An Approach for Systematic Decomposition of Complex LLM Tasks

Anonymous EACL submission

Abstract

Large Language Models (LLMs) suffer from reliability issues on complex tasks, as existing decomposition methods are heuristic and rely on agent or manual decomposition. This work introduces a novel, systematic decomposition framework that we call `Analysis of CONstraint-Induced Complexity` (`ACONIC`), which models the task as a constraint problem and leveraging formal complexity measures to guide decomposition. On combinatorial (SAT-Bench) and LLM database querying tasks (Spider), we find that by decomposing the tasks following the measure of complexity, agent can perform considerably better (10-40 percentage point).

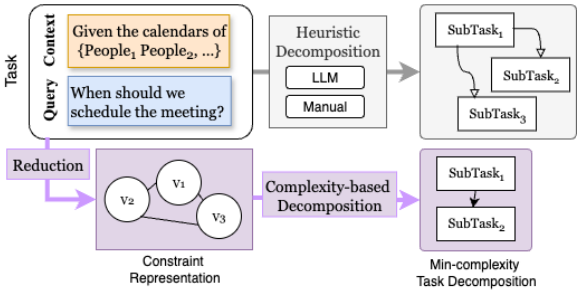


Figure 1: Given a scheduling task, existing approaches use heuristic LLM or manual task decomposition methods (top row). Our framework reduces the task into a constraint satisfaction problem that allows for a systematic decomposition that minimizes the problem complexity (bottom row).

1 Introduction

Large Language Models (LLMs) have demonstrated impressive competence across a wide range of reasoning, programming, and problem-solving tasks. Yet, when faced with *complex tasks* that require deep multi-step reasoning or combinatorial search, even state-of-the-art models often fail to produce correct results in a single forward pass.

A growing body of work addresses this limitation through *task decomposition*. Instead of solving a task monolithically, these methods break it into smaller, more tractable subtasks. One popular line of work, starting from chain-of-though (Wei et al., 2022), is to use LLMs for decomposition (Yao et al., 2023; Khot et al., 2023; Pourreza and Rafiei, 2023; Chen et al., 2024). Other methods rely on domain experts to decompose the task into workflows and provide access to tools that shoulder parts of the task (Wang et al., 2024; Singh et al., 2024). We summarize these methods as the grey path in Figure 1.

While decomposition seeks to break more complex tasks into workflows of simpler subtasks, existing approaches are largely heuristic. When is a task "complex"? How should it be decomposed? A principled measure of task complexity would enable systematic decomposition strategies and the ability to study tasks of comparable difficulty, and provide guidance on when tools are needed.

In this paper, we introduce a formal complexity framework for LLM tasks by reducing them into *constraint satisfaction problems* (specifically, 3-SAT). We use properties of the induced constraint graph (graph size and treewidth) as measures of task complexity. Building on these measures, we propose a decomposition method based on (Bodlaender, 1998) that minimizes subtask complexity under this formalization, yielding decompositions that preserve global satisfiability while maximizing local solvability. Along the purple path in Figure 1, we first reduce the task—modeled as a context that describes a set of constraints and a query that must reason over the constraints—into a formal constraint satisfiability problem. We then decompose the constraint problem and construct a workflow over the subtasks defined for each subproblem.

We use SAT-BENCH (Wei et al., 2025b) and the NL2SQL SPIDER (Yu et al., 2019) benchmark to study this framework. We find that the task measures define **frontiers of difficulty** that separate

tasks easily solvable by LLMs from those that are nearly impossible without structural assistance. In addition, decomposition based on complexity improves task accuracy by $9 - 40\%$ as compared to chain-of-thought decomposition.

2 Reduction to Constraint Representation

At a high level, our reduction happens in two steps. The first models the agent's task using a state-based framework that captures the interaction between the agent and its observations of the system environment. This enables us to model the agent's planning as a satisfiability (PaS) problem (Selman et al.), and further reduce this to a standard constraint satisfiability problem. In our experiments, we develop automatic reduction procedures for two benchmarks.

2.1 Reduction to Planning Problems

The state based framework of agent operations (e.g. Agent Path Finding (Surynek et al., 2016)) can be formulated as PaS problems (Selman et al.). The framework transfers world states and actions into finite state abstractions

$$P = \langle F, A, I, G \rangle,$$

where F is a finite set of propositional *fluents* describing world facts, A is a finite set of *actions*, and I, G determines the initial and goal fluents. For a given action a, $\mathcal{P}(a) \subseteq F$ determines the preconditions, $\mathcal{A}(a) \subseteq F$ determines the fluents that become true, and $\mathcal{D}(a) \subseteq F$ determines the fluents that become false. An agent task becomes determining a sequence of actions $\langle a_1, \ldots, a_k \rangle$ such that, starting from I, each a_i is applicable (its preconditions hold), and executing its effects (adding and deleting fluents) transfers current state to a new state. The plan is valid if the final state after executing all actions satisfies all goals in G.

Consider a toy meeting scheduling problem:

> *Motivation Example — Meeting Scheduling.*
> Alice needs to meet with Bob and Charlie separately and asks an agent to schedule both meetings. The system records show:
>
> - Alice: ["Alice is available at morning in office r", "Alice is available in the afternoon at office t"]
> - Bob: ["Bob is available in the afternoon at office t"]
> - Charlie: ["Charlie is available at morning in office r", "Charlie is available in the afternoon at office t"]
>
> **Task.** How should the agent compose the invitation emails to Alice–Bob and Alice–Charlie?

In the toy example above, let us define $x_p^{(t,\ell)}$ to be a Boolean action variable indicating that person p is invited to location ℓ at time t, where $\mathcal{P}$, $\mathcal{T}$, and $\mathcal{L}$ denote the global set of people, times, and locations, respectively. For each person p, let $\mathcal{A}_p \subseteq \mathcal{T} \times \mathcal{L}$ be the set of feasible (time, location) pairs. For each pair (p, p'), define the pairwise feasible set $\mathcal{F}_{p,p'} = \mathcal{A}_p \cap \mathcal{A}_{p'}$ as given in the system record.

We model the state as the set of value assignments $\{x_p^{(t,\ell)} \mid p \in \mathcal{P}, (t,\ell) \in \mathcal{T} \times \mathcal{L}, x_p^{(t,\ell)} \in \{0,1\}\}$, indicates whether participant p attends slot (t,ℓ). An action consists of assigning some $x_p^{(t,\ell)} = 1$.Then a target set can be constructed by requiring that for each desired meeting pair $(p,p') \in \{(A,B),(A,C)\}$, exactly one common feasible slot is chosen where both p and p' are present. Formally, defining meeting at ℓ, t for (p,p') to be $M_{p,p'}^{(t,\ell)} := x_p^{(t,\ell)} \wedge x_{p'}^{(t,\ell)}$, the goal condition can be written as:

$$G: \bigwedge_{(p,p')} \mathrm{EO}\{M_{p,p'}^{(t,\ell)} \mid (t,\ell) \in \mathcal{F}_{p,p'}\} \wedge \bigwedge_{(t,\ell)} \neg\big(M_{A,C}^{(t,\ell)} \wedge M_{A,B}^{(t,\ell)}\big), \tag{1}$$

where EO denotes *Exactly One*, which can be easily represented with at least one, and at most one. All $x_p^{(t,\ell)}$ initialized to be False.

The agent must produce a sequence of actions that comply with the preconditions and effects defined in $P = \langle F, A, I, G \rangle$, thus transiting the states toward a goal-satisfying state.

Our objective is to find a problem decomposition strategy that provide observations over the subset of the problem. It should ensure that the agent's local consistency over observed subgraphs guarantees global satisfiability of G.

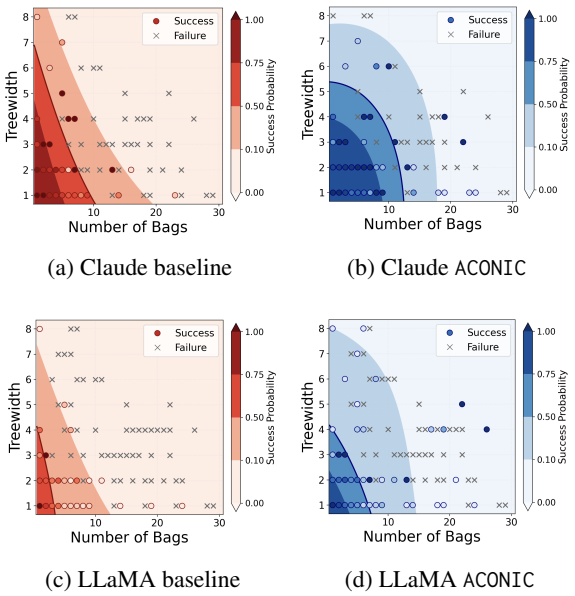
2.2 Reduction to Constraint Satisfaction

Given the planning-as-satisfaction formulation $P = \langle F, A, I, G \rangle$, we can reduce the problem to a constraint satisfaction instance by encoding the preconditions $f_p \in \mathcal{P}(a)$, add effects $f_a \in \mathcal{A}(a)$, and delete effects $f_d \in \mathcal{D}(a)$ of each action as Boolean dependent constraints between fluents and actions. Additionally, we include a k-wise constraint involving all variables in $\mathcal{P}(a) \cup \mathcal{A}(a) \cup \mathcal{D}(a)$ to enforce frame consistency and prevent anomalous models (Selman et al.) where fluents change truth values without any triggering action. By additionally encoding the goal states G as constraints, solv-

ing the resulting CSP is equivalent to solving the original PaS problem.

In constraint processing theory, tree decomposition (Bodlaender, 1998) on a constraint graph provides desirable guarantees of consistency and modularity: when traversing along the tree, any variable assignment made within one subgraph does not affect previously satisfied clauses except through shared boundary variables (which is always included in current subproblem). We therefore chose tree decomposition to separate the CSP into subproblems, the completion of which lead to global success of the problem. This ensures that local consistency over the bags implies global consistency of the full CSP solution.

(Bodlaender, 1998) show existence of a tree decomposition $\mathcal{D}^*$ with the smallest maximum bag size (treewidth), which characterizes the intrinsic problem complexity. Leveraging this prior work, ACONIC uses minimal tree decomposition to partition reasoning tasks into the minimal locally consistent subtasks while preserving global satisfiability.



(a) Claude baseline (b) Claude ACONIC

(c) LLaMA baseline (d) LLaMA ACONIC

Figure 2: Our complexity measures define **frontiers of difficulty**. SOTA models like Claude shifts the boundaries towards the right, while ACONIC's decomposition consistently pushes the frontiers towards more complex problems.

3 Experiments

We now evaluate our framework against chain-of-thought decomposition (Wei et al., 2022) on two benchmarks: SAT-based story problems (SAT-bench) and natural language to SQL (Spider).

3.1 SAT-Bench

SAT-Bench (Wei et al., 2025a) uses (i) an underlying SAT problem to construct (ii) a natural-language story describing the same constraints, and (iii) an alignment from story entities to their SAT representations. Instead of asking the LLM for an immediate satisfiability judgment, we prompt it to produce value assignments step-by-step to ensure a complete understanding of the assignment process. In this process, the state–action dependency is modeled by casting SAT as a PaS problem, where the goal state is defined by the original SAT-Bench CNF representation.

At each round of variable assignment, the agent is provided with the story, the variable to SAT mapping rules, and a set of observations. We tested agent with two different configurations of observations. In the chain-of-thought baseline, the agent observes all condition observations. In the tree-decomposition setup, the agent instead observes all conditions belonging to the same subproblem induced by the decomposition.

We evaluated Llama-3-70B on all tasks and Claude3.5-Sonnet on half the tasks (randomly sampled). Figure 2 plots the success/failure of each task by its complexity measures, and exhibits frontiers of difficulty beyond which tasks are too difficult or too simple. For the baseline, there appears to be a fixed "total task complexity" evidenced by the trade-off between problem treewidth and number of bags (left column). In contrast, ACONIC decomposition pushes the frontier outward and is able to successfully complete more complex tasks. Overall, ACONIC increases the task completion rate from $49.3\% \rightarrow 58.1\%$ using Claude, and $21.5\% \rightarrow 36.5\%$ using LLaMA—a $9 - 15\%$ improvement on both models.

3.2 Natural Language to SQL

Natural language to SQL (NL2SQL) tasks translate a natural language query over a database into a desired SQL query. A major challenge is constructing the join condition to connect tables referenced in the query. For example, the schema in Figure 3 contains 4 tables about singers and concerts. Although the NL query only refers to singers and concerts, the LLM needs to infer that `singer_in_concern` is needed for a valid join path.

The complexities of both the query (the number of tables referenced) and the database schema

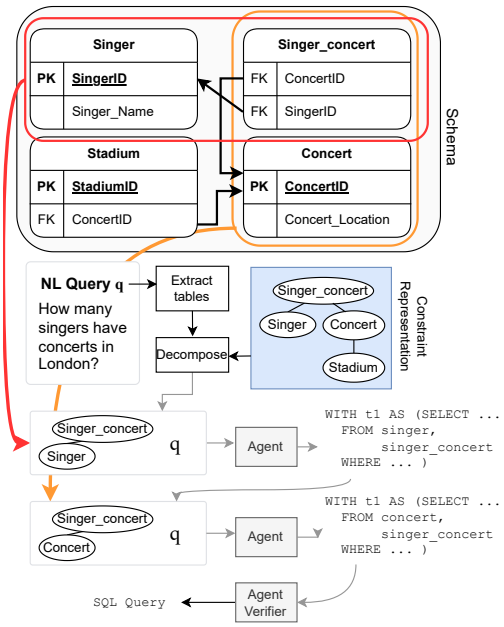


Figure 3: Instead of giving the full database schema and NL query q to an agent, ACONIC decomposes the NL task into two subtasks, each given a subset of the database schema, q, and the output of the previous subtask if any.

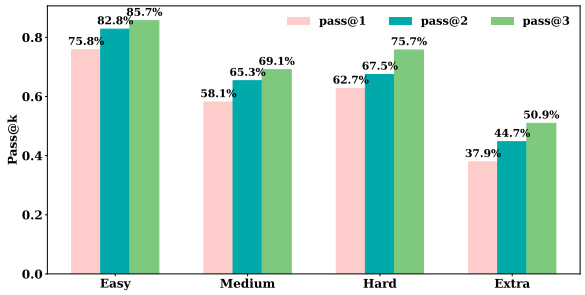


Figure 4: ACONIC result on Llama3-70B

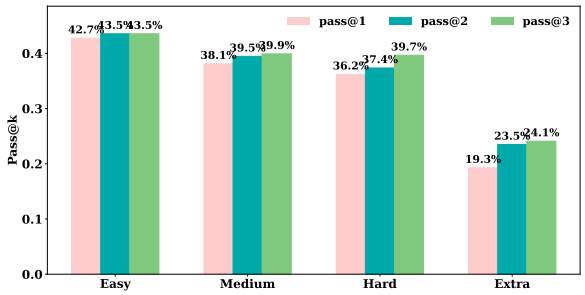


Figure 5: Baseline CoT on Llama3-70B

(number of tables and their foreign key graph) affect the agent's ability to construct queries with the appropriate join graphs. In addition, database theory (Gottlob et al., 2001) already models a database as a constraint graph (tables are nodes, and foreign keys are edges) and a query as a subgraph; this representation has also been used for NL2SQL approaches (Wang et al., 2019). Thus, we use NL2SQL to study decomposition.

We use the popular Spider NL2SQL benchmark dataset (Yu et al., 2019). It contains hundreds of databases, each containing up to 37 tables, 90 foreign keys, and potentially over 100 columns. Each task consists of a database schema S, the natural language (NL) query q, and a ground-truth SQL query q^*. The original benchmark submits (S, q) to the LLM, which can result in incorrect or invalid join conditions. Given the tables referenced in q, ACONIC decomposes the schema into subgraphs with minimized maximal complexitieså and constructs a workflow (grey arrows). For instance, the first subtask gives the agent q and the schemas for subgraph `singer-singer_concern` (red arrow), and asks it to construct the appropriate `WITH` clause. The next subtask does the same with the second subgraph (orange arrow), but is also provided the output from the previous subtask. A verification agent cleans up minor errors to build the final SQL query.

Figure 4 and Figure 5 respectively report accuracy for ACONIC and the CoT baseline. Each bar reports the best of 1, 2, or 3 agent trajectories. We find that while increasing the number of trajectories slightly improves the accuracy, decomposition using ACONIC increases the accuracy over CoT by $\approx 30 - 40\%$ across the conditions.

4 Conclusions

ACONIC is a principled framework for decomposing complex LLM tasks by reducing them to constraint satisfaction problems and quantifying complexity via treewidth. Unlike heuristic methods, ACONIC minimizes local complexity while preserving global satisfiability. Across SATBench and Spider, this yielded **frontiers of task difficulty** that define LLM reasoning limits. Complexity-based decomposition improved completion rates by up to 15% on SAT-bench and up to 40% on SPIDER. These results suggest a path towards theoretically grounded, reliable multi-step LLM systems.

5 Limitations

ACONIC provides a principled framework to analyze and reduce task complexity, but it does not yet constitute a fully autonomous decomposition or reasoning system. It is also not intended to be a complete solution for general task decomposition, but rather as a theoretical and empirical examination of how *constraint-induced complexity* affects the problem-solving ability of LLMs.

Our focus has been on evaluating how complexity-guided decomposition impacts performance relative to heuristic baselines such as chain-of-thought prompting, rather than on direct comparison with other theoretically grounded or learning-based decomposition frameworks. For this reason, we also do not claim that our final decomposition system is superior to existing solutions to the benchmark tasks. For instance, SAT-bench can be directly solved using a constraint solver. Similarly, the NL2SQL tasks can be readily solved using simple path-finding algorithms over the database schema.

The tasks that we evaluated can be conveniently modeled as constraint satisfiability problems. Although many other practical problems, such as deadlock detection in databases, resource scheduling in operating systems, task placement in distributed systems, and control/data-flow construction in agentic programming, can be formulated as constraint satisfiability problems, they often cannot be logically represented completely, either due to question ambiguity, lack of transparancy of the agent actions, or fuzzy contextual information. In these cases, future work might study hybrid decomposition approaches that mix logical and commonsense constraints in a task.

References

Hans L Bodlaender. 1998. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45.

Zhiruo Chen, Simeng Qiu, Victor Zhong, and Percy Liang. 2024. Divide-and-conquer prompting for complex reasoning. *arXiv preprint arXiv:2402.05359*.

Georg Gottlob, Nicola Leone, and Francesco Scarcello. 2001. Hypertree decompositions: A survey. In *International Symposium on Mathematical Foundations of Computer Science*, pages 37–57. Springer.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2306.04179*.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348.

Bart Selman et al. Planning as satisfiability.

Amanpreet Singh, Chenglei Si, Kurt Shuster, Jing Xu, and Jason Weston. 2024. Agent–computer interfaces: An approach for task-grounded agent evaluation. In *NeurIPS*.

Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. 2016. Efficient sat approach to multi-agent path finding under the sum of costs objective. In *Proceedings of the twenty-second european conference on artificial intelligence*, pages 810–818.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. *arXiv preprint arXiv:2402.01030*.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. 2025a. Satbench: Benchmarking llms' logical reasoning via automated puzzle generation from sat formulas.

Anjiang Wei, Yuheng Wu, Yingjia Wan, Tarun Suresh, Huanmi Tan, Zhanke Zhou, Sanmi Koyejo, Ke Wang, and Alex Aiken. 2025b. Satbench: Benchmarking llms' logical reasoning via automated puzzle generation from sat formulas. *arXiv preprint arXiv:2505.14615*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Pengfei Cao, Karthik Narasimhan, and Yingyu Song. 2023. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.