SKELETONPLAN: EXPERIENCE-GUIDED DECISION SKELETONS OVER COMPOSITIONAL CONSTRAINT PROGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neuro-symbolic mulit-agents methods have become a promising solution for Real-world agent planning which requires coordinating heterogeneous decisions under numerous explicit and implicit constraints. However, existing neuro-symbolic planners commonly expand planning requirements into lengthy collections of low-level constraints, obscuring the higher-level dependencies among decision families and increasing the representation burden placed on planning agents. They also solve the resulting models either jointly or through decompositions fixed before inference, preventing the solving structure from adapting to the feasibility bottlenecks of each instance; moreover, effective solving structures discovered for one instance are rarely abstracted and reused across related problems. To address these limitations, we propose SkeletonPlan, a neuro-symbolic multi-agent framework that combines a compositional representation of constraint structure with instance-adaptive organization of symbolic solving. First, SkeletonPlan represents recurring constraint patterns as executable compositions over typed and indexed variable families, making family-level dependencies explicit to planning agents while preserving deterministic compilation into the complete constraint model. Based on this structural representation, a constraint-guided skeleton agent selects interface families whose commitments are expected to shape downstream feasibility and constructs an instance-specific organization for symbolic solving; full-model completion verifies whether the selected commitments extend to a solution of the complete constraint system. Finally, SkeletonPlan learns an online structural memory from solver and evaluator feedback. The memory distills recurring observations, guidance, and exceptions into compact natural-language advice that informs route, interface, and support decisions on later instances, while concrete assignments and final validity remain the responsibility of the symbolic solver. Experiments on Travelplanner demonstrate that SkeletonPlan achieves state-of-the-art planning performance. while ablation studies confirm the respective contributions of compositional constraint representation, instance-adaptive skeleton construction, and online structural memory learning.

1 INTRODUCTION

Real-world decision tasks such as itinerary construction, meeting coordination, and calendar arrangement require a planner to determine multiple heterogeneous decision families simultaneously, including locations, transitions, resources, activities, time slots, and participants. These decisions are tightly coupled because a choice in one family can restrict the feasible values of several others, while the final plan must jointly satisfy temporal, spatial, resource, compatibility, and user-specific requirements (Xie et al., 2024; Zheng et al., 2024; Yang et al., 2025). We refer to this class of problems as wide-horizon planning, whose primary difficulty lies in coordinating a broad and interconnected assignment space rather than extending a single action sequence over many steps, as emphasized in long-horizon control. Solving such problems therefore requires both flexible semantic reasoning, so that natural-language requirements and task-specific information can be correctly interpreted, and exact global reasoning, so that interacting constraints can be enforced across all decisions. Neuro-symbolic multi-agent systems provide a natural foundation for this combination

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.

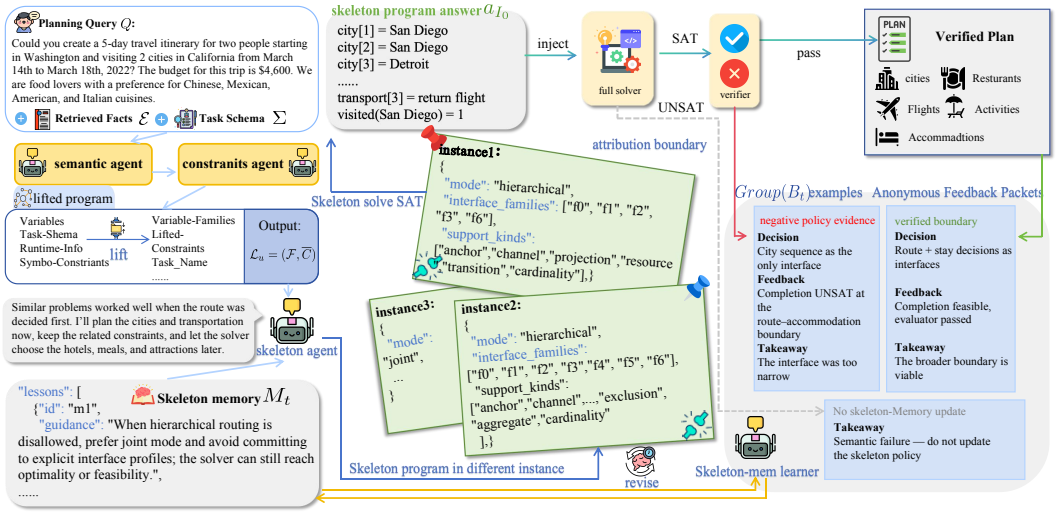

Figure 1: **Overview of SkeletonPlan. (A) Constraint Representation:** Semantic agents translate multimodal planning requirements into a lifted constraint program, which is compiled into an executable solver model. **(B) Adaptive Decision Skeleton:** Neural agents propose structural decisions over a value-free problem view, while a two-stage CP-SAT solver first determines skeleton commitments and then completes and verifies the full solution. **(C) Structural Memory Learning:** Solver diagnostics and evaluator feedback are summarized into a bounded memory that guides future skeleton generation, while symbolic completion ensures correctness.

by assigning semantic interpretation and task organization to language-based agents while retaining constraint enforcement and final feasibility verification within symbolic procedures.

Existing language-grounded planning methods do not yet realize this combination effectively. Purely neural planners can interpret underspecified requests, acquire information through external tools, and revise candidate plans through language-based reasoning, but the dependencies among their decisions remain distributed across generated trajectories and local repair steps, making global consistency increasingly difficult to maintain as the candidate space and the number of interacting constraints grow (Erdogan et al., 2025; Parmar et al., 2025; Ni et al., 2025). Neuro-symbolic planners improve reliability by compiling natural-language requirements into executable constraint or optimization models and delegating concrete decision-making to symbolic solvers, thereby enabling exact verification relative to the formalized requirements (Kambhampati et al., 2024; Hao et al., 2025b;a; Gundawar et al., 2024). Nevertheless, these systems commonly expose planning agents to explicitly grounded or otherwise low-level constraint descriptions, whose repeated expansion over indexed variables increases the agent-facing representation burden and obscures the structural roles of the underlying decision families. Moreover, the resulting model is typically solved either as a single coupled problem or through a decomposition designed before inference, even though the decisions that dominate feasibility may vary with the candidate set, resource availability, and constraint tightness of each instance. The unresolved challenge is therefore not merely to formalize planning requirements, but to expose their higher-level dependency structure and use that structure to organize symbolic solving adaptively without weakening full-model correctness.

To address this challenge, we propose SkeletonPlan (Figure 1), a neuro-symbolic multi-agent framework that makes the structural organization of a formalized planning problem explicit and adapts symbolic solving to the dependency pattern of each instance. First, SkeletonPlan represents structured constraint patterns as executable compositions over typed and indexed variable families, allowing planning agents to reason over compact family-level relations while a deterministic compiler preserves their complete solver-level semantics. Second, a constraint-guided skeleton agent analyzes these relations together with instance-specific information to identify the decision families most consequential to downstream feasibility, thereby constructing a solving structure that reflects the bottlenecks of the current problem rather than a hierarchy fixed across all instances. The proposed skeleton

determines how solving is organized but does not independently certify a plan: full-model completion accepts the resulting commitments only when they can be extended to a solution satisfying the complete constraint system. Finally, SkeletonPlan closes the loop with online structural memory learning. Each episode produces solver, completion, and official-evaluation feedback; batch-level clustering removes redundant evidence, and an LLM distills the remaining structural observations, guidance, and exceptions into a bounded natural-language memory. On a later instance, the skeleton agent reads the complete memory and directly chooses the route, interface, and support. The memory is advisory rather than executable, so full-model completion remains the boundary of correctness. Together, these components connect compact structural representation, instance-adaptive decomposition, and completion-based verification in a unified approach to wide-horizon planning.

Our contributions are threefold:

Executable compositional constraint representation. We introduce a constraint program over typed and indexed variable families that exposes compact family-level dependencies to planning agents while deterministically compiling into a complete executable model.

Completion-verified adaptive decision skeletons. We develop an instance-adaptive mechanism that identifies feasibility-critical decision families and organizes symbolic solving around them, while full-model completion verifies returned plans against the complete constraint system.

Online structural memory learning. We introduce an online learner that distills anonymized solver and evaluator feedback into bounded natural-language lessons, conditioning future route, interface, and support decisions on the accumulated memory.

2 Related Work

Language-grounded neuro-symbolic planning. Real-world planning benchmarks expose the difficulty of coordinating heterogeneous choices under interacting requirements, beyond generating a plausible sequence of actions (Xie et al., 2024; Zheng et al., 2024; Shao et al., 2024; Yang et al., 2025). Systems for this setting combine retrieval and tool use (Ni et al., 2025), multi-agent coordination (Choi et al., 2026; Liu et al., 2025; Zhang et al., 2024), and iterative generation or refinement (Erdogan et al., 2025; Parmar et al., 2025). A complementary line of work connects language understanding to formal planning and constraint solving. LLM+P translates language descriptions into PDDL for execution by a classical planner (Liu et al., 2023). LLM-Modulo and solver-augmented systems instead couple language models to external verifiers or encode real-world requirements as satisfiability and optimization problems (Kambhampati et al., 2024; Gundawar et al., 2024; Hao et al., 2025b;a; de la Rosa et al., 2024; Lee et al., 2025). Related neuro-symbolic reasoning systems also use symbolic solvers to check model-generated logical representations (Pan et al., 2023). These approaches establish a valuable separation between interpreting a request and enforcing its encoded requirements. SkeletonPlan builds on this separation and focuses on the organization of the resulting constraint problem: its compositional representation exposes dependencies among decision families, allowing an agent to select an instance-specific interface for symbolic solving. The distinction concerns how a formalized problem is organized, in addition to how language is translated into an executable model.

Decomposition and learned search guidance. Decomposition exploits problem structure by coordinating restricted subproblems, with the treatment of their shared decisions central to their interaction (Hooker, 2007). Learned guidance can also operate directly within combinatorial search. For example, ANYCSP learns a graph-based global search policy for constraint satisfaction (Tönshoff et al., 2022), whereas ReEvo combines LLM reflection with evolutionary search to generate heuristic programs (Ye et al., 2024). Program-search and model-reformulation approaches similarly use language models to discover algorithms or improve solver efficiency (Romera-Paredes et al., 2024; Michailidis et al., 2026). These approaches motivate learning how to search, but their learned objects differ from the decision skeleton studied here. SkeletonPlan selects a solving route, interface families, and existing support kinds over a typed constraint program. Concrete assignments are produced by the symbolic solver, and interface commitments are checked through full-model completion. This places learned guidance at the boundary between problem representation and solver execution, with an explicit distinction between proposing a useful organization and establishing that its assignments extend to a valid plan.

Experience learning in language agents. Experience can improve agent behavior without updating model parameters. ExpeL extracts natural-language knowledge from accumulated experience and recalls insights and prior examples at inference (Zhao et al., 2023); ICE consolidates interaction trajectories into reusable workflows and pipelines (Qian et al., 2024). Feedback can instead revise an agent's plan online (Sun et al., 2023; Gou et al., 2024), while self-evolving systems update symbolic knowledge, agent populations, or memory mechanisms over repeated episodes (Zhou et al., 2025; Yuan et al., 2025; Zhai et al., 2025; Zhang et al., 2025a; 2026). These works demonstrate that experience may be represented as both textual advice and procedural structure. SkeletonPlan studies experience at a specific decision boundary: the organization of symbolic solving. Its memory associates anonymous structural observations with guidance and exceptions, drawing on execution feedback whose scope distinguishes an infeasible interface commitment from a timeout or a task-level validation failure. The skeleton agent interprets this memory together with the current constraint structure to choose its route, interface, and support. Memory remains advisory; its aggregate evidence strength can be inspected, while the complete constraint model determines whether a returned assignment is acceptable.

3 SkeletonPlan: Learning to Organize Symbolic Planning

SkeletonPlan connects three operations: exposing the structure of a formalized planning problem, using that structure to organize symbolic solving, and consolidating execution experience for subsequent structural decisions. Figure 1 illustrates this process. a skeleton agent chooses how the encoded problem should be solved; a skeleton-mem learner updates compact structural advice from completed episodes. Deterministic compilation and full-model checking provide a common execution boundary for these neural components.

Planning problem. Let $u = (q, \mathcal{E}, \Sigma)$ denote a natural-language request, its retrieved facts, and a task schema. A semantic agent and a constraint agent work together to interprets these inputs and maps task-relevant concepts into symbolic variables. Further details of both agents are provided in Appendix **??**. Formalization yields

$$\mathcal{P}_u = (X, D, C, f), \qquad \Omega_u = \{z \in D : \; c(z) = 1 \; \forall c \in C\}, \tag{1}$$

where X is a finite set of variables, D their joint domain, C the encoded hard constraints, and f an optional objective to minimize over Ω_u. For feasibility tasks, $f \equiv 0$. Wide-horizon planning couples multiple heterogeneous decision families within this assignment space. Our objective is to organize the search for a feasible assignment under a finite computation budget. All symbolic validity statements are relative to C; task-level validation separately evaluates whether the decoded plan satisfies the original requirements, including requirements that formalization may have missed.

3.1 Executable Compositional Constraint Programs

A useful representation for structural planning must preserve the relationships that matter for solving while remaining compact enough for the agent to inspect. We therefore organize variables into typed families, compose their constraints through a checked lowering interface, and expose an anonymized structural view to the skeleton agent. Together, these elements retain executable semantics while making the interactions among decisions explicit.

Typed decision families. Structural decisions require an abstraction that preserves how groups of variables interact. The task schema therefore declares finite index sets and typed variable families $\mathcal{F} = \{F_1, \ldots, F_m\}$, with

$$F_i = \{x_{p_i(\boldsymbol{j})} : \; \boldsymbol{j} \in J_{i,1} \times \cdots \times J_{i,r_i}\}, \tag{2}$$

where p_i is a reference pattern. Daily locations, transitions, and resource selections can consequently be addressed as families rather than repeated lists of individual variables. For example, a transition relation links consecutive location decisions even as the number of days changes. Its family-level role provides a stable structural description across different instance sizes.

Executable compositions. The lifted program $\mathcal{L}_u = (\mathcal{F}, \overline{C})$ represents constraints through typed templates and their supported compositions. Each $\bar{c} \in \overline{C}$ records its arguments, family scope, hardness, and support kind $\kappa(\bar{c})$, such as transition, channel, compatibility, projection, or aggregate.

Family scope identifies the families touched by the constraint. Support kinds describe their role in organizing a skeleton; template arguments retain the information needed for execution. A deterministic registry Γ lowers each supported composition into solver constraints according to its declared semantics:

$$\mathrm{sat}(\bar{c}, z) = \bigwedge_{c \in \Gamma(\bar{c})} \mathrm{sat}(c, z). \tag{3}$$

The compiler checks template names, types, references, and finite domains before lowering. This contract concerns the encoded composition: it does not establish that the preceding language formalization captured every user requirement.

Agent-facing structural view. The skeleton agent receives an anonymized view $\mathcal{S}_u$ containing family roles and types, size buckets, index ranks, constraint incidence, and available support kinds. Selectable families receive local aliases; deferred resource families remain visible in the incidence structure but cannot be selected as interface decisions. Task-schema checks also indicate whether the required resource abstractions permit hierarchical solving. Concrete entity names and candidate assignments are omitted from this view. The agent can thus reason about where decisions interact while the full executable model retains the values and relations needed to solve them. Compactness here concerns the representation exposed to the agent; it does not imply a smaller grounded model or a lower worst-case solving complexity.

3.2 Completion-Verified Adaptive Decision Skeletons

With this structural view in place, SkeletonPlan can organize solving around the decisions that are most relevant to the current instance. The agent selects a route, interface, and support kinds; deterministic checks and support resolution execute that choice; and full-model completion verifies whether the resulting commitments can be extended to a valid plan. We then characterize the resulting soundness guarantees and the limits imposed by the search budget.

Instance-specific organization. For each instance, the skeleton agent reads $\mathcal{S}_u$ and the current structural memory M_t, introduced in Section 3.3, and proposes

$$d_u = \pi_\theta(\mathcal{S}_u, M_t) = (r, I_0, K), \tag{4}$$

where r selects joint or hierarchical solving, I_0 contains selected interface families, and K contains support kinds. The model parameters θ remain fixed during memory learning. A joint decision solves $\mathcal{P}_u$ directly and has an empty interface and support selection. A hierarchical decision selects a nonempty interface whose commitments will precede completion. The agent also supplies a structural rationale and records how it uses the memory. A deterministic action-schema check restricts decisions to the supplied families and kinds, requires support incident to each selected family, and enforces the task's hierarchical eligibility conditions. Invalid proposals are returned to the agent for bounded correction before solving.

The adaptive policy is evaluated against two fixed-interface controls. These controls are not additional learned modules; they are deterministic policies defined over the same anonymized structural view. Fixed-Narrow selects one smallest legal non-deferred family and its incident support kinds. It tests the failure mode in which the skeleton commits too little structure for the completion stage. Fixed-Broad ranks families by constraint incidence and selects a fixed target width of seven families, backing off only when the selected set would violate the legal proper-subset or support conditions. It tests whether a permanently wide interface can replace instance-specific organization. Because family aliases are resolved independently for every instance, these baselines fix the policy rule rather than copying a literal interface from one example to another. Together with Joint, which omits decomposition entirely, the two controls separate the value of decomposition, interface width, and adaptive selection.

Direct support resolution. The interface is executed as selected: $I = I_0$. The compiler resolves support kinds K to existing constraints incident to that interface,

$$\overline{C}_S = \{\bar{c} \in \overline{C} : \kappa(\bar{c}) \in K, \mathrm{fscope}(\bar{c}) \cap I_0 \neq \emptyset\}. \tag{5}$$

It neither recursively adds linked interface families nor automatically includes hard constraints solely because their scopes lie inside the interface. Applicable task-provided summaries are instantiated

only for selected support kinds and when their declared prerequisites hold. Domain and base constraints needed by participating variables complete the skeleton model C_S; auxiliary support variables do not enlarge the selected commitment interface. These operations bind existing constraint semantics rather than repair the agent's choice. Full-model completion remains responsible for checking whether those commitments admit a valid plan.

Symbolic commitment and completion. Let $X_{I_0} = \bigcup_{i \in I_0} F_i$. The skeleton solver obtains an assignment to C_S, and its determined interface values form a commitment a_J, where $J \subseteq X_{I_0}$ contains the variables actually fixed. The second stage solves the full model subject to this commitment:

$$\mathrm{Comp}_{\mathcal{P}_u}(a_J) = \{z \in \Omega_u : z|_J = a_J\}, \qquad \hat{z} \leftarrow \mathrm{Solve}(C \cup \{x = a_J(x) : x \in J\}). \tag{6}$$

A hierarchical plan is accepted only if this completion succeeds. For example, a location and transition interface may be inexpensive to solve, yet commit to a schedule for which the deferred resource choices have no feasible completion. The failed completion identifies a limitation of those commitments; it does not establish that the original planning instance is infeasible.

Correctness and search scope. Assuming faithful compilation and a sound solver, any returned completion $\hat{z}$ lies in Ω_u by Equation 6. This follows because the completion model contains every constraint in C, independently of how the interface was selected. The guarantee is acceptance soundness: a poor skeleton can still miss feasible plans under the available budget. When skeleton constraints are necessary conditions of the full model, a proved skeleton contradiction can establish global infeasibility; a contradiction under fixed interface values has only conditional scope. Timeouts establish neither. Similarly, optimality under a fixed interface does not imply global optimality. In the online memory protocol, unsuccessful hierarchical attempts are recorded without a joint fallback or a second solver run that repairs the chosen interface. This preserves the feedback from the organization actually selected by the agent.

3.3 Online Structural Memory Learning

The execution outcomes of these decisions provide evidence about which forms of organization work on recurring structures. We distinguish the scope of different feedback signals, consolidate repeated evidence into a compact memory, and reuse that memory as revisable advice on subsequent instances. The update protocol keeps the compiler and language model fixed while allowing experience to influence how symbolic search is organized.

Structurally grounded feedback. A single outcome has different implications depending on where it occurs. SkeletonPlan constructs an episode record from the pre-decision structural summary, the selected route and interface profiles, support kinds, solver and completion statuses, and task-level evaluator feedback. When available, completion assumption cores are summarized by their family profiles and the support kinds along the adjacent constraint boundary. This localizes the observed conflict without exposing concrete entities or values. Feedback preserves the distinction between globally scoped contradictions, failed interface commitments, censored timeout outcomes, and task-level failures after a feasible solve. In particular, a task-level failure calls for scrutiny of formalization or decoding; it is not evidence that a different skeleton was necessary. A successful episode provides an observation about the chosen boundary, without proving that its support choices were necessary or optimal.

Bounded structural memory. The learned state is a collection of natural-language lessons,

$$M_t = \{m_j\}_{j=1}^{L_t}, \qquad m_j = (o_j, g_j, e_j, n_j), \qquad L_t \le L_{\max}, \tag{7}$$

where o_j describes a structural observation, g_j gives decision guidance, e_j records an exception or uncertainty, and n_j counts accumulated supporting observations. Individual episode identifiers are not retained. The memory vocabulary is restricted to structural descriptions; lessons are instructed to exclude entity names, instance constants, local family aliases, and executable code or actions. The exception field gives the learner a place to preserve counterevidence and limits instead of turning every observation into an unconditional recommendation. We use $L_{\max} = 12$ to keep the complete memory available in the agent's context. This is an external memory update; the underlying language model and symbolic compiler are unchanged.

Online consolidation. Memory starts empty and remains fixed throughout each batch of training instances. At the batch boundary, feedback records with identical structural, decision, outcome, and

diagnostic fields are grouped, retaining only their counts. This is exact deduplication within each feedback batch. The update agent reads these groups together with the current memory and produces a replacement memory:

$$M_{t+1} = \mathcal{V}_{\text{schema}}(U_\theta(M_t, \text{Group}(B_t))), \tag{8}$$

where B_t contains the latest batch feedback. The updater is instructed to preserve useful lessons, revise contradicted guidance, merge duplicates, and remove unsupported observations. It cites supplied batch groups and earlier lessons; the system accumulates their counts into the returned lessons without storing episode-level provenance. The validator checks memory format, size limits, positive evidence counts, and supplied lesson or cluster references. It does not prove the truth or utility of a lesson, and updates require no counterfactual solver trials or implication checks.

Memory-conditioned reuse. On subsequent instances, the skeleton agent reads all retained lesson texts and assesses their applicability against the current structural view through Equation 4. It can use, adapt, or reject the supplied advice while choosing the route, interface, and support anew. This closes the loop from structural decisions to execution evidence and back to structural decisions. The memory supplies revisable guidance rather than a deterministic action transformation, and its empirical value must be established by comparing planning outcomes with and without learned memory. Regardless of that value, action-schema checks and full-model completion continue to govern execution and acceptance. Experience can therefore influence the organization of search without modifying the encoded planning requirements.

4 EXPERIMENTS

We evaluate four aspects of SkeletonPlan. First, we test whether learned structural memory improves held-out TravelPlanner validity over the same adaptive policy with empty memory (**Q1**). Second, joint and fixed-interface controls isolate the roles of adaptive construction, interface breadth, and memory (**Q2**). Third, we test task-specific memory induction on held-out Calendar, Meeting, and Trip Planning instances in NaturalPlan, with memory frozen at evaluation (**Q3**). Finally, we measure the latency and policy-token overhead of these validity gains.

4.1 EXPERIMENTAL SETUP

Benchmarks and splits. We evaluate SkeletonPlan on two real-world planning benchmarks, TravelPlanner (Xie et al., 2024) and NaturalPlan (Zheng et al., 2024). The TravelPlanner benchmark consists of 45 training, 180 validation, and 1000 test instances, while the NaturalPlan benchmark includes 3 different tasks. Following the official protocol, we report delivery rate, micro and macro pass rates for commonsense and hard constraints, and final pass rate. For NaturalPlan, we evaluate Calendar Scheduling, Meeting Planning, and Trip Planning following the official task-specific evaluation protocols and report accuracy.

Baselines. The completed comparison isolates memory conditioning within the same system. EMPTY-MEMORY runs the skeleton policy with initial memory, while **SkeletonPlan** uses the learned memory learned from the training instances online run. Both conditions use the same DeepSeek-Flash policy model, executable constraint programs, decoder, official evaluator, and 120-second per-instance solver limit. For the policy ablation, JOINT solves the complete model without a skeleton, while FIXED-NARROW and FIXED-BROAD use the deterministic interface rules defined above. The fixed controls use feasibility-only solving in the rapid ablation replay; the same hard constraints, decoder, evaluator, seeds, and completion acceptance criterion are retained across rows.

Implementation and budgets. The memory learner observes the training instances in batches of 10 and retains at most 12 lessons. The skeleton policy directly selects the solving route, interface families, and support kinds. No test-time memory update, programmatic decision transformation, repair run, or hidden fallback is used. the instance order is fixed with seed 2027. We report end-to-end latency and policy-token use so that the extra language-model context introduced by memory is not treated as free computation.

Table 1: TravelPlanner performance using the six official metrics. Panel A compares published planning systems with SkeletonPlan. Panel B reports diagnostics specific to completion-verified skeleton solving. Values are percentages except time and tokens.

Panel A: TravelPlanner official metrics

Model	Method	Delivery	Commonsense Pass Rate		Hard-Constraint Pass Rate		Final
			Micro	Macro	Micro	Macro	
Rule-based	Greedy Search (Xie et al., 2024)	100.00	74.40	0.00	60.80	37.80	0.00
GPT-4-Turbo	Direct (Xie et al., 2024)	100.00	80.40	17.20	47.10	22.20	4.40
Mistral-Large	RigorFormal (Hao et al., 2025a)	72.20	72.00	70.60	63.30	66.70	66.70
Claude-3		96.10	96.00	95.60	94.80	93.30	93.30
GPT-4		95.00	95.00	95.00	95.70	98.90	93.30
Gemini-2.5-Pro	ATLAS (Choi et al., 2026)	100.00	88.54	48.33	82.62	74.44	44.44
Claude-Sonnet-4		100.00	83.40	37.78	56.43	38.89	23.33
Qwen3-8B	HiMAP-Travel (Bui et al., 2026)	100.00	95.64	73.61	76.85	63.33	52.78
GPT-4	PMC (Zhang et al., 2024)	100.00	90.42	35.00	52.14	25.00	6.67
GPT-4	EvoAgent (Yuan et al., 2025)	100.00	81.50	21.10	31.40	18.90	7.20
DeepSeek-Flash	**SkeletonPlan**	**95.00**	**95.00**	**95.00**	**93.81**	**95.00**	**95.00**

Panel B: Skeleton-specific diagnostics and efficiency

Method	Hierarchical Solve	C-UNSAT	p50 (s)	p95 (s)	Mean Tokens
SkeletonPlan	**93.94**	**1.67**	6.59	117.87	4,736.5

4.2 LEARNED SKELETONPLAN IMPROVES VALID PLANNING

Overall planning validity. Table 1 shows that frozen learned memory raises final pass from 107/180 (59.44%) to 171/180 (95.00%), a gain of 35.56 percentage points. The corresponding pooled counts are 856/1,440 versus 1,368/1,440 for commonsense checks and 265/420 versus 394/420 for applicable hard constraints. Delivery equals final pass in both conditions: every decoded itinerary passes all of its official checks, and failures arise before an itinerary is delivered. All 107 empty-memory successes remain successful, while 64 previously failing instances become successful. The gain is concentrated in hierarchical episodes, whose solve rate improves from 60/132 (45.45%) to 124/132 (93.94%). Fixed-interface completion contradictions fall from 69/180 (38.33%) to 3/180 (1.67%). This improvement carries a cost: median latency increases from 2.78 to 6.59 seconds and mean policy usage from 3,317.6 to 4,736.5 tokens. After learning, learned memory can significantly improve the capability of the SkeletonPlan framework on the same dataset and reduce failure cases. Despite the increased latency and token consumption, this trade-off yields a substantial improvement in planning validity, demonstrating an efficient balance between computational cost and final success rate.

Generalization on NaturalPlan. Table 2 evaluates whether SkeletonPlan generalizes across three distinct planning tasks in NaturalPlan: Calendar Scheduling, Meeting Planning, and Trip Planning. These tasks involve different decision structures and constraint interactions, providing a broader test of whether the framework is applicable beyond a single planning setting. For each task, we stratify instances by the benchmark's task-size field and reserve 10% for task-specific memory induction using seed 2027; the remaining 90% form a disjoint evaluation subset. Memory starts empty, is updated only on the induction subset, and is frozen throughout evaluation. All three tasks use the same SkeletonPlan framework, policy model, evaluation protocol, and per-instance solver budget.

SkeletonPlan achieves 83.33% accuracy on Calendar Scheduling, 100.00% on Meeting Planning, and 96.67% on Trip Planning, yielding a macro-average of 93.33% across the three tasks. The published GPT-4o results span markedly different task profiles: SwarmAgentic is strongest among the retained references on Calendar (82.00%), whereas ADAS is strongest on Meeting (43.00%) and Trip (66.00%). These values provide context rather than a controlled comparison because the published systems use different backbones and may use different evaluation partitions. Nevertheless, SkeletonPlan maintains consistently high held-out accuracy across tasks with substantially different planning structures. This result demonstrates that the framework's structural representation, skeleton construction, and memory-learning procedure remain effective across multiple planning domains rather than being specialized to one task type.

Table 2: Accuracy (%) on held-out NaturalPlan evaluation subsets. Published GPT-4o results are included as external reference points. SkeletonPlan learns memory only from a disjoint 10% target-task induction subset and freezes it on the remaining 90%; it uses DeepSeek-Flash. Because published systems may use different random partitions, their values are contextual rather than paired comparisons.

Method	Backbone	Calendar	Meeting	Trip
SPP (Wang et al., 2024)	GPT-4o	1.30	33.00	44.00
EvoAgent (Yuan et al., 2025)	GPT-4o	1.90	38.00	52.00
SwarmAgentic (Zhang et al., 2025b)	GPT-4o	82.00	56.00	13.10
ADAS (Hu et al., 2025)	GPT-4o	3.10	43.00	66.00
SkeletonPlan	DeepSeek-Flash	83.33	100.00	96.67

4.3 Ablations and Mechanism Analysis

Table 3 summarizes the currently completed policy variants. JOINT solves without a skeleton. FIXED-NARROW always uses a small legal interface, whereas FIXED-BROAD uses a wider deterministic interface. The last two rows use the adaptive skeleton with empty or learned memory.

Table 3: Current policy ablation results. Values are percentages except median latency.

Variant	Official ↑	Hier. solve ↑	C-UNSAT ↓	p50 (s) ↓
Joint	90.56	–	–	3.65
Fixed-Narrow Skeleton	69.44	66.67	23.33	3.26
Fixed-Broad Skeleton	90.00	95.45	0.56	5.06
Adaptive Skeleton + Empty Memory	59.44	45.45	38.33	2.78
Adaptive Skeleton + Learned Memory	95.00	93.94	1.67	6.59

Ablation findings. The fixed variants show that decomposition quality depends on the interface. FIXED-NARROW reaches only 69.44% success and 23.33% C-UNSAT, indicating that an overly restrictive interface omits dependencies required by completion. Widening it in FIXED-BROAD raises success to 90.00% and reduces C-UNSAT to 0.56%, nearly matching undecomposed JOINT (90.56%). Thus, decomposition alone is insufficient: a fixed interface is either too narrow or broad enough to approximate joint solving.

Structural memory is what makes adaptation effective. With empty memory, the adaptive policy achieves 59.44% success and 38.33% C-UNSAT; with learned memory, these become 95.00% and 1.67%, respectively, while hierarchical solve rises to 93.94%. This 35.56-point gain shows that instance-wise adaptation requires learned guidance rather than flexibility alone.

The learned policy also exceeds JOINT and FIXED-BROAD by 4.44 points, although FIXED-BROAD has slightly better intermediate hierarchical-solve and C-UNSAT values. The final gain therefore cannot be attributed to interface breadth or any single intermediate metric. Instead, memory helps choose interfaces and supports that better balance early commitment with full-plan completion. Its higher median latency (6.59 seconds) means that the demonstrated advantage is validity, not uniformly lower cost.

5 Conclusion

We present SkeletonPlan, a neuro-symbolic multi-agent framework that organizes wide-horizon planning around instance-specific structural dependencies. Typed compositional constraints expose family-level relations to an adaptive skeleton agent; full-model completion checks that its commitments extend to valid plans; and bounded structural memory converts solver and evaluator feedback into reusable guidance. On TravelPlanner and NaturalPlan, this design improves held-out validity, remains effective under task-specific induction across distinct tasks, and surpasses joint and fixed-interface controls, with latency and token costs reported. More broadly, reliable planning can be achieved by learning how to organize exact symbolic solving, unifying neural coordination with symbolic correctness.

REFERENCES

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.

Jihye Choi, Jinsung Yoon, Jiefeng Chen, Somesh Jha, and Tomas Pfister. Atlas: Constraints-aware multi-agent collaboration for real-world travel planning. In *Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Tomas de la Rosa, Sriram Gopalakrishnan, Alberto Pozanco, Zhen Zeng, and Daniel Borrajo. Trip-pal: Travel planning with guarantees by combining large language models and automated planners. *arXiv preprint arXiv:2406.10196*, 2024.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-act: Improving planning of agents for long-horizon tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 15419–15462, 2025.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Chen, and Nan Duan. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2024.

Atharva Gundawar, Mudit Verma, Lin Guan, Karthik Valmeekam, Siddhant Bhambri, and Subbarao Kambhampati. Robust planning with LLM-modulo framework: Case study in travel planning. *arXiv preprint arXiv:2405.20625*, 2024.

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*, pp. 3434–3483, 2025a.

Yilun Hao, Yang Zhang, and Chuchu Fan. Planning anything with rigor: General-purpose zero-shot planning with llm-based formalized programming. *International Conference on Learning Representations*, 2025b.

John N. Hooker. *Integrated Methods for Optimization*. Springer, 2007.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *International Conference on Learning Representations*, 2025.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas P. 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*, pp. 22895–22907, 2024.

Christine Lee, David Porfirio, Xinyu Jessica Wang, Kevin Zhao, and Bilge Mutlu. Veriplan: Integrating formal verification and llms into end-user planning. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2025.

Binwen Liu, Jialin Ge, and Jiajun Wang. Vaiage: A multi-agent solution to personalized travel planning. *arXiv preprint arXiv:2505.10922*, 2025.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

Kostis Michailidis, Dimos Tsouros, Nguyen Dang, and Tias Guns. Llm-guided evolutionary search for constraint model reformulation to improve solver efficiency. *arXiv preprint arXiv:2607.28268*, 2026.

Haowei Ni, Fang Liu, Xuxin Ma, Lu Su, Sheng Wang, Dawei Yin, Hui Xiong, and Hao Liu. TP-RAG: Benchmarking retrieval-augmented large language model agents for spatiotemporal-aware travel planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12392–12418, 2025.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Yang Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. *arXiv preprint arXiv:2305.12295*, 2023.

Mihir Parmar, Xiao Liu, Palash Goyal, Yanfei Chen, Long Le, Swaroop Mishra, Hossein Mobahi, Jiuxiang Gu, Zhen Wang, Hootan Nakhost, Chitta Baral, Chen-Yu Lee, Tomas Pfister, and Hamid Palangi. Plangen: A multi-agent framework for generating planning and reasoning trajectories for complex problem solving. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20640–20666, 2025.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *arXiv preprint arXiv:2401.13996*, 2024.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

Jie-Jing Shao, Xiao-Yuan Li, Shen-Zhi Zhao, Zhi-Rui Shen, Yuan-Hao Wang, Peng-Hao Shen, Xiao-Yu Wang, Jin-Rong Li, Yu-Tong Fu, Shu-Yuan Zhou, et al. Chinatravel: An open-ended benchmark for language agents in chinese travel planning. *arXiv preprint arXiv:2412.13682*, 2024.

Haotian Sun, Yuchen Zhuang, Lingkai Ling, Xin Chen, Jiayan Chen, Heng Ji, and Chao Zhang. Adaplanner: Adaptive planning from feedback with language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jan Tönshoff, Berke Kisin, Jakob Lindner, and Martin Grohe. One model, any csp: Graph neural networks as fast global search heuristics for constraint satisfaction. *arXiv preprint arXiv:2208.10227*, 2022.

Zilong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 257–279, 2024.

Jian Xie, Kai Zhang, Jiangjie Chen, Ting-Hao 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*, pp. 54590–54613, 2024.

Deheng Yang, Chenguang Lu, Qian Wang, Xiang Ma, Yansong Gao, Yuwei Hu, and Hai Zhao. Wide-horizon thinking and simulation-based evaluation for real-world LLM planning with multifaceted constraints. In *Advances in Neural Information Processing Systems*, pp. 94064–94104, 2025.

Haoran Ye, Jiarui Wang, Zhiguang Cao, and Xavier Bresson. Reevo: Large language models as hyper-heuristics with reflective evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Dongsheng Li, and Deheng Yang. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 6192–6217, 2025.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. Memevolve: Meta-evolution of agent memory systems. *arXiv preprint arXiv:2512.18746*, 2025a.

Shengtao Zhang, Jiaqian Wang, Ruiwen Zhou, Junwei Liao, Yuchen Feng, Weinan Zhang, Ying Wen, Zhiyu Li, Feiyu Xiong, Yutao Qi, Bo Tang, and Muning Wen. Memrl: Self-evolving agents via runtime reinforcement learning on episodic memory. *arXiv preprint arXiv:2601.03192*, 2026.

Shun Zhang et al. Planning with multi-constraints via collaborative language agents. *arXiv preprint*, 2024.

Yifan Zhang, Cheng Lin, Shiyu Tang, Haoran Chen, Shizhe Zhou, Yuzhe 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, 2025b.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. Expel: Llm agents are experiential learners. *arXiv preprint arXiv:2308.10144*, 2023.

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.

Wangchunshu Zhou, Yixin Ou, Shengwei Ding, Long Li, Jialong Wu, Tiannan Wang, Jiamin Chen, Shuai Wang, Xiaohua Xu, Ningyu Zhang, Huajun Chen, and Yuchen Eleanor Jiang. Symbolic learning enables self-evolving agents. *AI Open*, 6:314–322, 2025.