Can Dependencies Induced by LLM-Agent Workflows Be Trusted?

Yu Yao[♠*], **Yiliao Song**[♣*], **Yian Xie**[♢], **Mengdan Fan**[♡], **Mingyu Guo**[♣], **Tongliang Liu**[♠✉]
[♠]Sydney AI Centre, The University of Sydney
[♣]Adelaide University [♢]Monash University [♡]Peking University
tongliang.liu@sydney.edu.au

Abstract

LLM-agent systems often decompose high-level objectives into subtask dependency graphs, assuming that each subtask's output is reliable and conditionally independent of others given its parent responses. However, this assumption frequently breaks during execution, as ground-truth responses are inaccessible, leading to *inter-agent misalignment*—failures caused by inconsistencies and coordination breakdowns among agents [1]. To address this, we propose SEQCV, a dynamic framework for *reliable* execution under violated conditional independence. SEQCV executes subtasks sequentially, each conditioned on all prior verified responses, and performs consistency checks immediately after agents generate short token sequences. At each checkpoint, a token sequence is accepted only if it represents shared knowledge consistently supported across diverse LLM models; otherwise, it is discarded, triggering recursive subtask decomposition for finer-grained reasoning. Despite its sequential nature, SEQCV avoids repeated corrections on the same misalignment and achieves higher effective throughput than parallel pipelines. Across multiple reasoning and coordination tasks, SEQCV improves accuracy by up to 30% over existing LLM-agent systems. Code is available at github.com/tmllab/2025_NeurIPS_SeqCV.

1 Introduction

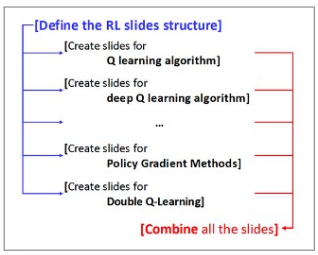
LLM-agent systems are designed to conceptualize the powerful large language models (LLMs) to perform a broad range of tasks [2, 3, 4]. On one hand, existing research focuses on improving the performance of LLM-agent systems across various applications. For example, some systems are tailored for specific domains such as software development [5, 6, 7], scientific discovery [8, 9, 10], and human behaviour simulation [11, 12, 13]. Others focus on augmenting LLM agents with external tools [14, 15, 16], or on establishing benchmarks to evaluate LLM-agent performance [17, 18, 19, 20].

On the other hand, recent studies have raised two critical questions: 1) what constitutes an LLM-agent system [21, 22], and 2) which attributes most influence its performance [1, 23]. Although there is no clear agreement on these two questions [24, 25], one certainty has emerged: *multi-LLM-agent systems* demonstrated superior performance than single LLM-agent systems, highlighting the importance of orchestrating collaboration among LLM-agents to optimize task execution [26, 21, 1].

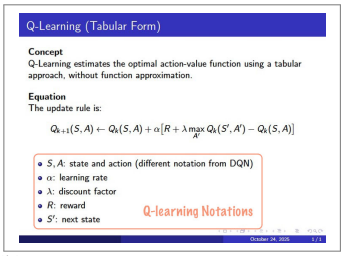
To enhance the reasoning capabilities of LLMs for complex tasks, *a high-level task objective is often decomposed* into a series of intermediate reasoning steps, such as chain of thought (CoT) [27], ReAct [28], tree of thought (ToT) [29], Reflexion [30]. In LLM-agent systems, these steps correspond to *subtasks*, as exemplified by AutoGPT [31], BabyAGI [32], LangChain [33], and Llama-index [34]. Recent studies suggest that *decomposing tasks as graphs* can further enhance performance, in both

[*] Equal Contributions. ✉ Corresponding Author.

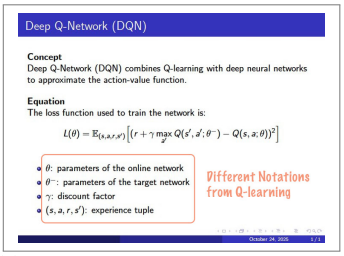
39th Conference on Neural Information Processing Systems (NeurIPS 2025).



(a) Task 1 defines the conceptual skeleton and corresponding LaTeX style, which should be inherited by later tasks.



(b) An intermediate subtask generates additional content following the initial structure.

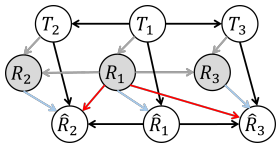


(c) Another parallel subtask produces a new style instead of inheriting the existing one, leading to inter-agent misalignment.

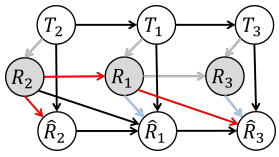
Figure 1: Inter-agent misalignment `[MASFT 2.5 [1]]` when generating reinforcement learning teaching slides in LaTeX. Task 1 defines the structure, while downstream parallel tasks independently create slides, leading to inconsistent styles.

single-agent systems, such as graph of thought (GoT) [35] and LangGraph [36], and multi-agent systems, including GPTSwarm [37], Flow [38], AFlow [39], AgentPrune [40], and AoT [41].

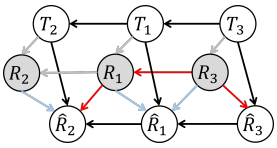
These *graph-as-workflow* orchestration systems often assume that each subtask produces a reliable response that depends only on its parent nodes. In practice, this assumption of *conditional independence* often fails, leading to the problem of *inter-agent misalignment*. When subtasks are executed in parallel without conditioning on one another's intermediate outputs, agents may produce inconsistent or incompatible results. Figure 1 illustrates such a case in a multi-agent setup for RL slide generation. The workflow first delegates the slide-structure definition to one agent, then launches parallel agents to generate algorithm-specific slides (e.g., Q-Learning and DQN). Because these subtasks run independently and the structure lacks a unified notation policy, their outputs use inconsistent symbols and formatting. A common mitigation strategy is to introduce an additional LLM agent to review and reconcile all outputs, but this approach substantially reduces efficiency.



(a) workflow: $T_1 \rightarrow T_2, T_1 \rightarrow T_3$
Ideal: *given* R_1, R_2 and R_3 are *independent*.
Real: *given* $\hat{R}_1$, $\hat{R}_2$ and $\hat{R}_3$ are *dependent*.



(b) workflow: $T_2 \rightarrow T_1 \rightarrow T_3$.
Ideal: *given* R_1, R_2 and R_3 are *independent*.
Real: *given* $\hat{R}_1$, $\hat{R}_2$ and $\hat{R}_3$ are *dependent*.



(c) workflow: $T_2 \leftarrow T_1 \leftarrow T_3$.
Ideal: *given* R_1, R_2 and R_3 are *independent*.
Real: *given* $\hat{R}_1$, $\hat{R}_2$ and $\hat{R}_3$ are *dependent*.

Figure 2: Examples of a high-level task decomposed to three subtasks $\{T_1, T_2, T_3\}$ as a graph. R_i is the latent *ground-truth answer* while $\hat{R}_i$ is response *actually observed*. Arrows indicate causal dependencies, where $\rightarrow$ indicate an explicit causal dependency in real execution; $\rightarrow$ indicate an ideal (theoretical) causal dependency; $\rightarrow$ indicates that the dependency can be either statistical or causal; $\rightarrow$ highlights the reason why conditional independence breaks. Intuitively, there exists at least one path where $\hat{R}_1$ and $\hat{R}_3$ remain dependent given $\hat{R}_2$.

Why does this misalignment happen? Decomposing a task into a directed acyclic graph (DAG) usually encode *dependencies* among subtasks. Consider the example of decomposition into three subtasks, where Figure 2 enumerates all possible DAGs. In theory, each subtask (node) is assumed to be *conditionally independent* of others given its parents. That is, with only ground-truth response R_i, removing its parent disconnects R_i from the rest. For example, in (a), deleting R_1 isolates R_2 from R_3, implying R_2 and R_3 are conditionally independent given R_1. However, in practice, only approximate responses $\hat{R}_i$ are available, each causally dependent on its ground-truth counterpart R_i. Therefore, $\hat{R}_2$ and $\hat{R}_3$ share hidden dependencies through R_1, *violating conditional independence*.

Motivated by this finding, we propose SEQCV, a dynamic framework that enables ***reliable execution*** under violated conditional independence assumptions. In SEQCV, subtasks are executed sequentially, each *conditioned* on *all* prior responses and *verified* via consistency checks immediately after agents generate a short token sequence. At each checkpoint, the generated token sequence is deemed reliable if it remains semantically consistent across multiple LLM models. When inconsistency is detected,

the unreliable sequence is discarded, and a recursive splitting mechanism is activated to decompose the subtask into smaller components. Despite its sequential execution, SEQCV avoids repeated inner-agent misalignment corrections on the same issue and achieves higher effective throughput than parallel pipelines. Across 8 creative agentic tasks, SEQCV improves accuracy by up to 30%.

Our contributions: (1) We inspire subsequent research in LLM-agent systems by investigating the root cause of inter-agent misalignment in graph-based orchestration. Our practical analysis identifies the violation of the conditional independence assumption as the core issue. We further highlight that this assumption cannot be satisfied in practice due to the nature of LLM-agent behavior. (2) We propose SEQCV as a practical solution to mitigate failures arising from this infeasible assumption. (3) Our SEQCV enables ensembling multiple LLM models by optimizing the selection of reliable outputs. We design a sequential execution and swift verification pipeline, coupled with a recursive splitting mechanism, to ensure tasks are decomposed into manageable subtasks aligned with LLM capabilities. Experiments on standard benchmarks and real-world tasks demonstrate consistent improvements, offering a promising direction for future research on accuracy and efficiency enhancement.

2 Methodology

We present SEQCV, a dynamic framework that employs a recursive splitting mechanism to ensure that tasks are always decomposed into manageable units aligned with the capabilities of LLMs. This allows multiple LLM-agents to execute subtasks and verify their outputs effectively within the system. Furthermore, SEQCV facilitates the ensembling of multiple LLM models by optimizing the selection of reliable responses. In the following, we articulate the problem formulation, conceptual overview. We then describe the execution (*i.e.,* how responses are generated, Section 2.1), the verification (cross-model validation, Section 2.2) and the recursive splitting mechanism (Section 2.2).

Problem Formulation Let $\mathcal{O}$ denote a high-level task objective and let $\mathcal{G} = (\mathcal{T}, \mathcal{E})$ be a directed acyclic graph (DAG) of subtasks $\mathcal{T} = \{T_1, \ldots, T_m\}$ as its nodes with edges $\mathcal{E}$ encoding parent–child dependencies among nodes. In prior graph-as-workflow systems, each T_i emits an approximate response $\hat{R}_i$ conditioned only on its parents $\{\hat{R}_j : j \in \mathrm{Pa}(i)\}$, assuming conditional independence from all other nodes. In practice, hidden correlations among LLM outputs violate this assumption, causing *inter-agent misalignment*. Our goal is to execute $\mathcal{G}$ reliably by verifying the reliability of each subtask's response before this response serves as input context for downstream subtasks.

Overview As shown in Figure 3, SEQCV decomposes a given high-level task into a sequence of subtasks and orchestrates a team of diverse LLM agents to execute these subtasks reliably by always conditioning on the complete history of verified outputs. Beginning with the first subtask, an agent from one LLM model generates a candidate response in the context of all previously accepted segments of responses, ensuring full awareness of prior decisions. This candidate response is then peer-reviewed by agents of other LLM models. Only when a majority consensus confirms its coherence and consistency, the candidate response is appended to the global history as an accepted response. If consensus is not achieved, the current subtask is automatically decomposed into simpler subtasks, each re-entering the same generate-verify-split loop. This process continues sequentially, recursing only when necessary, until all tasks yield reliable responses or a maximum recursion depth is reached. By combining full-trace conditioning with early, targeted splitting, SEQCV prevents hidden dependencies from triggering error cascades while maintaining end-to-end efficiency.

Advances of SEQCV As illustrated in Figure 2, latent dependencies can propagate along paths in the workflow graph that do not correspond to direct parent–child edges, since an LLM-generated response is only an approximation of ground truth. This implies that hidden dependencies generally exist even when the observed workflow graph suggests conditional independence. By constructing a task's context to encompass all previously generated responses, we capture these hidden dependencies and enforce high-level consistency in terminology, style, and factual content across subtasks. This approach replaces the summary and rerun-after-completion modules used in existing work [38, 39] to resolve conflicts and ensures that computational resources and token consumption are focused on generating extra information. In other words, our SEQCV allocates tokens efficiently and wisely.

To further improve reliability, we assess the reliability of each subtask. Note that this is challenging because we lack the ground truth of intermediate results. We propose that it can be approximated by

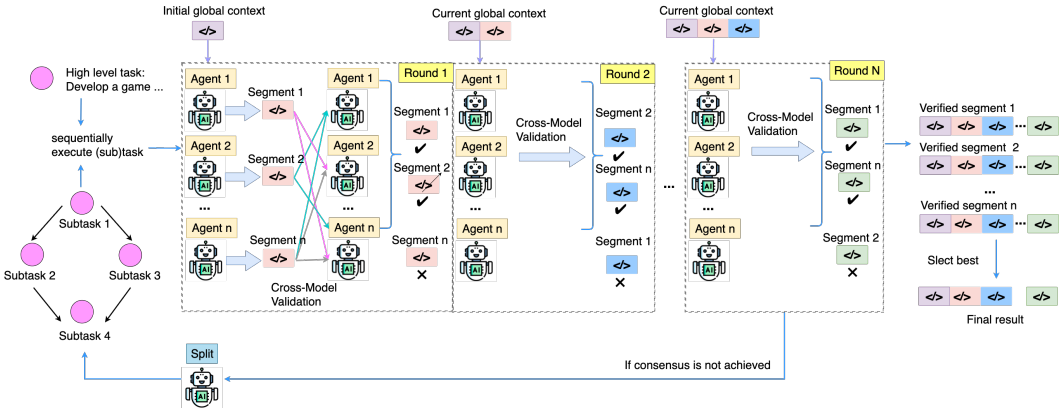


Figure 3: Overview of SEQCV. A high-level task is executed sequentially and split if necessary. During task execution, each agent generates a segment of candidate response conditioned on the verified context of all prior outputs. The generated segment is then peer-reviewed by other LLM-agents. If consensus approves, it is accepted as a reliable response; otherwise, the subtask splits into simpler ones (e.g., a task is split to four simpler subtasks), re-entering the generate-verify-split loop. This recursion continues until all tasks are resolved or a maximum depth is reached, ensuring reliability while maintaining efficiency. Algorithm 1 implements segment-level generation and validation is provided and Algorithm 2 implements dynamic splitting.

the extent to which its response reflects common sense across different models. We only split a task when this cross-model consensus indicates that splitting is necessary to resolve disagreement.

To further enhance runtime efficiency, SEQCV employs segment-level execution. In this way, validation occurs early via peer review, and if most results are incorrect, the process can be halted immediately, avoiding further wasted computation.

2.1 Response Generation

Segment-Based Response Generation For a subtask T_i, each agent $a \in \mathcal{A}$ receives the subtask context and global context prompt so far (see Subtask-Context Generation and Global-Context Construction below), and generate a full response segment as

$$S_{j|i,a}(l_0) \sim \text{Generate}(\text{LLM}, \pi_{j-1|i,a}, \Pi_i), \tag{1}$$

subject to a maximum length l_0. During code implementation, l_0 is a tunable hyperparameter that controls the granularity of subtask refinement. For ease of reading, we denote $S_{i,a}(l_0)$ simply as $S_{i,a}$ in the following sections. Each $S_{i,a}$ is verified via Cross-Model Validation (see Section 2.2). A passed $S_{j-1|i,a}$ will be accepted as a *full, verified response* $\hat{R}_{j-1|i,a}$ for subtask context generation.

Subtask-Context Generation The subtask-context generation means for each subtask T_i, we assemble a single prompt $\pi_{i,a}$ that includes the entire execution trace to date:

$$\pi_{j-1|i,a} = \left[T_i;\ \hat{R}_{1|i,a},\ \hat{R}_{2|i,a},\ \ldots,\ \hat{R}_{j-1|i,a}\right],$$

where each $\hat{R}_{\cdot|i}, a$ is a ***full, verified response*** (from $S_{\cdot|i,a}$) to address subtask T_i.

This subtask-context conditioning reduces inter-agent misalignment from any contradiction or mismatch between $\hat{R}_{\cdot|i}, a$ and its earlier outputs. As illustrated in Figure 2, latent correlations can propagate workflow graph paths beyond direct parent–child edges during practical execution, because the LLM-generated response is an approximation of ground-truth. By generating subtask context $\pi_{\cdot|i,a}$ that encompasses all previously verified segments of responses, we can: (a) capture such hidden dependencies; (b) enforce high-level consistency in terminology, style, and factual content across subtasks; and (c) mitigate the risk of a model "forgetting" context outside its own conditioning set.

Global-Context Construction With a collection of generated subtask-context $\{\pi_{\forall|i,a}\}$ ($_{\forall|i}$ means this subtask is completed) over all possible a, a majority voting will invoke to select the best π by

$$\pi^*_{\forall|i,a} = \text{Select}(\text{LLM}, \Pi_i), \tag{2}$$

where $\Pi_{i+1} = [\pi^*_{\forall|i,a}; \Pi_1, \Pi_2, \cdots, \Pi_{i-1}]$. That is, the global context up to $i+1$ (as T_i is completed) is constructed by concatenating the optimal subtask-context generations. While Select$(\cdot)$ invokes an LLM, it only needs to output a decision of which candidate to select, rather than generate a lengthy response. This design significantly improves efficiency and reduces computational cost.

2.2 Verification and Splitting

Because conditional independence among subtask responses rarely holds in practice, SEQCV executes subtasks sequentially, verifying each response segment before incorporating it into the global context for downstream tasks. While this guarantees reliability, naive sequential execution can lead to high end-to-end latency. To improve efficiency without compromising correctness, we introduce two complementary mechanisms: segment-level cross-model validation and dynamic recursive splitting of misaligned subtasks. These mechanisms jointly prune incorrect response segments at an early stage, enabling the system to focus computational resources on reliable content as much as possible.

Cross-Model Validation With regard to generating the j-th segment of subtask T_i, every agent a_l evaluates each peer candidate response $\{S_{j|i,1}, \ldots, S_{j|i,L}\}$ by returning a binary vote

$$v_{j|i,l}\big(S_{j|i,\ell}\big) = \begin{cases} 1, & \text{if } S_{j|i,l} \text{ is coherent and consistent with } \pi_i, \\ 0, & \text{otherwise,} \end{cases} \quad l = 1, \ldots, L. \tag{3}$$

We then consider each candidate response $S_{j|i,\ell}$ reliable for its generator a_ℓ if it receives at least τn votes, where $\tau = \lceil \frac{n}{2} \rceil / n$. In that case, agent a_ℓ commits $\hat{R}_{j|i,\ell} = S_{j|i,\ell}$. After verification via the above Cross-Model Validation, the Subtask-Context Generation is invoked, and each agent continues segment-based response generation based on its own verified subtask context of its own, rather than relying on a single "best" response.

If $S_{j|i,\ell}$ fails to achieve τn votes, then agent a_ℓ treats $S_{j|i,\ell}$ as unreliable generation and discard it. This segment-based per-agent validation preserves the diversity of all plausible solutions while ensuring that *no* agent propagates an unreliable response.

Dynamic Recursive Splitting If none of the candidate responses for subtask T_i pass the cross-model validation, we discard all candidate segments and invoke a learned decomposition model M_{split} on the global context Π_i. This yields a new directed subgraph

$$\mathcal{G}' = \big(\{T_1, \cdots, T_{i-1}, T'_i, \mathbf{T'_{i,1}}, \ldots, \mathbf{T'_{i,r_i}}, T_{i+1}, \cdots, T_m\},\, \mathcal{E}'_i, \mathcal{E}_i\big), \tag{4}$$

where each T' is a simpler subtask and $\mathcal{E}'_i$ encodes newly added internal dependencies. Namely, $\mathcal{G}$ is updated to a new DAG $\mathcal{G}'$ by inserting a small subgraph. To preserve the original ordering of nodes other than T_i, every parent of T_i (*i.e.,* $p \in \text{Pa}(T_i)$) is directed to each new subtask $\mathbf{T}'_{(i,\cdot)}$, and each $\mathbf{T}'_{(i,\cdot)}$ is then directed to every child of T_i (*i.e.,* $c \in \text{Ch}(T_i)$).

We recompute a topological ordering of $\mathcal{G}'$ and resume sequential execution at the first newly inserted subtask, processing each $\mathbf{T}'_{(i,\cdot)}$ at depth $d+1$ under the same segment-based response generation, cross-model validation, and splitting. By allocating efforts to correct the misaligned region and preserving the remainder of the workflow intact, this mechanism bounds the cost of recovery and prevents global pipeline re-execution. Recursion terminates when all subtasks verify successfully or when the maximum depth $d_{\max}$ is reached; in the latter case, we fill the original T_i by selecting the candidate response that achieved the highest total vote count.

3 Experiments

We conduct thorough experiments to evaluate SEQCV by performing extensive benchmark assessments on six standard datasets, comparisons with state-of-the-art (SOTA) reasoning models, efficiency comparison analyses, and investigations using **cross-validation modules and workflow decomposition modules**. Our key findings reveal consistent performance enhancements across various tasks, with particularly notable improvements in multi-hop reasoning. By comparing with leading reasoning models, we demonstrate SEQCV's efficacy as a versatile framework. Additionally, our efficiency comparison experiments further substantiate SEQCV's flexibility and efficiency. Finally, evaluations of critical components such as the DAG structure and decomposition mechanism through cross-validation modules and workflow decomposition modules verify the necessity of our design.

Algorithm 1 Segment-level Generation and Validation

Require: Task T_i, global context Π_i, agent pool $\mathcal{A}$, vote threshold τ, max round $J_{\max}$, max token per round l_0

function SEGGEN($T_i,\ \Pi_i,\ \mathcal{A},\ \tau,\ l_0,\ J_{\max}$)

  for $j = 1$ **to** $J_{\max}$ **do** ▷ 1. Each agent generates a full response segment

    for all $a \in \mathcal{A}$ **do**

      $S_{j|i,a} \sim \text{Generate}\big(\text{LLM},\ \pi_{j-1|i,a},\ l_0\big)$

    end for

    $\#success \leftarrow 0$ ▷ 2. Cross-model validation and per-agent context update

    for all $a \in \mathcal{A}$ **do**

      votes $\leftarrow \sum_{b\in\mathcal{A}} \text{Verify}\big(b, S_{j|i,a}, \pi_{j-1|i,b}\big)$ ▷ majority voting

      if votes $> |\mathcal{A}|/2$ **then**

        $\pi_{j|i,a} \leftarrow [\pi_{j-1|i,a}; S_{j|i,a}]$

        $\#success \leftarrow \#success + 1$

      end if

    end for

    if #success<$\tau|\mathcal{A}|$ **then**

      return (**failure**, $\emptyset$)

    end if

    $\pi_i^* \leftarrow$ MAXVOTE($\{\pi_{\forall|i,a}\}$)

    return (**success**, π_i^*)

  end for

end function

Algorithm 2 Running and Dynamic Splitting DAG

Require: DAG $\mathcal{G} = (\mathcal{T}, \mathcal{E})$, global context Π, depth d, agent pool $\mathcal{A}$, primary $\mathcal{A}$, vote threshold τ, max depth $d_{\max}$, split model M_{split}

function RUNGRAPH($\mathcal{G}, \Pi, d$)

  if $d > d_{\max}$ **then**

    $\pi_i^* \leftarrow$ MAXVOTE($\{\pi_{\forall|i,a}\}$) ▷ depth limit: force to pick best via Equation 2

    $\Pi_{i+1} \leftarrow [\pi^*; \Pi_i]$

    $\mathcal{G}[T_i] \leftarrow \Pi_{i+1}$

    return $\mathcal{G}$

  end if

  $ord \leftarrow$ TOPOORDER($\mathcal{G}$)

  for T_i **in** ord **do**

    $(status, \pi_i^*) \leftarrow$ SEGGEN($T_i, \Pi_i, \mathcal{A}, \tau, l_0$)

    if $status =$ success **then**

      $\Pi_{i+1} \leftarrow [\pi^*; \Pi_i]$ ▷ append verified segment

    else

      $\{T'_{i,1}, \ldots, T'_{i,r}\} \leftarrow$ SPLITGRAPH($M_{\text{split}}, T_i, \Pi$) ▷ split T_i into simpler subtasks

      $\mathcal{G} \leftarrow$ INSERTSUBTASKS($\mathcal{G}, T_i, \{T'_{i,j}\}$)

      return RUNGRAPH($\mathcal{G}, \Pi, d+1$) ▷ recurse on updated graph

    end if

  end for

  return $\mathcal{G}$

end function

3.1 Experimental Setup

Agentic Tasks We evaluate 8 tasks designed to test creativity, multi-step reasoning, and adherence to diverse constraints. The detailed task prompts can be found in Appendix E

- **NeurIPS Website**: The task requires creating a website with four core elements: a navigation menu, a paper submission section, a schedule display, and speaker profiles. The primary constraint is that the solution must use only HTML and CSS.
- **Lecture Slides Generation**: Create a 30-page LaTeX presentation on maximum likelihood estimation with motivation, problem statement, intuitive solution, detailed math equations. It must be implemented in LaTeX format.
- **Pac-Tank Arcade Game**: This game combines action and logic, requiring implementation of player movement, enemy AI, collision detection, a scoring system, and a game-over mechanism. It must be implemented in Python with Pygame, without sound or external images.
- **Electrical-Circuit Puzzle Game**: This puzzle involves a drag-and-drop interface, circuit validation, level progression, visual feedback, and a win condition. The task must be implemented using vanilla JavaScript with no libraries.
- **Snake-Chess Fusion**: This hybrid game merges snake mechanics with chess logic, requiring snake movement, chess piece mechanics, a growth mechanism, strategic gameplay, and a lose condition. The allowed setup is Python with Pygame and no sound.
- **RPG Task Manager**: This productivity tool introduces gamified task management through task CRUD operations, quest progression, character statistics, and save/load functionality. It must be implemented using HTML, CSS, and JavaScript, with no backend.
- **Tetris+Bejeweled Mash-up**: The task combines falling-block mechanics with match-3 detection, line clearing, score tracking, and level difficulty progression. It uses Python with Pygame, again without sound.
- **Family Travel Planning**: The goal is to produce a LaTeX-based travel plan including flight details, accommodation bookings, daily agenda, and cost estimates.

Baselines We compare the generated results against three recent baselines: Flow [38], AFlow [39] and Atom [41], as well as o4-mini-high [42] via OpenAI interface. We implement SEQCV by using a mixture of o4-mini and o3-mini to complete each task. **For each method, we run three times and select the best result.** For code tasks, we manually check if they implement the necessary functionality required. For all tasks, we use GPT to evaluate which one is better and identify the core weaknesses based on the prompt.

Table 1: Average performance across seven agentic tasks. HR: Hard Requirements, ES: Execution Success, CA: Constraint Adherence. Our method (**SEQCV**) substantially outperforms all baselines across all three metrics, demonstrating superior capability in handling complex creative and interactive tasks.

Method	HR	ES	CA	AVG
AFlow	36%	30%	40%	36%
Atom	26%	40%	30%	29%
Flow	56%	70%	40%	58%
o4-mini-high	81%	100%	70%	82%
SeqCV	**83%**	**100%**	**100%**	**88%**

Performance metrics Each method is evaluated on seven challenging agentic tasks using three key metrics. Hard Requirements (HR) measures the percentage of task-specific requirements fully satisfied. Execution Success (ES) represents the percentage of runs completing without major bugs. Constraint Adherence (CA) quantifies the percentage of runs respecting all specified constraints. We also compare the running time for different AI agent systems.

3.2 Quantified Results

Each method is executed three times, and the best run is selected for reporting. For code-generation tasks, we manually verify whether the produced code correctly implements the required functionalities. For all tasks, GPT-4.1 based evaluation assists in assessing correctness and identifying weaknesses based on the prompt. All method names are masked during evaluation to ensure a fair comparison.

Table 1 summarizes the average performance across all seven tasks. The results highlight consistent improvements by our proposed method.

SeqCV achieves the highest total score of 88%, outperforming o4-mini-high (82%) and Flow (58%). It attains perfect scores on both Execution Success and Constraint Adherence, indicating superior reliability and precision. Compared to baseline models, SeqCV shows gains of +6% over o4-mini-high, +30% over Flow, +52% over AFlow, and +59% over Atom. The results further reveal that SeqCV's model ensemble combining o4-mini and o3-mini outperforms the stronger individual model.

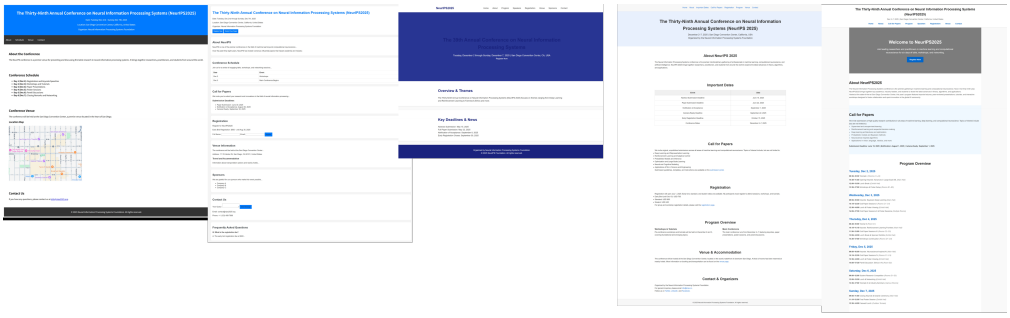
3.3 Per-Task Analysis

We provide detailed analysis for each agentic task, examining the major weaknesses of different methods. For each task, we present visual results and analyze the specific failure modes of baseline methods. We present detailed discussions of three tasks in the main text: *NeurIPS Website Development*, *Lecture Slides Generation*, and *Family Travel Planning*. Analyses for the remaining five tasks (*Pac-Tank Arcade*, *Electrical-Circuit Puzzle*, *Snake-Chess Fusion*, *RPG Task Manager*, and *Tetris+Bejeweled Mash-up*) are provided in Appendix B.

3.3.1 NeurIPS Website Development

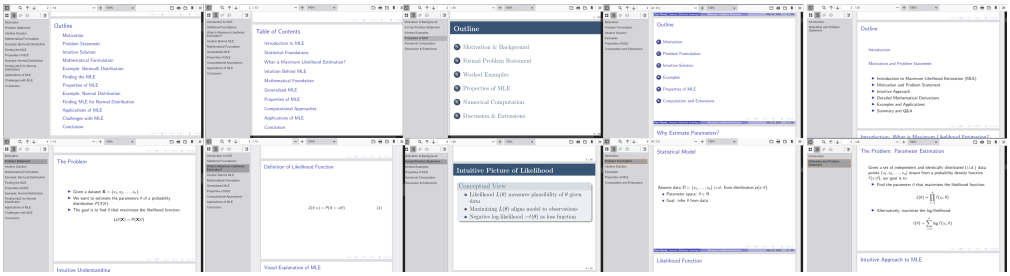
The task requires creating a conference website with navigation menu, paper submission section, schedule display, and speaker profiles, constrained to HTML+CSS format only. Figure 8 shows representative outputs from different methods.

Weaknesses Analysis **AFlow** generates an incomplete website missing the navigation menu and schedule sections. The submission form exists but lacks validation, and the HTML structure is poorly nested. **Atom** produces static placeholder content without interactivity, and the minimal CSS results in an unprofessional appearance. **Flow** outputs a well-structured HTML document but with limited styling and an unformatted schedule section. **o4-mini-high** achieves good layout and styling, yet omits detailed speaker biographies and required form fields. In contrast, **SEQCV** fulfills all requirements with comprehensive HTML structure, polished styling, and functional validation, though mobile responsiveness can be improved.



(a) AFlow (b) Atom (c) Flow (d) o4-mini-high (e) SEQCV

Figure 4: NeurIPS Website Development results. From left to right: outputs from different methods showing varying levels of completeness and styling quality.



(a) AFlow (b) Atom (c) Flow (d) o4-mini-high (e) SEQCV

Figure 5: Lecture Slides Generation results. Sample pages showing varying levels of completeness, mathematical rigor, and pedagogical structure across different methods.

3.3.2 Lecture Slides Generation

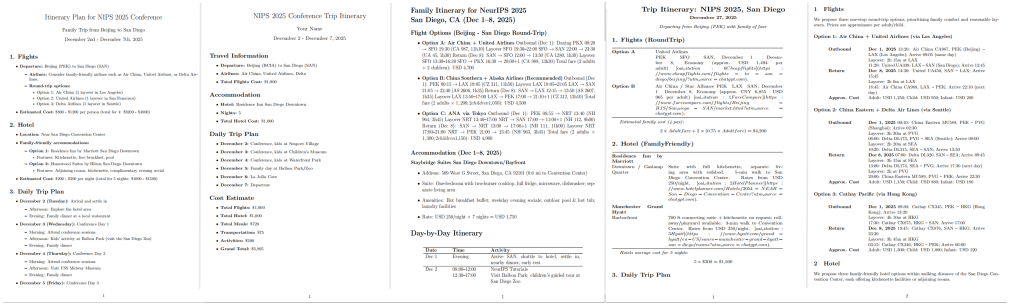
The task requires creating a comprehensive 30-page LaTeX presentation on maximum likelihood estimation suitable for a 2-hour research-level lecture. The slides must include motivation, problem statement, intuitive solutions, detailed mathematical derivations, and be self-contained. Figure 7 shows sample slides from different methods.

Weaknesses Analysis **AFlow** produces only 13 slides focusing on two distributions, lacking variety, intuition, and real-world examples. **Atom** creates 14 slides with shallow coverage, incomplete derivations, and placeholder text, omitting key theoretical properties and motivation. **Flow** delivers a full 30-slide deck with strong theory but no visual aids or case studies, limiting engagement. **o4-mini-high** provides 31 slides with solid fundamentals but lacks depth, numbering, and advanced topics such as EM or logistic regression. **SEQCV** generates a well-structured 30+ slide presentation with complete derivations, examples, and practical insights, though it would benefit from more visuals and intermediate mathematical steps for clarity.

3.3.3 Family Travel Planning

This task generates a NeurIPS 2025 travel itinerary with flight details, accommodation bookings, daily agenda, and cost estimates in LaTeX format. Figure 6 shows the generated itineraries.

Weaknesses Analysis **AFlow** omits flight times and airline details, provides vague accommodations, and contains calculation errors. **Atom** outputs placeholder bookings and unrealistic pricing with minimal agenda content. **Flow** includes detailed activities and realistic bookings but has LaTeX compilation issues from missing packages and faulty table syntax. **o4-mini-high** presents coherent content but lacks detailed schedules and a complete cost breakdown including taxes and contingencies. **SEQCV** produces a complete, well-formatted LaTeX document with accurate travel details, realistic budgeting, and professional presentation.



(a) AFlow (b) Atom (c) Flow (d) o4-mini-high (e) SEQCV

Figure 6: Family Travel Planning showing LaTeX-formatted itineraries with flight details, accommodations, daily schedules, and budget breakdowns.

Table 3: Average running time (seconds) of SEQCV compared to baselines over seven tasks.

Method	slides	nips-web	pac-tank	snake-chess	task-RPG	tetris-bejew	travel	avg
o4-mini-high	61s	47s	51s	65s	40s	59s	58s	54.43s
AFlow	202s	242s	170s	140s	115s	139s	118s	160.86s
Flow	1083s	354s	330s	1539s	571s	560s	218s	665.00s
Atom	464s	476s	533s	556s	668s	533s	428s	522.57s
SEQCV (Ours)	255s	173s	160s	285s	225s	217s	205s	217.14s

3.4 Misalignment Analysis

To directly measure the effect of misalignment reduction, we conducted a controlled experiment isolating the impact of our sequential conditioning mechanism. We used the same workflow produced by a parallel baseline method and modified only the execution mode: the parallel execution was replaced with sequential conditioning, while all other components (such as cross-verification and recursive splitting) were disabled. Thus, the only difference between the two settings was whether subtasks were executed concurrently or sequentially conditioned on prior outputs. Any performance difference can therefore be attributed to reduced inter-agent misalignment.

Table 2: Controlled comparison isolating the effect of inter-agent misalignment. Sequential conditioning improves accuracy by reducing dependency conflicts across agents.

Configuration	Score (%)
Parallel (baseline)	52.3
Sequential only	58.4

This controlled setup demonstrates that sequential conditioning mitigates inter-agent misalignment arising from dependency inconsistencies, resulting in measurable accuracy gains.

3.5 Running Time Efficiency

As shown in Table 3, SEQCVachieves an average runtime of 217 s across eight representative tasks, corresponding to a 2.5× speed-up over Flow (665 s) and a 2.1× speed-up over Atom (523 s). On the six most computationally intensive tasks (slides, nips-web, pac-tank, snake-chess, task-RPG and tetris-bejew), SEQCVreduces execution time by 51%–82% compared to Flow, shaving more than half off runtime. Although parallel pipelines can process multiple sub-tasks concurrently, they incur frequent misalignment corrections that erode effective throughput. By adopting a carefully optimized sequential strategy, SEQCVavoids these costly corrections and delivers higher end-to-end throughput on complex tasks.

4 Related Work

LLM-agent Systems The emergence of large language models (LLMs) has revolutionised agent capabilities, reshaping their role in artificial intelligence and expanding the scope of their applications [21]. With the rising adoption of LLM as an agent in the system, LLM-agent systems show great

potential in automation task execution [2, 3, 4] across a variety of domains, such as such as software development [5, 6, 7], scientific discovery [8, 9, 10], and human behaviour simulation [11, 12, 13]. The capability of LLM-agent can be further augmented with external tools and APIs [14, 15, 16], enabling a wide use in a diversity of scenarios and benchmarks [17, 18, 19, 20, 43, 44].

To enhance the reasoning capabilities of LLMs, complex objectives are typically decomposed into intermediate reasoning steps, including chain of thought (CoT) [27], ReAct [28], tree of thought (ToT) [29], and Reflexion [30]. Single-agent implementations of this approach include AutoGPT [31], BabyAGI [32], LangChain [33], and Llama-index [34]. Decompositing a high-level task into subtasks as a structured graph can further improve coordination and robustness of the task execution, both in single-agent settings (e.g., Graph of Thought (GoT) [35], LangGraph [36]) and in multi-agent settings (e.g., GPTSwarm [37], Flow [38], AFlow [39], AgentPrune [40], AoT [41]).

LLM-agent Systems Orchestration With regards to the orchestration of agents, a branch of approach applies a conversational collaboration and human-in-the-loop interaction. Examples include AutoGen [2], Reflexion [30], DSPy [45], CAMEL [4], and Voyager [46], which enable multi-round dialogues, reflective reasoning, and declarative composition for improved coordination [47]. However, this approach can be less effective in multi-agent systems such as AG2 [2] and ChatDev, where dialogue breakdowns often limit performance. Alternatively, role-based orchestration assigns predefined agent roles and structured workflows, as demonstrated by MetaGPT [3], which uses standard operating procedures (SOPs) to coordinate LLM agents on software development tasks. Instead of a predesigned pipeline, modular and dynamically executable workflows offer improved performance and flexibility. Systems such as GPT-Swarm [37], Flow [38], AutoFlow [48], and Cut-the-Crap [40] adopt DAG-based or communication-optimized designs to enable fault-tolerant and reconfigurable execution. Further enhancements in adaptability are achieved through graph-based agent communication protocols [49] and reinforcement learning-based role assignment strategies [50].

Task Verification Alongside Orchestration Alongside the orchestration, task verification is essential for ensuring the result quality and system robustness in LLM-agent workflows [1]. Existing techniques include self-consistency voting [51], reflective refinement [30], LLM-as-a-judge evaluation [52], verifier agents [53], and token-level consistency checks [1]. Hierarchical validation frameworks further incorporate syntactic, semantic, and goal-specific checks [38], while arbitration schemes resolve conflicts via voting or consensus [47]. Additional innovations include progressive refinement loops [54], structured debate or voting-based decision making [40], dual-layer validation pipelines [55], dynamic fallback completions [56], and role reassignment for failing agents [57]. Systems typically terminate when retry limits are exceeded [54], semantic drift is detected [1], or insufficient agent participation occurs [38]. Recovery mechanisms range from returning partial results to invoking fallback models or recursive decomposition [3, 37]. However, limited work has systematically compared the efficiency and token cost of these task verification methods.

5 Conclusion

Inter-agent misalignment is a critical factor contributing to the failure of multi-LLM-agent systems. In this study, we investigated its underlying cause. Through practical examination of state-of-the-art LLM-agent systems that employ graph-based orchestration, we identified the violation of the conditional independence assumption as the root issue. This assumption, which requires agents to generate ground-truth responses at each step, can NOT be satisfied in practice due to the inherent limitations of current LLMs. To address this, we proposed SEQCV, a framework that enhances output reliability via sequential execution and swift verification. SEQCV further introduces a recursive splitting mechanism to decompose tasks into smaller, manageable subtasks aligned with the LLMs' capabilities. Unlike existing methods, SEQCV directly concatenates verified responses without requiring full-context re-evaluation, significantly reducing token costs and improving system efficiency. We evaluated SEQCV on both standard benchmarks and real-world user tasks, demonstrating consistent improvements over existing state-of-the-art approaches in both accuracy and efficiency.

Acknowledgments

TLL is partially supported by the following Australian Research Council projects: FT220100318, DP220102121, LP220100527, LP220200949.

References

[1] Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent llm systems fail? In *Proceedings of the Building Trust Workshop at the 13th International Conference on Learning Representations (ICLR), Singapore, April 24-28, 2025*. University of California, Berkeley and Intenso Sao Paolo, 2025. Equal contribution by first three authors.

[2] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. In *First Conference on Language Modeling*, 2024.

[3] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[4] Guohao Li, Hasan Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[5] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[6] Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An Open Platform for AI Software Developers as Generalist Agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

[7] Genghan Zhang, Weixin Liang, Olivia Hsu, and Kunle Olukotun. Adaptive self-improvement LLM agentic system for ML library development. In *Workshop on Reasoning and Planning for Large Language Models*, 2025.

[8] Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The virtual lab: Ai agents design new sars-cov-2 nanobodies with experimental validation. *bioRxiv*, 2024.

[9] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[10] Michael Skarlinski, Tyler Nadolski, James Braza, Remo Storni, Mayk Caldas, Ludovico Mitchener, Michaela Hinks, Andrew White, and Sam Rodriques. Futurehouse platform: Superintelligent ai agents for scientific discovery. `https://www.futurehouse.org/research-announcements/launching-futurehouse-platform-ai-agents`, May 2025. Research Announcement, FutureHouse.

[11] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery.

[12] Ziyi Yang, Zaibin Zhang, Zirui Zheng, Yuxian Jiang, Ziyue Gan, Zhiyu Wang, Zijian Ling, Jinsong Chen, Martz Ma, Bowen Dong, Prateek Gupta, Shuyue Hu, Zhenfei Yin, Guohao Li, Xu Jia, Lijun Wang, Bernard Ghanem, Huchuan Lu, Chaochao Lu, Wanli Ouyang, Yu Qiao, Philip Torr, and Jing Shao. Oasis: Open agent social interaction simulations with one million agents, 2024.

[13] Chong Zhang, Xinyi Liu, Zhongmou Zhang, Mingyu Jin, Lingyao Li, Zhenting Wang, Wenyue Hua, Dong Shu, Suiyuan Zhu, Xiaobo Jin, et al. When ai meets finance (stockagent): Large language model-based stock trading in simulated real-world environments. *arXiv preprint arXiv:2407.18957*, 2024.

[14] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[15] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: solving ai tasks with chatgpt and its friends in hugging face. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2023. Curran Associates Inc.

[16] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 6491–6501, New York, NY, USA, 2024. Association for Computing Machinery.

[17] Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 126544–126565. Curran Associates, Inc., 2024.

[18] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[19] Tianqi Xu, Linyao Chen, Dai-Jie Wu, Yanjun Chen, Zecheng Zhang, Xiang Yao, Zhiqiang Xie, Yongchao Chen, Shilong Liu, Bochen Qian, Philip Torr, Bernard Ghanem, and Guohao Li. Crab: Cross-environment agent benchmark for multimodal language model agents, 2024.

[20] Yiheng Xu, Hongjin Su, Xing Chen, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, and Tianbao Xie. Lemur: Harmonizing natural language and code for language agents. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*, 2024.

[21] Bang Liu, Xinfeng Li, Jiayi Zhang, Jinlin Wang, Tanjin He, Sirui Hong, Hongzhang Liu, Shaokun Zhang, Kaitao Song, Kunlun Zhu, et al. Advances and challenges in foundation agents: From brain-inspired intelligence to evolutionary, collaborative, and safe systems. *arXiv preprint arXiv:2504.01990*, 2025.

[22] Julia Wiesinger, Patrick Marlow, and Vladimir Vuskovic. Agents. `https://www.kaggle.com/whitepaper-agents`, May 2025. Kaggle Whitepaper.

[23] Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, and Qingyun Wu. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. PMLR, 2025. Spotlight.

[24] Shaokun Zhang, Jieyu Zhang, Jiale Liu, Linxin Song, Chi Wang, Ranjay Krishna, and Qingyun Wu. Offline training of language model agents with functions as learnable weights. In *Forty-first International Conference on Machine Learning*, 2024.

[25] Shaokun Zhang, Jieyu Zhang, Dujian Ding, Jiale Liu, Mirian Del Carmen Hipolito Garcia, Ankur Mallick, Daniel Madrigal, Menglin Xia, Victor Rühle, Qingyun Wu, and Chi Wang. Ecoact: Economic agent determines when to register what action. In *Workshop on Reasoning and Planning for Large Language Models*, 2025.

[26] Jintian Zhang, Xin Xu, Ningyu Zhang, Ruibo Liu, Bryan Hooi, and Shumin Deng. Exploring collaboration mechanisms for LLM agents: A social psychology view. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14544–14607, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[28] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[29] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[30] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[31] Toran Bruce Richards. Auto-gpt: An autonomous gpt-4 experiment. `https://github.com/Significant-Gravitas/Auto-GPT`, 2023. GitHub repository.

[32] Yohei Nakajima. Babyagi: An experimental prototype framework for building self-building autonomous agents. `https://github.com/yoheinakajima/babyagi`, 2023. GitHub repository.

[33] Harrison Chase. Langchain: Build context-aware reasoning applications. `https://github.com/langchain-ai/langchain`, 2022. GitHub repository.

[34] Jerry Liu. Llamaindex: A data framework for llm applications. `https://github.com/jerryjliu/llama_index`, 2022. GitHub repository.

[35] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: solving elaborate problems with large language models. 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*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024.

[36] LangChain Inc. Langgraph: A framework for building stateful, multi-actor applications with llms. `https://langchain-ai.github.io/langgraph/`, 2025. GitHub repository.

[37] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. Gptswarm: Language agents as optimizable graphs. In *International Conference on Machine Learning (ICML)*, 2024.

[38] Boye Niu, Yiliao Song, Kai Lian, Yifan Shen, Yu Yao, Kun Zhang, and Tongliang Liu. Flow: Modularized agentic workflow automation. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

[39] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Adaptive workflow construction for llm-based agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

[40] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*, 2025.

[41] Fengwei Teng, Zhaoyang Yu, Quan Shi, Jiayi Zhang, Chenglin Wu, and Yuyu Luo. Atom of thoughts for markov llm test-time scaling, 2025.

[42] OpenAI. o4-mini-high. Large language model, 2025. Accessed: 2025-05-16, via OpenAI API.

[43] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

[44] Zhenran Xu, Senbao Shi, Baotian Hu, Jindi Yu, Dongfang Li, Min Zhang, and Yuxiang Wu. Towards reasoning in large language models via multi-agent peer review collaboration. *https://arxiv.org/abs/2311.08152*, 2023.

[45] Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*, 2024.

[46] 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. *Transactions on Machine Learning Research*, 2024.

[47] Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 257–279, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[48] Zelong Li, Shuyuan Xu, Kai Mei, Wenyue Hua, Balaji Rama, Om Raheja, Hao Wang, He Zhu, and Yongfeng Zhang. Autoflow: Automated workflow generation for large language model agents, 2024.

[49] Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *ICLR 2024 Conference*, 2024.

[50] Zelai Xu, Chao Yu, Fei Fang, Yu Wang, and Yi Wu. Language agents with reinforcement learning for strategic play in the werewolf game. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[51] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[52] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[53] Jiuzhou Han, Wray Buntine, and Ehsan Shareghi. Verifiagent: a unified verification agent in language model reasoning. *https://arxiv.org/abs/2504.00406*, 2025.

[54] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[55] Ningyuan Xi, Xiaoyu Wang, Yetao Wu, Teng Chen, Qingqing Gu, Yue Zhao, Jinxian Qu, Zhonglin Jiang, Yong Chen, and Luo Ji. Dual-layer training and decoding of large language model with simultaneously thinking and speaking, 2024.

[56] Maor Ivgi, Ori Yoran, Jonathan Berant, and Mor Geva. From loops to oops: Fallback behaviors of language models under uncertainty. *https://arxiv.org/abs/2407.06071*, 2024.

[57] Jen tse Huang, Jiaxu Zhou, Tailin Jin, Xuhui Zhou, Zixi Chen, Wenxuan Wang, Youliang Yuan, Michael R. Lyu, and Maarten Sap. On the resilience of llm-based multi-agent collaboration with faulty agents. *https://arxiv.org/abs/2408.00989*, 2024.

[58] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[59] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *CoRR*, abs/2110.14168, 2021.

[60] Qihao Zhao, Yangyu Huang, Tengchao Lv, Lei Cui, Qinzheng Sun, Shaoguang Mao, Xingxing Zhang, Ying Xin, Qiufeng Yin, Scarlett Li, and Furu Wei. MMLU-CF: A contamination-free multi-task language understanding benchmark. *CoRR*, abs/2412.15194, 2024.

[61] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13003–13051. Association for Computational Linguistics, 2023.

[62] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics, 2018.

[63] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics, 2024.

[64] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Trans. Assoc. Comput. Linguistics*, 10:539–554, 2022.

[65] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6609–6625. International Committee on Computational Linguistics, 2020.

[66] Michihiro Yasunaga, Xinyun Chen, Yujia Li, Panupong Pasupat, Jure Leskovec, Percy Liang, Ed H. Chi, and Denny Zhou. Large language models as analogical reasoners. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[67] Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolò Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, Renqian Luo, Scott Mayer McKinney, Robert Osazuwa Ness, Hoifung Poon, Tao Qin, Naoto Usuyama, Christopher M. White, and Eric Horvitz. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *CoRR*, abs/2311.16452, 2023.

[68] Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. *CoRR*, abs/2408.08435, 2024.

[69] Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. *CoRR*, abs/2412.09078, 2024.

[70] OpenAI. gpt-4o-mini. Large language model, 2025. Accessed: 2025-05-16, via OpenAI API.

[71] OpenAI. gpt-4.1-nano. Large language model, 2025. Accessed: 2025-05-16, via OpenAI API.

APPENDIX

Contents

A Limitation

By design, SEQCV linearizes the task graph $\mathcal{G} = (\mathcal{T}, \mathcal{E})$ into a directed acyclic sequence of subtasks. While this design suits many content generation and document authoring scenarios, it cannot naturally express cyclic or recurrent dependencies among agents. In domains such as continuous control, multi step planning with feedback loops, or systems where subtasks iteratively refine one another, a purely acyclic representation becomes restrictive. Although one could in principle introduce an explicit time index t to convert loops into a directed acyclic graph, this workaround often results in cumbersome encodings. More flexible architectures, such as OpenAI's SwarmAgent or general message passing frameworks, may be better suited to tasks that require intrinsic cyclic structures.

SEQCV conditions each subtask T_i on the full history of verified responses. In practice, large scale tasks can quickly exhaust an LLM's context window. While a hierarchical summarization step can compress earlier segments into a shorter representation, summarization inevitably introduces information loss and cannot guarantee preservation of all context necessary for downstream subtasks. As model context size limits increase, this constraint will lessen. Until then, careful trade offs between context fidelity and sequence length are required, and some applications may remain infeasible.

Our method fails to provide benefits for reducing inter agent misalignment only in a specific type of wide graph. This occurs when each independent branch (1) has no incoming edges from outside the branch (that is, no parent nodes pointing into it), and (2) has no connections to nodes in other branches. In such a strictly isolated structure, there is little opportunity for our method to reduce inter agent misalignment, as no misalignment exists.

B More Results on Agentic Tasks

Real users expect autonomous assistants to handle diverse, multi-step requests ranging from structured document synthesis to interactive code generation and personalized logistics planning with demonstrating strong domain expertise and consistently high-quality output. To evaluate whether SEQCV meets these expectations, we design eight end-to-end agentic tasks spanning:

- *Long-text generation & structure* (e.g., self-contained lectures, multi-page itineraries),
- *Multi-modal code synthesis* (e.g., LaTeX, HTML/CSS/JS, Python/Pygame),
- *Dynamic reasoning & planning* (e.g., game AI, resource management, travel logistics), and
- *Cross-domain adaptability* (e.g., from mathematical instruction to family travel planning).

These tasks are carefully crafted to probe the core competencies of a general-purpose agent system. The eight tasks cover a wide range of domains with different output modalities:

- **Lecture-slide generation:** 30-page LaTeX slide for a two-hour lecture about maximum-likelihood estimation for research students.
- **Conference website design:** HTML/CSS site for NeurIPS 2025.
- **Pac-Tank arcade game:** Python/Pygame implementation merging gaming rules from both Pac-Man and Tank City.
- **Electrical-circuit puzzle game:** Browser-based drag-and-drop interface using vanilla JavaScript.
- **Snake-Chess fusion:** Real-time Python/Pygame game fusing the gaming rules of Snake with simplified chess tactics.
- **RPG task manager:** A desktop-style "RPG" mapping to-do items to quests (HTML/CSS/JS).
- **Tetris+Bejeweled mash-up:** Python/Pygame game fusing the gaming rule of Match-3 and line-clearing game.
- **Family travel planning:** LaTeX-formatted itinerary for NeurIPS 2025, including flights, lodging, daily agenda, and cost estimates.

Baselines We implement SEQCV by using a mixture of o4-mini and o3-mini to complete each task. We compare the generated results against three recent baselines: Flow [38], AFlow [39] and Atom [41], as well as comparing to a plain Q&A results from o4-mini-high [42] via OpenAI interface. In each of the following subsections, we first present the prompt for the task description, followed by the performance results of SEQCV and the baselines.

Evaluation **For each method, we run three times and select the best result.** For code tasks, we manually check if they implement the necessary functionality required. For all tasks, we use GPT to evaluate which one is better and identify the core weaknesses based on the prompt. ***All method names are masked to ensure a fair comparison.*** After that, we will further manually check whether the weaknesses listed are correct.

Overall, our method significantly outperforms existing baselines. Notably, the framework enhances the model's reasoning ability with a mixture of o4-mini and o3-mini, it achieves even better performance than o4-mini-high.

B.1 Lecture Slides - Maximum Likelihood Estimation

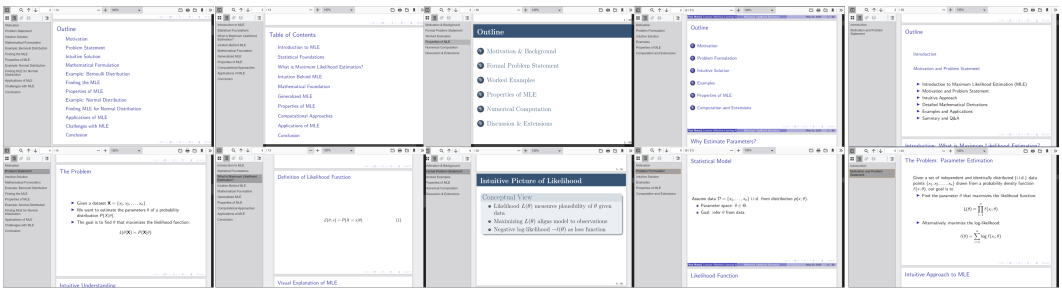
Task 1: Lecture Slides

Lecture slide:
I am a lecturer. I am teaching the machine learning course for research students. Please generate lecture slides for maximum likelihood estimation.

Note that:

1. The lecture duration is 2 hours, so we need to generate 30 pages.
2. The slides should include motivation, problem, intuitive solution, and detailed math equations.
3. Please make sure the lecture is well self-contained.

Output Format: LaTeX code



(a) AFlow **(b) Atom** **(c) Flow** **(d) o4-mini-high** **(e) SEQCV**

Figure 7: Illustration of lecture slides.

AFlow This slide deck covers the essentials of maximum likelihood estimation: it begins with a title and outline that includes motivation, problem statement, intuitive solution, mathematical formulation, examples for Bernoulli and Normal distributions, properties, applications, challenges, and a conclusion. It includes log-likelihood definitions and derivations for Bernoulli and Gaussian distributions, with a brief discussion of asymptotic properties. However, the deck includes only 13 topical slides, which are far fewer than the 30 required resulting in insufficient coverage for a two-hour lecture. The content is limited to just two examples and omits other important distributions such as Poisson, Exponential, and Gamma. Furthermore, while mathematical formulas are included, intuitive explanations and real-world applications are sparse, making the slides less accessible to students who benefit from contextual learning.

Table 4: Summary of Major Weaknesses in Lecture Slide Implementations (MLE)

Method	Major Weakness
AFlow	Covers only 13 slides, far short of the required 30. Focuses on just two distributions (Bernoulli and Normal), omitting others like Poisson or Exponential. Lacks sufficient intuitive explanation and applied context.
Atom	Provides only 14 slides with limited derivations. Contains placeholders and lacks worked examples. Does not clearly explain when or why to use MLE.
Flow	While it meets the 30-slide requirement and covers a wide range of MLE topics, it includes minimal discussion of applied use cases and gives only a brief treatment of model limitations.
o4-mini-high	Produces a complete slide count but with shallow treatment of advanced topics. Omits logistic regression and EM algorithm, and some slides are underdeveloped with limited explanation.
SEQCV	Offers the broadest distribution coverage and includes case studies and computational methods. Minor gaps include occasional skipped steps in derivations.

Atom This version opens with a "Comprehensive Guide" framing and includes a standard table of contents: Introduction, statistical foundations, intuition, mathematical foundations, generalized MLE, properties, computation, applications, and conclusion. It defines MLE formally and includes a placeholder for an example graph. Nonetheless, it contains only 14 slides and remains too brief to fill a two-hour session. The presentation does not include detailed explanations and lacks any worked examples beyond a placeholder. Only the basic likelihood function is shown while omitting important and profound concepts. It also does not provide a discussion of why or when MLE should be used, leaving a gap in forming good motivation for learning or practical guidance.

Flow The Flow output delivers a complete 30-slide deck that methodically addresses all major topics. It begins with motivation and background, followed by a formal problem statement and worked examples for Bernoulli, Gaussian, and exponential family distributions. It further discusses asymptotic properties such as consistency, normality, and efficiency, and explores Fisher information, the Cramér–Rao bound, and the invariance property. Numerical optimization techniques like Newton–Raphson and EM algorithms are also introduced, as are extensions to models like logistic regression. However, the deck relies almost entirely on text and equations. There are no figures or diagrams to enhance intuition. Additionally, while logistic regression is mentioned, other real-world case studies are largely absent, and topics like model misspecification and robust alternatives to MLE are only briefly touched upon.

o4-mini-high This submission produces a concise 31-slide deck that introduces MLE fundamentals including motivation, mathematical formulation, and properties. It defines the likelihood and log-likelihood functions and walks through closed-form solutions for Bernoulli, Gaussian, and Poisson distributions. The slides also touch on asymptotic properties and basic computational techniques like Newton–Raphson and MAP. That said, the deck has several shortcomings. It does not include slide numbers, which makes referencing and navigation difficult. The coverage of example distributions, while accurate, is shallow and does not extend to more advanced topics such as the EM algorithm or regression models. Additionally, many slides are sparsely populated, with only a formula and minimal surrounding explanation.

SEQCV Our method generates a richly detailed lecture deck that includes a well-organized structure beginning with an introduction and motivation, followed by a clear problem statement, intuitive explanation, and thorough step-by-step derivations. These derivations span a wide range of distributions including Normal, Bernoulli, Exponential, Poisson, and Gamma. The presentation also includes applied case studies, computational strategies like Newton–Raphson and EM, and discussions of robustness, limitations, and practical considerations such as logistic regression and comparisons with Bayesian methods. It concludes with a Q&A slide and reading suggestions. While comprehensive, the slides would benefit from the inclusion of intuitive visual aids such as likelihood surface plots or EM iteration diagrams to help learners better grasp abstract concepts. Moreover, although the slide

content is extensive, some derivations skip intermediate steps, which could be confusing for students less familiar with calculus-based proofs.

Overall Among the five methods evaluated, Flow and SEQCVare the most successful in meeting the task requirements. Flow achieves the target length with a precise 30-slide structure and provides a comprehensive theoretical treatment, yet it lacks visual illustrations and concrete, practical applications. SEQCVcovers the broadest scope, including advanced material and practical implementation tips, and is well-suited for a research-level audience. However, it would benefit from a clearer visual structure and improved use of diagrams. AFlow and Atom show promise with strong outlines and partial derivations but fall significantly short in slide count and content depth. o4-mini-high, although organized and conceptually accurate, is too minimal in detail.

B.2 Website Development – NeurIPS 2025

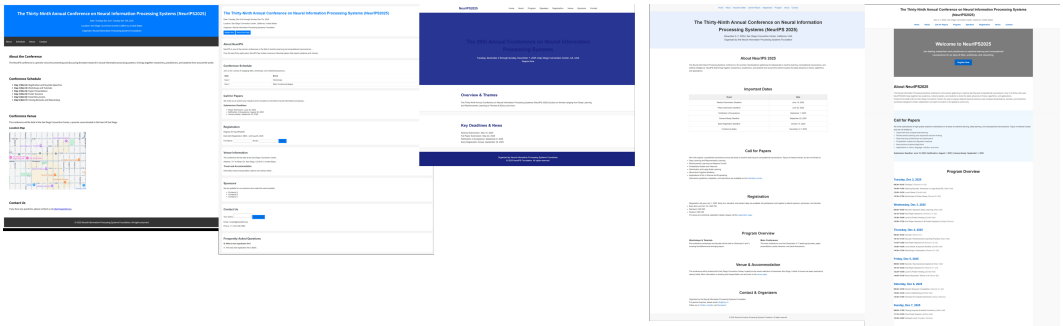
Task 2: NeurIPS2025 Website

I want to create a website for the following conference:

- **Conference Name:** The Thirty-Ninth Annual Conference on Neural Information Processing Systems (NeurIPS 2025)
- **Date:** Tuesday, Dec 2nd through Sunday, Dec 7th, 2025
- **Location:** San Diego Convention Center, California, United States
- **Organizer:** Neural Information Processing Systems Foundation

Please generate a detailed website structure and content for this conference. Ensure the content is professional, clear, and suitable for an international academic conference.

Output Format: HTML and CSS



(a) AFlow **(b) Atom** **(c) Flow** **(d) o4-mini-high** **(e) SEQCV**

Figure 8: Illustration of website Development.

AFlow The content is too short, and it does not meet the standards of the international academic conference.

Atom The Atom implementation only displays schedules for December 2nd and 3rd, omitting any other relevant dates. Furthermore, the provided content lacks sufficient depth and detail to meet the standards expected of an international academic conference.

Flow Flow meets the basic task requirements but fails to generate specific dates for each day as specified in the prompt. Additionally, the use of the designated key above is non-functional and

Table 5: Summary of major weaknesses in schedule generation implementations

Method	Major Weakness
AFlow	Content lacks the depth and quality expected for an international academic conference.
Atom	Displays only schedules for December 2nd and 3rd, and the content lacks the depth and quality expected for an international academic conference.
Flow	Although it meets basic task requirements, it fails to generate specific dates for each day, includes a broken key that triggers a page request, and presents insufficient content for academic standards.
o4-mini-high	No major weakness; all task requirements are fulfilled with appropriately formatted and complete academic scheduling content.
SEQCV	No major weakness; generates a correct and complete schedule aligned with international academic standards.

instead triggers an unintended page request. The overall content is not sufficiently detailed and does not align with the quality expected for international academic events.

o4-mini-high This version successfully fulfills all the outlined task requirements. The schedule is presented as expected and meets the standard for academic presentation formatting.

SEQCV Similar to o4-mini-high, SEQCVmeets all the specified task requirements, generating the schedule and content in a manner appropriate for the intended academic context.

Overall While Atom and Flow demonstrate partial task completion, both suffer from either incomplete date coverage or inadequate content quality. In contrast, o4-mini-high and SEQCVfully satisfy the prompt's demands and produce output suitable for academic presentation.

B.3 Game Development – Pac-Man fuses Tank City

The generation results are presented in Figure 9.

Task 3: Pac-Man fuses Tank City

Develop a single-player game that fuses mechanics from Pac-Man and Tank City.

The player controls a Pac-Man-like character navigating a maze, collecting pellets for points. Enemy tanks act as ghost-like chasers, actively pursuing the player using basic AI pathfinding.

Enemy tanks can shoot bullets in four directions. Unlike traditional bullets, these bullets do not disappear—instead, they remain in the maze as collectible pellets. The player must guide Pac-Man to eat as many of these bullets as possible to gain points and avoid tank power-ups.

If bullets are left uncollected, enemy tanks can absorb them to accelerate their shooting rate, making them more dangerous. Tanks can also switch between patrol and chase modes to track the player.

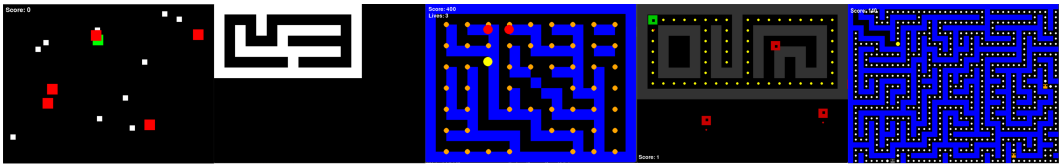
The player can collect special items that temporarily allow Pac-Man to eat enemy tanks for bonus points.

The game must support:

- Maze-based movement and pellet collection
- AI-controlled enemy tanks with patrol and chase behaviors
- Enemy tank shooting mechanics with persistent bullets
- Tank interaction with bullets (absorb and power-up)
- A GUI (using Pygame) showing the maze, player, enemies, pellets, and bullets
- Smooth animations and clear visual feedback

Sound effects are not required.

Output Format: Python



(a) AFlow (b) Atom (c) Flow (d) o4-mini-high (e) SEQCV

Figure 9: Illustration of Pac-Man fuses Tank City.

AFlow failed to meet the core requirements of the task. The game only implemented a few basic functions and lacked critical gameplay elements. The maze was not successfully generated, enemy tanks did not fire bullets, and the tanks remained stationary instead of exhibiting tracking or patrol behaviors.

Atom also did not fulfill the task objectives. The output consisted of an empty maze without any interactive elements. There was no player character, no pellets or bullets to collect, and no tanks present in the scene. As a result, the game was entirely non-functional.

Flow partially addressed the requirements. In this version, tanks were able to track the player using basic AI, but they could not shoot bullets. This left a significant portion of the gameplay mechanics such as persistent bullets and their interactions unimplemented.

Table 6: Summary of major weaknesses in pac-man + tank city fusion game implementations.

Method	Major Weakness
AFlow	Incomplete implementation; maze generation failed, tanks remain static, and no bullet mechanics or tracking behavior is present.
Atom	Game is non-functional; only an empty maze is rendered with no player, tanks, pellets, or bullets. Core gameplay elements are entirely missing.
Flow	Enemy tanks can track the player but cannot shoot bullets, leaving out key mechanics such as persistent projectiles and bullet interactions.
o4-mini-high	Players and tanks do not follow maze paths and can move freely across the screen; bullets vanish on hit instead of persisting, and patrol logic breaks at screen edges.
SEQCV	Mostly fulfills task requirements but suffers from flawed movement controls—arrow key inputs only move the player between corridor ends, preventing precise navigation.

o4-mini-high included several game components but deviated substantially from the task expectations. When bullets hit the player, the player would die immediately, but the bullets would not remain in the maze as collectibles. The maze existed visually but did not function as a constraint for movement. Both the player and tanks could move freely across the screen without following the maze paths. Additionally, enemy patrols disappeared off the edge of the screen, and bullets lacked the special interactions specified in the task.

SEQCV came closest to satisfying the task requirements. It included the necessary components, but player movement within the maze was flawed. Using the arrow keys moved the player from one end of a corridor to the other without allowing accurate navigation to specific positions, making it difficult to control the character effectively.

Overall, among the five methods evaluated, only SEQCVimplemented the required features to a meaningful extent. However, it still exhibited critical navigation issues within the maze, preventing precise control and detracting from the gameplay experience.

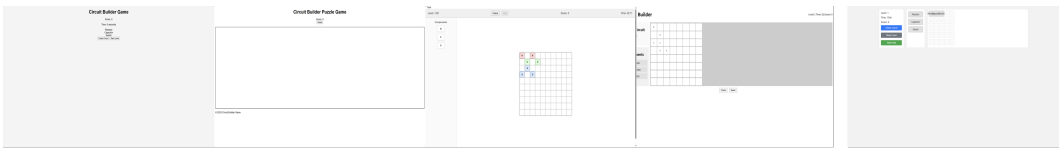
B.4 Web Game Development - Puzzle Game

Task 4: Puzzle Game

Create a casual browser puzzle game where players build electrical circuits from components.

Mechanics: Drag-and-drop resistors, capacitors, and switches onto a grid to complete a target circuit.
Levels: 20 puzzles of increasing difficulty.
Scoring: Based on correctness and build time.
Tech: HTML5 + CSS3 + vanilla JavaScript.

Output Format: HTML5 + CSS3 + vanilla JavaScript



(a) AFlow (b) Atom (c) Flow (d) o4-mini-high (e) SEQCV

Figure 10: Illustration of Puzzle Game.

Table 7: Summary of major weaknesses in puzzle game implementations.

Method	Major Weakness
AFlow	Incomplete implementation with no grid, drag-and-drop functionality, or playable levels; the game cannot be initiated.
Atom	Entirely non-functional: lacks grid, interactivity, and level structure; button presses yield no response.
Flow	No major weakness; the game is playable, visually clear, and meets the task requirements.
o4-mini-high	No major weakness; provides a clean, functional interface with color-coded components and full gameplay.
SEQCV	No major weakness. A minor issue is that that text slightly overlaps.

AFlow AFlow did not meet the task requirements. No playable game was produced, and the implementation appeared incomplete and overly simplistic. The interface only listed the names of the components resistors, capacitors, and switches, without any drag-and-drop functionality. Furthermore, there was no grid layout, and the game could not be initiated. Level progression was also entirely missing.

Atom Atom also failed to deliver a working puzzle game. The page lacked a grid and the basic mechanisms required to begin gameplay. No levels were implemented, and features such as a timer or an answer validation button were absent. Additionally, pressing buttons on the screen did not trigger any response, rendering the game unplayable.

Flow Flow fulfilled the task requirements and successfully produced a playable puzzle game. The game interface was visually appealing, and different colors were used to represent resistors, capacitors, and switches.

o4-mini-high The o4-mini-high version also met the task requirements and presented a fully functional game. The interface was clean and attractive, and the components were color-coded to enhance visual clarity.

SEQCV SEQCVproduced a playable game that satisfied the task criteria. A noteworthy enhancement in this version was the inclusion of a hint system, which added depth and improved gameplay. Nevertheless, the drag-and-drop interaction lacked polish, when components were placed on the grid, they retained their original labels and overlapped.

Overall In summary, AFlow and Atom failed to deliver playable games and did not fulfill the project requirements. Flow, o4-mini-high, and SEQCVmanaged to create playable games, though each had distinct shortcomings. Among these, Flow, o4-mini-high, and SEQCVstood out as the most successful implementations in terms of functionality and gameplay experience.

B.5 Game Development – Snake Fuses Chess

Task 5: Snake Chess

Develop a real-time arcade game that merges Snake with simplified chess tactics.

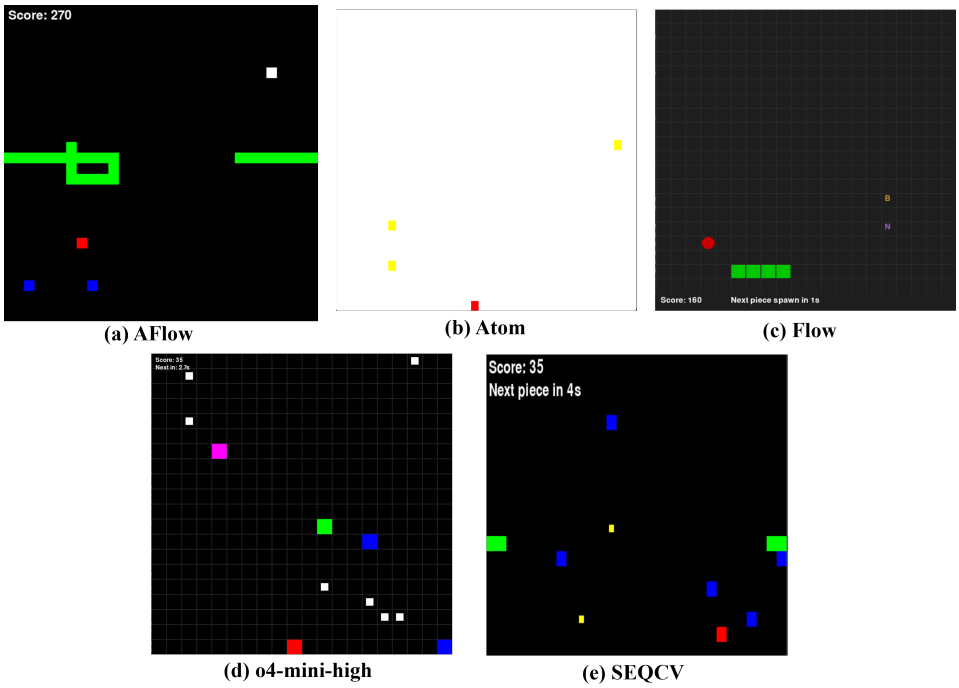
Player: Controls a "snake" that grows by eating apples.
Chess Pieces: Randomly spawning pawns, knights, and bishops move following their rules.

Mechanics:

1. If the snake's head collides with an apple → length+1, score+10.
2. If the snake's head lands on a chess piece's square → capture it: score + piece value (pawn=5, knight=10, bishop=15); snake doesn't grow.
3. Knights can shoot bullets. When the player is shot → length–1. If length = 0 → game over.
4. Bullets disappear when they hit the edges.
5. Chess pieces move one step every second; if they collide with the snake's body → game over.
6. Board wraps around edges.
7. No extra image files.

UI: Pygame window showing the grid, snake, apples, chess pieces, current score, and "next piece spawns in Xs."

Output Format: Python



(a) AFlow (b) Atom (c) Flow

(d) o4-mini-high (e) SEQCV

Figure 11: Illustration of Snake Fuses Chess.

AFlow The AFLow implementation suffers from critical omissions. Although three chess pieces appear at the start of the game, no further pieces spawn, rendering the "next piece spawns in Xs"

Table 8: Summary of major weaknesses in snake-chess implementations.

Method	Major Weakness
AFlow	Fails to spawn additional chess pieces and does not execute piece movement or self-collision detection, leaving key mechanics inactive after initial setup.
Atom	Omits critical mechanics i.e., boundary and knight projectiles.
Flow	Suffers from unmanageable speed, broken bullet mechanics, and lacks dynamic chess piece behavior, severely compromising playability.
o4-mini-high	No major weakness; implements all core features including wraparound, per-second movement, scoring, collisions, and UI elements correctly.
SEQCV	Core gameplay mechanics are solid and complete, but the missing background grid detracts from spatial clarity during play.

indicator non-functional. Additionally, pawns, knights, and bishops remain static, failing to move at one-second intervals as specified. While the apple collection mechanics work correctly, growing the snake and adding to the scor, self-collision does not lead to a game over, which deviates from the core rules of snake gameplay.

Atom Atom handles apple and piece scoring accurately, awarding points appropriately for apples and captured pieces. However, it lacks two fundamental features: edge wrapping and knight projectiles. When the snake crosses the game board's boundary, it disappears instead of wrapping around to the opposite side. Although knights spawn and move correctly, they never shoot bullets, omitting an important gameplay hazard.

Flow The Flow variant struggles significantly with game pacing and bullet behavior. The snake's movement speed is excessively high, making the game difficult to control. Furthermore, bullets emerge from off-screen positions and instantly end the game upon impact, rather than subtracting a single segment of the snake's length. Like AFLow, Flow fails to implement the per-second movement of chess pieces and does not spawn any additional pieces beyond the initial set.

o4-mini-high This implementation meets the full range of required mechanics. It correctly handles snake growth and scoring upon apple consumption and awards appropriate points for capturing chess pieces without increasing the snake's length. Chess pieces move at one-second intervals and spawn at regular timed intervals. The snake wraps around board edges, self-collision results in game over, and knights shoot bullets that reduce the snake's length by one segment. Bullets also disappear upon reaching the edge of the board. The UI includes both a score display and an accurate "next piece in Xs" timer.

SEQCV This version adheres to all specified mechanics except for a missing visual grid on the game board. Despite the absent background layout, the game successfully incorporates timed chess piece spawning, per-second piece movement, accurate wraparound behavior, and self-collision detection. Apples cause snake growth and score increase, and chess pieces are captured correctly without length gain. Knight bullets correctly subtract one segment from the snake and vanish when they reach the screen edge.

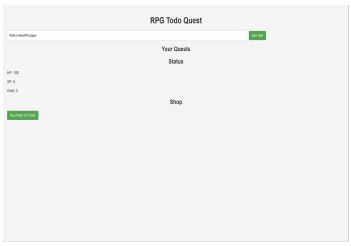
Overall AFLow and Flow both fail to implement timed chess piece behavior and do not handle self-collision correctly. Flow is further hindered by erratic game speed and flawed bullet mechanics. Atom handles scoring and piece motion but omits crucial features like edge wrapping and knight attacks. In contrast, o4-mini-high and SEQCVsuccessfully deliver the intended Snake-Chess hybrid gameplay. The only deviation in SEQCVis the lack of a grid background, while o4-mini-high satisfies every requirement, making these two implementations the most faithful to the design prompt.

B.6 Tool Development - Task Manager RPG

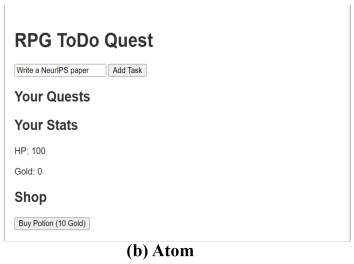
Task 6: Task Manager RPG

Build a desktop "RPG" where user's real-world todo items become in-game quests.
Features:
1. User adds a task → appears as a "quest" on the map.
2. Completing the task (marking done) triggers battle simulation against a monster.
3. Success grants XP and loot; failure reduces HP.
4. Shop uses gold to buy potions that restore HP.
5. Audios are not needed
6. No extra image files

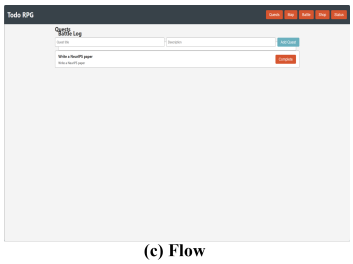
Output Format: HTML+CSS+JS



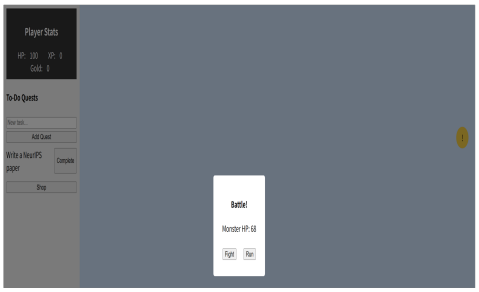
(a) AFlow



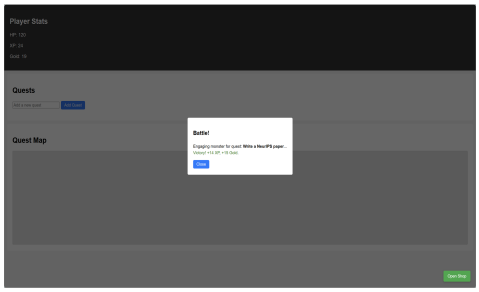
(b) Atom



(c) Flow



(d) o4-mini-high



(e) SEQCV

Figure 12: Illustration of Task Manager RPG.

Table 9: Summary of major weaknesses in task manager RPG implementations.

Method	Major Weakness
AFlow	Task input is non-functional, no tasks appear after submission, and no RPG features are triggered, likely due to missing event bindings or state logic.
Atom	Shares the same failure as AFlow: the interface accepts input but fails to reflect changes or create quests, indicating broken UI-state integration.
Flow	Allows task submission and rendering, but the "Complete" button has no effect which battle logic and stat progression are entirely unimplemented.
o4-mini-high	No major functional weakness; all core features are present. However, styling is minimal, and the UI lacks polish compared to alternatives.
SEQCV	No major functional weakness; all gameplay mechanics operate correctly. Minor improvements could be made in dynamic cost modeling and transport detail.

AFlow The AFlow implementation fails at the most basic level: when a task is entered and the "Add Task" button is clicked, no quest appears on the map. There is no visual or console feedback,

and the underlying application state remains unchanged. This indicates that essential wiring, such as input bindings, a task-tracking array, or event listeners is likely missing. As a result, the key functionality of mapping tasks to quests and triggering RPG mechanics upon completion is completely unimplemented.

Atom Atom's behavior closely mirrors that of AFlow. Although the UI displays an input field and a button, clicking "Add Task" does not result in any visual change or state update. No quests are rendered, and none of the game mechanics can proceed. Like AFlow, this suggests a disconnect between user input and the internal quest-tracking logic which possibly due to incomplete event handling or missing state integration.

Flow The Flow version succeeds in registering new tasks: clicking "Add Task" correctly appends the quest to the UI list. However, the quest's "Complete" button is non-functional. No battle modal is triggered, and RPG mechanics such as XP gain, HP loss, or gold rewards do not respond to quest completion. The core gameplay loop is thus partially implemented, but the combat simulation and stat updates are missing, likely due to a lack of handlers or modal logic associated with quest state changes.

o4-mini-high This version delivers a complete and fully functional implementation of all six specified features. Tasks are correctly transformed into quests on the map; completing a task launches a battle interface; outcomes appropriately modify XP, HP, and gold balances; and a shop enables gold to be exchanged for potions. While the user interface is visually basic, it is coherent and operational. Every critical user interaction performs as intended.

SEQCV SEQCVmatches the functionality of o4-mini-high but adds refinements. It correctly maps user tasks to quests, triggers polished battle dialogs upon completion, and manages game state updates across XP, HP, and gold. The shop is responsive and well-integrated. Visually, the interface is more polished and consistent than o4-mini-high, with smoother transitions and more intuitive dialogs.

Overall Among the five methods evaluated, only o4-mini-high and SEQCVsuccessfully implement the full RPG task manager experience. Flow partially satisfies the requirements by registering tasks as quests but fails to connect this to the rest of the gameplay loop. Both AFlow and Atom break down at the input stage, unable to transform tasks into game objects due to likely omissions in event binding or state logic.

In terms of prompt adherence and effectiveness:

Feature completeness: SEQCV$\approx$ o4-mini-high > Flow > Atom $\approx$ AFlow.

UI/UX quality: SEQCV> o4-mini-high > Flow > Atom / AFlow (not assessable).

The two fully functional solutions indicate proper use of event-driven design and state management, whereas the others likely suffer from poor component integration or uninitialized logic.

B.7 Game Development – Tetris Fuses Bejeweled

Task 7: Tetris-fuses Bejeweled

Develop a game that fuses Tetris and Bejeweled mechanics. This game needs to add keyboard control function. Falling tetrominoes should lock into a grid and transform into colored gems. The game must support both Tetris line-clearing and Bejeweled match-3 clearing, triggering chain reactions and bonus points. Include a GUI (using a framework like Pygame) that displays the game grid, current score, and next tetromino preview, along with smooth animations. No sound effects are needed.

Output Format: Python

AFlow The AFlow implementation correctly spawns and locks tetrominoes into the grid, but it omits any "game over" logic: new pieces continue appearing and stacking beyond the top boundary,

Table 10: Summary of major weaknesses in game implementation.

Method	Major Weakness
AFlow	Omits "game over" logic and never triggers Bejeweled-style match-3 eliminations which pieces stack beyond the top and identical gems remain static.
Atom	Fails to launch entirely (crashes or hangs on startup), preventing any assessment of gameplay mechanics or GUI.
FLow	Overly permissive elimination logic clears non-contiguous or unintended clusters, producing unpredictable chain reactions.
o4-mini-high	Gameplay and animations are smooth, but the score display is truncated or inconsistently updated during combos, undermining point tracking.
SEQCV	Core mechanics work correctly.

violating the core Tetris constraint. Moreover, the Bejeweled match-3 mechanic never activates, adjacent gems of the same color remain static, so no eliminations or chain reactions occur.

Atom Atom's build fails to launch: on startup the application either crashes or hangs indefinitely, preventing any gameplay. As a result, none of the required features that tetromino locking, line clearing, gem transformation, GUI display, or keyboard controls, can be evaluated.

Flow FLow implements both line-clearing and match-3 behaviors, but its elimination logic is overly permissive: it sometimes clears non-contiguous or unintended clusters, wiping out gems that aren't directly matched. This overreach causes unpredictable chain reactions and undermines player strategy.

o4-mini-high Among the five variants, o4-mini-high delivers the smoothest play experience: falling, locking, line clears, and match-3 mechanics all function correctly, and keyboard inputs are responsive, with fluid animations. However, its score display is truncated or inconsistently updated during combos, preventing players from accurately tracking points and bonuses.

SEQCV SEQCVsuccessfully integrates both Tetris and Bejeweled rules, with correct grid locking and match-3 elimination. Visually it matches the other implementations.

Overall None of the five implementations fully satisfies the original specification. Atom doesn't run, offering zero utility. AFlow lacks both game-over enforcement and gem clearance. FLow's match detection is overactive, compromising game integrity. o4-mini-high and SEQCVdelivers solid mechanics and visuals yet misses a robust scoring system.

B.8 Family Travel Plan

Task 8: Travel Plan

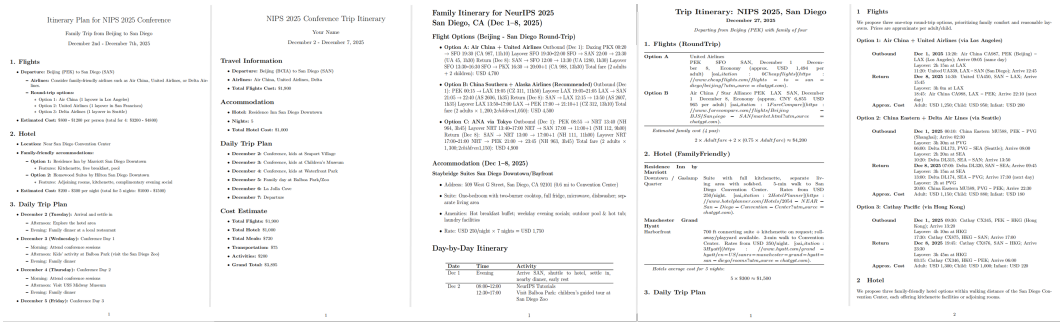
I need to attend the NeurIPS 2025 conference.

Dates: Tuesday, December 2nd through Sunday, December 7th, 2025.
Location: San Diego Convention Center, California, United States.

I want to bring my husband and two children with me. We will be departing from Beijing. Please help me plan:

1. **Flights:** Round-trip options from Beijing to San Diego, considering family-friendly airlines and layovers.
2. **Hotel:** Family-friendly accommodations near the convention center, preferably with kitchenette or adjoining rooms.
3. **Daily trip plan:** Sightseeing and activities suitable for children, balanced with my conference schedule.
4. **Cost estimate:** Total budget breakdown for flights, hotel, meals, local transportation, and activities.

Output Format: LaTeX itinerary plan



(a) AFlow (b) Atom (c) Flow (d) o4-mini-high (e) SEQCV

Figure 13: Illustration of Travel Plan.

Table 11: Summary of major weaknesses in itinerary methods

Method	Major Weakness
AFlow	Lacks specific flight schedules, concrete daily activity times, and detailed budget breakdowns.
Atom	Omits key travel details and lacks transparency in airfare and meal cost estimates.
Flow	Inflates the total budget without justification.
o4-mini-high	Includes inconsistent pricing and misalign with the conference schedule.
SEQCV	Minor issues like return flight date mismatch and oversimplified meal budgeting; lacks local transport comparison.

AFlow This itinerary correctly identifies the conference dates (December 2–7, 2025) and suggests family-friendly airlines, hotels with kitchenettes or adjoining rooms, a structured day-by-day activity plan, and an overall cost estimate. However, it lacks concrete details such as exact flight schedules and specific departure/return times from Beijing. The cost estimates are presented in broad ranges rather than broken down by adult and child, making budgeting less actionable. Daily activities are

described generically, for instance, “family dinner” appears without reference to timing or reservation logistics. Moreover, the itinerary omits supplementary but important considerations such as visa information, travel insurance, or packing recommendations.

Atom The "Atom" plan efficiently aligns the flights, hotel, daily itinerary, and budget components, and provides total figures for each category. Nevertheless, it omits key travel details, including layover cities, exact flight durations, and a transparent rationale for the total airfare of $1,900 for four passengers which an amount that appears inconsistent with typical market rates. The hotel section is limited to a single option without mentioning alternatives or amenities such as family dining options. The daily itinerary does not specify conference session times or consider the ages of the children when recommending activities. Additionally, some estimates, such as $720 allocated for meals over six days, lack justification and may appear inflated.

Flow This itinerary is branded with the NeurIPS name and includes three detailed flight options, each with carrier codes, schedules, and differentiated fares for adults and children. It also provides a well-organized day-by-day schedule and a proposed seven-night hotel stay. However, it incorrectly shifts the travel dates to December 1–8 rather than the specified December 2–7, and suggests a seven-night stay instead of the expected five. The total budget of $11,790 is notably excessive and not well justified, raising questions about its accuracy.

o4-mini-high This version includes concrete flight legs, pricing details, and two hotel recommendations with citations. Despite its thorough structure, it introduces significant errors. The itinerary title mistakenly lists “December 27, 2025,” which is unrelated to the actual conference. The flight dates span December 1–8, conflicting with the user’s request. Currency conversions are inconsistently applied, and hotel rates are uniformly set at $300 per night without reflecting differences in suite types or amenities like kitchenettes. The activity plan compresses the conference into five days but fails to align with actual session counts and lacks proper integration of child-friendly scheduling.

SEQCV This itinerary is the most comprehensive and well-aligned with the prompt. It correctly spans December 2–7, provides three well-differentiated flight options with complete schedules and per-passenger pricing, and offers three hotel choices near the venue, each with detailed features and family suitability. The daily schedule is rich in content and time-specific, and the cost table is clear and supplemented with general travel tips. Minor shortcomings include a return flight scheduled for December 8 rather than December 7, and a flat meal budget of $200 per day that does not distinguish between adult and child needs. Additionally, it lacks a comparison of local transportation options, such as the relative costs of public transit versus rideshares.

Overall All five methods address the core requirements of flight planning, lodging, daily itinerary, and budgeting, but their levels of detail and accuracy vary significantly. AFlow provides a sensible range of accommodations and correctly aligns with the conference dates but lacks precision in its flight information and day planning. Atom is concise and organized but falls short in logistical specificity and cost justification. Flow stands out for its exhaustive airline data and scheduling, yet mislabels the conference, adjusts the dates incorrectly, and overestimates the budget without explanation. o4-mini-high demonstrates strong structure and citation use but misaligns dates, misprices accommodations, and misrepresents the event timeline. In contrast, SEQCVmost closely adheres to the prompt, presenting a balanced, richly detailed itinerary with thoughtful accommodations and scheduling. While there is room for minor improvement, particularly in return date alignment and cost granularity, it remains the strongest overall solution.

C Evaluation on Other Benchmarks

Datasets We also evaluate SEQCV on commonly used reasoning benchmarks. Note that these benchmarks are not primarily designed to assess agentic behavior. We include these benchmarks only for completeness. For example, the GSM8K dataset states: “These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations.” Such problems can often be solved effectively by a single high-performance model with strong chain-of-thought reasoning capabilities, leaving limited room for

multi-agent advantages. Moreover, we observed that some samples in these datasets contain incorrect ground-truth answers.

The evaluation covers four categories of reasoning tasks: *Mathematical reasoning:* MATH [58] (617 level-5 problems from geometry, statistics, number theory, algebra, and calculus) and GSM8K [59]; *Knowledge-intensive reasoning:* MMLU-CF [60]; *Logical reasoning:* BBH [61]; *Multi-hop reasoning:* HotpotQA [62] and LongBench [63] (samples from MuSiQue [64] and 2WikiMultiHopQA [65] for long-context reasoning). For each benchmark dataset, we randomly sample 300 examples for evaluation. We follow the experimental setup of recent state-of-the-art studies [41, 39].

Baselines We compare SEQCV with two groups of baselines: (1) *Classical prompting methods:* Chain-of-Thought (CoT), CoT with Self-Consistency ($n = 5$) [27], MultiPersona [47], Self-Refine (up to 3 iterations) [54], Analogical Reasoning [66], and MedPrompt (3 answers, 5 votes) [67]; (2) *Advanced reasoning frameworks:* AFlow [39], ADAS [68], Forest of Thought (FoT) [69], Atom (AoT) [41], and Flow [38]. For FoT, we implement the Tree-of-Thoughts variant ($n = 3$) for general applicability. All results are averaged over three runs.

We use `gpt-4o-mini` [70] as the main backbone model for all baselines. For SEQCV, we evaluate two configurations: **same:** three `gpt-4o-mini` models; **different:** `gpt-4o-mini`, `gpt-4.1-mini`, and `gpt-4.1-nano` (Table 12). Following AoT, we provide each model with different types of prompts to encourage diverse reasoning processes and to avoid cache effects. We use `gpt-4.1-nano` [71] to automatically verify whether the model predictions match the ground truth and to compute accuracy. The results are presented in the following table.

Table 12: Performance comparison across tasks (%). We compare SEQCV with classical prompting methods and advanced reasoning frameworks. Results are reported as exact match accuracy for MATH, GSM8K, BBH, and MMLU-CF, and as F1 scores for HotpotQA and LongBench.

Method	MATH	GSM8K	BBH	MMLU-CF	HotpotQA	LongBench	Avg.
CoT	78.3	90.9	78.3	69.6	67.2	57.6	73.7
CoT-SC (n=5)	81.8	92.0	83.4	71.1	66.2	58.6	75.5
Self-Refine	78.7	91.7	80.0	69.7	68.3	58.2	74.4
MultiPersona	50.8	92.5	81.1	70.3	69.8	59.3	75.4
MedPrompt	50.0	93.2	82.0	70.8	70.5	60.1	76.1
Analogical Reasoning	65.4	87.2	72.5	65.8	64.7	52.9	68.1
ADAS	76.0	90.8	75.3	68.9	64.5	56.2	71.9
AFlow	83.0	93.5	76.0	69.5	73.5	61.0	76.1
FoT (n=8)	82.5*	94.0*	82.4	70.6	66.7	59.1	75.9
SEQCV (same)	80.5	92.0	84.5*	70.8*	76.5*	66.8*	77.2*
SEQCV(different)	**86.5**	**94.3**	**87.3**	**71.5**	**77.8**	**62.0**	**79.9**

D System Design

The system orchestrates multiple agents through an iterative three-phase cycle. First, agents execute tasks in parallel with shared memory access. Second, cross-agent validation using majority voting filters low-quality outputs. If the validation success rate falls below a threshold, the task is decomposed into smaller subtasks that are executed recursively, with results accumulated across subtasks. Otherwise, consensus voting selects the best result for archival. Shared memory is updated after each iteration, enabling progressive refinement until convergence or the maximum number of iterations is reached.

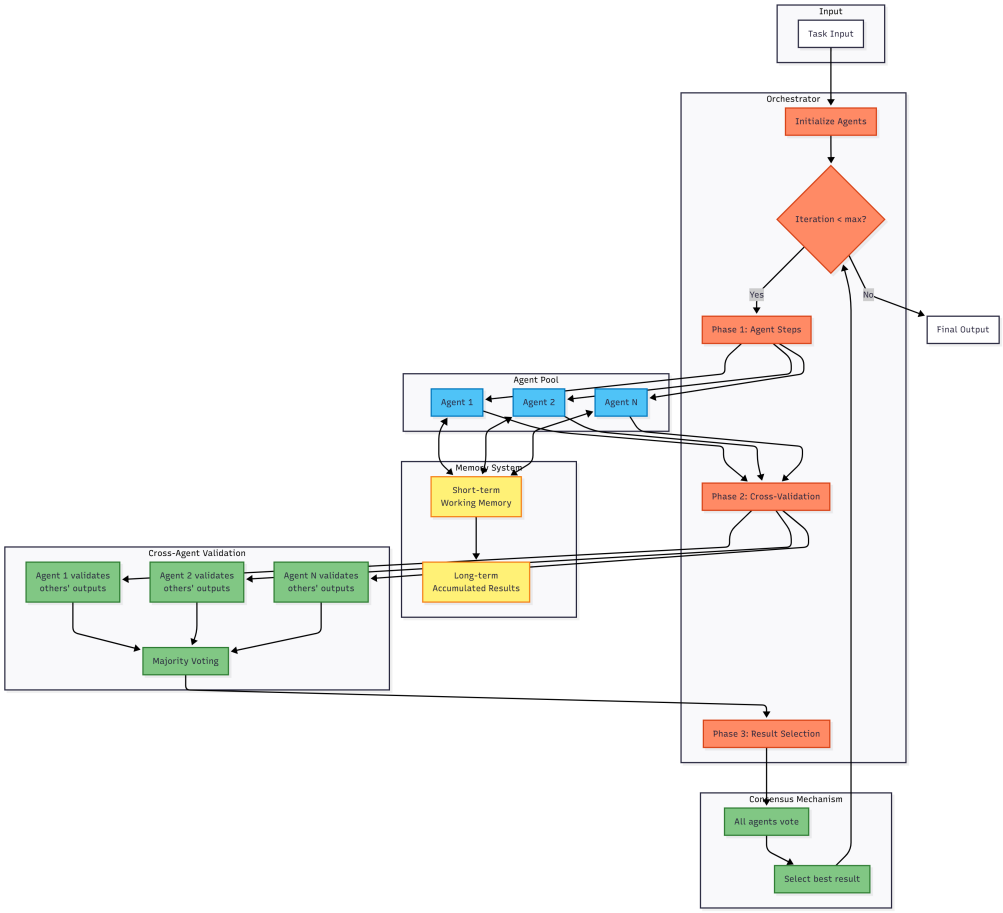


Figure 14: Multi-agent collaborative architecture.

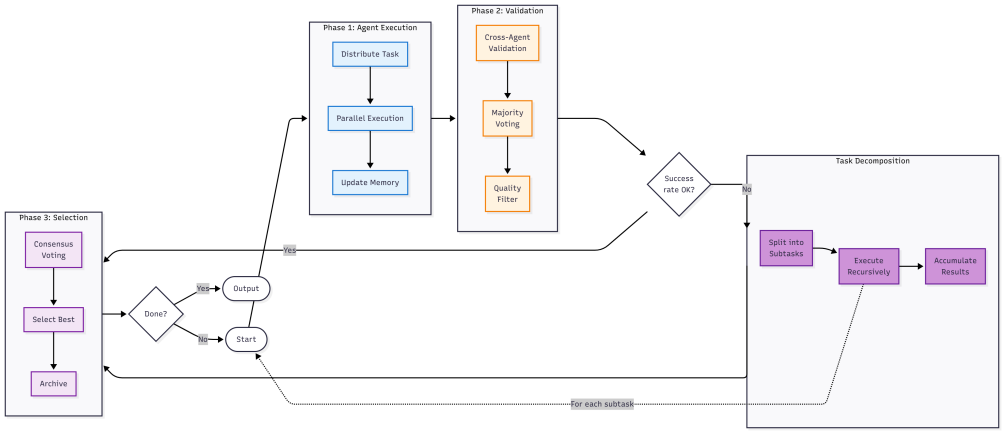


Figure 15: Three-phase execution cycle.

E Prompts for SEQCV

Prompt 1: Segment Generation Prompt

You are an **Incremental Task Developer** whose sole job is to continue the current task in sequential segments.

Policy

- **First iteration** (no prior output): generate the initial part of the result for the task.
- **Subsequent iterations** (prior output exists): generate the next part of the result by either choosing to override previously generated segments or seamlessly continue from where the last segment ended.
- Each `new_content` section should be around 300 tokens.
- If any changes to previously generated content are required, use **override**.
- Ensure the result is logically and syntactically coherent, needing no manual editing.
- Aim for quality and depth in each step; do not rush to completion.
- A maximum of 10 rounds is allowed.
- You may improve the *justify* section in future rounds.
- No omissions are permitted in the generated result.

Output Format
Return a single JSON object with exactly **four** keys:

1. `justify` (string):
   Explain what was done in this step, decisions made, and what remains. This guides future development.
2. `new_content` (string):
   Generate the next logical segment of the overall result.
   Must integrate smoothly with previous output and contain **no more than 300 tokens**.
3. `status` (string):
   `"complete"`: This output, when combined with any previous content, forms a full and correct implementation that meets the task requirements.
   `"ongoing"`: The implementation is still in progress and needs additional iterations.
   *Note: For code tasks, do **not** generate the main function or full framework yet if the status is "ongoing".*
4. `mode`: `"override"` or `"continue"`
   `"continue"`: Appends this segment to the previous output.
   `"override"`: Replaces all previously generated content with this result.

Sample response format:

```
{
  "justify": "",
  "new_content": "",
  "status": "",
  "mode": ""
}
```

Prompt 2: Task Decompose Prompt

You are a **task decomposer**. Your job is to split a task into a set of strictly simpler **interdependent subtasks**, forming a Directed Acyclic Graph (DAG).

Guidelines for DAG-Based Decomposition

- Each subtask should be **indivisible**, focused, and simpler than the original task.
- Define subtasks such that they can be executed **only after** their dependencies are completed. The overall structure must form a **valid DAG** (no cycles).
- **Do not** include any meta-planning subtasks (e.g., "Plan steps", "Outline", "Define framework"). The executor will handle all planning.
- All subtasks must adhere strictly to the **output format** defined in the original objective. No extra headers, annotations, or explanations are allowed.
- If subtasks produce partial outputs to be combined, their results should be **compatible and composable** in a way that aligns with the final task format.
- The system cannot create any files.

Output Format
Return a single JSON object with exactly one key:

1. `subtasks` (list): a list of all subtasks, each with its unique ID, objective, and a list of dependency IDs it depends on.

```
{
  "subtasks": [
    {
      "id": 0,
      "objective": "...",
      "depends_on": []
    },
    {
      "id": 1,
      "objective": "...",
      "depends_on": []
    },
    {
      "id": 2,
      "objective": "...",
      "depends_on": []
    },
    {
      "id": 3,
      "objective": "...",
      "depends_on": []
    }
  ]
}
```

Prompt 3: Cross-Model Validation Prompt

You are an intermediate-step validator. Your task is to assess whether the accumulated previous output and the current output fragment correctly align with the overall task and are error-free enough to continue generation.

Inputs:

1. **overall objective**
2. **current task description**
3. **current output fragment**

Validation Criteria

- Does the current fragment logically continue the task as described?
- Is it free from critical errors, omissions, or contradictions?
- Is it sufficient to serve as the basis for the next iteration?

Output Instructions
Return a single JSON object with exactly two keys:

1. `justify`: justify the error found
2. `result` (str): output `"true"` or `"false"` only

Prompt 4: Cross-Model Voting Prompt

You are a validator. Your task is to rank all of the accumulated previous outputs and determine which answer is correct.

Inputs:

1. **Overall objective**
2. **Different versions of answers**

Output Format
Return a single JSON object with exactly two keys:

1. `justify`: clearly reason in logic why you think the answer is better
2. `votes` (list): a list of **top-2** version indices (e.g. [2, 0]), ordered from the best to the worst. Use 0, 1, 2, ... to represent each version.

Sample response format:

```
{
  "justify": "",
  "votes": []
}
```

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 main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations have been discussed in the main content.

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: [NA]

   Justification: The paper does not include theoretical results.

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: The reproducibility is ensured by detailed instructions and the release of code.

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: [Yes]

   Justification: This paper releases anonymized data and code.

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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental setup and detailed settings are presented in the main context and Appendix as well as code.

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 paper does not report error bars.

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: [No]

   Justification: The paper does not require specific CPU or GPU resources.

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 conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics

10. **Broader impacts**

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

Answer: [NA]

Justification: The presented technique is intended for general use and, at this early stage, has no explicit connection to any societal impact.

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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

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: All the original papers and dataset used for producing experimental results have been cited appropriately.

13. **New assets**

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

    Answer: [Yes]

    Justification: The details of the new assets have been discussed. It does not use another asset.

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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

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: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

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 rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: The LLM is used only for writing, editing, or formatting purposes. The core method development in this research does not involve LLMs as any important, original, or non-standard components.