Hi-DeA: Hierarchical Decision Agents for Dynamic and Static Decision Making

Anonymous ACL submission

Abstract

Large language model agents can call tools and solve many simple tasks, but they still struggle with complex decision problems that involve long context, heterogeneous data, and changing constraints. In this paper, we propose **Hi-DeA**, a hierarchical decision agent framework for dynamic and static decision making. Hi-DeA separates the decision process into planning, routing, and execution. A Planner decomposes the user goal into a task DAG, a dynamic router assigns subtasks to suitable skill-based executors, and the final decision is made by integrating their outputs with user preferences and global constraints. We evaluate Hi-DeA on three representative scenarios: quantitative investment, supply chain allocation, and housing purchase. Experiments on investment, supply chain, and housing tasks show that Hi-DeA improves utility on two tasks, remains competitive on the third, and provides stronger completeness and structured reliability than ablated variants.

1 Introduction

Large language models (LLMs) are now often used as agents. These agents can reason, call tools, search documents, and interact with external environments (Yao et al., 2023b; Schick et al., 2023). This ability makes them useful for many practical tasks, such as question answering, data analysis, web search, and tool-based problem solving. However, many real-world decision problems remain hard for a flat agent. A flat agent usually receives all information at once and then tries to plan, execute, and explain in the same loop. This design is simple, but it can be weak when the task needs a long context, multiple data types, and several domain-specific operations.

In dynamic decision problems, the system must use long time-series data, text signals, and changing states. Examples include quantitative trading and supply chain allocation. In quantitative trading, the system may need to combine price trends, market news, social media sentiment, and portfolio constraints (Xu and Cohen, 2018; Malo et al., 2014). In supply chain allocation, it may need to consider demand, inventory, shipping cost, weather, and fuel price. In static but high-value decision problems, the system must compare many factors and explain the trade-offs. An example is buying a house, where a user may care about price, location, schools, commute, risk, budget, and future policy. These tasks are not only about generating a fluent answer. They require a structured decision process.

We argue that a single-agent design has three main problems in these settings. First, it may lose focus when the input context is long and multimodal. Prior work shows that LLMs can fail to use important information when it appears in a long context (Liu et al., 2024). In decision tasks, this means that the top-level goal can be hidden by many local details. Second, a general-purpose LLM may not have enough domain knowledge for time-series forecasting, sentiment modeling, policy analysis, logistics planning, or risk control. Third, many current tool-use systems depend on static function calling. They can call a tool, but they do not always change the execution plan when the environment or an intermediate result changes. Recent work also shows that dynamic model or tool selection is important when quality, cost, and latency change across inputs (Wang et al., 2025).

To address these problems, we propose **Hi-DeA** (Hierarchical Decision Agents). Hi-DeA follows a leader-executor architecture. The leader is a Planner. It understands the user goal, user preference, and global constraints, and then creates a directed acyclic graph (DAG) of subtasks. The executors are specialized agents. Each executor is linked to one or more skills, and each skill is defined by a structured JSON document. A dynamic router, called EvoRoute, selects executors for subtasks by using semantic matching, schema compatibility, cost, and historical feedback. After execution, the Planner

collects the results, checks whether they support the final decision, and can trigger correction if a result is weak or inconsistent.

The main idea is simple. The Planner should focus on the global decision process, not on every local calculation. Executors should focus on domain-specific work. The router should adaptively connect subtasks with suitable skills. The skill files make the system modular and easier to test. This design also makes the decision process more transparent: each subtask has an input contract, an output contract, and a clear role in the final answer. In this way, Hi-DeA tries to reduce context loss, improve specialization, and support closed-loop correction.

Experiments show that Hi-DeA improves utility on investment and housing tasks and remains competitive on supply chain allocation. It also provides better structured reliability and complete decision coverage. Ablation results further show that success feedback, schema validation, and persona modeling contribute to the final performance.

This paper makes the following contributions:

- We propose **Hi-DeA**, a hierarchical LLM-agent framework that decomposes complex decisions into task DAGs and solves them with skill-based executors.
- We design **EvoRoute**, a dynamic router that assigns subtasks by considering task-skill similarity, schema fit, cost, and feedback.
- We evaluate Hi-DeA on investment, supply chain, and housing tasks, showing improved utility on two tasks, competitive performance on the third, and stronger completeness and reliability.

2 Problem Definition

We define a decision problem as a tuple:

$$\mathcal{P} = (g, u, c, X, \mathcal{S}), \tag{1}$$

where g is the user goal, u is the user persona, c is the set of global constraints, X is the available data, and $\mathcal{S}$ is the skill library. The system must produce a final decision a and an explanation e. The Planner converts the problem into a task DAG:

$$G = (V, E), \tag{2}$$

where each node $v \in V$ is a subtask and each edge $(v_i, v_j) \in E$ means that task v_j depends on the output of task v_i. Each task has a required skill, input schema, output schema, and quality criteria.

3 Hi-DeA

3.1 System Overview

Figure 1 shows the overall design of Hi-DeA. Hi-DeA is designed for complex decision tasks that require multiple types of evidence and several rounds of reasoning. Instead of using one flat agent to read all information and directly produce an answer, Hi-DeA separates the process into planning, routing, and execution. This helps the system keep the global goal clear while using specialized analysis for local subtasks.

Hi-DeA has three layers:

- **Orchestration layer.** The Planner reads the user request, user persona, global constraints, and skill library. It creates a task DAG and later integrates executor outputs.
- **Routing layer.** EvoRoute selects suitable executors for subtasks by considering semantic match, schema compatibility, historical success, and cost.
- **Execution layer.** Executors run domain skills and return structured, schema-checked outputs. Each executor focuses on a specific type of evidence or analysis.

The Planner represents the decision process as a directed acyclic graph. This makes the decision path explicit, allows independent tasks to run in parallel, and lets later tasks reuse earlier evidence.

3.2 Task Representation

Each DAG node is a structured task object. It contains a task ID, a natural language description, dependencies, a required skill type, and expected output fields. The description explains what should be solved. The dependencies tell the executor which previous outputs to use. The required skill guides routing, while the expected fields define what the task must return.

For example, in investment, the Planner may create tasks for price trend analysis, sentiment aggregation, risk estimation, and final action selection. The price trend task may require `trend`, `momentum`, and `volatility`. In housing, the Planner may create tasks for affordability, school quality, neighborhood risk, and final ranking.

This representation gives EvoRoute enough information to select an executor. It also gives the Planner a clear way to verify the result. If required fields are missing or if a task output does not match its expected format, the system can detect the problem before the final decision is produced.

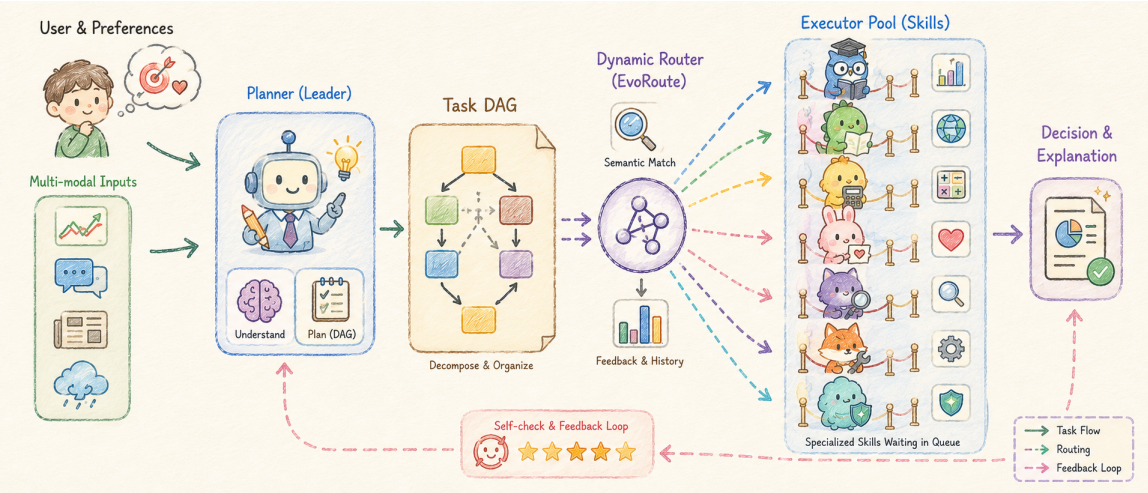

Figure 1: Architecture of Hi-DeA. The Planner decomposes the user goal into a task DAG. EvoRoute dynamically assigns subtasks to skill-based executors. The executor outputs are checked and integrated into the final decision.

3.3 Skill-Based Contracts

A skill is a structured contract between the Planner, router, and executor. It is stored as a JSON file and includes a skill ID, name, description, input schema, output schema, required context, domain logic, and quality checks. The skill file defines what an executor can do, what data it needs, and what result it must return.

The `input_schema` defines required fields, and the `output_schema` defines the result format. The `required_context` lists injected information, such as historical prices, user risk profile, inventory status, mortgage rate, or retrieved policy text. The `quality_checks` define validation rules, such as whether a score is in $[0, 1]$, a shipping ratio sums to 1, or a recommendation has supporting evidence.

We define three types of skills. **Table 1** gives examples of skills used in the framework.

Planner skills. A planner skill describes a high-level decision style. For example, a risk-aware planner gives higher weight to safety, uncertainty, and downside risk. A cost-first planner gives higher weight to cost and efficiency. A multi-objective planner balances several factors and is useful for decisions such as housing purchase.

Skill Type	Example Skill	Main Output
Planner	risk-aware planning	task DAG
User	buyer persona parsing	preference vector
Executor	time-series analysis	trend and risk
Executor	sentiment analysis	sentiment score
Executor	logistics optimizer	route plan
Executor	housing affordability	budget risk

Table 1: Example skills in Hi-DeA. Each skill has an explicit input schema, output schema, and quality check.

User skills. A user skill describes the user persona and preference, such as risk tolerance, budget, time horizon, liquidity needs, location preference, or family needs. The same evidence may lead to different decisions for different user personas.

Executor skills. An executor skill describes a domain function. Examples include time-series analysis, sentiment analysis, logistics cost estimation, weather risk analysis, housing affordability analysis, school quality analysis, and policy reading. Executor skills are designed to be small and composable, so a complex decision can be solved by combining several focused outputs.

3.4 Planner

The Planner acts as the leader in Hi-DeA. It does not directly solve all subtasks. Given the user goal, persona, constraints, data summary, and skill catalog, it creates and monitors a structured task DAG.

The Planner has three main responsibilities. First, it identifies key decision factors. In housing, these may include affordability, school quality, commute, mortgage rate, and policy risk. In trading, they may include price trend, volatility, sentiment, portfolio exposure, and downside risk. Second, it maps these factors to available skills. Third, it defines dependencies between tasks, avoiding a single long prompt that mixes all factors.

The Planner also monitors intermediate outputs. If an executor returns missing fields, low confidence, or weak evidence, the Planner can request repair or ask EvoRoute to choose another executor. After all tasks finish, it integrates evidence accord-

ing to the user skill and planner skill. For example, a risk-averse user gives more weight to downside risk, while a cost-first user gives more weight to cost and efficiency.

3.5 EvoRoute: Dynamic Skill Routing

EvoRoute selects an executor for each subtask. Unlike static function calling, where a task is mapped to a fixed tool by name, EvoRoute chooses among executors with different quality, cost, latency, and reliability.

For a task v and executor skill s, the routing score is:

$$R(v,s) = \lambda_1 \cdot \mathrm{sim}(v,s) + \lambda_2 \cdot \mathrm{schema}(v,s) + \lambda_3 \cdot \mathrm{success}(v,s) - \lambda_4 \cdot \mathrm{cost}(s), \tag{3}$$

where $\mathrm{sim}(v,s)$ is the semantic similarity between task and skill descriptions, $\mathrm{schema}(v,s)$ measures input-output compatibility, $\mathrm{success}(v,s)$ is the historical success score, and $\mathrm{cost}(s)$ is estimated cost.

The semantic score can be computed by embeddings or lightweight text similarity. The schema score checks whether the skill covers required fields. The success score is updated from schema validity, confidence, evidence quality, and Planner feedback. The cost term can represent API cost, runtime, or latency.

This design lets Hi-DeA adapt to context. For example, a finance sentiment task can be routed to a financial sentiment skill instead of a general sentiment skill. A policy task with long documents can be routed to a retrieval-based executor. Over time, EvoRoute can prefer executors that are more reliable for similar tasks.

3.6 Executor and Self-Check

Each executor receives the current task, required context, and outputs of dependency tasks. It runs its skill and returns a structured result with the main analysis, supporting evidence, confidence score, and diagnostics. The evidence field records which data source or dependency output supports the result. The diagnostics field records missing fields, warnings, and validation results.

Before returning the result, the executor performs a self-check. It compares the output with the skill output schema and checks simple constraints. For example, a probability should be between 0 and 1, a shipping ratio should sum to 1, and an affordability score should not be negative. If the output is invalid, the executor attempts local repair. If repair fails, it returns a failure state and the Planner can trigger re-routing.

This local check helps catch errors before they propagate to later tasks and affect the final decision.

3.7 Closed-Loop Correction

Hi-DeA uses a closed loop between the Planner, EvoRoute, and executors. The loop has 3 checks:

- **Schema check:** whether the output format is valid and required fields are present.
- **Evidence check:** whether the output is supported by data, retrieved evidence, or dependency outputs.
- **Decision check:** whether the output helps the final decision and follows the user persona and constraints.

If an output fails the schema check, the system first asks the executor to repair it. If repair fails, EvoRoute selects the next best executor. If the output has weak evidence or low confidence, the Planner can create an additional verification task. For example, if a housing policy result is uncertain, the Planner may request another policy-reading skill or lower the confidence of the final recommendation.

The Success Log stores task type, selected skill, routing score, validation status, Planner score, and error type. EvoRoute uses this log in future routing, making routing adaptive rather than fixed.

3.8 Final Decision Synthesis

After all DAG nodes are executed, the Planner produces the final decision. It combines executor outputs according to the user skill, planner skill, and global constraints. The synthesis step does not simply average all scores. It weights evidence by decision style. In investment, a risk-aware planner can reduce the effect of positive sentiment when volatility is high. In supply chain allocation, a cost-first planner may choose a cheaper route only if delay risk is acceptable. In housing, a family-first persona may give more weight to school quality.

The final output contains three parts: the recommended action, an evidence summary, and a risk summary. The recommended action gives the final choice. The evidence summary explains which executor outputs support the choice. The risk summary states uncertainty, missing information, or possible failure cases. This makes the decision process more transparent and easier to inspect.

Task	Main Dataset	Extra Data	Input Type	Output	Split
Investment	StockNet	Financial PhraseBank	price + text + portfolio	action + weight	chronological
Supply chain	DataCo	weather + fuel price	tabular + time series	mode allocation	chronological
Housing	Zillow + ACS + NCES	mortgage + policy	tabular + retrieved text	ranked options	region/listing

Table 2: Tasks and datasets.

4 Experiment

4.1 Research Questions

This section studies five research questions:

- **RQ1**: Does Hi-DeA improve decision utility over flat or less structured agent baselines?
- **RQ2**: How much does each key component contribute to the final performance?
- **RQ3**: Can EvoRoute assign subtasks to suitable executor skills?
- **RQ4**: Does skill-based schema validation improve output validity and completeness?
- **RQ5**: What is the overhead of Hi-DeA, and can it provide an interpretable decision path?

4.2 Tasks and Datasets

4.2.1 Tasks

We evaluate Hi-DeA on 3 tasks: investment, supply chain allocation, and housing purchase. Investment predicts buy, sell, or hold actions from prices, financial text, and portfolio state. Supply chain allocation chooses shipping-mode ratios from order, inventory, delivery, weather, and fuel features. Housing ranks houses from market, mortgage, school, and policy signals. We use StockNet, Financial PhraseBank, DataCo, Zillow Research, ACS 5-Year, and NCES CCD as data sources (Xu and Cohen, 2018; Malo et al., 2014; DataCo Global, 2019; Zillow Research, 2026; U.S. Census Bureau, 2024; National Center for Education Statistics, 2024).

Figure 2: Preprocessing pipeline. Raw data are cleaned, transformed into features, organized into chronological datasets, and saved as scenario bundles for Hi-DeA.

4.2.2 Data Preprocessing

Figure 2 shows the preprocessing pipeline. Raw data are first collected from the market, text, portfolio, and external sources. We then clean missing or noisy records, build technical, sentiment, and external features, and construct chronological train/validation/test splits. The final outputs are scenario bundles with features, labels, metadata, and split files. For time-sensitive tasks, only past information is used to avoid future leakage.

4.3 Experimental Setup

4.3.1 Compared Baselines

We compare Hi-DeA with flat-agent baselines and several ablated variants. These methods test the effects of DAG planning, dynamic routing, schema validation, feedback, and persona modeling. Table 3 summarizes the component differences.

Method	DAG	Dynamic Route	Schema	Feedback	Persona
Single Agent	–	–	–	–	–
ReAct Agent	–	–	–	–	–
Static Routing	✓	–	✓	–	✓
No-Feedback Hi-DeA	✓	✓	✓	–	✓
No-Schema Hi-DeA	✓	✓	–	✓	✓
No-Persona Hi-DeA	✓	✓	✓	✓	–
Hi-DeA	✓	✓	✓	✓	✓

Table 3: Comparison of methods.

4.3.2 Metrics

We use task-specific metrics for decision utility and general metrics for agent behavior. Investment is evaluated by action quality and risk-adjusted return, supply chain by cost, delay, and service level, and housing by ranking quality and constraint violations. General metrics include schema pass rate, logical completeness, routing accuracy, and cost. Details are in **Appendix B.1**.

4.3.3 Implementation Details

All methods use the same LLM backbone, skill library, and external data. EvoRoute ranks skills by semantic similarity, schema fit, historical success, and cost. We use chronological splits for time-sensitive tasks. Full settings are in **Appendix B.2**.

Method	Investment Utility	Supply Utility	Housing Utility	Schema Pass	Completeness	Route Score ↑
Single Agent	0.00%	12.83%	58.92%	–	–	–
ReAct Agent	77.62%	87.92%	95.38%	–	–	–
Static Routing	82.48%	48.29%	89.37%	–	–	–
Hi-DeA	100.00%	86.25%	100.00%	83.33%	100.00%	1.379
– success feedback	60.39%	75.38%	88.98%	83.33%	80.00%	1.489
– schema validation	78.48%	89.39%	95.38%	–	60.00%	1.434
– user persona	85.37%	86.25%	100.00%	83.33%	100.00%	1.382

Table 4: Main results and ablation study. Dashes indicate metrics that are not defined or not evaluated.

4.4 Main Results

This experiment answers **RQ1**. We compare Hi-DeA with Single Agent, ReAct Agent, and Static Routing on investment, supply chain allocation, and housing purchase. We also report system-level metrics, including Schema Pass, Completeness, and Route Score. Here, Route Score measures the average quality of the selected routes, so higher is better. The results are shown in **Table 4**.

Hi-DeA achieves the best utility on investment and housing, with 100.00% on both tasks. It obtains 86.25% on supply chain allocation, which is slightly lower than ReAct Agent at 87.92%, but much higher than Single Agent and Static Routing. Compared with Single Agent, Hi-DeA improves investment utility from 0.00% to 100.00%, supply utility from 12.83% to 86.25%, and housing utility from 58.92% to 100.00%. Compared with Static Routing, Hi-DeA also improves all three tasks. In addition, Hi-DeA reaches 83.33% Schema Pass and 100.00% Completeness, showing strong structured reliability and decision coverage.

Insight. The results suggest that hierarchical orchestration is useful when the decision requires multiple evidence types. In investment and housing, the system must combine trend, sentiment, risk, preference, and constraint information. A flat agent can miss some factors or mix them in one long context. Hi-DeA reduces this problem by letting the Planner decompose the decision into a DAG and letting executors focus on smaller domain tasks. The supply-chain result is more mixed: ReAct is slightly better in utility, possibly because the current supply-chain task is more rule-like and can be solved well by a single tool-use loop. Although Hi-DeA does not have the highest Route Score, it achieves a better balance between utility, completeness, and reliability.

4.5 Ablation Study

This experiment answers **RQ2**. We remove three components from Hi-DeA: success feedback, schema validation, and user persona modeling. The variant "– success feedback" removes the Success Log from EvoRoute. The variant "– schema validation" removes strict output checking. The variant "– user persona" removes user preference information during decision synthesis.

The results are included in **Table 4**. Removing success feedback causes the largest drop in investment utility, from 100.00% to 60.39%. It also reduces supply utility to 75.38% and housing utility to 88.98%. Its Route Score increases from 1.379 to 1.489, but the final utility drops, suggesting that a higher local routing score does not always lead to better end decisions without feedback calibration. Removing schema validation reduces investment utility to 78.48%, while supply and housing remain high. However, its Completeness drops to 60.00%, showing that the final decision becomes less complete when structured validation is removed. Removing the user persona reduces investment utility to 85.37%, while supply and housing stay close to the full model. Its Route Score is close to Hi-DeA, suggesting that persona mainly affects final synthesis rather than route selection.

Insight. The results show that the three components play different roles. Success feedback aligns route selection with final decision quality, not only with local task-skill match. Without feedback, EvoRoute may select skills with high route scores, but these choices can still be less useful for the final decision. Schema validation improves reliability and completeness, while persona modeling mainly changes how evidence is weighted in final synthesis. This matches the Hi-DeA design: EvoRoute controls task-skill assignment, while the Planner handles trade-off decisions.

4.6 Routing Analysis

This experiment answers **RQ3**. We report three routing statistics in **Table 5**: average Route Score, reroute count, and the number of selected skills. In this experiment, Route Score measures the quality

Scenario	Avg. Route Score	Reroute Count	Selected Skills
Investment	1.379	2	6
Supply Chain	1.581	0	5
Housing	1.483	1	6

Table 5: Routing analysis of EvoRoute. Higher Route Score indicates better route quality.

of selected routes after considering semantic match, schema fit, historical success, and execution cost. Therefore, higher is better. Reroute count shows how often the system changes the executor after weak or invalid outputs. Selected skills show how many different skills are used in each scenario.

Supply chain has the highest average Route Score, 1.581, and no rerouting. This suggests that its subtasks are easier to match with stable executor skills. Housing has a moderate Route Score and one reroute. Investment has the lowest Route Score, 1.379, and the highest reroute count, 2. This suggests that investment routing is more difficult because it combines heterogeneous signals such as price, sentiment, risk, and portfolio state. Investment and housing both use 6 skills, while supply chain uses 5 skills.

Insight. The routing results show that Route Score and reroute count capture different aspects of routing behavior. A high Route Score means the selected skill matches the current subtask well, while reroute count reflects how often correction is needed. Supply chain has high route quality and no rerouting because its tasks are more structured. Investment has lower route quality and more rerouting because its evidence sources are more diverse and uncertain. This supports the design of EvoRoute: dynamic routing is most useful when task-skill matching is uncertain.

4.7 Schema and Reliability Analysis

This experiment answers **RQ4**. We compare Hi-DeA with three ablated variants using four metrics: Schema Pass, Task Success, Completeness, and Repair Rate. Schema Pass measures whether the executor outputs satisfy the required skill schema. Task Success measures whether subtasks finish successfully. Completeness measures whether the final decision covers required factors. The repair rate measures how often a correction is needed.

As shown in **Figure 3**, Hi-DeA reaches 83.33% Schema Pass, 100.00% Task Success, and 100.00% Completeness, with 10.00% Repair Rate. Removing feedback keeps Schema Pass at 83.33%, but

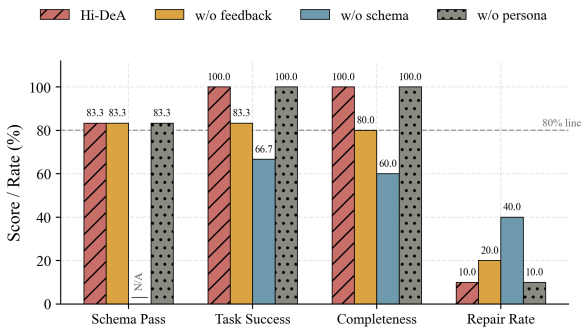

Figure 3: Schema and reliability analysis.

reduces Task Success to 83.33% and Completeness to 80.00%, while increasing Repair Rate to 20.00%. Removing schema validation gives the weakest reliability, with 66.67% Task Success, 60.00% Completeness, and 40.00% Repair Rate. Removing persona keeps the metrics close to Hi-DeA.

Insight. The results match the design of Hi-DeA. Skill contracts and executor self-checks catch invalid or incomplete outputs before final synthesis. Feedback helps EvoRoute avoid weak execution paths, which reduces repair needs. When schema validation is removed, the system can still generate answers, but the answers are less complete and require more correction. Persona modeling has little effect on reliability because it mainly changes final evidence weighting rather than executor validity.

4.8 Case Study

This study illustrates how Hi-DeA makes an interpretable decision in a risk-averse investment scenario. The goal is to generate a one-week stock investment decision for a risk-averse user. As shown in **Figure 4**, the workflow contains six main steps: defining the case goal, collecting data, running parallel analysis, applying a risk gate, constructing the

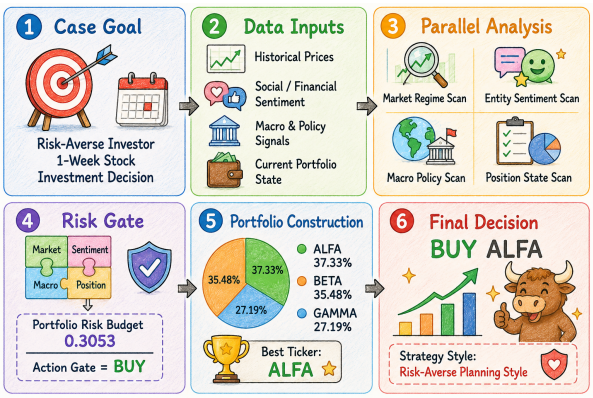

Figure 4: Quant investment case study.

portfolio, and producing the final decision.

Hi-DeA first collects prices, sentiment signals, macro-policy information, and portfolio state. The Planner then decomposes the decision into parallel tasks, including market regime scan, entity sentiment scan, macro policy scan, and position state scan. These outputs are merged by the Risk Gate before portfolio construction. The final decision is produced under a risk-averse planning style, where positive signals are not used alone but are filtered through risk budget and portfolio constraints. This shows the benefit of hierarchical orchestration: the decision is not made from a single long prompt, but from a traceable workflow where each intermediate result has a clear role in the final recommendation.

4.9 Cost and Latency

Hi-DeA introduces extra computation because it decomposes a decision into multiple subtasks and calls several executors. We therefore report calls, token usage, and latency in **Appendix C**. Overall, Hi-DeA has a higher cost than flat baselines, but the extra cost comes with better decision utility, stronger completeness, and more reliable outputs.

5 Related Work

LLM agents and tool use. LLM agents can use language reasoning to call tools and interact with external environments. ReAct combines reasoning and acting in one loop (Yao et al., 2023b), while Toolformer shows that LLMs can learn tool use from self-supervised signals (Schick et al., 2023). WebGPT uses browsing and evidence for question answering (Nakano et al., 2021). Reflexion (Shinn et al., 2023) and Voyager (Wang et al., 2023a) add reflection and memory. These methods often use a flat control loop. Hi-DeA separates planning, routing, execution, and feedback.

Planning and decomposition. Planning helps LLMs solve complex tasks step by step. Chain-of-thought makes intermediate reasoning explicit (Wei et al., 2022). Self-Ask, Plan-and-Solve, Tree of Thoughts, and Least-to-Most prompting further decompose problems into smaller parts or search over multiple reasoning paths (Press et al., 2023; Wang et al., 2023b; Yao et al., 2023a; Zhou et al., 2023). These methods improve reasoning inside one model. Hi-DeA uses decomposition at the system level. The Planner builds a task DAG, and each node can be handled by a suitable executor.

Multi-agent systems. Multi-agent systems assign different roles to different agents. CAMEL studies role-playing agents (Li et al., 2023). AutoGen supports multi-agent conversation for LLM applications (Wu et al., 2024). MetaGPT and ChatDev use role-based agents for software development (Hong et al., 2023; Qian et al., 2023). HuggingGPT connects an LLM controller with many expert models (Shen et al., 2023). Hi-DeA is related to these systems, but it uses explicit skill contracts.

Routing and mixture of experts. Routing is related to mixture-of-experts models, where only part of the model is activated for each input (Shazeer et al., 2017). GShard and Switch Transformer scale this idea to large Transformer models (Lepikhin et al., 2021; Fedus et al., 2022). Recent LLM systems also route queries across models or APIs to balance quality and cost (Chen et al., 2023; Hu et al., 2024). MixLLM formulates LLM routing as a dynamic contextual-bandit problem and considers changing model pools, quality, cost, and latency (Wang et al., 2025). Hi-DeA applies routing at the agent-skill level and selects executors.

Decision making with external data. Many decision tasks need both structured data and text. In finance, StockNet combines historical prices and tweets for stock movement prediction (Xu and Cohen, 2018). Financial PhraseBank and FiQA provide financial sentiment and opinion-mining data (Malo et al., 2014; Maia et al., 2018). Similar issues appear in supply chain and housing decisions, where the system must combine tables, time series, policy text, and user preferences. Hi-DeA handles these heterogeneous sources by assigning different evidence types to specialized executors.

6 Conclusion

We presented Hi-DeA, a hierarchical LLM-agent framework for complex decision making. Hi-DeA decomposes a user goal into a task DAG, routes subtasks to skill-based executors, and integrates schema-checked outputs into a final decision. Experiments show improved utility on two tasks, competitive performance on the third, and strong completeness and reliability. Ablations further show that feedback, schema validation, and persona modeling improve decision quality and stability. These findings suggest that hierarchical orchestration is useful for high-value decisions with multiple evidence sources and explicit trade-off reasoning.

Limitations

Hi-DeA has several limitations. (1) It depends on the quality of the skill library. If a skill description, schema, or quality check is incomplete, EvoRoute may select a weak executor. (2) It depends on the quality and coverage of external data. Missing or noisy financial, supply chain, or housing data can affect the final decision. (3) It introduces extra computation because a decision is decomposed into multiple subtasks and executor calls. (4) Some tasks, especially housing purchase, are hard to evaluate because there may be no single correct answer. (5) In high-stakes domains, Hi-DeA should be used as a decision-support tool rather than the only decision maker to avoid failure.

References

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

DataCo Global. 2019. Dataco smart supply chain for big data analysis. Mendeley Data.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Wang, Jinlin Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. MetaGPT: Meta programming for a multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*.

Qisheng Hu, Alex Bie, Yuxuan Li, Nan Jiang, Ben Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shyam Upadhyay. 2024. RouterBench: A benchmark for multi-llm routing system. In *Proceedings of the 41st International Conference on Machine Learning*.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. GShard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative agents for "mind" exploration of large language model society. In *Advances in Neural Information Processing Systems*.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. WWW'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, pages 1941–1942.

Pekka Malo, Ankur Sinha, Pekka Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *Journal of the Association for Information Science and Technology*, 65(4):782–796.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Ben Chess, and John Schulman. 2021. WebGPT: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

National Center for Education Statistics. 2024. Common core of data. Online data portal.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. HuggingGPT: Solving ai tasks with chatgpt and its friends in hugging face. In *Advances in Neural Information Processing Systems*.

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

U.S. Census Bureau. 2024. American community survey 5-year data. Online data portal.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023a. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023b. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. 2025. MixLLM: Dynamic routing in mixed large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 10912–10922, Albuquerque, New Mexico. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Adam White, Doug Burger, and Chi Wang. 2024. AutoGen: Enabling next-gen llm applications via multi-agent conversation. In *COLM*.

Yumo Xu and Shay B. Cohen. 2018. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*.

Zillow Research. 2026. Zillow research data. Online data portal.

A Skill Contract Example

Hi-DeA uses JSON skill files to describe the abilities and contracts of planners, users, and executors. Each specifies what the skill can do, what input it requires, what output it must return, and how the output should be checked. **Listing 1** shows an example for time-series analysis in the investment.

Listing 1: Example skill contract.

```
{
  "skill_id": "SKILL_FIN_TS_001",
  "name": "time_series_analysis",
  "type": "executor",
  "description": "Analyze historical stock
    prices and extract trend, momentum,
    volatility, and risk signals.",
  "domain": "quantitative_investment",

  "input_schema": {
    "ticker": "string",
    "price_history": {
      "date": "string",
      "open": "number",
      "high": "number",
      "low": "number",
      "close": "number",
      "volume": "number"
    },
    "lookback_window": "integer",
    "user_risk_profile": "string"
  },

  "output_schema": {
    "ticker": "string",
    "trend": "string",
    "momentum_score": "number",
    "volatility_score": "number",
    "risk_level": "string",
    "confidence": "number",
    "evidence": "array"
  },

  "required_context": [
    "historical_prices",
    "user_risk_profile",
    "portfolio_constraints"
  ],

  "domain_logic": {
    "use_case": "Estimate whether a stock has a
    positive trend under the user's profile.",
    "preferred_methods": [
      "moving_average",
      "return_statistics",
      "volatility_estimation"
    ]
  },

  "quality_checks": [
    "momentum_score must be in [0, 1]",
    "volatility_score must be in [0, 1]",
    "confidence must be in [0, 1]",
    "risk_level must be one of: low, med, high",
    "evidence must not be empty"
  ]
}
```

Metric Type	Metric	Scenario	Meaning
Task utility	Action accuracy / return	Investment	quality of trading action
Task utility	Cost / delay / service level	Supply chain	quality of logistics plan
Task utility	Ranking agreement / top-1 match	Housing	quality of house ranking
Reliability	Schema pass rate	All	valid structured outputs
Reliability	Task success rate	All	successful subtask execution
Reliability	Repair rate	All	frequency of correction or rerouting
Reasoning	Logical completeness	All	coverage of decision factors
Routing	Routing accuracy	All	correct skill selection
Efficiency	Cost / latency	All	computation overhead

Table 6: Evaluation metrics. Higher is better except cost, delay, drawdown, violation rate, and latency.

B Evaluation and Implementation Details

B.1 Evaluation Metrics

We use task-specific utility metrics and general agent metrics. Task-specific metrics evaluate whether the final decision is useful in each domain. For investment, we report action accuracy, return, Sharpe ratio, and maximum drawdown. For supply chain allocation, we report total cost, late delivery rate, average delay, and service level. For housing, we report ranking agreement, top-1 match with the expert score, and affordability violation rate.

General agent metrics evaluate the behavior of the agent system itself. The schema pass rate measures whether the executor outputs match the required output schema. Logical completeness measures whether the final answer covers the required decision factors. Routing accuracy measures whether the selected executor matches the task's needs. Cost and latency measure computation overhead, including token usage, number of calls, and runtime.

B.2 Implementation Details

All methods use the same LLM backbone, skill library, and external data to ensure a fair comparison. The skill library is fixed before evaluation. The Planner only sees skill descriptions and schema summaries, and it does not access hidden labels. EvoRoute computes the routing score from semantic similarity, schema compatibility, historical success, and cost. For time-sensitive tasks, we use chronological train, validation, and test splits to avoid future leakage.

Item	Setting
LLM backbone	*Qwen 3 8B*
Max repair attempts	2
Routing candidates	top-k skills
Chronological split	80 / 10 / 10

Table 7: Implementation settings.

For each method, we use the same random seeds when sampling or decoding is needed. We report the mean over repeated runs. The maximum number of repair attempts is limited to avoid unbounded loops.

C Cost and Latency Analysis

Hi-DeA may require more calls than flat baselines because it decomposes a decision into multiple subtasks and executes specialized skills. We therefore report the number of calls, token usage, and latency in **Table 8**. The goal of this analysis is not to show that Hi-DeA is cheaper, but to make the utility-cost trade-off explicit.

Method	Calls	Tokens	Latency ↓
Single Agent	1	423	fast
ReAct Agent	13	6048	moderate
Static Routing	11	5502	moderate
Hi-DeA	24	11824	slow

Table 8: Cost and latency comparison. Hi-DeA uses more calls and tokens because it decomposes the decision into subtasks.

As shown in **Table 8**, Hi-DeA has the largest computation cost. This is expected because it performs planning, routing, executor calls, validation, and final synthesis. In return, it improves decision utility on most tasks and provides stronger output structure. In practice, the cost can be reduced by parallel execution of independent DAG nodes, caching executor outputs, using smaller models for simple skills, and only triggering repair when validation fails.

D Ethical Considerations

Hi-DeA may be used in domains that affect money, housing, or access to services. The system should clearly state uncertainty, expose important risks,

and provide evidence for human review. User data should be protected, especially financial, location, and preference data. For housing tasks, the system should avoid using sensitive attributes in ways that may cause unfair treatment. For trading tasks, the system should not present its output as guaranteed financial advice.

E Artifact Use and Licenses

We use public or research datasets and follow their original access terms. StockNet and Financial PhraseBank are used for research on financial text and stock movement prediction. DataCo is used as a public supply-chain dataset. Zillow Research, ACS 5-Year, and NCES CCD are used through their public data portals or published access interfaces. We do not redistribute the original raw datasets. Our code only provides preprocessing scripts and expects users to download the data from the original sources when required. All derived scenario bundles should be used for research purposes and should respect the licenses and terms of the original data providers. We document the domains, input types, outputs, preprocessing steps, and evaluation files used by each scenario bundle.

F Software and Package Details

The implementation uses Python with common data-processing and evaluation packages, including pandas, NumPy, scikit-learn, and matplotlib. We fix random seeds when sampling or decoding is used.

G AI Assistant Use

We used AI assistants to help draft and revise text, generate code templates, debug implementation issues, and create illustrative figures. The authors reviewed, edited, and verified the final content, experiments, and claims.