories and interpolants, and* $\bar{Q}(s,\cdot)$ *is* G*-Lipschitz on the relevant action region. Assume the displayed second moments are finite. Set*

$$A(L) = \begin{cases} \sqrt{(e^{2L}-1)/(2L)}, & L > 0, \\ 1, & L = 0, \end{cases} \qquad B_K = A(L_v)\sqrt{\mathcal{E}_K(\psi)} + e^{L_v}\delta_K + \epsilon_{\text{num}}. \tag{22}$$

Then the action actually returned by the numerical policy satisfies

$$\left(\mathbb{E} \sum_{j=1}^{N_{\text{cand}}} w_j(s) \|\widehat{a}_B(s) - a_j^\star(s)\|_2^2 \right)^{1/2} \le B_K. \tag{23}$$

For any comparator action $b(s,\xi)$*, define*

$$m_b = \mathbb{E}\left[\sum_{j=1}^{N_{\text{cand}}} w_j(s)\bar{Q}(s,a_j^\star(s)) - \bar{Q}(s,b(s,\xi)) \right]. \tag{24}$$

The deployed action obeys

$$\mathbb{E}[\bar{Q}(s,\widehat{a}_B(s)) - \bar{Q}(s,b(s,\xi))] \ge m_b - GB_K. \tag{25}$$

Theorem 1 characterizes when the advantage of the Q-guided targets is preserved by the deployed BiFlow policy. Specifically, if the target-value margin m_b over a comparator b exceeds the deployment penalty GB_K, then the executed BiFlow action has strictly larger expected critic value than b. The error budget B_K separates the three factors that can prevent this transfer: flow-matching fitting error, mismatch between the lookahead source x_K used for training and the retained source c_{dep} used at deployment, and numerical integration error.

Corollary 1 (Conditional advantage over constituent policies)**.** *Let* $a_C(s) = c_{\text{dep}}(s)$ *denote the retained coarse action and let* $a_R(s,\xi)$ *denote a refinement-only comparator. Under Theorem 1, if*

$$GB_K < \min\{m_C, m_R\},$$

then

$$\mathbb{E}\,\bar{Q}(s,\widehat{a}_B(s)) > \max\left\{\mathbb{E}\,\bar{Q}(s,a_C(s)), \mathbb{E}\,\bar{Q}(s,a_R(s,\xi))\right\}.$$

Thus, whenever the Q-guided targets have sufficient advantage over both constituent policies to dominate the flow-fitting, source-mismatch, and numerical-execution errors, the deployed BiFlow action has higher expected critic value than either constituent policy. The proof and a more explicit sufficient condition for the coarse-only comparison are given in Appendix A.

5 EXPERIMENTS

We evaluate BiFlow on continuous-control benchmarks including Gymnasium MuJoCo tasks (Towers et al., 2024) and tasks from the DeepMind Control Suite (Tunyasuvunakool et al., 2020), and compare against various standard model-free baselines and recent online RL methods based on diffusion or flow-matching policies.

5.1 EXPERIMENTAL SETUP

The proposed algorithm, BiFlow, is implemented in JAX (Bradbury et al., 2018). Experiments were conducted on a server equipped with dual Intel(R) Xeon(R) Platinum 8480C CPUs and an NVIDIA H800 GPU. BiFlow hyperparameters, including the inner ascent steps, the candidate action count, and other algorithm-specific settings are listed in Appendix Table 4; default training and evaluation settings are listed in Appendix Table 5.

Tasks. We evaluate on eight tasks: Hopper-v4, Walker2d-v4, Swimmer-v4, Ant-v4, Humanoid-v4, Quadruped-walk, Dog-walk, and Dog-run. The first five are Gymnasium MuJoCo tasks, while the last three are from the DeepMind Control Suite. Most tasks are trained for 1M environment steps. Dog-walk and Dog-run are trained for 2M environment steps due to their locomotion difficulty.

Baselines. We compare several model-free RL algorithms including TD3 (Fujimoto et al., 2018), QVPO (Ding et al., 2024), QSM (Psenka et al., 2023), DACER (Wang et al., 2024), SDAC and DPMD (Ma et al., 2025), and QFlex (Wei et al., 2026). TD3 is a classic model-free RL baseline, while the remaining methods are recent diffusion- or flow-policy online RL baselines. All baseline results are rerun using public implementations under the same evaluation protocol as BiFlow. Implementation sources and evaluation details are provided in Appendix D.

Evaluation protocol. For each run, evaluation is performed throughout training at 20 checkpoints. Each checkpoint is evaluated over 20 episodes. Experiments are conducted over five random seeds.

Table 1 and Figure 2 summarize the main comparison. BiFlow achieves the best performance on all eight tasks. In particular, BiFlow improves over the strongest baseline by 22.7% on Ant-v4 and 81.0% on Dog-run, and also yields clear gains on Walker2d-v4 and Swimmer-v4.

Diffusion-based SOTA methods such as QSM, QVPO, and DACER perform competitively on selected environments but remain task-dependent, with degradation on harder locomotion tasks, such as Ant-v4, Quadruped-walk, Dog-walk, and Dog-run. QFlex is the strongest flow-matching baseline, yet BiFlow outperforms it on every task, especially on Ant-v4 and Dog-run. This consistent improvement supports our coarse-to-fine design: the lookahead source moves actions toward high-value regions.

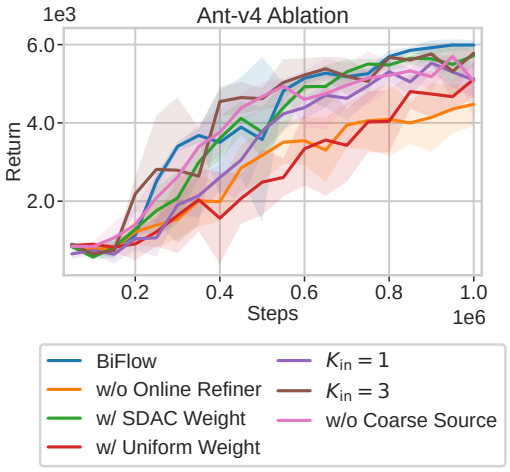

Figure 3: Ablation results on Ant-v4.

Ablation Study We ablate BiFlow on Ant-v4 under the main evaluation protocol. Variants remove the online refiner or learned coarse source, replace our weights with SDAC or uniform weights, or vary the number of inner ascent steps ($K_{\text{in}} \in \{1, 3\}$ versus the default $K_{\text{in}} = 5$). The no-online-refiner variant uses the refiner only at evaluation; the no-coarse-source variant starts flow matching from Gaussian noise.

Table 2 and Figure 3 show that BiFlow performs best. Removing the online refiner or learned coarse source reduces return by 25.3% and 15.4%, respectively, supporting their coupled use. SDAC weighting remains competitive, whereas uniform weighting degrades performance, showing the

benefit of improvement-aware weights. Varying the number of inner ascent steps shows a consistent, monotonic trend: $K_{\text{in}} = 1$ reaches 5093.2 ± 469.2, $K_{\text{in}} = 3$ improves to 5780.4 ± 53.6, and the default $K_{\text{in}} = 5$ performs best at 5989.6 ± 125.7, indicating a modest but steady benefit from additional lookahead steps.

Table 1: Performance on MuJoCo and DeepMind Control Suite tasks. The final evaluation return is reported as mean ± standard deviation over 5 random seeds after training. The best result for each task is highlighted in bold.

Method	Hopper-v4	Walker2d-v4	Swimmer-v4	Ant-v4
TD3	3214.8 ± 206.2	3731.2 ± 519.0	46.0 ± 3.5	2436.7 ± 155.9
QVPO	3140.6 ± 129.7	2650.6 ± 861.0	48.7 ± 4.2	2604.1 ± 380.8
QSM	2854.0 ± 562.7	3867.2 ± 634.0	68.0 ± 10.1	2161.6 ± 443.5
DACER	2091.0 ± 734.0	1550.3 ± 480.4	30.6 ± 15.5	3721.2 ± 773.3
SDAC	2608.0 ± 773.2	3801.7 ± 231.6	53.2 ± 8.9	2602.4 ± 192.6
DPMD	3244.9 ± 190.4	3564.2 ± 374.2	53.9 ± 6.5	3869.3 ± 528.0
QFlex	3032.1 ± 656.8	3916.5 ± 294.7	120.5 ± 2.8	4880.8 ± 294.5
BiFlow (Ours)	$\mathbf{3468.5 \pm 115.9}$	$\mathbf{4529.0 \pm 487.4}$	$\mathbf{136.4 \pm 14.6}$	$\mathbf{5989.6 \pm 125.7}$
Method	Humanoid-v4	Quadruped-walk	Dog-walk	Dog-run
TD3	553.2 ± 111.5	877.7 ± 82.1	695.0 ± 127.8	193.0 ± 20.2
QVPO	3728.8 ± 1031.8	377.0 ± 240.8	46.9 ± 21.3	56.3 ± 26.7
QSM	4895.2 ± 476.1	928.4 ± 24.6	734.0 ± 41.2	203.1 ± 20.4
DACER	632.0 ± 177.4	912.3 ± 19.0	403.9 ± 138.9	138.4 ± 20.5
SDAC	4830.4 ± 564.8	368.2 ± 139.0	693.0 ± 49.8	147.6 ± 12.7
DPMD	4818.2 ± 794.3	262.7 ± 114.5	729.4 ± 104.5	237.0 ± 25.6
QFlex	5966.5 ± 246.3	941.2 ± 18.1	836.6 ± 30.9	326.1 ± 25.1
BiFlow (Ours)	$\mathbf{6545.0 \pm 327.2}$	$\mathbf{961.2 \pm 11.1}$	$\mathbf{896.5 \pm 15.0}$	$\mathbf{590.4 \pm 42.7}$

Table 2: Ablation results on Ant-v4. The final evaluation return is reported as mean ± standard deviation over 5 random seeds after training.

Variant	Ant-v4
BiFlow($K_{\text{in}} = 5$)	$\mathbf{5989.6 \pm 125.7}$
w/o Online Refiner	4475.3 ± 553.7
w/ SDAC Weight	5719.6 ± 277.4
w/ Uniform Weight	5117.2 ± 439.8
BiFlow($K_{\text{in}} = 3$)	5780.4 ± 53.6
BiFlow($K_{\text{in}} = 1$)	5093.2 ± 469.2
w/o Coarse Source	5066.4 ± 969.5

Table 3: Average wall-clock cost per training cycle. TPC denotes time per cycle, where each cycle consists of 5 environment interactions and one update of all networks.

Method	DPMD	SDAC	BiFlow
TPC (ms)	24.0	30.5	31.0

6 Conclusion

BiFlow defines coarse-to-fine learning as a population bilevel problem coupled by the exact composed-policy value, and implements it with a first-order lookahead. Our deployment bound and experiments support this practical design while making its approximation scope explicit.

AI USE STATEMENT

Generative AI tools assisted with mathematical proof checking, code verification, and manuscript polish. They were not used to generate experimental data or results. The authors reviewed and verified all AI-assisted outputs and take full responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

REFERENCES

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Tianyi Chen, Haitong Ma, Na Li, Kai Wang, and Bo Dai. One-step flow policy mirror descent. *arXiv preprint arXiv:2507.23675*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Wonhyeok Choi, Shutong Ding, Minwoo Choi, Jungwan Woo, Kyumin Hwang, Jaeyeul Kim, Ye Shi, and Sunghoon Im. A review of online diffusion policy rl algorithms for scalable robotic control. *arXiv preprint arXiv:2601.06133*, 2026.

Shutong Ding, Ke Hu, Zhenhao Zhang, Kan Ren, Weinan Zhang, Jingyi Yu, Jingya Wang, and Ye Shi. Diffusion-based reinforcement learning via q-weighted variational policy optimization. *Advances in Neural Information Processing Systems*, 37:53945–53968, 2024.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1587–1596, 2018.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Lei Lv, Yunfei Li, Yu Luo, Fuchun Sun, Tao Kong, Jiafeng Xu, and Xiao Ma. Flow-based policy for online reinforcement learning. *arXiv preprint arXiv:2506.12811*, 2025.

Haitong Ma, Tianyi Chen, Kai Wang, Na Li, and Bo Dai. Efficient online reinforcement learning for diffusion policy. *arXiv preprint arXiv:2502.00361*, 2025.

Max Sobol Mark, Tian Gao, Georgia Gabriela Sampaio, Mohan Kumar Srirama, Archit Sharma, Chelsea Finn, and Aviral Kumar. Policy agnostic rl: Offline rl and online rl fine-tuning of any class and backbone. *arXiv preprint arXiv:2412.06685*, 2024.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. *arXiv preprint arXiv:2507.21053*, 2025.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099.

Hado van Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), 2016.

Yinuo Wang, Likun Wang, Yuxuan Jiang, Wenjun Zou, Tong Liu, Xujie Song, Wenxuan Wang, Liming Xiao, Jiang Wu, Jingliang Duan, et al. Diffusion actor-critic with entropy regulator. *Advances in Neural Information Processing Systems*, 37:54183–54204, 2024.

Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.

Yunyue Wei, Chenhui Zuo, and Yanan Sui. Scalable exploration for high-dimensional continuous control via value-guided flow. *arXiv preprint arXiv:2601.19707*, 2026.

Long Yang, Zhixiong Huang, Fenghao Lei, Yucun Zhong, Yiming Yang, Cong Fang, Shiting Wen, Binbin Zhou, and Zhouchen Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023.

Chubin Zhang, Zhenglin Wan, Feng Chen, Xingrui Yu, Ivor Tsang, and Bo An. Gorl: An algorithm-agnostic framework for online reinforcement learning with generative policies. *arXiv preprint arXiv:2512.02581*, 2025.

A Additional Theory and Proofs

Corollary 2 (Full version of conditional comparison with coarse-only and refine-only)**.** *Let $a_C(s) = c_{\mathrm{dep}}(s)$ be the retained coarse action and let $a_R(s,\xi)$ be a refinement-only comparator. Under Theorem 1,*

$$GB_K < \min\{m_C, m_R\} \quad \Longrightarrow \quad \mathbb{E}\bar{Q}(s,\widehat{a}_B(s)) > \max\{\mathbb{E}\bar{Q}(s,a_C(s)), \mathbb{E}\bar{Q}(s,a_R(s,\xi))\}, \tag{26}$$

where m_C and m_R are Eq. (24) *evaluated at $b = a_C$ and $b = a_R$, respectively.*

The target filter guarantees improvement relative to x_K, not necessarily relative to c_{dep}. In particular, define

$$\Delta_K = \mathbb{E}\sum_{j=1}^{N_{\mathrm{cand}}} w_j(s)[\bar{Q}(s,a_j^\star(s)) - \bar{Q}(s,x_K(s))] \geq 0. \tag{27}$$

Then $m_C = \Delta_K + \mathbb{E}[\bar{Q}(s,x_K(s)) - \bar{Q}(s,c_{\mathrm{dep}}(s))] \geq \Delta_K - G\delta_K$. Hence a sufficient condition for improvement over the retained coarse action is

$$\Delta_K > G\left[A(L_v)\sqrt{\mathcal{E}_K(\psi)} + (e^{L_v}+1)\delta_K + \epsilon_{\mathrm{num}}\right]. \tag{28}$$

For a refinement-only comparator, positivity of m_R is not guaranteed by target construction. The corollary transfers an existing target advantage to the deployed action; it does not establish unconditional dominance.

Why lookahead helps refinement. Let $\bar{a}_K(s) = \sum_{j=1}^{N_{\mathrm{cand}}} w_j(s)a_j^\star(s)$. At the reference field $v \equiv 0$,

$$\begin{aligned}\mathcal{E}_K(0) &= \mathbb{E}\sum_{j=1}^{N_{\mathrm{cand}}} w_j(s)\|a_j^\star(s) - x_K(s)\|_2^2 \\ &= \mathbb{E}\|x_K(s) - \bar{a}_K(s)\|_2^2 + \mathbb{E}\sum_{j=1}^{N_{\mathrm{cand}}} w_j(s)\|a_j^\star(s) - \bar{a}_K(s)\|_2^2.\end{aligned} \tag{29}$$

For fixed targets and weights, moving the source closer to their weighted mean reduces the first term. This identity alone proves neither faster optimization nor smaller final fitting error, and BiFlow regenerates its targets. The deployment bound also shows the tradeoff: a useful lookahead source must not create an excessive amplified mismatch from the retained source. All value comparisons use the same critic and state population ν; they are not discounted-return comparisons across different policy occupancies.

A.1 Proof of Theorem 1

Proof. Fix a state and candidate set. For target j, write $x = x_K(s)$, $y_j = a_j^\star(s)$, $X_{j,t} = (1-t)x + ty_j$, and

$$r_j(t) = v_\psi(s, X_{j,t}, t) - (y_j - x).$$

Let z_t solve $\dot{z}_t = v_\psi(s,z_t,t)$ from $z_0 = c_{\mathrm{dep}}(s)$ and put $e_{j,t} = z_t - X_{j,t}$. Its initial deviation is $e_{j,0} = c_{\mathrm{dep}}(s) - x_K(s)$, and spatial Lipschitz continuity gives

$$\|e_{j,t}\|_2 \leq \|e_{j,0}\|_2 + \int_0^t (L_v\|e_{j,u}\|_2 + \|r_j(u)\|_2)du.$$

Gronwall's inequality with this nonzero initial error yields

$$\|F_\psi(s, c_{\mathrm{dep}}(s)) - y_j\|_2 \leq e^{L_v}\|c_{\mathrm{dep}}(s) - x_K(s)\|_2 + \int_0^1 e^{L_v(1-t)}\|r_j(t)\|_2 dt. \tag{30}$$

By Cauchy–Schwarz in time,

$$\int_0^1 e^{L_v(1-t)}\|r_j(t)\|_2 dt \leq A(L_v)\left(\int_0^1 \|r_j(t)\|_2^2 dt\right)^{1/2}.$$

Taking the weighted L^2 norm over states, candidate sets, and $j = 1, \dots, N_{\text{cand}}$, then applying Minkowski's inequality, gives

$$\left(\mathbb{E} \sum_{j=1}^{N_{\text{cand}}} w_j \| F_\psi(s, c_{\text{dep}}(s)) - a_j^\star(s) \|_2^2 \right)^{1/2} \leq e^{L_v} \delta_K + A(L_v) \sqrt{\mathcal{E}_K(\psi)}.$$

Normalization of the weights makes the norm of the shared initial-error term exactly $e^{L_v} \delta_K$. Weighted Minkowski applied between the numerical and exact endpoints adds ϵ_{num}, proving Eq. (23).

For any comparator $b = b(s, \xi)$, Lipschitz continuity and normalized nonnegative weights imply

$$\bar{Q}(s, \widehat{a}_B) - \bar{Q}(s, b) \geq \sum_{j=1}^{N_{\text{cand}}} w_j [\bar{Q}(s, a_j^\star) - \bar{Q}(s, b)] - G \sum_{j=1}^{N_{\text{cand}}} w_j \| \widehat{a}_B - a_j^\star \|_2.$$

Averaging and applying Cauchy–Schwarz under the normalized weights yields Eq. (25). No independence between the comparator and targets is required beyond a joint distribution with the stated marginals. □

A.2 Proof of Corollary 2

Proof. Apply Eq. (25) with $b = a_C = c_{\text{dep}}$ and $b = a_R$. The stated condition makes both lower bounds strictly positive. For Eq. (28), insert and subtract $\bar{Q}(s, x_K(s))$ in m_C. Lipschitz continuity and Cauchy–Schwarz give

$$\mathbb{E}[\bar{Q}(s, x_K(s)) - \bar{Q}(s, c_{\text{dep}}(s))] \geq -G \delta_K.$$

Therefore the deployed-over-coarse advantage is at least

$$\Delta_K - G \left[A(L_v) \sqrt{\mathcal{E}_K(\psi)} + (e^{L_v} + 1) \delta_K + \epsilon_{\text{num}} \right],$$

which is positive under Eq. (28). □

B Algorithm

We denote by $\text{ODESolve}(v_\psi, s, a_0)$ the numerical integration of $\frac{da_t}{dt} = v_\psi(s, a_t, t)$ from $t = 0$ to $t = 1$, with initial condition a_0.

In implementation, ODESolve is realized by Euler integration:

$$a_{n+1} = a_n + \Delta t \, v_\psi(s, a_n, t_n), \quad \Delta t = \frac{1}{N}, \quad n = 0, \dots, N-1.$$

Algorithm 1: BiFlow Training

Input: Replay buffer $\mathcal{D}$, coarse actor c_ϕ, refiner v_ψ, critics $Q_{\theta_1}, Q_{\theta_2}$, inner steps K_{in}, candidates N.

for *each training iteration* **do**

Collect transition with $a = \mu_{\psi,\phi}(s) + \epsilon$ and store it in $\mathcal{D}$

Sample mini-batch $\mathcal{B}$ from $\mathcal{D}$

`// Critic update`

Construct the target action

$$\tilde{a}' = \text{clip}\big(\mu_{\psi,\phi}(s') + \epsilon', -1, 1\big).$$

Compute the TD target

$$y = r + \gamma(1-d)\min_i Q_{\bar{\theta}_i}(s', \tilde{a}').$$

Update $Q_{\theta_1}, Q_{\theta_2}$ by Bellman regression and set

$$\bar{Q}(s,a) = \min_i Q_{\theta_i}(s,a).$$

`// Inner lookahead coarse adaptation`

Set

$$\phi^{(0)} = \phi.$$

for $k = 0, \ldots, K_{\text{in}} - 1$ **do**

$$\phi^{(k+1)} = \phi^{(k)} + \eta_\phi \nabla_{\phi^{(k)}} \mathbb{E}_{s\sim\mathcal{B}}\left[\bar{Q}\big(s, c_{\phi^{(k)}}(s)\big)\right].$$

Define the lookahead source

$$a_{\text{src}}(s) = c_{\phi^{(K_{\text{in}})}}(s).$$

`// Q-guided target construction`

foreach $s \in \mathcal{B}$ **do**

Sample N candidates around $a_{\text{src}}(s)$ and improve them by action-gradient ascent on $\bar{Q}$

Obtain targets $\{a_j^\star(s)\}_{j=1}^N$

Compute

$$\Delta_j(s) = \bar{Q}(s, a_j^\star(s)) - \bar{Q}(s, a_{\text{src}}(s)).$$

Set weights by softmax over positive improvements:

$$w_j(s) = \mathbf{1}[\Delta_j(s) > 0]\frac{\exp(\Delta_j(s)/T_\alpha)}{\sum_{k:\Delta_k(s)>0}\exp(\Delta_k(s)/T_\alpha)}.$$

If no $\Delta_j(s) > 0$, use the stay target $a^\star = a_{\text{src}}$ with weight 1

`// First-order refiner update`

Sample $t \sim U[0,1]$ and set

$$a_{j,t} = (1-t)a_{\text{src}}(s) + t a_j^\star(s).$$

Update ψ by minimizing

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{s,t}\sum_j w_j(s)\left\|v_\psi(s, a_{j,t}, t) - \big(a_j^\star(s) - a_{\text{src}}(s)\big)\right\|_2^2.$$

Treat a_{src} and $a_j^\star$ as detached when computing the refiner update; no gradient from $\mathcal{L}_{\text{FM}}$ is propagated through the inner adaptation trajectory

`// First-order retained coarse meta-update (FOMAML)`

Using the current updated refiner ψ, compute

$$g_\phi^{\text{FO}} = \nabla_{\phi^{(K_{\text{in}})}}\mathbb{E}_{s\sim\mathcal{B}}\left[\bar{Q}\Big(s, F_\psi\big(s, c_{\phi^{(K_{\text{in}})}}(s)\big)\Big)\right].$$

Apply this first-order meta-gradient to the retained coarse parameters:

$$\phi \leftarrow \phi + \eta_\phi g_\phi^{\text{FO}}.$$

Do not backpropagate through the K_{in} inner updates when computing g_ϕ^{FO} (i.e., use the FOMAML approximation)

Polyak update target critics

return $a = \text{ODESolve}(v_\psi, s, c_\phi(s))$.

C HYPERPARAMETER SETTINGS.

The default settings used in the experiments are summarized in Appendix Tables 4 and 5.

Toy example details. We construct a two-dimensional state-conditioned bandit to visualize the role of the lookahead source. The initial action is sampled from a Gaussian distribution centered at $(0, 8)$ with standard deviation 0.70. The action space contains three high-value target modes located at $(-4, -7)$, $(0, -7)$, and $(4, -7)$, and each target action is sampled around its corresponding mode with standard deviation 0.35. The state $s \sim \mathcal{N}(0, 1)$ determines the optimal mode using thresholds -0.43 and 0.43. The toy Q-function is defined as

$$Q(s, a) = -\frac{1}{2}\|a - \mu_s\|_2^2,$$

where μ_s denotes the target center selected by state s. For BiFlow, the lookahead source is obtained by $K = 5$ lookahead steps from the initial action with step size 2.1, followed by an 8-step Euler flow refinement. The single-stage FM baseline uses the same number of Euler steps but directly transports the initial action to the target action. Both flow models use a three-layer MLP velocity field with hidden dimension 128, and are trained for 3000 steps with batch size 512 and learning rate 2×10^{-3}.

Table 4: Default BiFlow algorithm hyperparameters. These values specify the coarse-to-fine optimization, Q-guided target construction, and flow-matching refiner used in all experiments unless otherwise stated.

Hyperparameter	Default value	Description
Inner ascent steps K_{in}	5	Number of temporary coarse-policy ascent steps used to construct the lookahead source $a_{\text{src}}(s) = c_{\phi^{(K_{\text{in}})}}(s)$.
Inner coarse-policy learning rate η_ϕ	5×10^{-4}	Learning rate for the coarse/source actor update.
Candidate action count N_{cand}	32	Number of perturbed candidate actions sampled around each lookahead source.
Candidate perturbation scale σ_c	0.03	Standard deviation of the Gaussian perturbation used to initialize local candidate actions around the lookahead source.
Action-space Q-ascent steps L	5	Number of gradient-ascent steps applied to each candidate action during Q-guided local target construction.
Action-space ascent learning rate η_a	10^{-3}	Step size for improving candidate actions by ascending the critic aggregate in action space.
Value-aware weighting temperature T_α	initialized at 1.0; clipped to $[0.15, 1.0]$	Temperature used in the softmax over positive critic improvements in the value-aware target weighting rule. The temperature is adaptive during training.
Refiner learning rate η_ψ	9×10^{-5}	Learning rate for the conditional flow-matching refiner.
Flow refinement steps	8	Number of numerical integration steps used when refining the coarse action into the executed action.
Flow-matching hidden width	256	Hidden width of the flow-matching refiner network.
Critic and shared hidden layers	3 layers, width 256	Network size used by the critic and shared policy components.
Coarse/source actor hidden layers	2 layers, width 256	Network size of the coarse source actor that produces the initial action proposal.

Table 5: Default training and evaluation hyperparameters. These values define the online RL optimization and evaluation protocol used by BiFlow unless otherwise stated.

Hyperparameter	Default value	Description
Critic learning rate	3×10^{-4}	Learning rate for the twin Q-functions used in Bellman regression and Q-guided target construction.
Discount factor γ	0.99	Discount factor used in the Bellman target.
Target-network Polyak rate τ	0.005	Exponential moving-average rate for updating target critics and target policy components.
Reward scale	0.2	Multiplicative scale applied to rewards before critic updates.
Batch size	256	Number of transitions sampled from the replay buffer for each training update.
Initial random interaction steps	30,000	Number of warm-up environment steps collected with random actions before policy updates dominate data collection.
Total environment steps	1,000,000	Default interaction budget for most tasks. Dog-walk and Dog-run use 2,000,000 steps due to their longer-horizon locomotion difficulty.
Evaluation checkpoints	20	Number of uniformly spaced evaluation checkpoints during training.
Exploration noise scale	0.1	Standard deviation of Gaussian exploration noise added to the refined action during online data collection.
Target smoothing noise standard deviation and clip	0.2, 0.5	TD3-style target smoothing noise applied when constructing critic targets; the noise is clipped to the specified range.

D Baseline Implementation Details

To ensure a fair and reproducible comparison, all baseline results reported in Table 1 and Figure 2 are obtained by rerunning the corresponding methods under the same evaluation protocol as BiFlow. Specifically, each method is evaluated with five random seeds, 20 uniformly spaced evaluation checkpoints, and 20 evaluation episodes per checkpoint.

For TD3 (Fujimoto et al., 2018), QVPO (Ding et al., 2024), QSM (Psenka et al., 2023), DACER (Wang et al., 2024), SDAC, and DPMD (Ma et al., 2025), we use the public Flow-RL codebase[1]. For QFlex (Wei et al., 2026), we use the official implementation released by the authors[2]. Unless otherwise stated, we keep the default algorithm-specific hyperparameters provided by the corresponding public implementations, including optimizer settings, batch size, update frequency, and method-specific diffusion or flow parameters.

All methods are run on the same benchmark environments and under the same environment-level settings, including the task versions, action range, evaluation procedure, training budget, and random-seed protocol. For most tasks, we train each method for 1M environment steps; for Dog-walk and Dog-run, we use 2M environment steps due to their longer-horizon locomotion difficulty. All methods are executed on the same compute server and use a single NVIDIA H800 GPU for training.

[1] https://github.com/typoverflow/flow-rl/

[2] https://github.com/LNSGroup/Qflex