Conformal Constrained Policy Optimization for Cost-Effective LLM Agents

Wenwen Si, Sooyong Jang, Insup Lee, Osbert Bastani

PRECISE Center, Department of Computer and Information Science
University of Pennsylvania
{wenwens, sooyong, lee, obastani}@seas.upenn.edu

Abstract

While large language models (LLMs) have recently made tremendous progress towards solving challenging AI problems, they have done so at increasingly steep computational and API costs. We propose a novel strategy where we combine multiple LLM models with varying cost/accuracy tradeoffs in an agentic manner, where models and tools are run in sequence as determined by an orchestration model to minimize cost subject to a user-specified level of reliability; this constraint is formalized using conformal prediction to provide guarantees. To solve this problem, we propose Conformal Constrained Policy Optimization (CCPO), a training paradigm that integrates constrained policy optimization with off-policy reinforcement learning and recent advances in online conformal prediction. CCPO jointly optimizes a cost-aware policy (score function) and an adaptive threshold. Across two multi-hop question answering benchmarks, CCPO achieves up to a 30% cost reduction compared to other cost-aware baselines and LLM-guided methods without compromising reliability. Our approach provides a principled and practical framework for deploying LLM agents that are significantly more cost-effective while maintaining reliability.

Extended version — https://arxiv.org/pdf/2511.11828

1 Introduction

While large language models (LLMs) have made tremendous progress towards solving challenging tasks, they often require significant cost to do so. Human decision-makers can reduce cost by employing a meta-level strategy where they monitor their own uncertainty, seek targeted help, and choose to try again precisely when the potential benefit outweighs the cost of additional attempts. Bringing this capability to LLM agents is essential for cost-effective deployment. LLMs range from smaller LLMs that are cheap but inaccurate, to larger LLMs that can be far more accurate but significantly more costly. A natural strategy is to use a small, cheap LLM when it is confident, switching to the large LLM otherwise to ensure reliability. Beyond this simple strategy, we can consider more complex combinations of the two LLMs based on the intermediate results they produce.

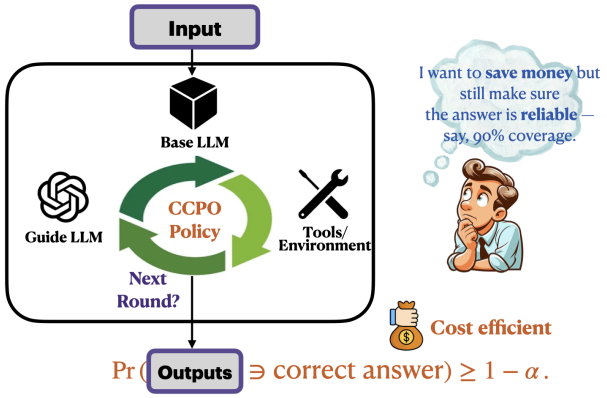

Figure 1: We propose a framework for training a policy designed to orchestrate a fast, inaccurate LLM agent and a slow, accurate LLM to minimize cost while maintaining reliability (as formalized by conformal prediction).

Specifically, we consider an agentic setting, where a policy orchestrates a small LLM and a large LLM across a series of rounds to answer the question, with the goal of minimizing the expected cost subject to a user-specified reliability level. To formalize the reliability constraint, we use conformal prediction (Shafer and Vovk 2008), which provides theoretical guarantees by modifying the system to output sets of labels (called *prediction sets*) instead of individual labels. Conformal prediction provides the *coverage guarantee*, which says the prediction set contains the true label with high probability. In our setting, we can only cover the true answer if some orchestration strategy does so; thus, our reliability constraint says the following holds with high probability: if some sequence of actions produces the correct answer, then our prediction set contains the true label.

Thus, the orchestration policy must select among LLM agents to satisfy the coverage guarantee while minimizing average cost (we implicitly constrain prediction set size). Prior work has addressed only parts of this goal. For instance, chain-of-thought prompting (Wei et al. 2022), ReAct (Yao et al. 2023), debate protocols, and RL-based LLM agent frameworks (Zhou et al. 2025; Bensal et al. 2025) improve accuracy but ignore cost. Alternatively, FrugalGPT (Chen, Zaharia, and Zou 2024) reduces API expenses

but does not guarantee reliability. Finally, uncertainty-aware methods (Ren et al. 2023; Han, Buntine, and Shareghi 2024) calibrate model confidences but rely on ad-hoc techniques to perform model selection and do not seek to minimize cost. No existing approach unifies these desiderata.

We propose *Conformal Constrained Policy Optimization (CCPO)*, which combines recent advances in online conformal prediction (Angelopoulos, Barber, and Bates 2024) with constrained policy optimization (Achiam et al. 2017). Specifically, we seek to train a conformal policy that outputs a prediction set over actions to minimize cost subject to a conformal constraint; an optional penalty can also be imposed on prediction set size. However, training a set-based action policy is infeasible; instead, we convert the conformal policy into a stochastic one (i.e., uniform distribution over the action set) and train this policy instead. A key challenge is that the conformal policy visits a different state-action distribution compared to the stochastic policy; to correct for this shift, we use importance weighted V-trace targets and policy gradients (Espeholt et al. 2018). Finally, we apply online conformal prediction (Angelopoulos, Barber, and Bates 2024) to scale the action sets to achieve coverage.

We empirically evaluate our approach on the free-form HotpotQA (Yang et al. 2018) and multiple-choice MMLU (Hendrycks et al. 2021) datasets, showing that CCPO reduces total cost by up to 30% while satisfying target coverage guarantees, outperforming state-of-the-art cost-minimization baselines.

To summarize, our contributions are three-fold. First, we propose a formal framework for cost-effective and reliable LLM-agent deployment. Second, we apply V-trace off-policy corrections between the behavioral score function and the conformal-wrapped target policy, thereby sidestepping the exponential search over action sets. That is, rather than enumerating all set-valued mappings, we optimize stochastic surrogates that softly enforce both cost and coverage objectives and allow efficient gradient updates. Third, across diverse models and datasets, our method consistently achieves the lowest cost while meeting the target coverage level, demonstrating its practical effectiveness.

2 Related Work

LLM Agents. One way to improve the accuracy of LLM agents is to instruct them to perform reasoning. Chain-of-Thought (CoT) explicitly elicits step-by-step explanations (Wei et al. 2022), and ReAct interleaves reasoning with tool calls (Yao et al. 2023). Extensions incorporate uncertainty quantification—e.g., UALA estimates predictive variance before committing to an answer (Han, Buntine, and Shareghi 2024). They also propose to reduce cost by making early-exit decisions in multi-turn chains (Lu et al. 2025) and role-based multi-agent editing pipelines (Wan et al. 2025).

Rather than using hand-written prompts, an alternative strategy is to train control policies. Min et al. (2025) first trains a Process Reward Model to score each intermediate thought and then learns a binary tabular policy—"request help" vs. "proceed". SWEET-RL (Zhou et al. 2025) tackles multi-turn tasks by using external feedback as contextual states, yielding better credit assignment. Bensal et al. (2025) leverages a self-reflection mechanism for self-improvement. However, while these methods improve reliability or fluency, they do not provide any guarantees. In contrast, we aim to provide reliability guarantees using conformal prediction.

Safe & Off-Policy RL. There has been work on imposing safety constraints in reinforcement learning. TRPO constrains each update by a KL trust region (Schulman et al. 2015); CPO augments this approach with an explicit cost bound for one-step feasibility (Achiam et al. 2017). Shielding methods block actions that violate specifications (Alshiekh et al. 2018), while Lagrangian penalties trade reward and risk (Tessler, Mankowitz, and Mannor 2019). Bastani, Li, and Xu (2021) combines a learned policy with a backup policy, sacrificing reward when necessary to ensure safety.

A complementary line of work tackles stable learning from off-policy data. Retrace clips importance ratios for stable Q-learning from replay (Munos et al. 2016); IMPALA's V-trace extends this to distributed actor–learner systems (Espeholt et al. 2018). In purely offline data, CQL (Kumar et al. 2020) and IQL (Kostrikov, Nair, and Levine 2022) suppress over-optimistic estimates with regularization.

Online Conformal Prediction. Online conformal prediction extends classical, exchangeability-based CP to sequential or adversarial data streams by tracking the conformity threshold, largely through online optimization. ACI (Gibbs and Candes 2021) introduces an adaptive quantile estimator that preserves marginal coverage when the underlying distribution evolves. MVP (Bastani et al. 2022) achieves adversarially robust, group-conditional ("multivalid") coverage efficiently with tight prediction sets. Angelopoulos, Barber, and Bates (2024) propose a weight-decayed quantile tracker that gives online conformal prediction "best-of-both-worlds" guarantees—$O(\sqrt{T})$ coverage regret for adversarial streams and near-optimal set sizes for i.i.d. data.

Learning Conformal Score Functions. Stutz et al. (2022) proposes a method to simulate split-conformal prediction within each mini-batch and backpropagate through the procedure, enabling the training of optimal score functions for classification problems. A final batch calibration step restores the coverage guarantee, yielding tighter confidence sets than conventional post-hoc conformal prediction. However, optimizing the conformal score function remains unexplored in sequential settings such as ours.

3 Cost-Effective LLM Agents

We propose a collaborative LLM-agent framework where uncertainty-guided decisions govern interactions among multiple LLMs (possibly augmented with external tools) to solve a task. Specifically, we consider a question-answering task with question-answer pair distribution $(Q, Y^*) \sim \mathcal{D}$.

LLM Agent Orchestration. We assume given both (i) a *base agent*, which is relatively cheap (e.g., a small open-source LLM) but with weaker performance, and (ii) a *guide agent*, which offers substantially better capabilities (e.g., more accurate answers, better reasoning skills) and uncertainty measurements at a higher cost. Intuitively, we want

to switch to the guide agent if the base agent is uncertain. However, the base agent may not be effective at estimating its own uncertainty. Thus, we make use of the fact that input tokens tend to be substantially cheaper than output tokens, which enables us to have the guide agent assess confidence based on the base agent's reasoning trace. Specifically, we consider the following orchestration strategy: on each step (i) run the base agent to generate a reasoning trace and answer, (ii) run the guide agent to evaluate the base agent's generation and produce a correct answer, and (iii) have the policy to decide the next action (choose the base answer, guide answer, or continue for another round). Note that (ii) is cheap since it mostly uses input tokens and very few output tokens. This process is repeated until the policy chooses an answer or a maximum number of rounds is reached.

In more detail, the guide model is given the question along with the base model's reasoning chain and answer (for the current round), and outputs a binary judgment (yes/no) and corrected answer (without performing any reasoning itself). The base model's context includes the reasoning traces and answers from previous rounds and the guide model's corresponding output. When the base model is LLaMA-2-7B and the guide model is GPT-4o, we find that this strategy achieves comparable or even stronger performance than using GPT-4o with the chain-of-thought reasoning.

POMDP formulation. We formalize this strategy as a finite-horizon Partially Observable Markov Decision Process (POMDP) with T steps. At each step, we run the base model on the context so far, and the policy π receives observation $o_t \in \mathcal{O}$, which includes the base model's context, the guide model's judgment and uncertainty, the index of the current round, and cumulative guide model token usage. Then, the policy selects an action $a_t \in \mathcal{A}$, where

$$\mathcal{A} = \{\text{guide answer}, \text{base answer}, \text{next round}\}.$$

This process continues until the policy chooses an answer, at which point the episode terminates. The goal is to learn a policy that achieves a (i) *coverage guarantee*—i.e., a set of answers generated by the orchestration process contains the true answer if there is some sequence of actions that produces the true answer, and (ii) *cost minimization*—i.e., minimize the total monetary cost (e.g., API usage fees). To this end, we introduce the *conformal policy* $C : \mathcal{O} \to 2^{\mathcal{A}}$, which maps an observation $o_t \in \mathcal{O}$ to an action set $A_t = C(o_t) \subseteq \mathcal{A}$. Rather than producing a single rollout, this strategy produces a set of rollouts by taking every possible action $a \in A_t$, and recursively collecting rollouts across each. Since only the action "next round" results in additional choices, the maximum number of rollouts is $2T$ (where T is the horizon), so we do not suffer exponential blowup.

Next, let $\mathcal{Y}(q)$ denote the set of answers that can arise from all possible action sequences—i.e., letting the base and guide answers on round t be $\hat{y}^t_{\text{base}}, \hat{y}^t_{\text{guide}} \in \mathcal{Y}$, respectively, where $\mathcal{Y}$ is the answer space (here, we assume the policy proceeds to round T), then $\mathcal{Y}(q) = \{\hat{y}^t_{\text{base}}(q)\}_{t=1}^T \cup \{\hat{y}^t_{\text{guide}}(q)\}_{t=1}^T$. Now, we aim to solve the following:

$$\begin{aligned} &\min_C \mathbb{E}[J(C, Q) + \lambda |C(Q)|] \\ &\text{s.t. } \Pr[\mathbb{1}\{Y^* \in C(Q) \vee Y^* \notin \mathcal{Y}(Q)\}] \geq 1 - \alpha, \end{aligned} \tag{1}$$

where $J(C, q)$ denotes the (random) cumulative reward of C given question q (in our setting, it is the cumulative API cost incurred in that episode), $C(Q) \in 2^{\mathcal{Y}}$ denotes the (random) set of answers obtained by applying C to Q, α is the user-specified reliability level, and λ is a hyperparameter. The objective optionally includes the added term $\lambda|C(Q)|$ to impose a penalty on larger prediction sets; since the structure of our problem imposes the constraint $|C(Q)| \leq 2T$ (where T is the horizon), we can also take $\lambda = 0$. The constraint says that with high probability, if $Y^* \in \mathcal{Y}(Q)$, then $Y^* \in C(Q)$ as well—i.e., if there is any chance of getting the correct answer Y^*, then the prediction set $C(Q)$ includes Y^*.

4 Conformal CPO

We propose Conformal Constrained Policy Optimization (CCPO) to solve Eq. (1). The key challenge is that directly searching over all conformal policies C is challenging due to the combinatorial action space $2^{\mathcal{A}}$. Instead, we parameterize C by a stochastic policy $\pi(a \mid o)$ and a threshold $\kappa \in [0, 1]$; the corresponding conformal policy is

$$C_{\pi,\kappa}(o) = \{a \in \mathcal{A} : \pi(a \mid o) \geq \kappa\}. \tag{2}$$

Then, by updating κ using online conformal prediction and assuming π converges, we guarantee that $C_{\pi,\kappa}$ will asymptotically achieve the desired $1 - \alpha$ coverage.

To update π, we adapt the trust-region method from CPO (Achiam et al. 2017), yielding consistent improvement in our objective while maintaining our coverage constraint. Taking into account the KL-ball search in CPO and the contractive validity of V-trace (see Section 6), we optimize the stochastic conformal policy $S_{\pi,\kappa}$ as the target policy. Concretely, for any score function π and threshold κ, we define

$$S_{\pi,\kappa}(a \mid o) = \frac{\mathbb{1}\{\pi(a \mid o) \geq \kappa\}}{\sum_{a' \in \mathcal{A}} \mathbb{1}\{\pi(a' \mid o) \geq \kappa\}} \tag{3}$$

i.e., $S_{\pi,\kappa}$ places uniform probability over the actions in the conformal set $C_{\pi,\kappa}(o)$. We then employ an actor–critic architecture in which both the value function V and constraint function V_C (i.e., the value function with the reward replaced by the constraint value), and the policy updates are computed with respect to the stochastic conformal policy $S_{\pi,\kappa}$.

Critic update. To bridge the gap between π (which is used to collect trajectories) and $S_{\pi,\kappa}$, we employ V-trace off-policy correction as in Espeholt et al. (2018). At each iteration, rollouts under π produce tuples (o_t, a_t, r_t, c_t), where r_t and c_t are the reward and constraint values on step t, respectively. In our POMDP in Eq. (1), we have reward values

$$r_t = J_t(C, q) + \mathbb{1}\{t = t_f\} \cdot \lambda |C(q)|,$$

where $J_t(C, q)$ is the API cost on step t and t_f is the step on which the policy π selects an answer, and constraint values

$$c_t = \mathbb{1}\{t = t_f\} \cdot \mathbb{1}\{y^* \in C(q) \vee y^* \notin \mathcal{Y}(q)\}.$$

Now, given $\bar{\rho} > 0$, we define truncated importance weights

$$\rho_t = \min\left\{\bar{\rho}, \frac{S_{\pi,\kappa}(a_t \mid o_t)}{\pi(a_t \mid o_t)}\right\}$$

Algorithm 1: Conformal Constrained Policy Optimization

Input: Initial behavior policy π_0, initial threshold κ_0, user-specified reliability level α, dataset $D = \{(q_i, y_i)\}_i$.
Output: Conformal policy $C_{\pi,\kappa}$.

1: **for** $k \in \{0, 1, ...\}$ **do**
2: Sample $(q, y) \sim D$ and collect rollouts with π
3: Update value/constraint function estimates via Eq. (5)
4: Compute π_{k+1} via Eq. (7)
5: Compute κ_{k+1} via Eq. (9)
6: **end for**

to correct for the discrepancy between π and $S_{\pi,\kappa}$. We use $\bar{\rho} = 1$, which clips large weights but not small ones; see Section 6 for a discussion of our choice. Then, the V-trace target for the value function estimate V_θ is

$$v_t = V_\theta(o_t) + \delta_t V + \rho_t(v_{t+1} - V_\theta(o_{t+1})) \quad (4)$$
$$\delta_t V = \rho_t(r_t + V_\theta(o_{t+1}) - V_\theta(o_t)),$$

and update θ via gradient descent on the L_2 loss to v_t:

$$\theta \leftarrow \theta - \sum_{t=1}^{T} \nu(v_t - V_\theta(o_t))\nabla_\theta V_\theta(o_t), \quad (5)$$

where ν is the learning rate. We apply the same gradient update rule to train a constraint function estimate $V_{C,\phi}$, where r_t is replaced by c_t. In addition, for each sampled action, we obtain the advantage function

$$\hat{A}_t^{S_{\pi,\kappa}} = r_t + v_{t+1} - V_\theta(o_t), \quad (6)$$

and analogously for the constraint advantage function $\hat{A}_{C,t}^{S_{\pi,\kappa}}$ by replacing r_t by c_t and V_θ with $V_{C,\phi}$.

Policy Update. Next, we describe our trust–region policy update. We let $S_k = S_{\pi_k,\kappa_k}$ and $C_k = C_{\pi_k,\kappa_k}$ denote the stochastic and conformal policies, respectively, on the kth iteration of optimization. Then, as in Achiam et al. (2017), we iteratively solve the following approximation of Eq. (1) (we describe how κ is updated later):

$$\min_\pi \mathbb{E}\left[\sum_{t=1}^{T} \hat{A}_t^{S_{\pi,\kappa}}\right] \quad (7)$$
$$\text{s.t. } \bar{J}_C^{\pi,\kappa} \geq 1 - \alpha, \quad D_{\mathrm{KL}}(S_{\pi,\kappa} \| S_k) \leq \delta.$$

Here, we have used the fact that our original objective is equivalent to the sum of advantages $A_t^{S_{\pi,\kappa}}$; furthermore, the constraint is replaced by a constraint $\bar{J}_C^{\pi,\kappa} \geq 1-\alpha$ and a KL constraint that keeps the new policy close to the current one, where $\bar{J}_C^{\pi,\kappa}$ is an upper bound on the constraint value:

$$\bar{J}_C^{\pi,\kappa} \geq J_C^{\pi,\kappa} = \Pr[Y^* \in C_{\pi,\kappa}(Q) \vee Y^* \notin \mathcal{Y}(Q)].$$

To derive $\bar{J}_C^{\pi,\kappa}$, first note that

$$J_C^{\pi,\kappa} \leq \Pr[Y^* \in C_{\pi,\kappa}(Q)] + 1 - \Pr[Y^* \in \mathcal{Y}(Q)].$$

The second term is easy to estimate; for the first, we have

$$\begin{aligned}
&\Pr[Y^* \in C_{\pi,\kappa}(Q)] \\
&\leq \mathbb{E}_S\left[\left(\prod_{t=1}^{T} |C_{\pi,\kappa}(o_t)|\right) \mathbb{1}\{S_{\pi,\kappa}(Q) = Y^*\}\right] \\
&\approx \mathbb{E}_\pi\left[\left(\prod_{t=1}^{T} \rho_t \cdot |C_{\pi,\kappa}(o_t)|\right) \mathbb{1}\{S_{\pi,\kappa}(Q) = Y^*\}\right]
\end{aligned}$$

where $S_{\pi,\kappa}(Q)$ is the random answer obtained by using stochastic policy $S_{\pi,\kappa}$ in the POMDP on question Q; the approximation in the second line comes from the fact that we are clipping our importance weights ρ_t. Thus, we use

$$\bar{J}_C^{\pi,\kappa} = \bar{J}_{C,0}^{\pi,\kappa} + 1 - \Pr[Y^* \in \mathcal{Y}(Q)]$$
$$\bar{J}_{C,0}^{\pi,\kappa} = \mathbb{E}_\pi\left[\left(\prod_{t=1}^{T} \rho_t \cdot |C_{\pi,\kappa}(o_t)|\right) \mathbb{1}\{S_{\pi,\kappa}(Q) = Y^*\}\right].$$

As in CPO, we then compute the optimal Lagrange multiplier for the constraint, and use conjugate gradient descent to solve for the natural gradient step that satisfies the KL radius δ; the resulting step gives the new policy π_{k+1}.

A remaining caveat is that the definition of $S_{\pi,\kappa}$ in Eq. (3) involves the indicator function, which is non-differentiable. Thus, we replace it with a smooth sigmoid approximation; given $\epsilon > 0$, we instead use a softmask function

$$\text{softmask}(a, o; \kappa) = \sigma\left(\frac{\pi(a \mid o) - \kappa}{\epsilon}\right), \quad (8)$$

where $\sigma(z) = (1 + e^{-z})^{-1}$ is the sigmoid function. As $\epsilon \to 0$, $\text{softmask}(a, o; \kappa)$ recovers the binary indicator.

Threshold Calibration. The threshold κ is updated after each episode to provably maintain the coverage guarantee without relying on convergence of the policy. When solving for π_{k+1} in Eq. (7), we fix $\kappa = \kappa_k$. Afterwards, we update κ_k based on the coverage of C_{π_{k+1},κ_k} using the algorithm in Angelopoulos, Barber, and Bates (2024), that κ_{k+1} is

$$\kappa_k + \eta_k\left(1 - \mathbb{1}\{Y^* \in C_{\pi,\kappa}(Q) \vee Y^* \notin \mathcal{Y}(Q)\} - \alpha\right). \quad (9)$$

Assuming $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$, then Angelopoulos, Barber, and Bates (2024, Theorem 4 and Corollary 2) show that in the i.i.d. setting, the coverage converges to $1 - \alpha$ even though π is being updated in a streaming fashion, as long as it converges. If additionally $\eta_t \propto t^{-1/2-\xi}$ for some $\xi \in (0, 1/2)$, then in the adversarial setting the long-run coverage error is bounded as $O(T^{-1/2+\xi})$ (Angelopoulos, Barber, and Bates 2024, Theorem 1). While we guarantee coverage for the final policy π_K assuming π_k converges, we perform a batch calibration on π_K (i.e., use traditional conformal prediction (Shafer and Vovk 2008) on a held-out calibration set to select a final value κ_{K+1}) to ensure coverage for π_K even without this assumption.

5 Experiments

5.1 Experimental Setup

Environment. We set the horizon to $T = 4$ since questions that cannot be solved in this time are generally too difficult. For conformal prediction, we use $\alpha \in \{0.1, 0.2\}$.

Policy	Base-Guide	α-aware	Conf-ormal	Point-wise	RL	LLM Policy
LLM-EXIT				✓		✓
Random	✓			✓		
GPT-4o/ LLaMA	✓	✓		✓		✓
UALA	✓					
CPO	✓	✓		✓	✓	
CPO batch/ online	✓	✓	✓		✓	
CCPO (ours)	✓	✓	✓		✓	

Table 1: Summary of key features of each approach.

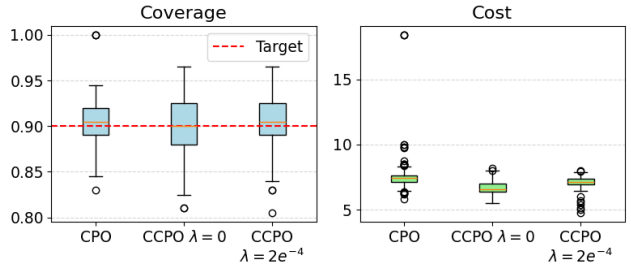

Figure 2: HotpotQA coverage and cost with LLaMA-2-7B base model over 100 splits, $\alpha = 0.1$.

Training. All our policy networks are three-layer neural networks with 64 hidden units per layer. We train them from scratch for 1500-2000 steps, using a learning rate of 10^{-3} and batch size of 10. We set the KL divergence constraint to $\delta = 0.01$, soft masking parameter $\epsilon = 0.01$, and conformal decay step $\xi = 0.1$. For optimizing κ, we perform a grid search over the threshold with a granularity of 10^{-6}.

Datasets. We evaluate our approach on two challenging question-answering benchmarks. HotpotQA (Yang et al. 2018) is a commonsense QA dataset requiring multi-hop reasoning and often multiple Wikipedia searches to answer each question. MMLU (Hendrycks et al. 2021) is a multiple-choice QA dataset spanning academic and professional subjects such as biology, mathematics, and physics. Following UALA (Han, Buntine, and Shareghi 2024), we use Wikipedia search as a tool for HotpotQA. For MMLU, UALA uses the Google API as a tool. However, the cost of Google API is $5 per 1,000 calls, which far exceeds our budget; thus, we apply CoT with deterministic decoding (temperature 0.0) in the first step and temperature 1.0 in the subsequent steps. For HotpotQA, we train on 1,000 examples, while for MMLU, we train on 560 examples. For both, we use 200 examples for batch calibration and 200 for testing.

Models. We consider both LLaMA-2-7B (8-bit) and LLaMA-3.2-3B as base models, and use GPT-4o as the guide model. However, because LLaMA-3.2-3B achieves notably stronger performance in some cases, we deliberately set a stricter target ($\alpha = 0.05$) for it to avoid trivial results.

Metrics. We evaluate the performance of our method using four primary metrics. "Cost" measures the cumulative cost of API usage over the test set. "Coverage" measures the coverage rate $\Pr[Y^* \in C(Q) \vee Y^* \notin \mathcal{Y}(Q)]$. "Avg. Len." is the average episode length (i.e., the average number of steps taken before the policy chooses an answer; for conformal policies, it is the maximum length across branches). "Set Size" is the average prediction set size $|C(q)|$.

Policy	Cost (cents)	Coverage	Avg. Len.	Set Size
GPT-4o EXIT	827.0	0.908	2.39	2.39
LLaMA-2 EXIT	0.000	0.653	2.55	2.55
Random	6.811	0.578	1.44	1.00
GPT-4o	18.65	0.780	1.20	1.00
LLaMA-2-7B	4.551	0.675	1.03	1.00
UALA	9.153	0.923	2.00	2.00
CPO	4.704	0.832	1.004	1.00
CPO batch	7.835	0.905	1.70	2.19
CPO online	7.484	0.897	1.58	2.38
CCPO ($\lambda = 0$)	**6.552**	0.902	**1.34**	2.35
CCPO ($\lambda = 2e^{-4}$)	7.026	0.903	1.48	2.22

Table 2: HotpotQA results with LLaMA-2-7B, $\alpha = 0.1$.

Baselines. We consider two kinds of baselines: LLM agents and CPO methods. First, we summarize our LLM agent baselines:

- Random policy: We apply a uniform policy π that randomly selects among the three actions in $\mathcal{A}$.
- LLM policy: We replace the parameterized policy π with an LLM, supplying it with the question, each model's answer, and uncertainty scores derived from GPT-4o. We consider using both of the LLaMA models as well as GPT-4o; these are denoted as GPT-4o, LLaMA-3.2-3B, and LLaMA-2-7B, in our results.
- LLM-EXIT (Lu et al. 2025): This baseline uses a single LLM both to provide answers and to decide whether to proceed to the next round. [1]
- GPT-4o-guided UALA: In UALA (Han, Buntine, and Shareghi 2024), both uncertainty estimates and answers are generated by the same model, which tend to be biased and inaccurate for smaller models. To align with our approach, we adopt GPT-4o to quantify uncertainty. Then, this baseline follows the same steps as our approach, except instead of using a learned policy, it applies fixed thresholds on uncertainty to choose the action.

Next, our policy learning baselines use CPO to solve

$$\min_{\pi} \mathbb{E}[J(\pi, Q)]$$
$$\text{s.t. } \Pr[Y^* \in \pi(Q) \vee Y^* \notin \mathcal{Y}(Q)] \geq 1 - \alpha,$$

with the following variations:

- CPO: Vanilla CPO, where we train a stochastic policy using the CPO algorithm without any conformal prediction.

[1] For exit policies, we report *matched coverage* by summing the percentages of correct and unsolvable questions, ensuring a fair comparison.

- CPO batch: Vanilla CPO, but perform batch conformal prediction on the held-out calibration set to select a value of κ for the final stochastic policy to obtain a conformal policy that guarantees coverage.
- CPO online: Vanilla CPO, but where we use online conformal prediction to convert the stochastic policy to obtain a conformal policy that guarantees coverage (using the same episodes as in CCPO training).

Table 1 summarizes key features of the different approaches: "Base-Guide" indicates whether it uses our strategy for composing a base and a guide LLM, "α-aware" indicates whether it is given the user-specified reliability level α, "Conformal" indicates whether it provides a coverage guarantee, "Pointwise" indicates whether it outputs a single label (instead of prediction set), "RL" indicates whether it uses reinforcement learning to train a policy, "LLM Policy" indicates whether it uses a prompted LLM as the policy. Compared to CPO online/batch, CCPO (our approach) tightly integrates conformal prediction into CPO.

5.2 Results and Discussion

Results. Our results on both HotpotQA and MMLU are shown in Tables 2, 3, 4, & 5; in addition, results for an alternate choice of α are provided in the appendix. The first two methods do not use base-guide composition, the next five use this strategy but do not provide conformal guarantees, and the last four use this strategy and provide conformal guarantees. We also show box plots of coverage and cost for both CPO and CCPO, computed across 100 random evaluation splits, in Figures 2, 3 4, & 5, and for an alternate choice of α in the appendix.

Discussion. First, cost-aware policy optimization efficiently minimizes cost while maintaining the desired coverage, achieving orders of magnitude cost reduction compared to LLM-based methods with similar coverage. The uncertainty-thresholding method with UALA is also effective, consistent with the findings of Chen, Zaharia, and Zou (2024). Nevertheless, policy optimization methods still offer a clear performance advantage, as they explicitly incorporate cost information and can potentially leverage additional contextual features for even stronger decision-making.

On the other hand, pointwise methods, whether LLM-guided or RL, fail to achieve higher coverage. Moreover, LLM-guided approaches built on small base models offer only limited control over agent collaboration and provide marginal improvements over a random policy. It is worth noting that GPT-4o serves as a strong judgment model, but inevitably incurs significantly higher inference cost.

Next, CCPO achieves the lowest cost at the desired coverage level. Its advantage is clear, even when compared to variants of CPO-based policy training. Notably, the CPO batch/online baselines, our strongest competitors, incorporate trust-region optimization to jointly enforce coverage and minimize cost with guarantees. Yet, CCPO reduces cost by 12% to 27% compared to this baseline. Additionally, CCPO outperforms the GPT-guided UALA on optimal batch-mode uncertainty calibration, further demonstrating its strong cost efficiency under coverage constraints.

Policy	Cost (cents)	Coverage	Avg. Len.	Set Size
GPT-4o EXIT	827.0	0.908	2.39	2.39
LLaMA-3 EXIT	0.000	0.718	4.00	4.00
Random	1.411	0.61	1.51	1.00
GPT-4o	18.09	0.850	1.15	1.00
LLaMA-3.2-3b	4.449	0.615	1.00	1.00
UALA	8.012	0.923	2.00	2.00
CPO	4.655	0.829	1.00	1.00
CPO batch	8.007	0.932	1.79	3.58
CPO online	8.829	0.875	1.856	3.11
CCPO ($\lambda = 0$)	**7.061**	0.901	**1.59**	3.18
CCPO ($\lambda = 1e^{-4}$)	7.397	0.902	1.66	**2.78**

Table 3: HotpotQA results with LLaMA-3.2-3b, $\alpha = 0.1$.

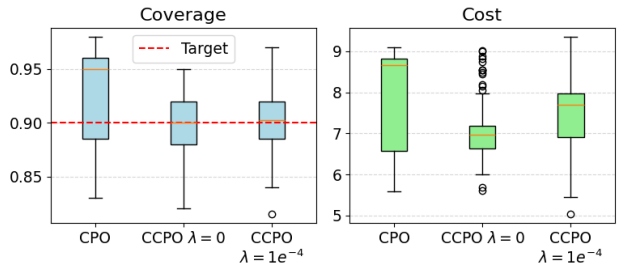

Figure 3: HotpotQA coverage and cost with LLaMA-3.2-3b base model over 100 splits, $\alpha = 0.1$.

Finally, CCPO produces reasonable prediction set sizes. While cost efficiency and coverage are our primary goals, it is also important that the resulting answer sets remain non-trivial and informative. Compared to the CPO variants, CCPO achieves similar answer set sizes while offering significantly greater cost efficiency. Notably, even when compared to the more costly GPT-4o-based methods, CCPO maintains comparable set sizes, further underscoring the informativeness and quality of the learned policy.

Comparison to CPO. We find that CPO often fails to find an optimal policy, especially at higher coverage levels. Moreover, when CPO already struggles to satisfy the coverage constraint, it becomes particularly sensitive to issues such as noisy or limited data, which can lead to severe divergence during training. In contrast, CCPO aims for a conformal policy, allowing multiple actions to be taken at each stage and thus flexibly achieving any desired coverage level. Empirically, we also observe that CCPO delivers more stable training performance under these data challenges.

Furthermore, since CPO optimizes for pointwise performance, successful training often produces sharp action distributions with extreme confidence. This can lead to overfitting, making the policy sensitive to noise and difficult to calibrate. In contrast, CCPO simultaneously updates the threshold κ, resulting in a more separable policy geometry and avoiding the abrupt coverage jumps often seen during calibration on sequential data. As a consequence, CCPO achieves tighter and more consistent coverage performance.

Policy	Cost (cents)	Coverage	Avg. Len.	Set Size
GPT-4o EXIT	104.6	0.960	1.39	1.39
LLaMA-2 EXIT	0.000	0.790	3.14	3.14
Random	12.99	0.545	1.53	1.00
GPT-4o	25.35	0.930	1.08	1.00
LLaMA-2-7B	8.682	0.660	1.11	1.00
UALA	16.99	0.920	2.00	2.00
CPO	9.174	0.70	1.195	1.00
CPO batch	22.25	0.902	2.86	4.20
CPO online	19.07	0.901	2.48	3.46
CCPO ($\lambda = 0$)	**7.949**	0.920	**1.08**	2.14
CCPO ($\lambda = 2e^{-4}$)	8.071	0.907	1.11	**1.48**

Table 4: MMLU results with LLaMA-2-7B, $\alpha = 0.1$.

Policy	Cost (cents)	Coverage	Avg. Len.	Set Size
GPT-4o EXIT	104.6	0.960	1.39	1.39
LLaMA-3 EXIT	0.000	0.840	3.97	3.97
Random	12.74	0.607	1.55	1.00
GPT-4o	26.87	0.913	1.19	1.00
LLaMA-3-2-3b	8.540	0.580	1.03	1.00
UALA	16.10	0.893	2.00	2.00
CPO	6.846	0.860	1.01	1.00
CPO batch	12.12	0.917	1.51	2.36
CPO online	11.76	0.878	1.51	2.53
CCPO ($\lambda = 0$)	**10.82**	0.901	**1.34**	2.66
CCPO ($\lambda = 1e^{-4}$)	10.90	0.900	1.35	2.70

Table 5: MMLU results with LLaMA-3.2-3b, $\alpha = 0.1$.

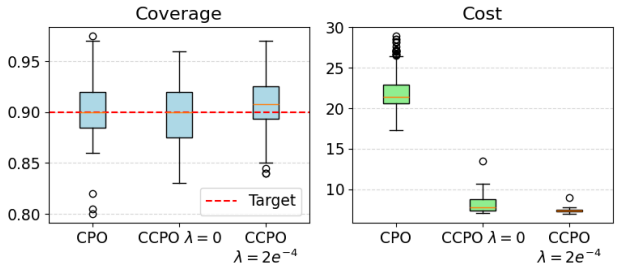

Figure 4: MMLU coverage and cost with LLaMA-2-7B base model over 100 splits, $\alpha = 0.1$.

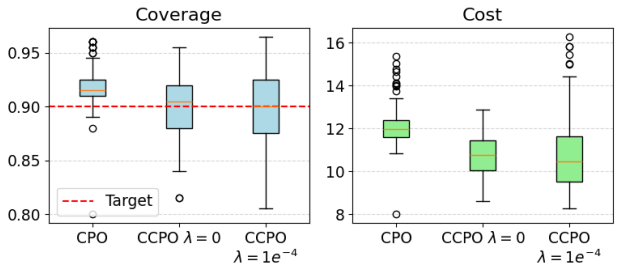

Figure 5: MMLU coverage and cost with LLaMA-3.2-3b base model over 100 splits, $\alpha = 0.1$.

Comparison to LLM policies. LLMs have demonstrated great potential as language-based controllers. However, in our setting, large models such as GPT-4o are required for strong performance, but these incur high costs. In contrast, smaller models like LLaMA-2-7B and LLaMA-3.2-3B exhibit limited ability to assess the correctness of responses, whether their own or from other models, often performing comparably to random. Among LLM policy strategies, LLM-EXIT performs best, likely due to the inherent strength of LLMs in binary decision-making tasks.

6 Discussion

Clipping. To guarantee convergence and a unique fixed point for the critic, we clip the importance weights so that the V-trace operator becomes a contraction mapping. Specifically, Espeholt et al. (2018, proof of Remark 3) show that with $\gamma\bar{c} < 1$, the V-trace operator $\mathcal{R}V$ is a contraction mapping in the sup-norm. Thus, setting $\bar{c} = \bar{\rho} = 1$ preserves the contraction property and introduces only negligible bias in the critic update. For policy updates, we apply clipping only to the reward gradient to reduce variance, while leaving the cost gradient unclipped. Clipping g_c would introduce bias, so we avoid it. Overall, this clipping scheme ensures that the critic remains stable and that the CPO gradients are both low-variance and nearly unbiased.

Need for stochastic policy. We need a stochastic policy both for the CPO policy update and for the stability of the V-trace estimate. First, CPO imposes a trust-region constraint on policy updates via the Kullback–Leibler (KL) divergence $D_{\mathrm{KL}}\left(\pi_{\text{new}} \,\|\, \pi_{\text{old}}\right) \leq \delta$, which ensures that each policy update remains within a local neighborhood where surrogate approximations are valid. However, KL divergence is only well-defined when π_{new} is absolutely continuous with respect to π_{old} and has the same support. Thus, the KL divergence for a conformal policy may become infinite or undefined. Using a stochastic policy ensures valid policy updates.

Second, Espeholt et al. (2018, proof of Remark 3) argue that for $\mathcal{R}V$ to be a contraction mapping, the policy must satisfy $\mathbb{E}_{\mu}\left[\frac{\pi_{\text{new}}(a_t|o_t)}{\pi_{\text{old}}(a_t|o_t)}\right] = 1$. Using a stochastic policy ensures the importance weights are bounded, thereby preserving convergence of the V-trace method.

7 Conclusion

We have proposed Conformal Constrained Policy Optimization (CCPO), an algorithm that unifies constrained policy optimization and online conformal prediction to orchestrate LLM agents to minimize cost while achieving a user-specified reliability level α. We formalized the problem as a finite-horizon reinforcement learning problem where we jointly optimize a stochastic policy and a threshold. Experiments on multi-hop question answering benchmarks demonstrate that our approach can reduce total computational and API costs by up to 30% compared to state-of-the-art cost-aware baselines at the target coverage levels. These results demonstrate that our approach is a promising strategy for reliable and cost-effective LLM agents.

Acknowledgements

This work was supported in part by NIH R01EY037101 and NSF Award CCF-2338777.

References

Achiam, J.; Held, D.; Tamar, A.; and Abbeel, P. 2017. Constrained policy optimization. In *International conference on machine learning*, 22–31. PMLR.

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Angelopoulos, A. N.; Barber, R.; and Bates, S. 2024. Online conformal prediction with decaying step sizes. In *International Conference on Machine Learning*, 1616–1630. PMLR.

Bastani, O.; Gupta, V.; Jung, C.; Noarov, G.; Ramalingam, R.; and Roth, A. 2022. Practical adversarial multivalid conformal prediction. *Advances in neural information processing systems*, 35: 29362–29373.

Bastani, O.; Li, S.; and Xu, A. 2021. Safe Reinforcement Learning via Statistical Model Predictive Shielding. In *Robotics: Science and Systems*, 1–13.

Bensal, S.; Jamil, U.; Bryant, C.; Russak, M.; Kamble, K.; Mozolevskyi, D.; Ali, M.; and AlShikh, W. 2025. Reflect, Retry, Reward: Self-Improving LLMs via Reinforcement Learning. arXiv:2505.24726.

Chen, L.; Zaharia, M.; and Zou, J. 2024. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Transactions on Machine Learning Research*.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. 2018. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, 1407–1416. PMLR.

Gibbs, I.; and Candes, E. 2021. Adaptive conformal inference under distribution shift. *Advances in Neural Information Processing Systems*, 34: 1660–1672.

Han, J.; Buntine, W.; and Shareghi, E. 2024. Towards Uncertainty-Aware Language Agent. In Ku, L.-W.; Martins, A.; and Srikumar, V., eds., *Findings of the Association for Computational Linguistics ACL 2024*, 6662–6685. Bangkok, Thailand and virtual meeting: Association for Computational Linguistics.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.

Kostrikov, I.; Nair, A.; and Levine, S. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *International Conference on Learning Representations*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. *Advances in neural information processing systems*, 33: 1179–1191.

Lu, Q.; Ding, L.; Cao, S.; Liu, X.; Zhang, K.; Zhang, J.; and Tao, D. 2025. Runaway is Ashamed, But Helpful: On the Early-Exit Behavior of Large Language Model-based Agents in Embodied Environments. In Christodoulopoulos, C.; Chakraborty, T.; Rose, C.; and Peng, V., eds., *Findings of the Association for Computational Linguistics: EMNLP 2025*, 24014–24027. Suzhou, China: Association for Computational Linguistics. ISBN 979-8-89176-335-7.

Min, S. Y.; Wu, Y.; Sun, J.; Kaufmann, M.; Tajwar, F.; Bisk, Y.; and Salakhutdinov, R. 2025. Self-Regulation and Requesting Interventions. arXiv:2502.04576.

Munos, R.; Stepleton, T.; Harutyunyan, A.; and Bellemare, M. 2016. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29.

Ren, A. Z.; Dixit, A.; Bodrova, A.; Singh, S.; Tu, S.; Brown, N.; Xu, P.; Takayama, L.; Xia, F.; Varley, J.; Xu, Z.; Sadigh, D.; Zeng, A.; and Majumdar, A. 2023. Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. In *7th Annual Conference on Robot Learning*.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897. PMLR.

Shafer, G.; and Vovk, V. 2008. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(3).

Stutz, D.; Dvijotham, K. D.; Cemgil, A. T.; and Doucet, A. 2022. Learning Optimal Conformal Classifiers. In *International Conference on Learning Representations*.

Tessler, C.; Mankowitz, D. J.; and Mannor, S. 2019. Reward Constrained Policy Optimization. In *International Conference on Learning Representations*.

Wan, D.; Chen, J.; Stengel-Eskin, E.; and Bansal, M. 2025. MAMM-refine: A recipe for improving faithfulness in generation with multi-agent collaboration. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 9882–9901.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Yang, Z.; Qi, P.; Zhang, S.; Bengio, Y.; Cohen, W.; Salakhutdinov, R.; and Manning, C. D. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, 2369–2380.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhou, Y.; Jiang, S.; Tian, Y.; Weston, J.; Levine, S.; Sukhbaatar, S.; and Li, X. 2025. SWEET-RL: Training Multi-Turn LLM Agents on Collaborative Reasoning Tasks. arXiv:2503.15478.