CoE-Agent: Co-Evolving Patient-Doctor Agents via Interactive Policy Graph Optimization for Clinical Decision Making

Guolin Huang
College of Computer Science
and Software Engineering,
Shenzhen University
Shenzhen, China
a1760050788@163.com

Wenting Chen*
Department of Radiation Oncology,
Stanford University
Stanford, California, USA
wentingchen168@gmail.com

Linlin Shen*
Intelligent Computing Center,
School of AI,
Shenzhen University
Shenzhen, China
llshen@szu.edu.cn

Abstract

Clinical Decision Making (CDM) requires integrating heterogeneous information across sequential diagnostic stages while interacting with patients whose behaviors are often non-stationary and unreliable. However, existing medical agents are typically trained against cooperative patient simulators and tackle each clinical task in isolation, making them fragile to dynamic patient behaviors and unable to capture inter-task dependencies essential for globally coherent decision-making. To address these issues, we propose CoE-Agent, a co-evolving patient–doctor agentic framework that learns globally coherent CDM strategies via interactive policy graph optimization. First, Dynamic Patient–Doctor Co-Evolutionary Learning pairs a Dynamic Patient Agent, which maintains an evolving cognition state to simulate non-stationary behaviors, with an Interactive Policy Graph-Based Doctor Agent that generates DAG-structured policy graphs unifying interactions, tool invocations, and reasoning steps. The doctor agent is then optimized via Interactive Policy Graph Optimization (IPGO), which jointly enforces structural validity and task-aligned correctness through reinforcement learning. Second, Cross-Task Policy Graph Consolidation and Unified Model Learning prunes low-value nodes, aggregates high-value nodes across tasks while filtering redundancy, and distills a unified policy planner with task-specific decision models for globally coherent decision-making. Experiments on MedChain and ClinicalBench show that CoE-Agent surpasses state-of-the-art baselines by 11.13% and 23.63% in average score, while achieving up to 19.5× faster inference. Source code is to be released.

1 Introduction

Clinical Decision Making (CDM) stands as one of medicine's most complex challenges at the intersection of AI and healthcare. It requires integrating diverse data sources, continuously assessing

*Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

evolving clinical scenarios, and making evidence-based diagnostic and treatment judgments. Recent advances in large language models (LLMs) have spurred the development of medical agents to assist this process. On one hand, multi-agent systems such as MedChain-Agent [12] and ClinicalAgent [25] explicitly model sequential diagnostic procedures and complex clinical tasks through collaborative reasoning among specialized roles. On the other hand, self-evolving paradigms have emerged, with frameworks like AgentHospital [10] and MedAgentSim[1] introducing simulated clinical environments where agents autonomously refine their diagnostic strategies and medical expertise through repeated interactions. Despite their promise in streamlining clinical workflows, these medical agents still face some challenges.

Although existing medical agents perform well only when interacting with cooperative, well-behaved patients, they struggle when *patient behavior becomes unstable or unpredictable*. In clinical practice, patients may misunderstand questions, misreport symptoms, provide inconsistent information, or express strong preferences and emotions [15]. Such misreporting is far from rare; in fact, it occurs in approximately 15–20% of real-world cases [7, 13], often introducing non-negligible interference into clinical conversations [22]. Recognizing the significance of this issue, several recent studies have begun to address it. For instance, Li *et al.* [11] introduce CPB-Bench, a medical benchmark designed to evaluate LLM performance under challenging patient behaviors, revealing that current medical LLMs consistently fall into recurring failure patterns, particularly when confronted with contradictory patient information and abnormal clinical values. To mitigate this, systems like PaMis have been proposed to identify misreports via clarifying questions, while frameworks such as EvoPatient [6] focus on improving the factual consistency of simulated patient responses. However, these efforts are largely directed at improving the patient simulator rather than strengthening the doctor model's ability to adapt to dynamic patient, which leaves the doctor model insufficiently robust. Therefore, it is essential for current medical agents to learn more robust decision-making strategies that can accommodate ***dynamic patient behaviors***.

Another challenge lies in the fact that current medical agents tackle each task in the CDM process independently, leading to ***suboptimal solutions for the entire clinical workflow***. For instance, MedChain-Agent [12] adds a feedback mechanism at each stage to reduce error propagation, while ClinicalAgent [25] decomposes diagnosis into six stages and assigns relevant departments to optimize each locally.

In real-world practice, CDM is an inherently sequential and interdependent process, in which the outcome of each task is contingent upon the preceding ones, and the quality of a downstream task can in turn reveal deficiencies in upstream decisions. When unexpected findings emerge in later stages (e.g., inconclusive laboratory results or contradictory imaging evidence), clinicians often roll back to revisit earlier steps, such as re-collecting the patient's history or ordering additional examinations. As a result, the responses across different tasks are deeply intertwined and should be jointly optimized rather than treated in isolation. However, existing medical agents for CDM scenarios mainly resolve each task in a forward-only manner, ignoring the bidirectional dependencies among tasks and thereby yielding suboptimal end-to-end solutions. Thus, it is essential to explicitly ***model the inter-task relationships*** and design a ***globally coherent decision-making strategy*** that supports dynamic revision and information flow across the entire diagnostic pipeline.

To address these challenges, we propose **CoE-Agent**, a co-evolving patient-doctor agentic framework that learns globally coherent clinical decision-making strategies through interactive policy graph optimization, comprising two complementary stages (Fig. 1). In the first stage, *Dynamic Patient–Doctor Co-Evolutionary Learning* lets the doctor agent optimize against an evolving patient agent to handle ***dynamic patient behaviors***. Specifically, we construct a *Dynamic Patient Agent* that maintains a continuously evolving cognition state, which is initialized from the case summary, updated through doctor-patient interactions, and transferred across tasks, thereby faithfully simulating the non-stationary and potentially unreliable nature of real patients. In parallel, an *Interactive Policy Graph-Based Doctor Agent*, consisting of a policy planner, a critic model, and a decision model, generates a directed acyclic graph (DAG)-structured policy graph that unifies patient interactions, tool invocations, and reasoning steps for each clinical task. To optimize the policy planner against the dynamic patient agent, we further design *Interactive Policy Graph Optimization (IPGO)*, which formulates the policy graph generation as a reinforcement learning problem. IPGO jointly leverages a Policy Graph-Grounded Reward to regularize the structural validity of the policy graph (e.g., node scale and tool-node correctness) and a Task-Aligned Reward to align the executed graph with the final task outcome. In the second stage, *Cross-Task Policy Graph Consolidation and Unified Model*

Learning is designed to ***model inter-task relationships*** and yield a ***globally coherent strategy*** across the entire clinical workflow. Concretely, *Reward-Based Policy Graph Pruning* first removes low-contribution nodes from each task-specific policy graph based on their marginal reward contribution; *Unified Policy Graph Construction* then aggregates high-value nodes across tasks, filters semantically redundant ones via inter-node similarity, and assembles a unified policy graph that captures cross-task dependencies; finally, *Unified Policy Planning and Decision Learning* performs supervised fine-tuning to obtain a unified policy planner together with task-specific decision models, enabling consistent and globally optimized decision-making across the entire CDM pipeline. Extensive experiments on two CDM benchmarks, i.e. MedChain and ClinicalBench, show that CoE-Agent significantly outperforms state-of-the-art models, validating its robustness to dynamic patient interactions and its effectiveness in globally coherent CDM. Our contributions are summarized as follows:

- We propose CoE-Agent, a co-evolving patient–doctor agentic framework that learns globally coherent clinical decision-making strategies via interactive policy graph optimization.
- For dynamic patient behaviors, we design a dynamic patient–doctor co-evolutionary learning stage, where doctor agent is iteratively improved against patient agent via reinforcement learning for robust decision-making. To consider inter-task dependencies, we further introduce a cross-task policy graph consolidation and unified model learning stage to aggregate task-specific policy graphs into a unified one for globally coherent decision-making across the entire CDM pipeline.
- Extensive experiments on two CDM benchmarks shows the superiority of CoE-Agentover the state-of-the-art medical agents.

2 Related Works

2.1 Medical Agents in Clinical Decision Making

Existing LLM-based medical agents for clinical decision-making (CDM) fall into two paradigms: *multi-agent collaboration*, where specialized roles interact to solve medical tasks (e.g. ClinicalAgent [25], MedChain-Agent [12]), and *self-evolving simulation*, where doctor agents improve through repeated interactions in simulated hospitals (e.g., Agent Hospital [10], MedAgentSim [1]). However, these methods handle clinical stages in isolation with forward-only information flow and assume cooperative patients, making them brittle to the misreporting and contradictions typical in real consultations [7, 13]. In contrast, CoE-Agent co-evolves a dynamic patient agent with an interactive policy-graph-based doctor agent via reinforcement learning, unifying task-specific policy graphs into a globally coherent CDM strategy that is robust to non-stationary patient behaviors and aware of cross-task dependencies.

2.2 Self-Evolving Agents

Self-evolving agents have been widely investigated and can be categorized based on their evolution targets into four groups, including models, context, tools, and architectures. First, agents evolve models by updating internal parameters; for example, SCA [32] fine-tunes weights via self-generated tasks, while TextGrad [27] uses textual feedback for policy improvement. Second, agents adapt their context through memory and prompt optimization. Systems like Mem0 [4] and Expel [30] dynamically update knowledge and rules. Third, agents expand their tools by synthesizing new capabilities, as demonstrated by Voyager's [23] autonomous skill library and SkillWeaver's [31] API generation. Finally, agents optimize their agentic architectures and workflows. This ranges from single-agent code self-modification, like the Darwin Gödel Machine [28], to multi-agent workflow optimization frameworks such as AFlow [29]. Despite these advancements, these self-evolving agents primarily optimize isolated capabilities in static or cooperative environments, struggling with dynamic patient behaviors and bidirectional inter-task dependencies. To bridge this gap, CoE-Agent extends the self-evolving paradigm into a *co-evolutionary* setting. By optimizing an interactive policy graph via reinforcement learning against a dynamic patient agent, CoE-Agent adapts to unpredictable behaviors and consolidates task-specific graphs into a globally coherent clinical strategy.

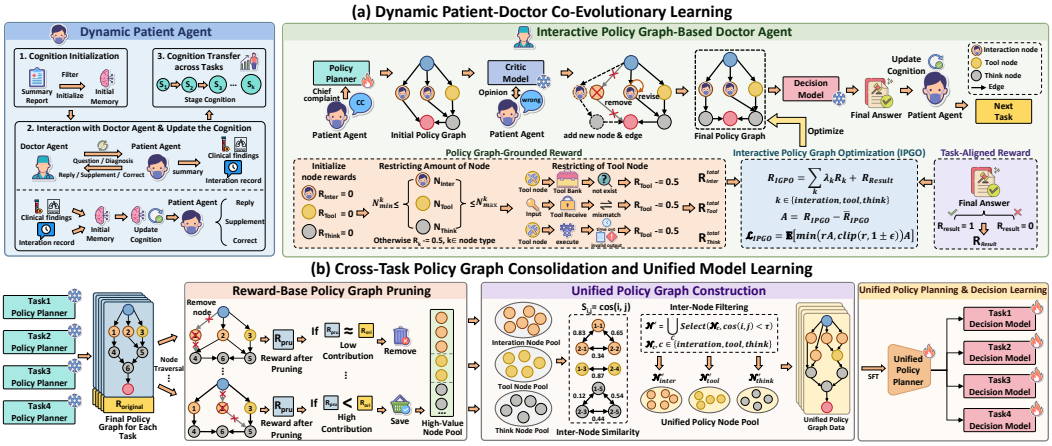


Figure 1: **Overview of CoE-Agent**, consisting of (a) ***dynamic patient-doctor co-evolutionary learning*** to optimize task-specific policy graphs through patient-doctor interaction, and (b) ***cross-task policy graph consolidation and unified model learning*** to build a unified policy planner for multi-task clinical decision-making

3 Method

Overview. In Fig. 1, CoE-Agent is a co-evolving patient–doctor agentic framework that aims to acquire robust and globally coherent CDM strategies through interactive policy graph optimization, including two complementary stages. In the first stage, *Dynamic Patient-Doctor Co-Evolutionary Learning* (Sec. 3.1) constructs a *Dynamic Patient Agent* that simulates the dynamic behaviors of real patients, and an *Interactive Policy Graph-Based Doctor Agent* that produces a directed acyclic graph (DAG)-structured policy graph for each clinical task. The doctor agent is co-evolved with the patient agent through *Interactive Policy Graph Optimization (IPGO)*, which jointly leverages a Policy Graph-Grounded Reward and a Task-Aligned Reward to ensure both structural validity and task-aligned correctness of the policy graph. In the second stage, *Cross-Task Policy Graph Consolidation and Unified Model Learning* (Sec. 3.2) prunes low-value nodes, aggregates high-value nodes across tasks into a unified policy graph, and finally trains a unified policy planner with task-specific decision models via SFT, yielding a globally coherent strategy spanning the entire CDM pipeline.

3.1 Dynamic Patient-Doctor Co-Evolutionary Learning (CoEL)

To mitigate the challenge of *dynamic patient behaviors*, CoEL models the patient as a continuously evolving environment and iteratively optimizes the doctor agent against it. It consists of a *Dynamic Patient Agent* that maintains an updatable cognition state to simulate non-stationary patient behaviors, and an *Interactive Policy Graph-Based Doctor Agent* that generates DAG-structured policy graphs and is co-evolved with the patient via Interactive Policy Graph Optimization (IPGO).

3.1.1 Dynamic Patient Agent (DPA)

The dynamic patient agent is built around an updatable cognition state c_t, which represents the patient's current understanding of the symptoms, prior examinations, and overall condition at time step t. The cognition state is initialized from the basic case information available at the first visit, continuously updated through doctor–patient interactions, and transferred across clinical tasks.

Cognition Initialization. To prevent information leakage and reflect the limited data available at first visit, c_t is initialized using only the chief complaint and basic medical history, excluding any unobservable future information: $c_0 = \mathcal{I}(x_{\text{case}})$, where x_{case} is the full case record and $\mathcal{I}(\cdot)$ filters for the subset observable at admission.

Interaction with Doctor Agent and Cognition Update. During doctor–patient interaction, the patient's cognition is continuously reshaped by the doctor's actions, including questions, examination orders, and diagnostic conclusions. Let a_t denote the doctor's action and o_t the patient's response

at step t. The cognition is updated as $c_{t+1} = \mathcal{U}(c_t, a_t, o_t) + \epsilon_t$, where $\mathcal{U}(\cdot)$ is the cognition update function and ϵ_t captures drift from misdiagnoses or unreasonable exam requests. Thus, flawed decisions bias subsequent feedback, while sound ones drive c_t toward a faithful state. Conditioned on the updated cognition, the patient agent chooses to *reply*, *supplement*, or *correct* the doctor, yielding realistic non-stationary behavior.

Cognition Transfer Across Tasks. CDM consists of multiple sequential tasks $\{T_1, T_2, \ldots, T_K\}$ (e.g., history-taking, examination recommendation, diagnosis, treatment). The patient's cognition naturally accumulates across these stages: the final cognition state of the previous task serves as the initial state of the next, namely $c_0^{(k)} = c_T^{(k-1)}$. Combining the within-task and cross-task updates, the cognition trajectory over the entire multi-stage pipeline is uniformly written as:

$$c_{t+1}^{(k)} = \begin{cases} \mathcal{U}\left(c_t^{(k)}, a_t^{(k)}, o_t^{(k)}\right) + \epsilon_t^{(k)}, & t < T_k, \\ \mathcal{I}\left(c_{T_k}^{(k)}\right), & t = T_k,\ k \to k+1. \end{cases} \tag{1}$$

This design enables the patient agent to amplify the long-range impact of early decisions: a wrong early-stage decision can propagate biased cognition into later tasks, posing a more challenging and realistic environment for the doctor agent.

3.1.2 Interactive Policy Graph-Based Doctor Agent (IPG-DA)

To enable flexible and composable decision-making in such a dynamic environment, we introduce IPG-DA, to model the doctor agent's reasoning as a *structured policy graph generation* problem rather than a fixed sequential pipeline. The doctor agent is composed of three complementary models: a **policy planner** for generating policy graphs, a **critic model** for revising them through patient interaction, and a **decision model** for executing them and producing final outputs.

Policy Planner. The policy planner π_θ generates a DAG-structured policy graph $G = \pi_\theta(x, c), G = (V, E, \tau)$, where x is the current task input, c is the patient cognition state, V is the set of nodes, E is the set of dependency edges, and $\tau : V \to \mathcal{T}$ assigns each node a type from $\mathcal{T} = \{\text{interaction}, \text{tool}, \text{think}\}$, corresponding to doctor–patient interactions, tool invocations, and intermediate reasoning steps, respectively. During execution, the policy graph is unrolled in topological order. For each node $v \in V$, its output is computed as $h_v = f_{\tau(v)}(\{h_u \mid u \in \text{Pa}(v)\},\ x,\ c)$, where $f_{\tau(v)}$ is the operation function determined by the node type and $\text{Pa}(v)$ denotes the parent nodes of v. The collection $\mathcal{T}(G) = \{h_v \mid v \in V\}$ forms the complete execution trajectory.

Critic Model. The critic model $\mathcal{C}_\phi$ evaluates the trajectory of the entire policy graph and identifies issues such as redundant structures, unreasonable dependencies, or missing critical information. To further verify its judgement, the critic actively interacts with the patient agent and obtains feedback $r = \mathcal{P}(\mathcal{T}(G), c)$, where $\mathcal{P}(\cdot)$ is the patient response function determined by cognition state c. Based on G, $\mathcal{T}(G)$ and r, the critic refines the policy graph through node insertion ($V \leftarrow V \cup \{v_{\text{new}}\}$), node deletion ($V \leftarrow V \setminus \{v\}$), node modification (updating $\tau(v)$ or its content), and edge revision ($E \to E'$), yielding a refined policy graph $G' = \mathcal{C}_\phi(G,\ \mathcal{T}(G),\ r)$.

Decision Model. Given the refined graph G', the decision model $\mathcal{F}_\psi$ aggregates the rich context produced along the execution trajectory, including symptom descriptions from interactions, returned results from tool invocations, and intermediate reasoning conclusions, into a structured medical analysis context, and produces the final task output: $\mathcal{H}(G') = \text{Aggregate}(\mathcal{T}(G')), y = \mathcal{F}_\psi(\mathcal{H}(G'),\ x)$.

3.1.3 Interactive Policy Graph Optimization (IPGO)

To optimize the policy planner against the dynamic patient agent, we propose IPGO, a reinforcement learning algorithm tailored for structured policy graph generation. IPGO employs two complementary reward signals: a *Policy Graph-Grounded Reward* that regularizes graph structure at the process level, and a *Task-Aligned Reward* that aligns the final output with the task-specific objective.

Policy Graph-Grounded Reward. This reward enforces the structural rationality of the policy graph based on node-level criteria. We first partition the nodes of $G' = (V, E, \tau)$ by type: $V_t = \{v \in V \mid \tau(v) = t\}, t \in \mathcal{T}$. The reward decomposes into three type-specific terms initialized as $R_{\text{int}} = R_{\text{tool}} = R_{\text{think}} = 0$, and is constrained from two aspects:

(i) Restricting Amount of Nodes. For each node type t, we encourage the count $|V_t|$ to fall within a reasonable range $[N_t^{\min}, N_t^{\max}]$: $R_t \mathrel{-}= \lambda_1 \cdot \mathbf{1}\big(|V_t| \notin [N_t^{\min}, N_t^{\max}]\big)\,, t \in \mathcal{T}$, penalizing graphs that are either too sparse or unnecessarily large.

(ii) Restricting Tool Nodes. For each tool node $v \in V_{\text{tool}}$, we further check whether the tool exists in the tool bank, whether its inputs match the expected schema, and whether the execution is successful (no time-out or invalid output). We define

$$\delta(v) = \begin{cases} 1, & \text{if } v \text{ is illegal or fails to execute,} \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

and update the tool reward as $R_{\text{tool}} \mathrel{-}= \lambda_2 \sum_{v \in V_{\text{tool}}} \delta(v)$, where λ_2 is a penalty coefficient. The overall reward is $R_{\text{graph}}(G') = R_{\text{int}} + R_{\text{tool}} + R_{\text{think}}$.

Task-Aligned Reward. To align the optimization with the downstream clinical task, we keep the decision model $\mathcal{F}_\psi$ frozen during this stage and evaluate the executed graph through the task metric: $R_{\text{result}}(y) = \mathcal{M}(y, y^*)$, where y^* is the reference answer and $\mathcal{M}(\cdot)$ is a task-specific evaluator (e.g., accuracy or matching score). In practice, we adopt a binary form $R_{\text{result}} \in \{0, 1\}$.

IPGO Objective. The total reward combines the structural and outcome signals: $R_{\text{IPGO}} = \sum_{k \in \{\text{int, tool, think}\}} \lambda_k R_k + R_{\text{result}}$. We estimate the advantage by group-wise normalization $A = R_{\text{IPGO}} - \bar{R}_{\text{IPGO}}$ and update the policy planner with the clipped surrogate loss:

$$\mathcal{L}_{\text{IPGO}} = \mathbb{E}[\min(rA,\ \text{clip}(r, 1 \pm \epsilon)\, A)]\,, \tag{3}$$

where r denotes the importance ratio between current and old policies and ϵ is the clipping threshold. By keeping the decision model frozen, IPGO ensures that the optimization focuses on improving the quality of generated policy graphs rather than overfitting to a particular downstream output.

3.2 Cross-Task Policy Graph Consolidation and Unified Model Learning (PGC-UML)

After the first stage, each task T_k obtains an IPGO-optimized policy planner that produces locally optimal policy graphs. However, these graphs are tied to specific tasks and execution trajectories, and lack consistency or global optimality across the entire CDM pipeline. To address the challenge of *inter-task dependencies*, we propose PGC-UML, which extracts high-value sub-structures from task-specific policy graphs, reorganizes them into a unified graph, and finally distills a unified policy planner via supervised fine-tuning.

3.2.1 Reward-Based Policy Graph Pruning (PGP)

To prune the graph nodes with low value, we introduce PGP to identify and retain the nodes that contribute most to the final decision for each task. Specifically, we evaluate every node by measuring its marginal effect on the final task reward. Given a policy graph $G = (V, E, \tau)$, removing a node $v \in V$ yields a pruned graph $G^{-v} = (V \setminus \{v\},\ E^{-v},\ \tau)$, where E^{-v} is the corresponding edge set that preserves graph executability. We then re-execute both G and G^{-v} through the decision model: $y = \mathcal{F}_\psi(G, x), y^{-v} = \mathcal{F}_\psi(G^{-v}, x)$, and compute the corresponding total rewards $R(G) = R_{\text{graph}}(G) + R_{\text{out}}(y)$ and $R(G^{-v}) = R_{\text{graph}}(G^{-v}) + R_{\text{out}}(y^{-v})$. The contribution of node v is defined as $\Delta R_v = R(G) - R(G^{-v})$. A node is regarded as a *low-value node* if $\Delta R_v \approx 0$ whose removal barely affects performance, and a *high-value node* if $\Delta R_v > \varepsilon$ whose removal causes a notable drop, where ε is a threshold. We retain only high-value nodes, yielding a compact subgraph $V_k^* \subseteq V_k$, that preserves the core decision skeleton for task T_k while discarding redundant structures.

3.2.2 Unified Policy Graph Construction

To capture cross-task dependencies, we further integrate high-value nodes from all tasks into a unified policy graph through three steps.

Cross-Task Node Aggregation. We first aggregate high-value nodes across all K tasks: $V^* = \bigcup_{k=1}^{K} V_k^*$. Each node in V^* retains its type label $\tau(v) \in \mathcal{T} = \{\text{interaction, tool, think}\}$.

Type-wise Node Pooling. Based on $\tau(\cdot)$, we partition V^* into three type-specific node pools: $\mathcal{P}_t = \{v \in V^* \mid \tau(v) = t\}, \quad t \in \mathcal{T}$, yielding $\{\mathcal{P}_{\text{interaction}}, \mathcal{P}_{\text{tool}}, \mathcal{P}_{\text{think}}\}$, which respectively correspond to the interaction node pool, tool node pool, and think node pool in Fig. 1(b).

Table 1: Performance comparison on the MedChain dataset.

Framework	Methods	Specialty Referral		History taking	Examination	Diagnosis	Treatment	Average	
		Level1	Level2						
Proprietary LLMs	GPT5.1 [17]	0.5157	0.3370	0.5059	0.5638	<u>0.5199</u>	0.5210	0.4939	↑ 0.14
	GPT-o3 [14]	0.5495	0.2866	0.3493	–	0.4891	0.3343	0.4018	↑ 0.23
	DeepSeek V3.2 [5]	0.5321	0.3530	0.4593	–	0.5126	0.5769	0.4868	↑ 0.14
	Kimi-K2 [19]	0.5227	0.3288	0.3667	0.4717	0.4297	0.5235	0.4405	↑ 0.19
Open-Source LLMs	MedGemma-27B [16]	0.5571	0.3665	0.4901	–	0.4879	0.4496	0.4702	↑ 0.16
	Baichuan-M2-32B [21]	0.5067	0.3337	0.4730	0.4914	0.5058	0.5882	0.4831	↑ 0.15
	Qwen3-32B [26]	0.5432	0.3628	0.5171	0.4819	0.5144	0.5153	0.4891	↑ 0.14
	Lingshu-32B [20]	0.5469	0.3673	0.4217	0.5457	0.4944	0.4934	0.4782	↑ 0.15
	HuatuoGPT-o1-8B [2]	0.4989	0.3333	0.3402	0.4522	0.4496	0.2445	0.3865	↑ 0.24
Multi-Agent System	MedAgent [18]	0.3830	0.2039	0.4454	0.4456	0.4102	0.3673	0.3759	↑ 0.26
	MDAgent [9]	0.4754	0.2063	0.4412	0.5346	0.4198	0.4371	0.4191	↑ 0.21
	MedChain-Agent [12]	0.5873	0.3505	<u>0.5836</u>	<u>0.6566</u>	0.4807	0.4613	<u>0.5200</u>	↑ 0.11
	MedAgent-Pro [24]	<u>0.5891</u>	<u>0.3714</u>	0.4607	0.5477	0.4520	0.5215	0.4904	↑ 0.14
Self-Evolving Agent	MDTeamGPT [3]	0.2943	0.3464	0.4446	0.5551	0.4118	0.4691	0.4202	↑ 0.21
	EvoClinician [8]	0.5412	0.3591	0.3812	0.5226	0.4268	0.5214	0.4587	↑ 0.17
	CoE-Agent	**0.7281**	**0.4911**	**0.5900**	**0.7248**	**0.5472**	**0.7064**	**0.6313**	–

Inter-Node Filtering via Similarity. Different tasks may produce semantically or functionally similar nodes (e.g., similar questioning utterances or repeated reasoning steps). Direct aggregation would inflate the policy graph and degrade efficiency. We therefore compute pairwise cosine similarity within each pool using a node embedding $\mathbf{h}_v = \Phi(v)$: $S_{ij} = \cos(\mathbf{h}_{v_i}, \mathbf{h}_{v_j}) = \frac{\mathbf{h}_{v_i}^\top \mathbf{h}_{v_j}}{\|\mathbf{h}_{v_i}\| \, \|\mathbf{h}_{v_j}\|}$, and filter out highly similar nodes by a threshold γ: $\tilde{\mathcal{P}}_t = \{v \in \mathcal{P}_t \mid \forall v' \in \tilde{\mathcal{P}}_t,\ \cos(v, v') < \gamma\}, \quad t \in \mathcal{T}$.

Unified Policy Graph Assembly. After de-redundancy, we assemble the unified policy graph by selecting nodes from each pool and reconstructing edges according to their cross-task execution dependencies:

$$\tilde{V} = \bigcup_{t \in \mathcal{T}} \tilde{V}_t, \quad \tilde{V}_t \subseteq \tilde{\mathcal{P}}_t, \qquad \tilde{E} = \{(v_i, v_j) \mid v_i, v_j \in \tilde{V},\ v_i \prec v_j\}, \tag{4}$$

where $v_i \prec v_j$ denotes a topological dependency between two nodes inherited from their original task graphs. The resulting unified policy graph is $G^* = (\tilde{V}, \tilde{E}, \tau)$. Then, we leverage G^* to construct a cross-task supervision dataset $\mathcal{D}_{\text{policy}} = \{(x_i, c_i, G_i^*, y_i^*)\}_{i=1}^N$ for unified model learning. Through this process, redundant information already covered by previous tasks is eliminated, while critical historical interactions and reasoning steps that benefit downstream decisions are explicitly preserved, leading to a compact yet globally consistent decision structure.

3.2.3 Unified Policy Planning and Decision Learning (UPPDL)

To model inter-task relationships and yield a globally coherent strategy, we introduce UPPDL, to learn a unified policy graph multiple tasks for the dynamic patients. Specifically, we train a *unified policy planner* π_θ^u via supervised fine-tuning (SFT) on a cross-task supervision dataset $\mathcal{D}_{\text{policy}}$. To make decision model fit the output formats and evaluation criteria of each downstream task, we further fine-tune task-specific decision models $\{\mathcal{F}_\psi^{(k)}\}_{k=1}^K$ using the trajectories produced by the unified policy planner. Each decision model takes the rich context $\mathcal{H}(G^*)$ collected during the execution of G^* as input. This yields a unified policy planner with the specialized decision models, which jointly support consistent and globally coherent clinical decision-making across the entire CDM pipeline.

Table 2: Performance comparison on the ClinicalBench dataset.

Framework	Methods	Department Guide		Clinical Diagnosis						Imaging	Overall			
		Acc	DIFR	PD	DB	DD	FD	PT	TP	Diagnosis	DWR	CDR	Acceptability	Avg.
Proprietary LLMs	GPT5.1 [17]	0.7000	1.0000	0.4087	0.3779	0.2853	0.6304	0.3051	0.2169	0.4023	8.2609	0.5870	0.2233	0.4807 ↑0.29
	GPT-o3 [14]	0.7391	0.9993	0.4022	0.4213	0.3548	0.6065	0.3504	0.2290	0.3947	10.6087	0.5565	0.2256	0.4997 ↑0.27
	DeepSeek V3.2 [5]	0.7043	0.9926	0.4587	0.4508	0.3432	0.6370	0.3086	0.2780	0.5041	11.4783	0.5739	0.2477	0.5197 ↑0.25
	Kimi K2 [19]	0.5804	0.9133	0.3022	0.3763	0.2500	0.5217	0.3428	0.3599	0.4075	5.3913	0.4543	0.1822	0.4504 ↑0.32
Open-Source LLMs	MedGemma-27B [16]	0.6826	0.9847	0.3261	0.4597	0.3337	0.5587	0.3692	0.2851	0.3411	6.5652	0.4783	0.2002	0.4823 ↑0.30
	Baichuan-M2-32B [21]	0.6978	0.9953	0.4161	0.4628	0.3482	0.5643	0.3157	0.2656	0.4156	10.5652	0.5543	0.2274	0.4979 ↑0.27
	Qwen3-32B [26]	0.6108	1.0000	0.3543	0.4379	0.3273	0.4848	0.3282	0.2579	0.4166	7.2609	0.4978	0.2034	0.4687 ↑0.30
	Lingshu-32B [20]	0.6695	0.9820	0.3870	0.4018	0.4289	0.5522	0.3873	0.4069	0.3795	13.2174	0.5717	0.2626	0.5105 ↑0.26
	HuatuoGPT-o1-8B [2]	0.5173	0.9960	0.1369	0.1929	0.1887	0.3344	0.2218	0.3891	0.1190	1.0000	0.2543	0.0689	0.3440 ↑0.43
Multi-Agent System	MedAgent [18]	0.6760	0.9935	0.3122	0.4371	0.3394	0.4163	0.2801	0.2815	0.4195	5.3478	0.4130	0.1664	0.4617 ↑0.31
	MDAgent [9]	0.6543	0.9978	0.3311	0.4259	0.3347	0.4561	0.3171	0.2835	0.3369	4.3043	0.4783	0.1853	0.4597 ↑0.31
	MedChain-Agent [12]	0.8239	0.9978	0.4783	0.4421	0.3423	0.6000	0.3541	0.2666	0.5043	13.9565	0.6065	0.2645	0.5344 ↑0.24
	MedAgent-Pro [24]	0.8586	0.9870	0.5065	0.3159	0.2761	0.5935	0.3195	0.2687	0.2997	6.0435	0.5717	0.1892	0.4917 ↑0.28
Self-Evolving Agent	MDTeamGPT [3]	0.7260	0.9935	0.3457	0.4346	0.3321	0.4978	0.3422	0.2659	0.3454	7.1304	0.5391	0.2084	0.4759 ↑0.29
	EvoClinician [8]	0.6413	1.0000	0.4370	0.4285	0.3225	0.6370	0.3247	0.2591	0.3922	9.3913	0.6130	0.2457	0.4936 ↑0.28
	CoE-Agent	**0.8717**	**1.0000**	**0.5369**	**0.9802**	**0.5966**	**0.7241**	**0.8514**	**0.6752**	**0.7001**	**15.4783**	**0.6391**	**0.4058**	**0.7707** –

4 Experiments

4.1 Datasets and Experimental Settings

For **datasets**, we evaluate our method on two public medical CDM benchmarks: MedChain [12] (9,724 training / 2,439 test cases) and ClinicalBench [25] (1,040 training / 460 test cases). MedChain is a multimodal benchmark and ClinicalBench is text-only benchmark. Covering both multi-stage clinical decision-making and standardized medical reasoning tasks, they enable a comprehensive evaluation across diverse clinical stages. We follow the original evaluation protocols of each benchmark for fair comparison. For **implementation details**, both the patient and doctor agents are built upon Qwen3-4B. The doctor agent includes three components: a policy planner, a critic model, and an inference model, which share the same backbone architecture. In the first stage, the decision model remains frozen and serves as a stable evaluator to provide task-aligned reward signals, while the policy planner is optimized via IPGO. Source code is to be released. See Appendix Sec. E for details.

4.2 Comparison with the State-of-the-Arts

We conduct comparisons on both benchmarks against four types of approaches: proprietary LLMs, open-source LLMs, medical multi-agent systems, and self-evolving agents. As shown in Table 1 and Table 2, CoE-Agent consistently surpasses all competitors with average scores of 0.6313 on MedChain and 0.7707 on ClinicalBench, outperforming the strongest baseline MedChain-Agent by 11.13% and 23.63%, respectively. Despite being built on a compact Qwen3-4B backbone, CoE-Agent exceeds much larger proprietary and medical-specialized LLMs by 11%–26%, suggesting that robust interaction strategies and globally coherent policy structures matter more than parameter scaling or medical pre-training. Compared with multi-agent baselines, CoE-Agent yields the largest gains on tasks requiring long-range cross-stage reasoning, e.g., Specialty Referral Level1 (+13.90%) and Treatment (+18.49%) on MedChain, and Disease Basis (+25.46%) and Prescription of Treatment (+19.34%) on ClinicalBench, validating the benefit of modeling inter-task dependencies via the unified policy graph. Compared with self-evolving agents, CoE-Agent achieves 17–21% higher average scores, indicating that Interactive Policy Graph Optimization provides a more principled self-improvement signal than prompt or memory refinement. Moreover, on interaction-heavy tasks such as History Taking and Examination, where most baselines drop to 0.34–0.52 and several proprietary models even fail to produce valid outputs, CoE-Agent still attains 0.5900 and 0.7248, confirming its robustness to dynamic patient behaviors and establishing a new state of the art for globally coherent clinical decision-making.

4.3 Ablation Study

To assess the effectiveness of each component in CoE-Agent, we conduct an ablation study on the MedChain benchmark by progressively enabling the Dynamic Patient Agent (DPA), the Interactive Policy Graph-Based Doctor Agent (IPG-DA) within Co-Evolutionary Learning (CoEL), and the Cross-

Table 3: Ablation study on the MedChain dataset.

CoEL		PGC-UML	Average
DPA	IPG-DA		
			0.4465
✓			0.5528
	✓		0.5220
		✓	0.5500
✓	✓	✓	**0.6313**

Table 4: Computational efficiency Comparison.

Framework	Method	Inference Time / s
Multi-Agent System	MedAgent [18]	98.3
	MDAgent [9]	162.6
	MedChain [12]	94.6
	MedAgent-Pro [24]	154.8
Self-Evolving Agent	MDTeamGPT [3]	341.2
	EvoClinician [8]	180.7
	CoE-Agent	**17.5**

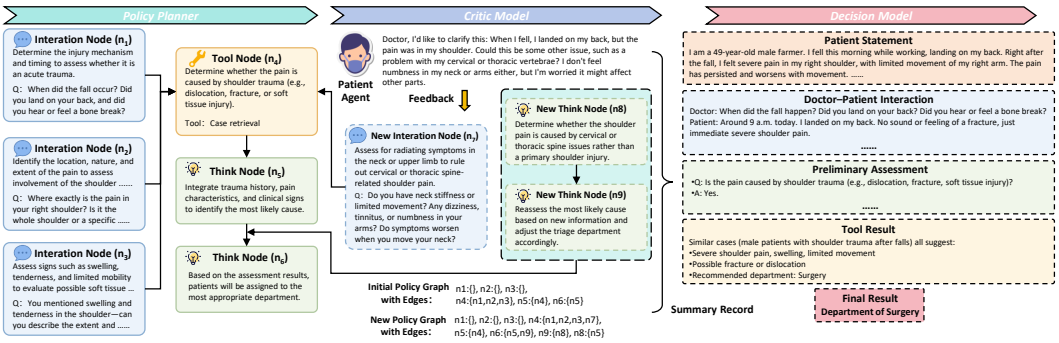


Figure 2: Visualization of Interactive Policy Graph-Based Doctor Agent.

Task Policy Graph Consolidation and Unified Model Learning (PGC-UML) in Table 3. Starting from the baseline (0.4465), adding DPA alone improves the average to 0.5528, indicating that simulating non-stationary and unreliable patient behaviors fosters more robust decision-making. Incorporating IPG-DA further enables structured policy graph generation with critic-driven revision, yielding additional gains over fixed sequential pipelines (0.5220). Applying PGC-UML alone raises performance to 0.5500, confirming that modeling inter-task dependencies via node pruning, cross-task aggregation, and unified planner distillation is essential for globally coherent decision-making. Finally, combining all three components achieves the best result of 0.6313 (+18.48% over the baseline), showing that the modules are highly complementary.

4.4 Computational Efficiency

Beyond accuracy, inference efficiency is critical for deploying medical agents in real clinical settings. We compare the average per-case inference time of CoE-Agent with seven baselines on MedChain. In Table 4, CoE-Agent requires only 17.5s per case, achieving a $5.4\times$–$9.3\times$ speedup over multi-agent systems and up to $19.5\times$ speedup over self-evolving agents. This efficiency mainly stems from the unified policy planner that replaces costly multi-role coordination with a single planning pass, together with reward-based pruning that removes redundant nodes to yield compact policy graphs.

4.5 Visualization of Interactive Policy Graph-Based Doctor Agent

To visualize the interpretability of doctor agent, Fig. 2 visualizes case about right scapular fracture from MedChain, walking through the three components of the doctor agent, i.e. policy planner, critic model, and decision model. Given a first-person chief complaint from the patient agent, policy planner emits a 6-node DAG-structured initial policy graph comprising three node types, each defined by an explicit purpose and a type-specific operation: interaction nodes (n_1–n_3) actively elicit information via targeted queries (e.g., *"When did the fall occur? Did you land on your back?"*); the tool node (n_4) invokes an external tool to verify the trauma hypothesis; and think nodes (n_5, n_6) perform internal sub-reasoning over upstream outputs to integrate evidence toward triage. After execution, critic model collects patient feedback and supports three revision operations (e.g. add, delete, modify) of which this case illustrates add: when the patient questions a possible *"cervical or thoracic"* origin, critic model inserts an interaction node n_7 (*"Assess radiating symptoms"*) and think nodes n_8, n_9

(*"Rule out spinal origin"*), and reconstructs edges to preserve DAG validity, yielding the new policy graph (n_4:$\{n_1, n_2, n_3, n_7\}$, n_6:$\{n_5, n_9\}$, n_9:$\{n_8\}$, n_8:$\{n_5\}$). Finally, decision model aggregates the refined trajectory into four structured fields, and outputs *Department of Surgery*, consistent with the ground truth, jointly verifying that policy model produces semantically complete DAGs, critic model converts patient feedback into valid structural edits, and decision model derives globally coherent clinical conclusions.

5 Conclusion

In this paper, we presented CoE-Agent, a co-evolving patient–doctor agentic framework for CDM. It integrates a CoEL stage, where a DPA with an evolving cognition state interacts with an IPG-DA optimized via IPGO for structural validity and task alignment, and a PGC-UML stage that prunes, aggregates, and distills task-specific policy graphs into a unified planner for globally coherent decision-making. Extensive experiments show that CoE-Agent significantly outperforms current baselines with substantial inference speedups. While CoE-Agent surpasses much larger proprietary LLMs on a single Qwen3-4B backbone, its behavior at larger scales (e.g., 32B or 70B) or across heterogeneous backbones remains future work. Broadly, CoE-Agent has the potential to assist medical education, support clinical reasoning training, and alleviate diagnostic workloads in under-resourced settings through interpretable, policy-guided decision-making.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grant 62576216, Guangdong Provincial Key Laboratory under Grant 2023B1212060076, and supported by the Intelligent Computing Center of Shenzhen University.

References

[1] Mohammad Almansoori, Komal Kumar, and Hisham Cholakkal. Self-evolving multi-agent simulations for realistic clinical interactions, 2025.

[2] Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. Huatuogpt-o1, towards medical complex reasoning with llms, 2024.

[3] Kai Chen, Xinfeng Li, Tianpei Yang, Hewei Wang, Wei Dong, and Yang Gao. Mdteamgpt: A self-evolving llm-based multi-agent framework for multi-disciplinary team medical consultation, 2025.

[4] Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory, 2025.

[5] DeepSeek-AI, Aixin Liu, et al. Deepseek-v3.2: Pushing the frontier of open large language models, 2025.

[6] Zhuoyun Du, Lujie Zheng, Renjun Hu, Yuyang Xu, Xiawei Li, Ying Sun, Wei Chen, Jian Wu, Haolei Cai, and Haochao Ying. LLMs can simulate standardized patients via agent coevolution. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17278–17306, Vienna, Austria, July 2025. Association for Computational Linguistics.

[7] Avi Fleischer, Alan D Mead, and Jialin Huang. Inattentive responding in mturk and other online samples. *Industrial and Organizational Psychology*, 8(2):196–202, 2015.

[8] Yufei He, Juncheng Liu, Zhiyuan Hu, Yulin Chen, Yue Liu, Yuan Sui, Yibo Li, Nuo Chen, Jun Hu, Bryan Hooi, Xinxing Xu, and Jiang Bian. Evoclinician: A self-evolving agent for multi-turn medical diagnosis via test-time evolutionary learning, 2026.

[9] Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik Siu Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae Won Park. Mdagents: An adaptive collaboration of llms for medical decision-making, 2024.

[10] Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, and Yang Liu. Agent hospital: A simulacrum of hospital with evolvable medical agents, 2025.

[11] Yahan Li, Xinyi Jie, Wanjia Ruan, Xubei Zhang, Huaijie Zhu, Yicheng Gao, Chaohao Du, and Ruishan Liu. Beyond idealized patients: Evaluating llms under challenging patient behaviors in medical consultations, 2026.

[12] Jie Liu, Wenxuan Wang, Zizhan Ma, Guolin Huang, Yihang SU, Kao-Jung Chang, Wenting Chen, Haoliang Li, Linlin Shen, and Michael Lyu. Medchain: Bridging the gap between llm agents and clinical practice with interactive sequence, 2025.

[13] Harald Merckelbach, Brechje Dandachi-FitzGerald, Daniel van Helvoort, Marko Jelicic, and Henry Otgaar. When patients overreport symptoms: More than just malingering. *Current Directions in Psychological Science*, 28(3):321–326, 2019.

[14] OpenAI. Openai o3 and o4-mini system card, 2025.

[15] ISABEL N Schuermeyer, Erin Sieke, Leah Dickstein, Tatiana Falcone, and Kathleen Franco. Patients with challenging behaviors: Communication strategies. *Cleve Clin J Med*, 84(7):535–542, 2017.

[16] Andrew Sellergren, Sahar Kazemzadeh, Tiam Jaroensri, Atilla Kiraly, Madeleine Traverse, Timo Kohlberger, Shawn Xu, et al. Medgemma technical report, 2026.

[17] Aaditya Singh, Adam Fry, Adam Perelman, et al. Openai gpt-5 system card, 2026.

[18] Xiangru Tang, Anni Zou, Zhuosheng Zhang, Ziming Li, Yilun Zhao, Xingyao Zhang, Arman Cohan, and Mark Gerstein. Medagents: Large language models as collaborators for zero-shot medical reasoning, 2024.

[19] Kimi Team, Yifan Bai, Yiping Bao, Y. Charles, Cheng Chen, et al. Kimi k2: Open agentic intelligence, 2026.

[20] LASA Team, Weiwen Xu, Hou Pong Chan, Long Li, Mahani Aljunied, Ruifeng Yuan, Jianyu Wang, Chenghao Xiao, et al. Lingshu: A generalist foundation model for unified multimodal medical understanding and reasoning, 2025.

[21] M2 Team, Chengfeng Dou, Chong Liu, Fan Yang, Fei Li, Jiyuan Jia, Mingyang Chen, Qiang Ju, Shuai Wang, et al. Baichuan-m2: Scaling medical capability with large verifier system, 2025.

[22] Marsha L Vanderford, Terry Stein, Robert Sheeler, and Susan Skochelak. Communication challenges for experienced clinicians: topics for an advanced communication curriculum. *Health Communication*, 13(3):261–284, 2001.

[23] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models, 2023.

[24] Ziyue Wang, Junde Wu, Linghan Cai, Chang Han Low, Xihong Yang, Qiaxuan Li, and Yueming Jin. Medagent-pro: Towards evidence-based multi-modal medical diagnosis via reasoning agentic workflow, 2025.

[25] Weixiang Yan, Haitian Liu, Tengxiao Wu, Qian Chen, Wen Wang, Haoyuan Chai, Jiayi Wang, Weishan Zhao, Yixin Zhang, Renjun Zhang, et al. Clinicallab: Aligning agents for multi-departmental clinical diagnostics in the real world. *arXiv preprint arXiv:2406.13890*, 2024.

[26] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, et al. Qwen3 technical report, 2025.

[27] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

[28] Jenny Zhang, Shengran Hu, Cong Lu, Robert Tjarko Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. In *The International Conference on Learning Representations*, 2026.

[29] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. Aflow: Automating agentic workflow generation, 2025.

[30] Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, pages 19632–19642, 2024.

[31] Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

[32] Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 113959–113991. Curran Associates, Inc., 2025.

Appendix for CoE-Agent

Abstract.

Appendix A presents the comprehensive ablation study on the MedChain dataset.

Appendix B presents the computational efficiency analysis, comparing our method with medical multi-agent systems and self-evolving approaches, and demonstrating the advantage of our unified policy mechanism in reducing inference cost.

Appendix C provides a detailed visualization of the Patient and Doctor Agents, including the construction of the dynamic cognition state, its update process, and how error propagation occurs across sequential clinical tasks.

Appendix D illustrates the policy graph for each task and the unified policy graph, highlighting how high-value decision structures are extracted, merged, and reused across stages to achieve global policy optimization.

Appendix E provides the datasets and implementation details.

Appendix F and G include the prompts used for the Patient Agent, Doctor Agent, and Unified Policy Planner, respectively, providing additional implementation details for reproducibility.

A Ablation Study

To validate the effectiveness of each component, we conduct systematic ablation studies on the key modules of our framework, as shown in Table 5. Overall, our method consists of three core components: the Dynamic Patient Agent, the Interactive Policy Graph-Based Doctor Agent, and Cross-Task Policy Graph Consolidation and Unified Model Learning. Notably, the latter is tightly coupled with the Doctor Agent; when used independently, it degenerates into standard SFT on the Decision Model.

Starting from the base setting without any modules (i.e., using the raw model with static inputs), the overall performance is relatively low (0.4465), indicating that direct decision-making based on static inputs is insufficient for complex clinical workflows. When only patient dialogue data is introduced (i.e., replacing raw formatted input with policy graph representation), performance improves to 0.5528, demonstrating that policy graph modeling is highly effective. Even without optimization, using policy graphs alone yields a 10.63% improvement.

Further, applying SFT alone (without policy graph modeling) results in a performance of 0.5500, showing limited gains and suggesting that static supervised learning is insufficient for modeling complex decision processes. When combining policy graph modeling with SFT, performance increases to 0.5881, achieving an additional 3.81% improvement, which indicates that structured policy representation is more effective than static structured outputs. When both the Patient Agent and Doctor Agent are introduced, performance further improves significantly to 0.6288, validating the effectiveness of combining structured policy modeling with reinforcement learning optimization.

Finally, integrating all components yields the best performance (0.6313). Compared to the setting without Cross-Task Policy Graph Consolidation and Unified Model Learning, performance improves by 0.25%. This result shows that cross-task policy consolidation can effectively integrate high-value decision structures across tasks, enabling global policy optimization and further improving performance in multi-stage clinical workflows. At the same time, it unifies multi-stage policy graphs into a single model, significantly reducing computational cost and resource consumption while maintaining strong performance.

B Computational Efficiency

We further evaluate the computational efficiency of different methods, as shown in Table 6. The compared approaches mainly include medical multi-agent systems (MMAS) and self-evolving methods. The reported results correspond to the average inference time for the Specialty Referral task on the MedChain dataset. Since a complete clinical workflow in MedChain consists of five sequential tasks, the overall inference cost of these methods can be approximated as single-task time ×

Table 5: Ablation study on the MedChain dataset.

CoEL		PGC-UML	Specialty Referral		History	Exam	Diag	Treat	Avg
DPA	IPG-DA		Level1	Level2					
			0.5153	0.3234	0.4441	0.4734	0.4750	0.4479	0.4465
✓			0.6478	0.4225	0.4392	0.6658	0.5050	0.6367	0.5528
	✓		0.6022	0.4610	0.3863	0.5537	0.5137	0.6153	0.5220
		✓	0.6929	0.4634	0.5452	0.4808	0.4665	0.6513	0.5500
✓	✓	✓	**0.7281**	**0.4911**	**0.5900**	**0.7248**	**0.5472**	**0.7064**	**0.6313**

Table 6: Computational Efficiency Comparison (Inference Time in Seconds)

Framework	**Method**	**Inference Time / s**
MAS	MedAgent	19.66 * 5
	MDAgent	32.51 * 5
	MedChain	18.92 * 5
	MedAgent-Pro	30.96 * 5
Self-evolution	MDTeamGPT	68.24 * 5
	EvoClinician	36.16 * 5
	Ours	**17.45 * 1**

5. Methods such as MedAgent and MedChain rely on multi-agent collaboration to perform stage-wise decision-making, while MDAgent and MedAgent-Pro further introduce complex coordination and tool usage. Self-evolving approaches (e.g., MDTeamGPT and EvoClinician) additionally require iterative optimization and experience accumulation, leading to even higher inference costs.

In contrast, our method demonstrates a clear advantage in computational efficiency. For a single task, our approach achieves the lowest inference time, requiring only 17.45 seconds. This indicates that although our framework involves a relatively complex policy graph, its actual execution cost remains low while still delivering significant performance gains, validating the effectiveness of our policy graph modeling. More importantly, our method completes the entire clinical workflow in 17.45 seconds (×1), whereas other methods require repeated execution across multiple stages (e.g., MedAgent: 19.66 × 5, MDAgent: 32.51 × 5). Overall, our method achieves substantially lower inference time compared to all baselines.

This efficiency gain mainly stems from our proposed Cross-Task Policy Graph Consolidation and Unified Model Learning mechanism. Through this module, we integrate high-quality decision structures across tasks and learn a unified global policy. During inference, this unified policy models the entire preceding decision process in a single pass, eliminating the need to repeatedly execute full multi-agent interactions or self-evolving procedures for each subtask. Subsequently, the final outputs for each stage are obtained by simply parsing the execution results using the Decision Model. Experimental results demonstrate that our method not only outperforms existing approaches in performance but also achieves superior computational efficiency. This "generate once, reuse across tasks" paradigm significantly enhances practicality and scalability for real-world clinical applications.

C Visualization of Patient and Doctor Agents

The core of the Patient Agent lies in a dynamically updated cognition state, which serves as a unified representation of all information the patient "perceives" at a given moment, including symptom descriptions, medical history, examination results, and the patient's understanding of their own condition. Unlike traditional approaches that model the patient as a static information provider, we treat it as a continuously evolving cognitive entity driven by interaction. The overall construction of the Patient Agent, along with the processes of cognition updating and information propagation, is illustrated in Figure 3.

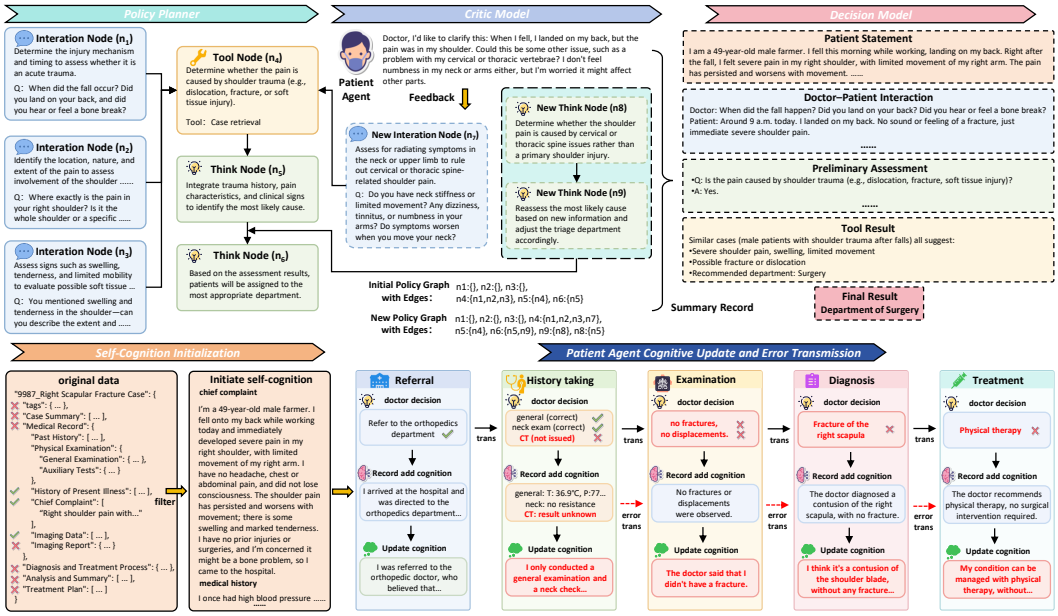


Figure 3: Visual analysis of doctor agent in triage task of the MedChain dataset.

At the initialization stage, the cognition state is constructed solely from information available at the initial consultation, such as the chief complaint and basic medical history. All future information (e.g., examination results and final diagnoses) is strictly removed to prevent information leakage. The filtered patient information is then reformulated in the first-person perspective and provided to the Patient Agent for role-playing, closely simulating real clinical scenarios. This primarily includes the current chief complaint and past medical history, enabling the Patient Agent to form an initial understanding of its condition and support subsequent interactions.

In the multi-stage clinical workflow, the cognition state is continuously propagated and accumulated across tasks, forming a cross-stage trajectory of cognitive evolution. This design allows early decisions to exert long-term effects on subsequent tasks, explicitly modeling error propagation and information dependency in real-world clinical settings. Specifically, the doctor's questions, examination recommendations, and diagnostic conclusions act as external inputs that influence and update the patient's cognition. These updates not only incorporate newly acquired information (e.g., examination results) but may also adjust or even distort existing cognition. For instance, when the doctor makes an incorrect judgment, the patient may develop a misunderstanding, leading to inconsistent or misleading responses in later interactions. In contrast, accurate clinical decisions help progressively refine the patient's cognition, resulting in more stable and reliable information expression. As shown in Fig. 3, we present an example from the MedChain dataset ("Case 9987: Right Scapular Fracture") to illustrate cognition initialization, updating, and error propagation. In this case, during the Specialty Referral stage, the Doctor Agent makes correct decisions, and the execution results of the policy graph are incorporated into the Patient Agent's cognition and carried forward to subsequent tasks. However, in the History Taking stage, if the Doctor Agent orders incorrect or incomplete examinations (e.g., missing a CT scan), the corresponding results cannot be retrieved, leading to errors that are incorporated into the cognition state and subsequently affect downstream tasks. More critically, if the Doctor Agent makes an incorrect diagnosis or description at a key stage, it can cause catastrophic failure in all subsequent tasks. For example, in the Examination stage of this case, the Doctor Agent incorrectly interprets the imaging results as showing no fracture or displacement. This incorrect information is then integrated into the Patient Agent's cognition, which directly leads to an incorrect diagnosis in the Diagnosis stage (e.g., diagnosing a shoulder contusion instead of a fracture), and consequently results in an inappropriate treatment plan (e.g., choosing physical therapy instead of surgery). Compared to MedChain-Agent, the error propagation in our setting is not limited to a single subsequent input but persists throughout all following interaction stages.

From an interpretability perspective, the behavior of the Patient Agent can be viewed as an external manifestation of its cognition state: each response is generated based on the current cognition, and changes in cognition directly reflect the flow of information and the impact of decisions during doctor–patient interactions. Therefore, this design not only enhances the realism of the simulated environment but also provides a clear and traceable explanation pathway for analyzing model behavior in complex clinical interactions.

D Visualization of Policy Graph for Each Task and Unified Policy Graph

To intuitively illustrate the behavior of the proposed method in multi-stage clinical decision-making, we conduct a comprehensive visualization analysis of the policy graphs generated for each subtask (Task 1–Task 5) as well as the final unified policy graph. A representative case from the MedChain dataset, "Case 9974: Left Femoral Neck Fracture," is selected for detailed analysis, as shown in Figure 4. For clarity, this case study focuses on the construction of the policy graph at each stage and the unified policy graph, while omitting the Critic Model process.

In the Specialty Referral stage (Task 1), the policy graph primarily revolves around the patient's initial complaint. The graph consists of three types of nodes. The Policy Planner first generates a complete policy graph based on the patient's statement and prior context, and the execution results are then passed to the Decision Model for final prediction. Typically, the graph begins with interaction nodes, which acquire key clinical information—such as injury mechanism, pain location, and severity—through multi-turn questioning (e.g., "Confirm the injury site and pain characteristics after the fall, and determine whether it is acute…", "Assess post-injury mobility limitations and whether basic function is affected."). In this setting, all relevant information must be actively obtained through doctor–patient interaction (e.g., "Q: Do you have dizziness, nausea, chest discomfort, abdominal pain, fever, or any urinary abnormalities?"). After sufficient interaction, the model obtains critical clinical insights. These interaction nodes are followed by either tool nodes or think nodes. Tool nodes specify the use of external tools and return corresponding results, while think nodes perform intermediate reasoning based on previously collected information, aiming to deepen reasoning and rule out alternative conditions. After all nodes are executed, the aggregated information is fed into the Decision Model to produce the final output. Importantly, the decisions made by the Doctor Agent at each stage directly influence the Patient Agent's cognition and the overall task outcome, thereby affecting the accuracy of information available in subsequent stages. For instance, if incorrect examinations are prescribed in the History Taking stage (Task 2) or misinterpretations occur in the Examination stage (Task 3), such errors (as illustrated in the "Medical examination results" in Figure 4) will propagate to all subsequent tasks, leading to compounded failures.

Within the Cross-Task Policy Graph Consolidation and Unified Model Learning framework, we further construct a unified policy planner that generates a single policy graph covering the entire five-stage clinical workflow. Compared to generating separate policy graphs for each task, the unified graph is more compact in structure, often with fewer or comparable nodes but significantly higher information density. This is achieved by retaining only high-value decision nodes during the consolidation process and merging semantically or functionally redundant nodes across tasks, thereby reducing redundancy. Moreover, the unified policy graph explicitly models cross-stage dependencies, allowing key decision nodes to serve multiple stages without repeated generation.

Specifically, using interaction nodes as examples, we analyze the correspondence between the unified policy and task-specific policies to validate its effectiveness. The interaction nodes in the unified policy can be clearly traced back to multiple task stages, demonstrating that the unified policy is not newly generated but rather a compression and reorganization of high-value decision patterns. For instance, the unified node "Check for any recent head injury or trauma" can be traced to core inquiries in Task 1 and Task 2 regarding injury mechanisms. In Task 2, this corresponds to "Understand the details of the fall, including timing and landing mechanism," while in Task 1 it relates to determining whether the injury is acute. These nodes repeatedly appear across tasks, highlighting their fundamental importance in clinical reasoning and justifying their inclusion in the unified policy.

Similarly, the unified node "Identify other systemic symptoms or conditions" also has cross-task origins. In Task 1, it corresponds to "Rule out possible visceral or systemic injuries," serving as an initial risk screening step, while in Task 4 it appears as "Identify neurological or systemic symptoms" to support diagnosis and differential analysis. This demonstrates that the node spans multiple stages

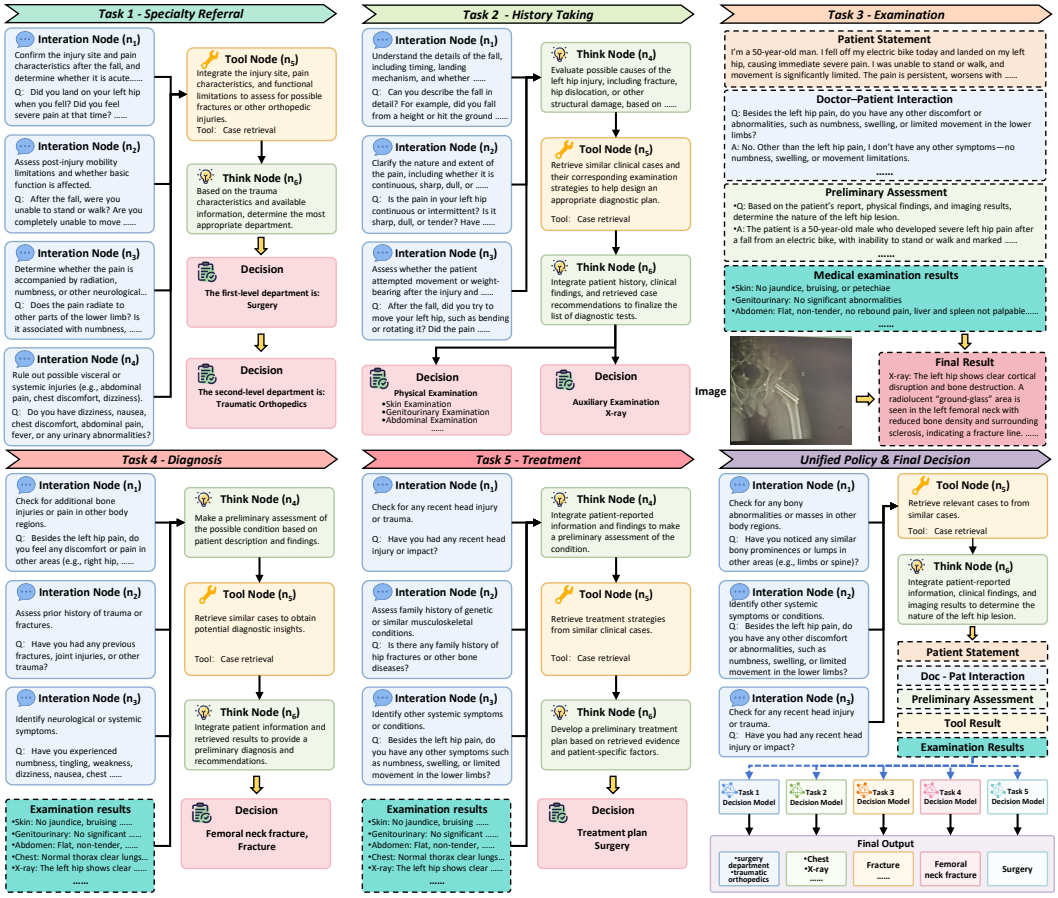


Figure 4: Visual analysis of doctor agent across all the tasks from the MedChain dataset.

from early screening to diagnosis, and is therefore abstracted into a more general form in the unified policy. Finally, the node "Check for any bony abnormalities or masses in other body regions" closely aligns with "Check for additional bone injuries or pain in other body regions" in Task 4, serving as an extension of diagnostic reasoning to avoid missing multiple injuries or related conditions. These observations show that nodes in the unified policy are not isolated but originate from repeatedly occurring, high-impact decision patterns across tasks. By abstracting and merging these nodes, the unified policy preserves essential cross-task decision logic while significantly reducing redundancy, thereby improving both efficiency and generalization without sacrificing performance.

E Datasets and Implementation Details

Datasets and Evaluation Metrics. We evaluate the proposed method on two public medical benchmarks: MedChain and ClinicalBench. These datasets cover both multi-stage clinical decision-making processes and standardized medical reasoning tasks, enabling a comprehensive assessment of model performance across different clinical stages. Importantly, they exhibit strong sequential dependency, where decisions made in earlier stages directly influence the availability of information and decision pathways in subsequent stages, making them well-suited for evaluating cross-stage dependency modeling and global policy optimization. To ensure fair comparison, we follow prior work and adopt their evaluation protocols. Notably, we use the Chinese version of the datasets. In ClinicalBench, we evaluate department guide task via accuracy and Department Instruction Following Rate (DIFR). In clinical diagnosis, PD task is assessed via accuracy, and the rest of task is evaluated through the average results of BLEU, ROUGE, and BERTScore. For the overall assessment, we compute the average (Avg.) score over all the 9 scores under the 8 tasks, and use Department Win Rate (DWR) and Comprehensive Diagnostic Rate (CDR) to evaluate.

Implementation Details. For MedChain, we split the dataset into training and test sets with a ratio of 8:2. For ClinicalBench, the training set contains 1,040 samples and the test set contains 460 samples. Both the Patient Agent and Doctor Agent are built upon Qwen3-4B. The Doctor Agent consists of three core components: a policy model, a critic model, and an inference model, which share the same backbone architecture but serve different roles. For training, our proposed Interactive Policy Graph Optimization (IPGO) is implemented based on Group Relative Policy Optimization (GRPO) to optimize the policy graph generated by the Policy Planner. During this process, the Decision Model remains frozen and serves as a stable evaluator to provide outcome-aligned reward signals, while the Policy Planner is updated under both structural constraints and task feedback. In the second stage, we perform supervised fine-tuning (SFT) on the Unified Policy Planner using the recomposed high-quality policy graph data. Furthermore, we leverage the execution trajectories of the policy graphs to conduct task-aligned training of the Decision Model, improving its performance on downstream tasks. The hyperparameters of IPGO are configured as follows. We adopt a learning rate of 5×10^{-6} with the AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.99$, weight decay $= 0.1$), and employ a cosine learning rate scheduler with a warmup ratio of 0.1. The model is trained using bfloat16 precision with a per-device batch size of 4 and gradient accumulation steps of 4. During training, the maximum prompt length and completion length are both set to 1024, and the number of generations for GRPO sampling is set to 4. Gradient clipping is applied with a maximum gradient norm of 0.1. For parameter-efficient fine-tuning, we employ LoRA with rank $r = 32$, LoRA alpha $= 64$, and dropout $= 0.05$. The LoRA modules are applied to the attention and feed-forward layers, including `q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, and `down_proj`. All experiments are based on the Qwen3-4B model initialized in bfloat16 format. We use DeepSpeed ZeRO-3 offloading for memory optimization during training. All experiments are conducted on 4 NVIDIA A6000 GPUs, and the entire training process is completed in less than two days.

F Prompts for Patient Agent

In this appendix, we further provide the detailed prompts used for the construction of the Patient Agent. Specifically, we present the prompts for two core components: (1) the patient response generation process during doctor–patient interaction, and (2) the patient feedback generation process for evaluating and commenting on the triage report. As shown in Fig. 5 and Fig. 6

G Prompts for Doctor Agent

In this appendix, we further present the detailed prompts used in the construction of the Doctor Agent. The Doctor Agent mainly consists of three components: the Policy Planner, the Critic Model, and the Decision Model. The following presents their specific prompts. As shown in Fig. 7, Fig. 8 and Fig. 9.

```
## Task
- You are now required to act as a patient communicating with a doctor, helping the doctor understand your condition. The conversation history and historical memory up to this point are provided below to assist your responses.
- The current simulation process has actually already happened; we are only replaying the medical consultation scenario from the beginning.
- "Historical Records" represent information that actually occurred during this real consultation, and you may use these records to support your responses.
- However, your responses must strictly avoid revealing any information that would only become known later in the consultation process, such as subsequent diagnosis results, examination findings, treatment plans, or doctor decisions.
- You may only answer based on information that is already known at the current stage or before the consultation.
- If there is no corresponding information available for the doctor's question, simply reply with phrases such as "I'm not sure" or "I don't know."
- Your output should only answer the doctor's current question, without any additional explanation or commentary.

## Historical Records
{history}

## Patient Self-Report
{dialogue_record}

## Current Doctor Question
{question}
```

Figure 5: The prompt of the patient's responses during the doctor-patient interaction.

Task
- You are currently playing the role of a patient visiting a hospital, and your current medical condition is described below.
- You have now received a triage report provided by the hospital. The report records the interaction between you and the doctor during the medical triage stage (presented in the form of a directed acyclic graph), as well as the final triage department result.
- Please read the report carefully. If there are any errors, missing information, or additional details that should be supplemented, provide a brief description. If no modification or supplementation is needed, simply output "None".
- Your response should be concise and written strictly from the patient's own perspective.
- Note that your current task is to check whether the report contains omissions, inaccuracies, or missing descriptions, rather than questioning the correctness of the final triage department.
- The currently available hospital departments are listed under "Available Primary Departments". Only one department can be selected for consultation. Since this is only a primary-level triage, it is sufficient as long as the general triage direction is appropriate. More detailed secondary department classification will follow later.
- Because you are acting as a patient, your response should sound as if you are directly replying to the doctor from the patient's perspective.

Available Primary Departments
['Nursing Department', 'Pharmacy Department', 'Dentistry', 'Pediatrics', 'Medical Imaging Department', 'Ophthalmology', 'Laboratory Department', 'Surgery', 'Dermatology & Venereology', 'Psychiatry', 'General Practice', 'Otorhinolaryngology', 'Internal Medicine', 'Emergency Department', 'Oncology', 'Traditional Chinese Medicine', 'Rehabilitation Department', 'Obstetrics & Gynecology', 'Psychology Department']

Patient Condition
Patient Statement: {description}

Triage Report
{report}

Figure 6: The prompt of the patient's opinion on the policy.

Task
- You are now required to complete a medical triage task. Based on the patient's current description, you need to assign the patient to the most appropriate primary department.
- Before making the triage decision, you should first create a decision-making plan describing how you intend to complete the task. The plan should be organized step-by-step in a directed acyclic graph (DAG)-style structure.
- During the planning process, the only thing you are allowed to do is communicate with the patient. In the "question" section, you may write the questions you would like to ask the patient, or "None" if no question is needed.
- Since this is only a triage process, you only need to plan how to assign the patient to an appropriate department based on the current information. Do NOT suggest or order any examinations or tests.
- During the planning process, you may use existing tools to assist your judgment. The currently available tool format is shown below. If additional tools are needed, describe them in the same format. If no tools are needed, output "None".
  > Tool Name: Case Retrieval
  > Tool Description: Input the patient's current condition and return triage outcomes from similar historical cases.
- Each planning step, patient question, and additional tool requirement can be treated as a node in the graph. You should also clarify the dependencies between nodes, and describe the graph connections in the "edges" section of the output example.
- Node types are divided into two categories:
  - "search": the node task mainly involves information gathering or retrieval.
  - "solve": the node task mainly involves reasoning or decision-making.
- Please note that your responsibility is only to comprehensively think through the task and consider different possibilities. You do NOT need to produce the final reasoning result or give the final department assignment.
- Only output the structured content shown in the "Output Example". Do not output any additional explanations.

Patient Statement
{description}

Output Example
```
{
  "nodes": [
    {
      "node_id": "n1",
      "task_type": "search",
      "task_desc": "First, confirm the details of the swallowed foreign object, including its material, size, and shape, in order to evaluate possible risks and management strategies.",
      "question": "Can you describe the shape and size of the swallowed metal object? Did it have any sharp edges?",
      "tools": "Tool name and description"
    },
    {
      "node_id": "n2",
      "task_type": "search",
      "task_desc": "Understand the patient's symptoms after swallowing the foreign object, including pain location, severity, and duration, to assess potential complications.",
      "question": "Where exactly is your abdominal pain located? Is the pain continuous or intermittent? How severe is it?",
      "tools": "Tool name and description"
    },
    {
      "node_id": "n3",
      "task_type": "solve",
      "task_desc": "Assess whether the patient has other symptoms after swallowing the foreign object, such as bleeding, hematemesis, or melena, in order to evaluate the possibility of gastrointestinal injury.",
      "question": "After swallowing the metal object, have you experienced vomiting blood, black stools, or any other signs of gastrointestinal bleeding?",
      "tools": "Tool name and description"
    },
    {
      "node_id": "n4",
      "task_type": "solve",
      "task_desc": "Evaluate the patient's mental and physical condition after swallowing the foreign object to assess possible complications.",
      "question": "You mentioned feeling mentally unwell. Could you describe the symptoms in more detail? Have you experienced dizziness or fatigue?",
      "tools": "Tool name and description"
    },
    {
      "node_id": "n5",
      "task_type": "solve",
      "task_desc": "Integrate all collected information to evaluate the risks and possible complications caused by swallowing the foreign object, and determine the most appropriate department for further management.",
      "question": "None",
      "tools": "Tool name and description"
    },
    {
      "node_id": "n6",
      "task_type": "solve",
      "task_desc": "Based on the patient's specific condition, assign the patient to the most appropriate department.",
      "question": "None",
      "tools": "Tool name and description"
    }
  ],
  "edges": [
    {"from_node": "n1", "to_node": "n5"},
    {"from_node": "n2", "to_node": "n5"},
    {"from_node": "n3", "to_node": "n5"},
    {"from_node": "n4", "to_node": "n5"},
    {"from_node": "n5", "to_node": "n6"}
  ]
}
```

Figure 7: The prompt of Policy Planner.

Task
- The current process is a medical triage task, where the patient should be assigned to the most appropriate department based on their condition.
- You are now acting as a medical expert. Based on the patient's condition, you need to review the current triage plan and execution process again.
- Your task contains two objectives:
  1. Determine whether the primary-level triage decision is correct.
  2. If the triage decision is incorrect, explain the issue and repair the plan by adding new nodes. The updated report will later be executed to generate the final answer.
- To support your reflection, the current triage plan and results have already been shown to the patient. The patient's feedback is provided below for your reference.
- However, your judgment must still primarily come from the perspective of a medical expert. The patient's opinion should only be used as a reference, and you should critically evaluate it rather than blindly following it.
- If you determine the triage decision is correct, simply output:
  "Correct"
  without any additional explanation or analysis.
- If the triage decision is incorrect, output:
  "Incorrect"
  and then supplement the report by adding new nodes to improve the reasoning process.
- Please note that at this stage you are NOT allowed to order or suggest any medical examinations or tests. You may only gather information through dialogue with the patient in order to complete your reasoning.

Available Primary Departments
['Nursing Department', 'Pharmacy Department', 'Dentistry', 'Pediatrics', 'Medical Imaging Department', 'Ophthalmology', 'Laboratory Department', 'Surgery', 'Dermatology & Venereology', 'Psychiatry', 'General Practice', 'Otorhinolaryngology', 'Internal Medicine', 'Emergency Department', 'Oncology', 'Traditional Chinese Medicine', 'Rehabilitation Department', 'Obstetrics & Gynecology', 'Psychology Department']

Triage Report Description
The triage plan and results are presented in the form of a directed acyclic graph (DAG). The nodes are mainly divided into two categories:
- "search" nodes: used to ask the patient questions. The "question" field contains the specific question content.
- "solve" nodes: used for thinking, analysis, and reasoning tasks at the current step.

Triage Report
{report}

Patient Feedback
{opinion}

Output Example (when the report is correct)
Correct

Output Example (when the report is incorrect)
Incorrect

Additional Nodes:
- {'task_type': 'search', 'task_desc': '.....', 'question': '.....'}
- {'task_type': 'solve', 'task_desc': '.....', 'question': '.....'}
- {'task_type': 'solve', 'task_desc': '.....', 'question': '.....'}
.....

Figure 8: The prompt of Critic Model.

Task
- You are now required to complete a medical triage task. Based on the patient's current description, assign the patient to the single most appropriate primary department from the provided department list.
- Below is a series of doctor-patient interaction records. Please make the final triage decision based on all the provided information.
- You should only output the final primary department name, without any explanation or additional content.

Known Information
Patient Statement
{description}

Doctor-Patient Conversation Records
{dialogue_record}

Some Diagnoses
{diagnosis}

Tool Call Results
{tools_result}

Available Primary Departments
['Nursing Department', 'Pharmacy Department', 'Dentistry', 'Pediatrics', 'Medical Imaging Department', 'Ophthalmology', 'Laboratory Department', 'Surgery', 'Dermatology & Venereology', 'Psychiatry', 'General Practice', 'Otorhinolaryngology', 'Internal Medicine', 'Emergency Department', 'Oncology', 'Traditional Chinese Medicine', 'Rehabilitation Department', 'Obstetrics & Gynecology', 'Psychology Department']

Output Example
Dermatology & Venereology

Figure 9: The prompt of Decision Model.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (Sec. 1) clearly state the two challenges (dynamic patient behaviors and inter-task dependencies) and the two corresponding contributions (CoEL with IPGO in Sec. 3.1, and PGC-UML in Sec. 3.2), which are supported by the experimental results on MedChain and ClinicalBench reported in Sec. 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our work in the Conclusion section (Sec. 5)

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results or theorems that require proofs; the contributions are methodological and empirical.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sec. 3 fully describes the method (policy graph formulation, IPGO objective, reward design, pruning, and unified planning), and Sec. 4.1 specifies datasets (MedChain, ClinicalBench), the Qwen3-4B backbone, training stages, and evaluation protocols sufficient to reproduce the main results. We will release the code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Both benchmarks (MedChain and ClinicalBench) are publicly available and cited.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4.1 provides dataset splits, the shared Qwen3-4B backbone for the policy, critic, and decision models, the frozen-decision-model setting during IPGO, and the adherence to each benchmark's original evaluation protocol. Additional hyperparameter details are provided in the Appendix Sec. E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The reported results in Tables 1 and 2 are single-run scores following the evaluation protocols of MedChain and ClinicalBench. Multi-run error bars are not included due to the high computational cost of running full multi-agent CDM pipelines.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We mention these details in the Appendix Sec. E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. We use only publicly available de-identified medical benchmarks (MedChain, ClinicalBench), do not involve direct interactions with real human subjects, and the simulated patient agents do not expose any personally identifiable information.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The potential impact is clarified in conclusion section.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release pretrained generative models, scraped datasets, or any artifacts with high risk of misuse. All experiments rely on already-public benchmarks and a released base model (Qwen3-4B); no new high-risk model or data is introduced.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the original papers for all existing assets used, including the MedChain [12] and ClinicalBench [25] benchmarks and the Qwen3-4B [26] backbone, and we use them in accordance with their released licenses and terms of use.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or pretrained models at submission time. Source code will be released upon acceptance with accompanying documentation.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or any direct research with human subjects. All "patient" interactions are produced by a simulated Dynamic Patient Agent built on publicly available de-identified benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The study does not involve human subjects research; it only uses public de-identified medical CDM benchmarks and simulated patient agents, so IRB approval is not applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are a core, non-standard component of this research: Qwen3-4B is used as the shared backbone for the policy planner, critic model, decision model, and dynamic patient agent, as described in Sec. 4.1 and Sec. 2. Any LLM usage for writing or editing purposes does not affect the core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.