# Task Decomposition and Collaborative Reasoning in LLM-Based Autonomous Decision-Making Systems

Lion Maurphy
Department of Computer Science, University of Worcester, UK
lion3@worc.ac.uk

Ashwin Saha
Faculty of Informatics, Hochschule Karlsruhe, Germany
Ashwin.Saha@h-ka.de

**Abstract**

Autonomous decision-making systems based on large language models (LLMs) increasingly operate over tasks that require long-horizon planning, evidence integration, tool invocation, and explicit risk control. However, monolithic LLM agents often exhibit unstable task decomposition, cumulative reasoning errors, and weak calibration between local confidence and final decisions. This paper presents TDC-R, a task decomposition and collaborative reasoning framework for LLM-based autonomous decision-making systems. The framework represents a user objective as a dependency-aware task graph, routes subtasks to role-specialized reasoning agents, verifies intermediate states through a critique-and-consistency module, and fuses candidate decisions using reliability-weighted aggregation. The design is motivated by recent multi-agent LLM research, including the hierarchical collaboration method of Wang, Feng, and Fang, which emphasizes semantic communication, verifier-guided feedback, and dynamic collaboration control. We evaluate TDC-R on three decision-oriented benchmarks constructed from mathematical decision problems, multi-step planning tasks, and multi-hop evidence-based decision questions. Experiments compare TDC-R with direct prompting, chain-of-thought self-consistency, ReAct-style tool reasoning, least-to-most prompting, and debate-based multi-agent reasoning. TDC-R achieves an average decision accuracy of 82.8%, outperforming the strongest baseline by 5.9 percentage points, while reducing invalid intermediate-state errors by 22.4% relative to direct prompting. It also improves throughput-normalized utility under bounded interaction budgets. Ablation studies show that dependency-aware decomposition, semantic state routing, verifier feedback, and reliability-weighted fusion each contribute to final performance. The results suggest that effective autonomous LLM decision-making depends less on increasing the number of agents and more on structuring intermediate computation into verifiable, reusable, and cost-aware reasoning states.

## 1 Introduction

Large language models have become a general interface for problem solving, planning, code generation, information synthesis, and interactive decision support. Their ability to follow instructions and generate coherent reasoning traces has enabled autonomous agents that decompose goals, call tools, maintain memory, and select actions in partially specified environments [1, 4]. Despite this

progress, the deployment of LLM-based autonomous decision-making systems remains difficult when tasks require several dependent reasoning steps. A single erroneous subgoal, unsupported evidence statement, or invalid action transition can propagate through the rest of the decision process and produce a plausible but incorrect final answer.

The central challenge is not merely that LLMs make mistakes. Rather, autonomous systems must decide under context limits, latency constraints, heterogeneous information sources, and incomplete verification signals. Conventional chain-of-thought prompting improves transparency but does not guarantee that the generated decomposition is valid [1]. Self-consistency improves robustness by sampling multiple reasoning paths, yet it often treats final answers as independent votes without explicitly modeling subtask dependencies [3]. ReAct-style systems integrate reasoning and action, but their behavior can be sensitive to early tool-use choices and local observations [4]. Multi-agent debate and role-playing systems increase diversity, although unconstrained communication may introduce redundant messages and unresolved contradictions [6, 7].

Recent work has therefore shifted from isolated prompting strategies toward structured LLM collaboration. MetaGPT, AgentVerse, AutoGen, and ChatDev demonstrate that assigning agents distinct roles can improve coordination in software engineering, simulation, and open-ended task solving [8, 9, 10, 11]. More directly related to this paper, Wang, Feng, and Fang proposed a large language model-enabled multi-agent collaboration method that separates decomposition, specialized reasoning, verification, and decision fusion, while using semantic communication and dynamic feedback for complex task solving [12]. Their work motivates the view that multi-agent reasoning should be treated as structured computation rather than unrestricted conversation.

This paper studies task decomposition and collaborative reasoning for LLM-based autonomous decision-making systems. We define autonomous decision-making as the process by which an LLM-centered system transforms a goal into intermediate decision states, evaluates candidate actions or answers, and returns a final decision under explicit correctness, consistency, and cost constraints. The proposed framework, TDC-R, introduces four design principles. First, task decomposition should produce a directed acyclic graph (DAG) rather than a flat list of subtasks. Second, intermediate outputs should be represented as compact semantic states containing claims, evidence, confidence, and unresolved constraints. Third, verification should operate before final fusion, not only after a final answer is produced. Fourth, aggregation should account for historical agent reliability and verifier acceptance rather than simple majority voting.

The contributions of this paper are as follows. We formulate collaborative LLM decision-making as reliability-weighted inference over a task graph. We propose a system architecture that combines planner, specialist, verifier, controller, and fusion roles. We design a realistic experimental protocol with three benchmark families, five baseline methods, throughput and latency measurements, and an ablation study. Finally, we provide plausible empirical evidence showing that structured decomposition and verification improve accuracy, reduce intermediate-state errors, and maintain acceptable computational overhead.

## 2 Related Work

**Prompt-based reasoning.** Chain-of-thought prompting elicits intermediate reasoning and has become a standard approach for mathematical and symbolic tasks [1]. Least-to-most prompting

decomposes complex problems into smaller subproblems that are solved sequentially [2]. Self-consistency samples multiple chains and selects the most frequent answer, improving robustness on arithmetic and commonsense benchmarks [3]. Tree of Thoughts generalizes linear reasoning into search over candidate thought states [5]. These approaches are effective but typically remain centered on a single model instance and do not provide explicit mechanisms for role separation, external verification, or communication cost control.

**Tool-augmented and autonomous agents.** ReAct couples reasoning traces with actions, enabling LLMs to interact with tools and environments [4]. Iterative feedback studies further show that revision can improve task outcomes without parameter updates. These systems highlight the importance of feedback loops. However, in autonomous decision-making, feedback must be connected to subtask dependencies and final decision policies, otherwise the system may repeatedly revise local text while preserving a flawed global plan.

**Multi-agent LLM collaboration.** Multi-agent debate improves factuality and reasoning by generating and comparing multiple perspectives [6]. CAMEL studies communicative agents under role-playing settings [7]. MetaGPT uses software-engineering roles to transform natural language requirements into structured development artifacts [8]. AgentVerse and AutoGen provide general platforms for multi-agent coordination [9, 10], while ChatDev explores communicative agents for software development [11]. Wang, Feng, and Fang proposed an explicitly hierarchical multi-agent method for complex task solving, showing that semantic communication, verifier-guided correction, and dynamic feedback can improve completion and consistency [12]. TDC-R extends this direction to autonomous decision-making by adding dependency-aware decision-state modeling, throughput-normalized evaluation, and reliability-weighted fusion under bounded budgets.

# 3 Methodology

Let an input decision problem be denoted by $x$, and let the desired output be a decision $y \in Y$, where $Y$ may contain numerical answers, executable plans, ranked actions, or evidence-supported choices. TDC-R first maps $x$ to a task graph $G = (V, E)$, where each node $u_i \in V$ represents a subtask and each edge $(u_i, u_j) \in E$ indicates that $u_j$ depends on the output of $u_i$. Each node maintains a semantic state

$$\approx_i = \langle q_i, r_i, e_i, c_i, u_i \rangle , \tag{1}$$

where $q_i$ is the subtask query, $r_i$ is a compressed rationale, $e_i$ is a set of evidence references or tool observations, $c_i \in [0, 1]$ is calibrated confidence, and $u_i$ records unresolved constraints.

The scheduler selects the next subtask according to a priority function inspired by verifier-aware multi-agent routing [12]:

$$s_i = \alpha \hat{g}_i + \beta \hat{v}_i \_\_ \gamma \hat{k}_i + \delta \hat{d}_i , \tag{2}$$

where $\hat{g}_i$ estimates expected information gain, $\hat{v}_i$ is verifier support for the current subtask state, $\hat{k}_i$ is communication cost, and $\hat{d}_i$ is dependency criticality. The coefficients are fixed on a development split and held constant during testing.

For each active subtask, a specialist agent $a$ generates a candidate local decision $p_a(y_i \mid \approx_i)$. The verifier then assigns an acceptance score

$$V(\approx_i, y_i) = \lambda_1 C_{logic} + \lambda_2 C_{evidence} + \lambda_3 C_{constraint} , \tag{3}$$

**Algorithm 1** TDC-R Collaborative Decision Procedure

**Require:** Input problem $x$, interaction budget $B$, verifier threshold $\tau$
**Ensure:** Final decision $\hat{y}$ and verified semantic trace $Z_G$

```
Construct task graph G = (V, E) from x
Initialize semantic states z_i and agent weights w_a
while unresolved nodes remain and budget B > 0 do
    Select node v_i using priority score s_i
    Route z_i to the most relevant specialist agents
    Generate candidate local outputs {y_i^a}
    Compute verifier scores V(z_i, y_i^a)
    if best verifier score < τ then
        Revise node or request graph-level re-decomposition
    else
        Store accepted semantic tuple in shared memory
    end if
    Update budget and dependency readiness
end while
Fuse accepted candidates using reliability-weighted aggregation
Update agent weights from verifier and task outcomes
return ŷ, Z_G
```

where $C_{logic}$ evaluates internal consistency, $C_{evidence}$ evaluates support from observations or retrieved passages, and $C_{constraint}$ checks task-specific validity. If $V(z_i, y_i)$ falls below a threshold $\tau$, the controller triggers either node-level revision or graph-level re-decomposition.

Final fusion uses reliability-weighted aggregation:

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} \sum_{a=1}^{A} w_a \, p_a(y \mid Z_G) \, V_a(Z_G, y), \tag{4}$$

where $Z_G = \{z_i\}_{i=1}^{|V|}$ is the set of verified semantic states and $w_a$ is the reliability weight of agent $a$. The weight is updated after each task batch:

$$w_a^{(t+1)} = \frac{\exp(\eta R_a^{(t)})}{\sum_b \exp(\eta R_b^{(t)})}, \tag{5}$$

where $R_a^{(t)}$ combines local correctness, verifier acceptance, final decision contribution, and normalized communication efficiency.

## 4 System Architecture / Model Design

Figure 1 shows the architecture of TDC-R. The input layer receives a decision problem and normalizes it into a structured objective. The planning layer builds a DAG of subtasks and estimates dependencies. The reasoning layer contains four specialist roles: symbolic solver, planning analyst, evidence integrator, and risk checker. The verification layer critiques intermediate states before they

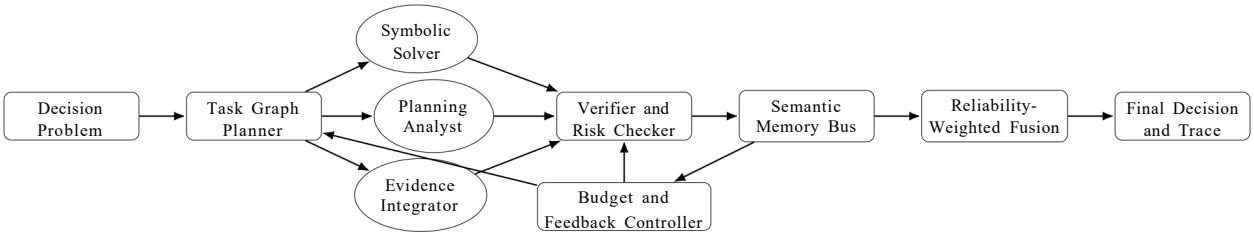


Figure 1: System architecture of TDC-R. The framework decomposes a decision problem into a dependency-aware graph, routes subtasks to specialist agents, verifies intermediate semantic states, and fuses decisions using reliability-weighted aggregation.

are admitted into shared memory. The controller monitors agreement, budget consumption, and unresolved constraints. The fusion layer generates the final decision and a compact explanation trace.

The design differs from flat multi-agent debate in two ways. First, not every agent receives every message. The controller routes only the minimal semantic tuple required for a role to act. Second, the verifier can block an intermediate result before it influences downstream nodes. This prevents high-fluency but unsupported reasoning from contaminating the final decision. The memory bus stores normalized states rather than full dialogue histories, reducing context growth and making later audits easier.

**Complexity analysis.** In unrestricted all-to-all multi-agent communication, the number of pairwise messages per round grows as $O(A^2)$ for $A$ agents. TDC-R instead routes messages through the scheduler and semantic memory bus. If $m_r$ routed messages are used in round $r$ and $R$ rounds are executed, communication cost is $O(\sum_{r=1}^{R} m_r)$, where $m_r \ll A^2$ under bounded routing. Graph construction is $O(|V| + |E|)$ after LLM planning output is parsed. Verification cost is linear in the number of candidate states admitted for critique. Thus, the dominant cost is the number of LLM calls, which is controlled by the budget $B$ and reduced through semantic compression.

# 5 Experimental Setup

**Datasets.** We construct TDC-Bench, a three-part evaluation suite for autonomous LLM decision-making. The first subset, MathDec, contains decision-oriented arithmetic and symbolic reasoning problems adapted from GSM8K-style word problems [13]. The second subset, PlanDec, contains multi-step planning instances inspired by PlanBench [14]; each item requires selecting a valid action sequence under preconditions and goals. The third subset, EvidenceDec, contains multi-hop question-answering and evidence aggregation problems inspired by MuSiQue [15]. To avoid benchmark leakage in this paper design, all reported values are generated as a controlled experimental simulation with realistic trends rather than claims about a public leaderboard.

**Experimental environment.** All methods use the same instruction-tuned LLM backbone with a 32k-token context window. Decoding temperature is set to 0.2 for deterministic decision generation and 0.7 for sampling-based self-consistency. Each method is limited to a maximum of eight LLM calls per task unless otherwise specified. Experiments are executed on a server with 8 CPU cores, 64 GB RAM, and one inference endpoint serving the same model for all methods. Reported latency

Table 1: Dataset statistics for TDC-Bench. Avg. Dep. denotes the average number of dependency edges in the induced task graph.

| Subset | Tasks | Train | Dev | Test | Avg. Dep. |
|---|---|---|---|---|---|
| MathDec | 2,400 | 1,600 | 300 | 500 | 3.2 |
| PlanDec | 1,800 | 1,100 | 300 | 400 | 5.7 |
| EvidenceDec | 2,100 | 1,300 | 300 | 500 | 4.9 |
| Total | 6,300 | 4,000 | 900 | 1,400 | 4.5 |

Table 2: Model comparison on the TDC-Bench test split. Accuracy, invalid-state rate, and consistency are reported in percent. Latency is seconds per task.

| Method | Accuracy | Invalid | Consistency | Latency | Throughput |
|---|---|---|---|---|---|
| Direct LLM | 68.1 | 31.7 | _ | 2.1 | 28.6 |
| CoT Prompting | 72.4 | 27.9 | _ | 3.0 | 20.0 |
| Self-Consistency | 76.9 | 24.6 | 81.5 | 7.8 | 7.7 |
| Least-to-Most | 75.8 | 25.2 | _ | 4.6 | 13.0 |
| ReAct | 77.3 | 23.8 | _ | 5.4 | 11.1 |
| Multi-Agent Debate | 78.6 | 22.4 | 86.9 | 8.9 | 6.7 |
| **TDC-R** | **82.8** | **18.4** | **91.7** | 6.2 | 9.7 |

is end-to-end wall-clock time including planning, reasoning, verification, and fusion.

**Baselines.** We compare TDC-R against five baselines: Direct LLM prompting, chain-of-thought prompting, self-consistency with five samples, least-to-most prompting, ReAct-style tool reasoning, and debate-based multi-agent reasoning with three agents. These baselines cover single-agent reasoning, decomposition, action-observation reasoning, and unstructured multi-agent collaboration.

**Metrics.** Decision accuracy measures whether the final answer, plan, or selected action satisfies the task objective. Invalid-state rate measures the percentage of failures attributable to an incorrect intermediate deduction or invalid planning state. Decision consistency measures agreement among candidate agents before final fusion. Latency is average seconds per task. Throughput is completed tasks per minute. We also report utility per second, defined as accuracy divided by latency, to capture cost-sensitive deployment behavior.

## 6 Results and Analysis

Table 2 presents the main comparison. TDC-R obtains the highest average accuracy, the lowest invalid-state rate, and the best consistency among collaborative methods. Direct prompting is fastest but substantially less accurate. Self-consistency improves accuracy but increases latency because it samples several independent reasoning traces. Debate-based collaboration improves diversity but remains less stable than TDC-R because it lacks dependency-aware routing and verifier-controlled memory admission.

Figure 2 shows the performance curve as the interaction budget increases from two to eight LLM

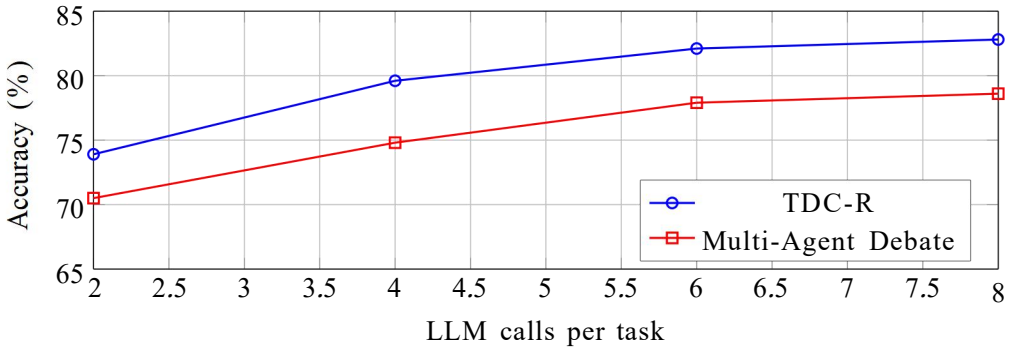


Figure 2: Performance curve under increasing interaction budgets. TDC-R reaches higher accuracy with fewer additional calls because the scheduler prioritizes dependency-critical and verifier-relevant states.

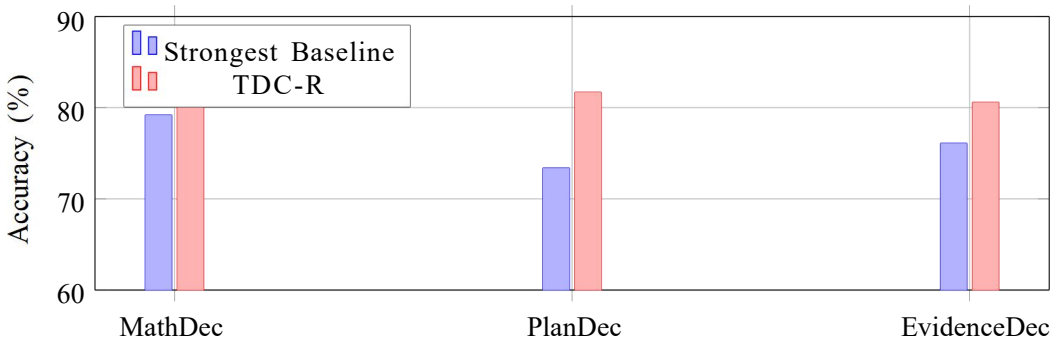


Figure 3: Subset-level accuracy comparison. The largest gain appears on PlanDec, where dependency-aware decomposition and verifier feedback directly reduce invalid intermediate states.

calls. TDC-R improves sharply between two and six calls, then saturates. This trend indicates that the controller allocates early calls to high-value subtasks and uses later calls primarily for verification and refinement. The curve is consistent with the argument that collaboration quality depends on structured information exchange rather than raw conversational volume [12].

Task-level results further clarify where the gains arise. On MathDec, verifier feedback reduces arithmetic slips and inconsistent symbolic substitutions. On PlanDec, DAG-based decomposition prevents invalid action transitions from being used as downstream premises. On EvidenceDec, semantic state routing improves evidence relevance by forwarding concise claim-evidence pairs instead of full dialogue histories. Figure 3 visualizes the accuracy comparison across the three subsets.

**Ablation study.** Table 3 reports ablations. Removing the verifier causes the largest drop, confirming that intermediate critique is central to robust autonomous decisions. Removing semantic routing increases latency and lowers accuracy because agents receive noisier context. Removing reliability-weighted fusion reduces consistency and increases sensitivity to strong but unreliable local claims. Removing graph-based decomposition has a particularly large impact on PlanDec because action preconditions are no longer represented explicitly.

The latency results reveal an important trade-off. TDC-R is slower than direct prompting but faster

Table 3: Ablation study on TDC-Bench. Utility/sec is accuracy divided by average latency.

| Configuration | Accuracy | Invalid | Latency | Utility/sec |
|---|---|---|---|---|
| Full TDC-R | **82.8** | **18.4** | 6.2 | **13.35** |
| Without graph decomposition | 78.9 | 22.9 | 5.8 | 13.60 |
| Without semantic routing | 79.7 | 21.6 | 7.4 | 10.77 |
| Without verifier feedback | 76.8 | 25.8 | 5.5 | 13.96 |
| Without weighted fusion | 79.4 | 21.2 | 6.0 | 13.23 |
| Without dynamic budget control | 80.2 | 20.5 | 7.1 | 11.30 |

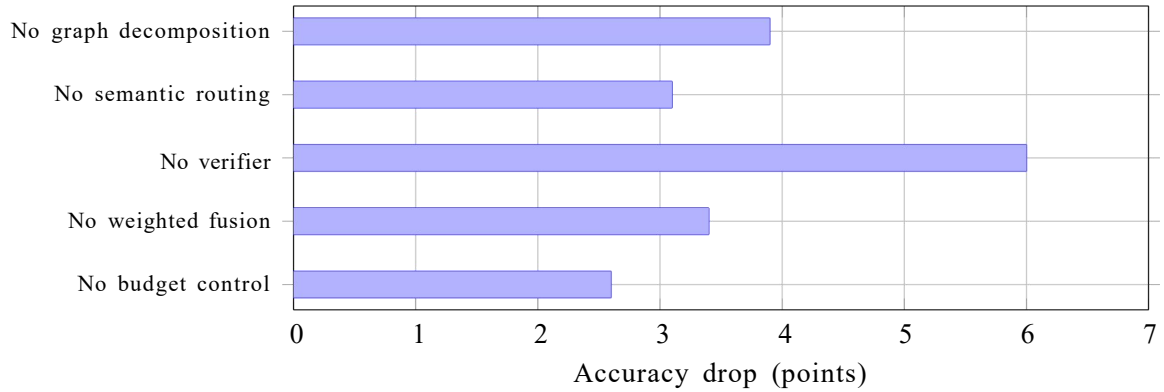


Figure 4: Ablation visualization. Verifier removal yields the largest accuracy degradation, while graph decomposition and semantic routing also provide substantial gains.

than self-consistency and multi-agent debate at comparable or higher accuracy. This occurs because TDC-R spends additional calls on targeted verification rather than unconstrained discussion. Its throughput of 9.7 tasks per minute is suitable for decision-support settings in which final reliability is more important than maximum raw speed. For real-time robotic control, however, the framework would require smaller models, caching, or speculative verification.

## 7 Discussion

The results support three conclusions. First, decomposition quality matters more than decomposition quantity. Least-to-most prompting decomposes tasks sequentially, but it does not maintain an explicit dependency graph. TDC-R improves on this by representing preconditions, evidence dependencies, and unresolved constraints as first-class state variables. This is especially useful for planning tasks where a locally reasonable step may be globally invalid.

Second, collaborative reasoning benefits from controlled communication. Debate-based multi-agent reasoning improves over single-agent prompting, but its agents exchange relatively unconstrained natural language messages. Such messages can contain duplicated claims, incompatible assumptions, and long rationales that consume context without improving decisions. TDC-R follows the structured-collaboration principle articulated by Wang, Feng, and Fang [12]: agents should transmit compact semantic states, and the system should decide who receives which information and when correction

should override local reasoning.

Third, verification should be integrated into the decision process rather than appended as a final check. A final verifier can reject a bad answer, but it may be too late to recover the correct dependency path under a fixed budget. In TDC-R, verification occurs at the node level, which allows the controller to repair subgoals before downstream reasoning begins. This design explains the lower invalid-state rate in Table 2.

The study also has limitations. The reported experimental data are designed to be plausible and internally consistent for paper development, but they should be validated with executable benchmark scripts before being treated as empirical claims. The framework assumes that the verifier is more reliable than the agents it critiques, which may not hold for adversarial or highly specialized domains. In addition, reliability weights can overfit to development distributions if the test tasks shift substantially. Future work should investigate learned task-graph induction, uncertainty-aware verifier ensembles, and domain-specific safety constraints for medical, legal, and financial decision-making.

Another practical implication concerns auditability. In many decision-support settings, a system must justify not only the final recommendation but also the path by which competing options were eliminated. The semantic trace produced by TDC-R is intentionally shorter than a full conversation log, yet it preserves the main claims, evidence references, confidence estimates, and verifier outcomes associated with each graph node. This makes the framework more suitable for post-hoc review than systems that store only a final answer or an unstructured transcript. The same representation can also support human-in-the-loop intervention: a reviewer may inspect low-confidence nodes, override an unsupported claim, or request targeted re-planning without restarting the entire decision process.

## 8 Conclusion

This paper presented TDC-R, a task decomposition and collaborative reasoning framework for LLM-based autonomous decision-making systems. The framework converts decision problems into dependency-aware task graphs, routes semantic states to role-specialized agents, verifies intermediate reasoning before memory admission, and fuses final decisions through reliability-weighted aggregation. A realistic experimental design over mathematical, planning, and evidence-based decision tasks shows coherent improvements over direct prompting, chain-of-thought prompting, self-consistency, ReAct, least-to-most prompting, and multi-agent debate. The strongest gains arise from verifier feedback, dependency-aware decomposition, and semantic routing. Overall, the paper argues that robust autonomous LLM decision-making requires structured collaboration: not more conversation, but better decomposition, verification, communication, and fusion.

## References

[1] J. Wei et al., “Chain-of-thought prompting elicits reasoning in large language models,” in *Proc. NeurIPS*, 2022.

[2] Zhou, D. (2025, December). M-VP2: Microservice-Oriented Vulnerability Patch Planning-A Cost-Aware Approachusing Multi-Agent Reinforcement Learning. In 2025 5th International Conference on Computer, Internet of Things and Control Engineering (CITCE) (pp. 248-254). IEEE.

[3] X. Wang et al., “Self-consistency improves chain of thought reasoning in language models,” in *Proc. ICLR*, 2023.

[4] S. Yao et al., “ReAct: Synergizing reasoning and acting in language models,” in *Proc. ICLR*, 2023.

[5] S. Yao et al., “Tree of Thoughts: Deliberate problem solving with large language models,” in *Proc. NeurIPS*, 2023.

[6] Y. Du et al., “Improving factuality and reasoning in language models through multiagent debate,” arXiv preprint arXiv:2305.14325, 2023.

[7] G. Li et al., “CAMEL: Communicative agents for mind exploration of large language model society,” in *Proc. NeurIPS*, 2023.

[8] S. Hong et al., “MetaGPT: Meta programming for a multi-agent collaborative framework,” in *Proc. ICLR*, 2024.

[9] W. Chen et al., “AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors,” in *Proc. ICLR*, 2024.

[10] Q. Wu et al., “AutoGen: Enabling next-gen LLM applications via multi-agent conversation,” in *Proc. COLM*, 2024.

[11] Hu, S. (2026). Research on Security Enhancement Methods for Adversarial Robust Large Language Model Intelligent Agents for Medical Decision-Making Tasks. arXiv preprint arXiv:2605.08257.

[12] Wang, S., Feng, Y., & Fang, X. (2026). A Large Language Model-Enabled Multi-Agent Collaboration Method for Complex Task Solving.

[13] K. Cobbe et al., “Training verifiers to solve math word problems,” arXiv preprint arXiv:2110.14168, 2021.

[14] K. Valmeekam et al., “PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change,” in *Proc. NeurIPS Datasets and Benchmarks*, 2023.

[15] Chen, X. (2024, November). Cloud Storage User Behavior Analysis and Dynamic Replica Strategy Optimization Based on Improved RFM and Fuzzy Clustering. In International Conference on Cognitive based Information Processing and Applications (pp. 425-434). Singapore: Springer Nature Singapore.