HYPERGRAPH-GROUNDED ROLLOUT SEARCH FOR MULTI-CONSTRAINT LLM PLANNING

Anonymous authors
Paper under double-blind review

ABSTRACT

While large language models (LLMs) have made significant strides in handling complex reasoning tasks, existing planning methods still face substantial challenges with multiple fine-grained constraints. Fundamentally, relying on unstructured textual context to track dynamic resources and global constraints makes resource selection difficult to audit and failures hard to localize. We propose HyperPlan, which introduces an instance-level directed heterogeneous hypergraph as a formal, structured planning representation. Through explicit operator families, the hypergraph integrates constraints, actual candidate resources, temporal states, and partial plans, capturing high-order dependencies and multi-step state transitions. Rather than generating the final plan, the LLM serves as a semantic interface, parsing open-ended requests into formal entities and n-ary relations. These elements enable hypergraph-based Monte Carlo rollouts and lexicographic feasibility assessment. Failed simulations are attributed to specific hypergraph elements; bounded failure memory guides later search without overriding hard constraints. On the TravelPlanner test set, HyperPlan achieves 100.00% delivery and 82.40% final pass rate with GPT-4-Turbo. With the same backend on NATURAL PLAN, it achieves 75.00% exact match on Trip Planning, 100.00% accuracy on Meeting Planning, and 62.00% solve rate on Calendar Scheduling. The system boundary is clear: the LLM handles language parsing, while explicit hypergraph search makes verifiable decisions.

1 INTRODUCTION

Constraint awareness and adherence are foundational to reasoning and problem solving in real-world environments. Despite the progress LLMs have made in reasoning and tool use (Wei et al., 2022; Yao et al., 2023a;b), their reliability remains limited on practical tasks involving complex, multifaceted constraints. Recent systematic evaluations likewise find that language models struggle with action change and long-horizon plan validity (Valmeekam et al., 2023; Kambhampati et al., 2024). LLMs rely heavily on unstructured textual context to track dynamic resources and global constraints. This implicit state tracking conflates language understanding with combinatorial optimization, rendering resource choices difficult to audit and failures difficult to localize.

Direct generation requires the LLM to maintain all states and constraints in natural-language context. Early requirements may be forgotten, and local decisions can compromise global feasibility. Figure 2 illustrates this in multi-day travel planning: a direct plan violates a minimum-stay requirement, omits essential Day 2 details, and reuses the outbound flight as the return flight.

Multi-agent frameworks introduce feedback loops (Zhang et al., 2025a; Choi et al., 2026; Bui et al., 2026), yet LLM-driven plan synthesis can leave the causes of violations opaque. Formal verification improves rigor (Hao et al., 2025), but requires a solver-specific representation that may not directly preserve instance-level candidate resources.

Figure 1 summarizes the resulting closed loop. We introduce HyperPlan around three principles: (1) the LLM extracts schema-constrained entities and n-ary relations instead of generating the final plan; (2) these elements populate an instance-level directed heterogeneous temporal hypergraph for Monte Carlo rollouts and lexicographic feasibility assessment; and (3) failed simulations are attributed to graph elements and recorded in bounded memory to guide subsequent search without overriding hard constraints.

Our contributions are threefold. First, we decouple language parsing from planning decisions using an instance-level heterogeneous hypergraph. Second, we develop feasibility-first Monte Carlo roll-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

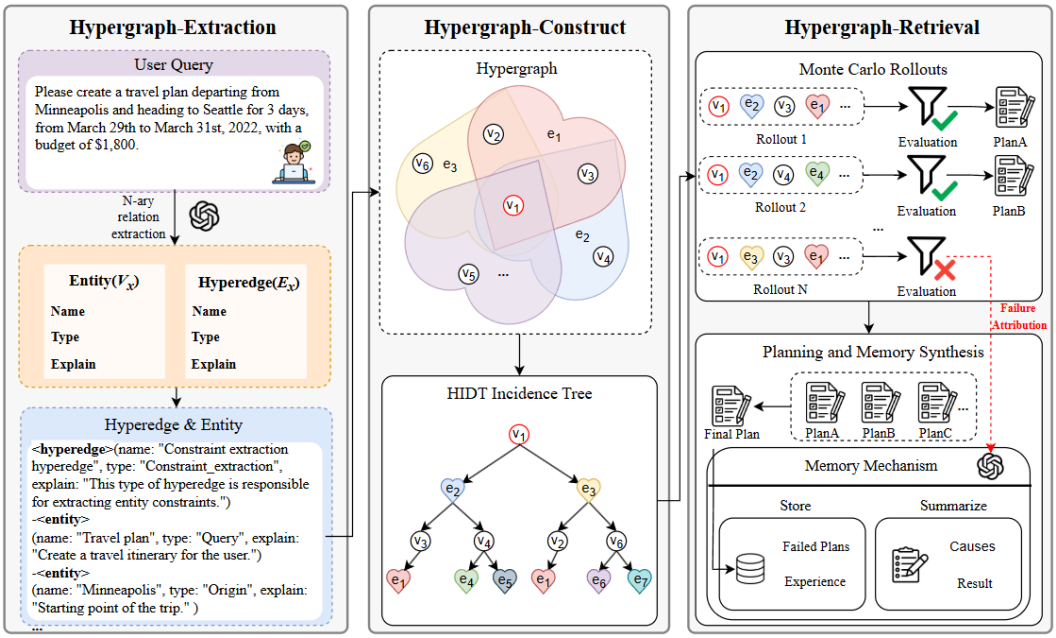


Figure 1: Overview of HyperPlan. The LLM parses an open-ended request into typed entities and n-ary relations; an instance-level directed heterogeneous hypergraph grounds those objects and their joint dependencies; feasibility-aware Monte Carlo rollouts construct partial plans; lexicographic verification attributes failures to graph elements; and bounded memory feeds diagnoses back into subsequent search. The final plan is produced only after delivery and hard-constraint checks succeed.

out search with explicit failure attribution that guides exploration while preserving hard constraints. Third, we evaluate the framework on TravelPlanner and NATURAL PLAN: it attains 100.00% delivery and 82.40% final pass on the TravelPlanner test set and 75.00%, 100.00%, and 62.00% on the three NATURAL PLAN subtasks, respectively.

2 Related Work

2.1 Planning with Language Agents

Language agents using chain-of-thought, search-based reasoning, and reason-act frameworks improve multi-step decision making (Wei et al., 2022; Yao et al., 2023a;b; Shinn et al., 2023). Language models have also been used as zero-shot planners and as interfaces to embodied feedback loops (Huang et al., 2022; 2023). In long-horizon tasks with multiple constraints, however, they primarily maintain planning states, constraints, and candidate decisions in natural-language context. This unstructured state tracking is susceptible to context drift, constraint forgetting, and inconsistent local decisions. HyperPlan externalizes these objects in an instance-level hypergraph, relieving the LLM of the need to memorize the entire plan or make final resource decisions.

2.2 Multi-Agent Collaboration

Multi-agent frameworks use role assignment, feedback-based revision, and iterative critique. Multi-Constraint Collaborative Planning assigns hierarchical subtasks to collaborative agents (Zhang et al., 2025a). ATLAS uses dynamic constraint management, critique, and interleaved search for travel planning (Choi et al., 2026); HiMAP-Travel employs hierarchical planning and monitoring to maintain global constraints (Bui et al., 2026). Although these methods improve task organization, LLM-driven plan synthesis can make it difficult to track resource selection and violations systematically. HyperPlan establishes a clearer boundary: the LLM provides structured semantic interpretation, while a search mechanism selects the final route and schedule.

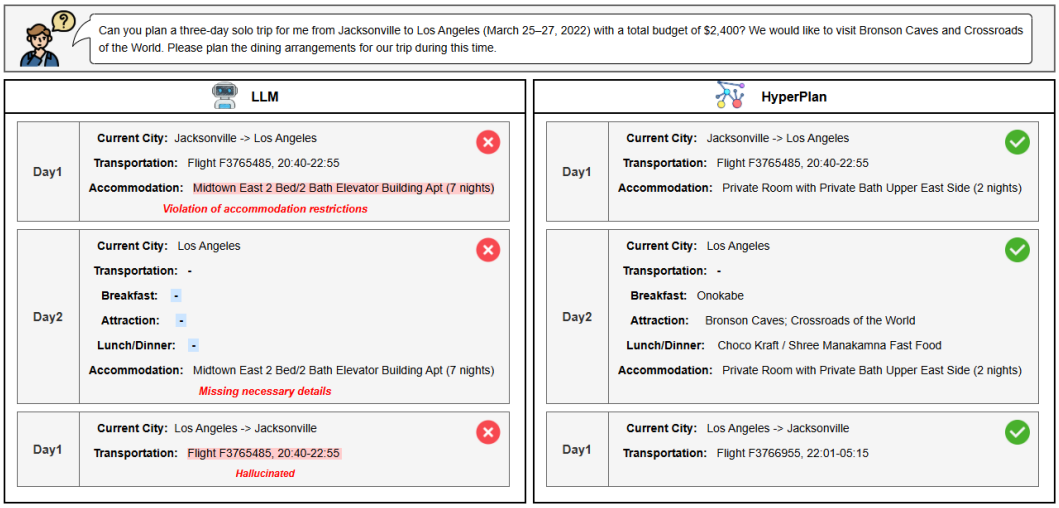


Figure 2: A TravelPlanner example contrasting direct LLM generation with structured planning. The direct plan violates a minimum-stay restriction, omits Day 2 details, and reuses the outbound flight for the return journey. The structured plan resolves these illustrated errors. The source figure uses the earlier label "HG-Plan-RS" for HyperPlan.

2.3 Structured and Symbolic Planning

Tree of Thoughts and Graph of Thoughts enable exploration and backtracking among candidate reasoning trajectories (Yao et al., 2023a; Besta et al., 2024), but primarily organize reasoning paths rather than planning objects and global constraints. Other work couples LLMs with executable programs or symbolic solvers (Gao et al., 2023; Chen et al., 2023; Pan et al., 2023; Huang et al., 2023). The LLM-Modulo perspective explicitly argues for separating language-model proposals from external planning and verification components (Kambhampati et al., 2024); PlanBench provides a complementary evaluation protocol for action change and plan validity (Valmeekam et al., 2023). Such approaches may require substantial formalization to handle large candidate-resource spaces and multi-step state dependencies. HyperPlan structures the planning objects themselves: resources, constraints, partial plans, and feasibility checks become nodes and hyperedges, enabling joint resource composition and precise error localization.

3 Method

3.1 Problem Definition and Framework Overview

We formulate planning as a constrained sequence-construction problem, following the separation between state-transition operators and explicit goal tests used in classical planning (Ghallab et al., 2004). Given a natural-language query q, external candidate resources $\mathcal{R}$, and a finite planning horizon $\mathcal{T} = \{1, \ldots, H\}$, the system constructs a plan

$$y = (a_1, \ldots, a_H), \tag{1}$$

where each action a_t describes a selected resource, a schedule, and a corresponding executable operation. A valid plan must satisfy delivery requirements (e.g., destinations, duration, and output format) and hard constraints (e.g., temporal consistency, resource availability, budget, and feasible transfers). Soft preferences such as cost, convenience, and diversity rank plans that satisfy the hard constraints.

Let $D(y)$ denote satisfaction of delivery requirements and $C_k(y)$ the kth hard constraint. The constrained objective is

$$y^* \in \arg\max_{y \in \mathcal{Y}} S(y) \quad \text{subject to} \quad D(y) = 1, \quad C_k(y) = 1 \ \forall k, \tag{2}$$

where $\mathcal{Y}$ is the candidate-plan space and S scores user preferences. The explicit feasibility conditions prevent a high preference score from compensating for a violated hard constraint.

The central challenge is keeping every intermediate decision consistent with entities, temporal states, and global constraints. HyperPlan therefore separates semantic parsing, structured plan construction, feasibility search, and final verification. The LLM extracts structured relationships; search and verification produce the final plan.

3.2 Semantic Parsing and Relation Extraction

The parser converts q into a structured planning description: entities such as origins, destinations, activities, time intervals, and candidate resources; typed temporal, spatial, budgetary, and dependency relations; and explicit constraints such as required visits, prohibited transitions, and duration limits. It outputs typed records

$$Z = \{z_i\}_{i=1}^{N}, \qquad z_i = (\text{name}_i, \text{type}_i, \text{attributes}_i). \tag{3}$$

For example, an origin record can name Minneapolis, while a temporal-constraint record specifies an interval or duration. Relations may involve more than two entities, allowing a requirement to bind a location, activity, time, and resource simultaneously.

The LLM performs semantic parsing rather than generating the final plan. This division prevents decisions from being implicitly encoded in free-form text and allows the extracted entities and constraints to be used by verification and search. Parsing errors can nevertheless propagate downstream.

3.3 Instance-Level Hypergraph Representation

We represent a planning instance x as a typed directed hypergraph

$$G_x = (V_x, E_x, \tau_V, \tau_E), \tag{4}$$

where V_x and E_x are node and hyperedge sets, and τ_V and τ_E assign their types. Nodes represent query entities, physical resources such as flights or hotels, temporal states, and intermediate planning states. A directed hyperedge

$$e = (I_e, O_e), \qquad I_e \subseteq V_x, \quad O_e \subseteq V_x, \tag{5}$$

connects a set of prerequisites to a set of resulting objects or states. Using O_e for outputs distinguishes an edge's outputs from the overall node set.

The representation retains joint dependencies. For example, choosing a feasible transfer can depend simultaneously on the current location, departure time, transportation resource, and remaining budget. A lodging decision may depend on arrival and departure dates, room rules, capacity, and a minimum-stay restriction. Hyperedges encode these dependencies as one typed relation, rather than treating every pairwise compatibility check as sufficient for the joint decision.

An action is admissible only when its associated transition is consistent with the current state and the constraints checked at that step. Global conditions are checked on complete candidates. The same graph supplies diagnostic units: a failed requirement can be linked to its constraint node, selected resource nodes, and relevant transition edges. Explicit representation improves traceability, while the quality of feasibility judgments still depends on the coverage and correctness of the encoded checks.

3.4 Hypergraph-Based Monte Carlo Rollouts

After constructing G_x, the searcher produces candidate trajectories

$$\Pi_m = (v_{m,1}, v_{m,2}, \ldots, v_{m,T_m}) \tag{6}$$

and their corresponding plans y_m. The use of Monte Carlo sampling is related to bandit-based tree search (Kocsis & Szepesvári, 2006), but our admissibility filter and lexicographic evaluator operate on grounded planning objects rather than unconstrained language trajectories. At state s_t, it chooses among admissible successors

$$\mathcal{A}(s_t) = \{v \in V_x : \text{the transition from } s_t \text{ to } v \text{ satisfies local hard checks}\}. \tag{7}$$

The transition refers to the applicable hyperedge and all its prerequisites; it is not an unconstrained pairwise hop. Rollouts explore combinations of grounded resources and temporal assignments. Local admissibility filters the search space, but a locally valid prefix may still fail a global budget or cross-step requirement when completed.

Each completed candidate receives a utility

$$J(\Pi_m) = \lambda_{\text{del}} J_{\text{del}}(\Pi_m) + \lambda_{\text{soft}} J_{\text{soft}}(\Pi_m) - \lambda_{\text{cost}} J_{\text{cost}}(\Pi_m), \tag{8}$$

subject to $J_{\text{hard}}(\Pi_m) = 1$ and satisfaction of delivery requirements. Utility is used only within the feasible candidate set. Consequently, a high soft-preference score cannot rescue a plan that violates a checked hard constraint.

A rollout ends when it yields a complete candidate or reaches a state with no admissible successor. Search stops when a candidate satisfying the requirements is found under the stopping rule or the rollout budget is exhausted. The best feasible candidate encountered is retained; otherwise the system returns failure information. Monte Carlo exploration is stochastic, so exact repeatability requires fixed sampling settings and random seeds in addition to deterministic checks.

3.5 Verification, Failure Attribution, and Memory

The evaluator tracks four components,

$$L(y) = \big(L_{\text{del}}(y), L_{\text{hard}}(y), L_{\text{soft}}(y), L_{\text{cost}}(y)\big). \tag{9}$$

The comparison is feasibility-first: delivery and hard requirements take precedence, with preferences and costs used to distinguish admissible candidates. This ordering separates the validity decision from scalar utility optimization.

For failed candidates, an attribution map connects violations to responsible nodes and edges:

$$\phi : \mathcal{V}_{\text{viol}} \longrightarrow \mathcal{P}(V_x) \times \mathcal{P}(E_x). \tag{10}$$

Here $\mathcal{P}$ is the power set, and the product allows a violation to reference both node and edge sets. A temporal conflict, for example, can be attributed jointly to a temporal node and a transportation hyperedge. The corresponding graph pattern and reason are stored in bounded memory,

$$\mathcal{M} = \{(g_j, r_j)\}_{j=1}^{K}, \qquad K \le K_{\max}. \tag{11}$$

Subsequent rollouts consult this memory to down-rank recurring failed choices. A transition can be pruned when its invalidity is established for the applicable state and constraints; a failure in one context alone is not evidence that the same resource is invalid in every context. Memory changes exploration, not the definition of feasibility.

The system's validation is relative to its encoded requirements and checkers. It is distinct from passing an external benchmark evaluator. This distinction matters empirically: perfect delivery does not imply a 100% final pass rate, and residual failures can reflect incomplete constraint recovery, imperfect checks, or unsuccessful search. Attribution makes those failures inspectable but does not, on its own, prove their elimination.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate HyperPlan on two planning benchmarks. TravelPlanner evaluates executable multi-day itineraries using realistic candidate resources (Xie et al., 2024). It tests budget constraints, transportation connections, accommodation rules, dining preferences, and commonsense constraints. We use its validation set of 180 instances and official test set of 1,000 instances. Related benchmarks such as Flex-TravelPlanner and ChinaTravel broaden the same evaluation question to flexible preferences and a different geographic/data distribution (Oh et al., 2025; Shao et al., 2024). NATURAL PLAN covers Trip Planning, Meeting Planning, and Calendar Scheduling (Zheng et al., 2024). These tasks require consistent, conflict-free decisions regarding participant availability, location transitions, duration requirements, and candidate time slots.

Baselines and base models. We compare four representative planning paradigms. Direct generation produces the whole plan in one interaction, relying on natural-language context to maintain states and constraints. Two-stage generation drafts a high-level outline before filling in resource details. Multi-Agent Collaboration (PMC) assigns hierarchical subtasks to collaborating language agents and refines the plan through critique and feedback (Zhang et al., 2025a). FormalVerify compiles natural-language constraints into a solver-specific representation and verifies plan feasibility (Hao et al., 2025). The TravelPlanner comparisons report GPT-3.5-Turbo and GPT-4-Turbo backends. NATURAL PLAN additionally includes LLM, SwarmAgentic (Zhang et al., 2025b), and PATIENT-VoT (Cho et al., 2025); Table 2 preserves the reported backend of each row.

Evaluation metrics. Following the benchmark conventions, TravelPlanner uses Delivery Rate, Commonsense Pass Rate (micro/macro), Hard Constraint Pass Rate (micro/macro), and Final Pass Rate, the percentage of plans satisfying all commonsense and hard constraints. For NATURAL PLAN, we report Trip Planning exact match (EM), Meeting Planning accuracy, and Calendar Scheduling solve rate. All table entries are percentages.

4.2 TRAVELPLANNER RESULTS

Validation. As shown in Table 1, HyperPlan achieves a 100.00% delivery rate with both backends. Its final pass rate reaches 98.89% with GPT-3.5-Turbo and 99.44% with GPT-4-Turbo, exceeding the listed generative and multi-agent baselines on this split. We use validation for prompt/schema development, checker calibration, and early debugging; it is not treated as an estimate of held-out benchmark performance.

Test. On the larger 1,000-instance test set (Table 1), delivery remains at 100.00%. HyperPlan achieves 78.60% final pass with GPT-3.5-Turbo and 82.40% with GPT-4-Turbo. The wider gap between delivery and final constraint satisfaction compared with validation demonstrates the increased difficulty of this split. HyperPlan exceeds the listed direct, two-stage, and PMC systems, although FormalVerify attains a higher GPT-4-Turbo final pass rate of 90.20%.

Table 1: TravelPlanner validation and test results. CS: commonsense; HC: hard constraints. All values are percentages. Bold identifies column maxima, and underline identifies the second highest within each backend.

Dataset	Base Model	Method	Delivery ↑	Commonsense ↑		Hard Constraint ↑		Final Pass ↑
				Micro	Macro	Micro	Macro	
Validation (#180)	GPT-3.5-Turbo	Two-stage	86.70	54.00	0.00	0.00	0.00	0.00
		Direct	**100.00**	60.20	4.40	11.00	2.80	0.00
		PMC	91.67	63.54	1.67	0.71	1.67	0.00
		HyperPlan	**100.00**	**100.00**	**100.00**	**99.29**	**98.89**	**98.89**
	GPT-4-Turbo	Two-stage	89.40	61.10	2.80	15.20	10.60	0.60
		Direct	**100.00**	80.40	17.20	47.10	22.20	4.40
		FormalVerify	95.00	95.00	95.00	95.70	98.90	93.30
		PMC	96.67	87.29	43.33	47.14	46.67	33.33
		HyperPlan	**100.00**	**100.00**	**100.00**	**99.76**	**99.44**	**99.44**
Test (#1000)	GPT-3.5-Turbo	Two-stage	91.80	57.90	0.00	0.50	0.60	0.00
		Direct	**100.00**	59.50	2.70	9.50	4.40	0.60
		PMC	84.09	57.51	1.95	1.37	1.30	0.65
		HyperPlan	**100.00**	**99.39**	**95.50**	**87.07**	**78.60**	**78.60**
	GPT-4-Turbo	Two-stage	93.10	63.30	2.00	10.50	5.50	0.60
		Direct	**100.00**	80.60	15.20	44.30	23.10	4.40
		FormalVerify	91.50	91.40	91.10	**91.30**	**90.20**	**90.20**
		PMC	97.40	91.46	50.00	53.96	45.12	42.68
		HyperPlan	**100.00**	**99.40**	**95.50**	88.95	82.40	82.40

Validation versus test. The two splits answer different questions. Validation measures whether the representation, extraction schema, and checkers work on a development distribution; the held-out test measures robustness after those choices are fixed. Delivery is unchanged at 100.00% for both backends, but final pass decreases by 20.29 percentage points (98.89% to 78.60%) for GPT-3.5-Turbo and by 17.04 points (99.44% to 82.40%) for GPT-4-Turbo. This gap is informative rather than a reason to merge the splits: it exposes harder resource combinations and longer-range interactions that are underrepresented in validation. We therefore report the validation results in the main paper for transparency and retain the test set as the primary generalization claim.

4.3 NATURAL PLAN TRANSFER

To test whether the representation-search boundary generalizes beyond travel, we evaluate HyperPlan on the three NATURAL PLAN subtasks. With GPT-4-Turbo, the framework reaches 75.00% exact match on Trip Planning, 100.00% accuracy on Meeting Planning, and a 62.00% solve rate

on Calendar Scheduling (Table 2). These results support the transfer of the representation-search architecture across distinct planning domains. SwarmAgentic with GPT-4-Turbo reaches a higher Calendar Scheduling result, so the gains are not uniform across tasks and backends.

Table 2: NATURAL PLAN results (%). TP: Trip Planning EM; MP: Meeting Planning accuracy; CS: Calendar Scheduling solve rate. Rows are grouped by reported backend. Bold denotes column maxima, and underline identifies the second highest within each backend.

Method	Backend	TP	MP	CS
GPT-3.5-Turbo				
SwarmAgentic	GPT-3.5-Turbo	<u>13.10</u>	<u>23.00</u>	<u>28.00</u>
LLM	GPT-3.5-Turbo	7.30	19.10	19.90
HyperPlan	GPT-3.5-Turbo	**74.38**	**100.00**	**60.00**
GPT-4-Turbo				
LLM	GPT-4-Turbo	31.10	47.00	41.20
PATIENT-VoT	GPT-4-Turbo	<u>32.50</u>	38.30	48.90
SwarmAgentic	GPT-4-Turbo	13.10	<u>56.00</u>	**82.00**
HyperPlan	GPT-4-Turbo	**75.00**	**100.00**	<u>62.00</u>

The backend grouping makes two patterns visible. First, HyperPlan is stable across the two reported Turbo backends on Meeting Planning and retains a large Trip Planning advantage over the other GPT-3.5-Turbo rows. Second, the Calendar Scheduling maximum belongs to SwarmAgentic with GPT-4-Turbo, so the transfer result is not uniformly superior and should not be read as a claim that the representation dominates every backend or subtask.

4.4 Cross-Benchmark Discussion

The results support a narrower and more useful claim than universal superiority. HyperPlan consistently makes the requested plan deliverable, while the harder question is whether every grounded resource and cross-day dependency is valid. The validation/test comparison shows that this distinction remains important when the candidate pool becomes more difficult: delivery is preserved, but final-pass robustness degrades. In practice, a system that exposes this gap and attributes the residual failures is more actionable than one that reports only a single aggregate success rate.

The NATURAL PLAN transfer study tests whether the representation-search boundary survives when travel-specific resources are replaced by meetings and calendars. The answer is mixed but informative. Strong Trip Planning and Meeting Planning scores suggest that typed entities, temporal relations, and feasibility-first selection are reusable abstractions. The lower Calendar Scheduling score, together with the stronger SwarmAgentic baseline, indicates that parser coverage and backend behavior still matter. This motivates reporting backend-specific results rather than averaging across models.

Our tables also expose what is not yet measured. We do not claim lower latency, token use, or monetary cost because the current record does not include runtime traces or repeated-run variance. A complete reproducibility report should add the random seed, rollout budget, candidate-pool size, memory capacity, model version, and per-instance cost. These quantities determine whether the additional structure is practical at larger horizons, and they are orthogonal to the accuracy comparisons reported here.

4.5 Ablation Study

To isolate the three main components, we conduct ablations on the TravelPlanner test set using GPT-3.5-Turbo (Table 3). The full method is listed first as the reference, followed by variants sorted by their decrease in Final Pass. Removing the instance-level hypergraph causes the largest observed drop (32.40 points), followed by removing the LLM (21.00 points) and replacing the searcher (12.20 points). These are sensitivity diagnostics under one backend, not independent causal effect estimates: each variant changes interactions among extraction, representation, and search, and no repeated-run variance is available. The results nevertheless identify the hypergraph as the most consequential component in this configuration.

Table 3: Ablations on TravelPlanner test (1,000 instances), using GPT-3.5-Turbo. All values are percentages. Bold denotes column maxima.

	Delivery	CS		HC		Final	Δ Final (pp)
Variant		Micro	Macro	Micro	Macro	pass	vs. full
HyperPlan (full)	**100.00**	**99.39**	**95.50**	**87.07**	**78.60**	**78.60**	—
w/o Hypergraph	**100.00**	96.00	88.60	64.00	46.20	46.20	-32.40
w/o LLM	**100.00**	**99.39**	**95.50**	74.32	57.60	57.60	-21.00
w/o Searcher	**100.00**	98.78	90.70	78.55	66.40	66.40	-12.20

4.6 Constraint Failure Analysis

The diagnostic heatmap in Figure 3 shows that commonsense checks such as city-route consistency and restaurant diversity are satisfied almost perfectly. Most errors arise from hard constraints, particularly accommodation requirements, which account for more than half of the reported failure incidents. Longer planning horizons amplify constraint coupling: failure rates rise as itineraries extend from three days to five and seven days, exposing empirical limitations in real-world planning.

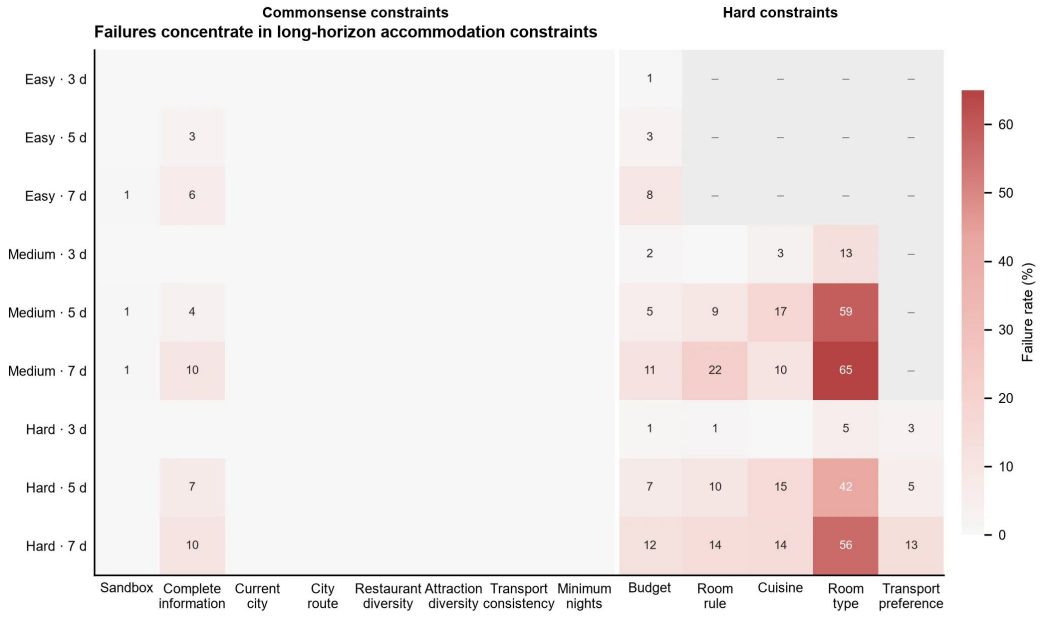


Figure 3: Constraint-failure heatmap reproduced from the source analysis. Rows group instances by difficulty and trip length; columns distinguish commonsense and hard constraints. Cell annotations and color scale report failure rates (%). Dashes and blank cells are retained as supplied. Accommodation-related hard constraints dominate the displayed pattern, particularly on longer trips.

4.7 Discussion and Limitations

The failure heatmap and the ablation table point to a common bottleneck: the system is usually able to produce a syntactically complete itinerary, but global constraints that couple multiple days and resources remain fragile. Accommodation is a representative case because a locally plausible room choice can violate a minimum-stay rule only after the complete horizon is known. The hypergraph makes this dependency explicit, while failure attribution records the offending room, stay edge, and affected temporal state. This improves diagnosis, but it does not guarantee that a different rollout will find a repair within a finite budget.

The proposed memory should therefore be interpreted as a bounded search heuristic rather than a learned world model. It can down-rank a recurring invalid pattern and preserve the reason for rejection, but it must remain scoped to the state and constraints under which the failure was observed.

Over-generalizing a failure would incorrectly prune a resource that is valid in another context. This is why the evaluator, rather than the memory, remains the authority for hard feasibility.

There are also important evaluation boundaries. The internal checker validates the encoded requirements and candidate bindings; it is not an independent proof that the benchmark's external evaluator will agree. The validation/test gap and the higher FormalVerify test score make this limitation concrete. Future work should compare checker decisions against external evaluator traces, report confidence intervals over repeated seeds, and include a dedicated memory ablation. These additions would separate better representation from better search luck and quantify how much attribution actually improves recovery.

Finally, the current study focuses on offline candidate pools and English planning requests. It does not address stale resources, API failures, multilingual extraction, privacy-sensitive itineraries, or interactive changes after a plan has been delivered. Extending the typed schema and hyperedge operators to those settings is promising, but would require new data and experiments rather than extrapolation from the present tables.

5 Conclusion

We presented HyperPlan, a framework that decouples semantic interpretation from constrained planning decisions. The LLM converts an open-ended request into typed entities and relations, while a directed heterogeneous hypergraph represents resources, constraints, temporal states, and partial plans. Feasibility-first Monte Carlo rollouts construct candidates, and structured validation attributes failures to graph elements so bounded memory can guide subsequent exploration. The TravelPlanner and NATURAL PLAN results demonstrate strong delivery and constraint adherence across several settings. Remaining failures involving accommodation rules and longer horizons identify directions for further improvement.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17682–17690, 2024. URL https://ojs.aaai.org/index.php/AAAI/article/view/29720.

The Viet Bui, Wenjun Li, and Yong Liu. HiMAP-Travel: Hierarchical multi-agent planning for long-horizon constrained travel. *arXiv preprint arXiv:2603.04750*, 2026. URL https://arxiv.org/abs/2603.04750.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023. URL https://arxiv.org/abs/2211.12588.

Ikhyun Cho, Changyeon Park, and Julia Hockenmaier. The power of bullet lists: A simple yet effective prompting approach to enhancing spatial reasoning in large language models. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 3047–3057, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.findings-naacl.166/.

Jihye Choi, Jinsung Yoon, Jiefeng Chen, Somesh Jha, and Tomas Pfister. ATLAS: Constraints-aware multi-agent collaboration for real-world travel planning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mIYGiBf9Pm. Poster.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023. URL https://proceedings.mlr.press/v202/gao23f.html.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, San Francisco, CA, 2004.

Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. Large language models can solve real-world planning rigorously with formal verification tools. 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*, pp. 3434–3483, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.naacl-long.176/.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 9118–9147. PMLR, 2022. URL https://proceedings.mlr.press/v162/huang22a.html.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205, pp. 1769–1782. PMLR, 2023. URL https://proceedings.mlr.press/v205/huang23c.html.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B. Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 22895–22907. PMLR, 2024. URL https://proceedings.mlr.press/v235/kambhampati24a.html.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pp. 282–293. Springer, 2006.

Juhyun Oh, Eunsu Kim, and Alice Oh. Flex-TravelPlanner: A benchmark for flexible planning with language agents. *arXiv preprint arXiv:2506.04649*, 2025. URL https://arxiv.org/abs/2506.04649.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, Singapore, 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-emnlp.248/.

Jie-Jing Shao, Xiao-Wen Yang, Bo-Wen Zhang, Baizhi Chen, Wen-Da Wei, Guohao Cai, Zhenhua Dong, Lan-Zhe Guo, and Yu-feng Li. ChinaTravel: A real-world benchmark for language agents in chinese travel planning. *arXiv preprint arXiv:2412.13682*, 2024. URL https://arxiv.org/abs/2412.13682.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/7a92bcdede88c7afd108072faf5485c8-Abstract.html.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837, 2022. URL https://arxiv.org/abs/2201.11903.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, 2024. URL https://arxiv.org/abs/2402.01622.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023a. URL https://arxiv.org/abs/2305.10601.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023b. URL https://arxiv.org/abs/2210.03629.

Cong Zhang, Xin Deik Goh, Dexun Li, Hao Zhang, and Yong Liu. Planning with multi-constraints via collaborative language agents. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 10054–10082, Abu Dhabi, UAE, 2025a. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.672/.

Yao Zhang, Chenyang Lin, Shijie Tang, Haokun Chen, Shijie Zhou, Yunpu Ma, and Volker Tresp. SwarmAgentic: Towards fully automated agentic system generation via swarm intelligence. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 1778–1818, Suzhou, China, 2025b. Association for Computational Linguistics. URL https://aclanthology.org/2025.emnlp-main.93/.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. NATURAL PLAN: Benchmarking LLMs on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024. URL https://arxiv.org/abs/2406.04520.

A TravelPlanner Jacksonville–Los Angeles Case Study

This appendix presents the intermediate outputs for the TravelPlanner comparison shown in Figure 2. The direct plan and the HyperPlan plan are evaluated for the same request, daily schema, candidate-resource bindings, route consistency, accommodation rules, and budget constraints. The purpose is to make the source of the planning errors explicit; the case does not claim an unsupported causal account of individual rollouts.

A.1 Query

> Can you plan a three-day solo trip for me from Jacksonville to Los Angeles (March 25–27, 2022) with a total budget of $2,400? We would like to visit Bronson Caves and Crossroads of the World. Please plan the dining arrangements for our trip during this time.

A.2 Constraint Manager Output

1. The itinerary starts in Jacksonville, travels to Los Angeles, and returns to Jacksonville.
2. The travel dates are March 25–27, 2022 (three days total).
3. The estimated total cost must not exceed $2,400.
4. The itinerary must include Bronson Caves and Crossroads of the World.
5. Under the daily-schema requirements, breakfast, lunch, and dinner fields must be filled.
6. The accommodation block must account for the two planned overnight stays.
7. Outbound and return legs must use flight resources matching the direction of travel.

A.3 Hypergraph Instantiation Output

- Query node: `trip-query`.
- Constraint nodes: `origin`, `destination`, `travel-date`, `days`, `budget`, `requested-attraction`, `meal-completeness`, `accommodation-feasibility`, and `route-specific-transport`.
- Resource pools: flights, ground transport, restaurants, attractions, and accommodations.
- Operator hyperedges: constraint extraction; resource retrieval and expansion; daily-level combination with state transitions; plan closure; and feasibility checks.
- Activated route edges: Jacksonville → Los Angeles and Los Angeles → Jacksonville.

A.4 Direct Planner Output

Day 1: Jacksonville → Los Angeles. Transportation: Flight F3765485, 20:40–22:55.
Accommodation: Midtown East 2 Bed/2 Bath Elevator Building Apt; minimum stay of seven nights.

Day 2: Los Angeles. Transportation: none. Breakfast, attraction, lunch, and dinner fields: empty.
Accommodation: Midtown East 2 Bed/2 Bath Elevator Building Apt; minimum stay of seven nights.

Day 3: Los Angeles → Jacksonville. Transportation: Flight F3765485, 20:40–22:55.

A.5 Evaluator Output for the Direct Plan

Decision: invalid under the case audit.

- **Accommodation failure:** The selected accommodation requires a minimum stay of seven nights, whereas the itinerary uses only two nights.
- **Completeness failure:** Although the request includes dining arrangements and designated attractions, the corresponding Day 2 fields remain empty.
- **Transport-resource binding failure:** Outbound flight F3765485 is reused for the return leg; the correct return flight is F3766955.
- **Attribution:** The failures arise from global accommodation feasibility, daily-field completeness, and directional resource binding.

A.6 Search Advisor and Failure Memory

The rollout-by-rollout advisor messages are not shown in the source figure. The evidence-supported repair summary is:

- select accommodation with a minimum-stay requirement of no more than two nights;
- fill the dining and sightseeing fields for Day 2;
- keep the accommodation block consistent across both nights; and
- retrieve a return flight with matching routing rather than duplicating the outbound resource.

A.7 HyperPlan Planner Output

Day 1: Jacksonville → Los Angeles. Transportation: Flight F3765485, 20:40–22:55.
Accommodation: Private Room with Private Bath Upper East Side; planned stay of two nights.

Day 2: Los Angeles. Breakfast: Onokabe.
Attractions: Bronson Caves; Crossroads of the World.
Lunch/Dinner: Choco Kraft / Shree Manakamna Fast Food.
Accommodation: Private Room with Private Bath Upper East Side; planned stay of two nights.

Day 3: Los Angeles → Jacksonville. Transportation: Flight F3766955, 22:01–05:15.
Accommodation: none.

A.8 Evaluator Output for the HyperPlan Plan

Decision: valid under the official checks applied to the preserved plan.

- Minimum stay: one night; planned stay: two nights.
- Route linkage: outbound and return flights match their respective directions.
- Daily schema: the Day 2 attraction and dining fields are populated.
- Estimated cost: $2,376; budget overrun: $0.
- Final Pass: `true`.

A.9 Final Delivered Plan

Day 1. Current city: Jacksonville → Los Angeles.
Transportation: Flight F3765485, 20:40–22:55.
Accommodation: Private Room with Private Bath Upper East Side; two-night stay.

Day 2. Current city: Los Angeles.
Transportation: none.
Breakfast: Onokabe.
Attractions: Bronson Caves; Crossroads of the World.
Lunch/Dinner: Choco Kraft / Shree Manakamna Fast Food.
Accommodation: Private Room with Private Bath Upper East Side.

Day 3. Current city: Los Angeles → Jacksonville.
Transportation: Flight F3766955, 22:01–05:15.
Accommodation: none.

A.10 Audit Note

This comparison highlights three directly generated planning issues: (i) an infeasible multi-day accommodation choice; (ii) incomplete daily-level fields; and (iii) reuse of an outbound resource for the return route. HyperPlan addresses these issues through resource binding and explicit feasibility checks over the complete daily plan.

B Use of Language Models in Manuscript Preparation

Generative AI assisted with editing the supplied English manuscript, organizing the presentation, formatting references, and preparing the LaTeX source. It was not used in this conversion to run new experiments or generate additional measurements. Numerical results in this version are retained from the existing manuscript and tables. The authors should review the final wording, reference metadata, and experimental reporting before submission.