2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

A Natural Language Interface for Multi-Constraint Spatiotemporal Planning via LLM-Parameterized Mixed-Integer Scheduling and A*

Sean Ye, Matthew B. Luebbers*, and Matthew Gombolay

***Abstract*— Spatiotemporal planning is critically important in fields like robotics, logistics, and naval operations, especially for problem specifications involving multiple constraints. Traditional approaches place the burden on end users to manually specify cost functions, constraints, or model parameters, a time-consuming and laborious process often resulting in less-than-ideal plans. We present a novel architecture integrating an LLM-based natural language interface with MILP scheduling and A* motion planning for multi-constraint spatiotemporal planning. We validate our LLM-planning approach through a within-subjects user study using a simulated maritime route-planning domain against manual control, and against autonomous planning with classical template-based constraint specification. Results showed our LLM-planning approach not only improved usability and reduced workload over alternative input modalities but also maintained the path optimality of traditional constraint specification interfaces while decreasing planning time. These findings demonstrate that bridging LLM-powered interfaces with robust schedulers and motion planners can enhance human-autonomy interaction in complex planning tasks, potentially making advanced spatiotemporal planning tools more practical for a broader range of users.**

I. Introduction

Effective task scheduling and motion planning, known collectively as spatiotemporal planning, is a fundamentally important component of solving complex, multi-objective planning tasks in fields such as robotics [1], logistics [2], manufacturing [3], and transportation [4]. Scheduling determines when and in what order tasks should be performed [5], while motion planning focuses on finding optimal paths through space, often subject to various constraints [6].

Traditional approaches to these problems typically require rigid syntactic specifications and expert knowledge, particularly in crafting precise cost functions for motion planning and defining appropriate constraints for scheduling. While these methods excel at finding solutions within well-defined problem spaces and offer strong guarantees on solution quality, interacting with and parameterizing them remains a time-consuming and laborious process, often resulting in plans misaligned with user intention [7]. As the number of variables and constraints increase, scheduling and motion planning problems can quickly become overwhelming for both algorithms and human planners. This inherent complexity makes it difficult to find optimal or even satisfactory solutions, especially when dealing with multiple objectives and intricate constraints. Thus, there is a pressing need for automated systems that can efficiently tackle these NP-hard [8] planning scenarios while remaining accessible and intuitive for users to interact with and guide.

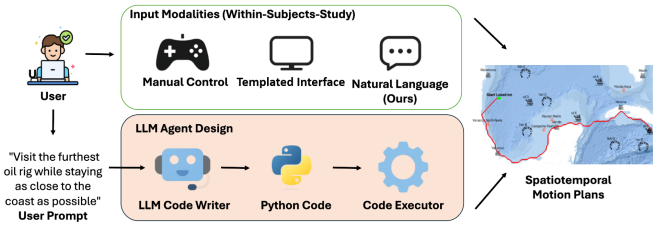

Fig. 1. *Top:* We conducted a within-subjects user study comparing three input modalities for spatiotemporal planning: manual control, templated constraint specification, and our natural language interface powered by a large language model (LLM). *Bottom:* Our LLM agent takes a user's prompt, translates the intention into Python code via the LLM Code Writer, and executes it through the Code Executor. The generated code leverages a Mixed-Integer Linear Program (MILP) for scheduling and an A* planner for motion generation.

Advancements in Large Language Models (LLMs) have revolutionized natural language interfacing with automated systems. In the field of robotics, LLMs have enabled users to specify tasks and plans through natural language interfaces, better bridging the gap between human intent and machine execution [9]–[11]. Recent work has additionally demonstrated that providing LLMs with classical planners to leverage as tools can produce significantly better plans than an LLM could on its own [7].

Our work expands upon, and addresses a specific gap within this state-of-the-art: integrating an LLM-powered interface with a combined scheduling and motion planning architecture, thus allowing users to flexibly solve multi-constraint temporospatial planning problems. Additionally, our work is among very few existing papers to rigorously validate such an LLM-planning approach from a human-factors perspective, considering subjective and objective metrics in a human-subjects study to determine user preferences [12].

Our work advances the technical capabilities of planning systems while also providing concrete evidence of the usability and preference for LLM-based interfaces in real-world robotic scheduling and motion planning scenarios (Figure 1). These findings pave the way for more intuitive and accessible planning systems. Our contributions are three-fold:

- We design a language-to-spatiotemporal planning interface, mapping unstructured user-input natural language to formal planning specification.
- We formulate a novel neurosymbolic architecture bridging LLM-generated constraints and cost functions with a MILP scheduler and A* motion planner.

*Corresponding author: matthew.luebbers@gatech.edu
All authors are affiliated with the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA.
This work was supported by ONR grant N00014-23-1-2887 and Naval Research Lab grants N00173-21-1-G009 and N00173-25-1-0050.

- Through a 1×3 within-subjects study ($n = 30$), we show that our LLM-planning interface and architecture reduces cognitive workload ($p < .0001$) and improves usability ($p < .0001$) over manual control and classical template-based constraint specification, while maintaining similar path optimality and reducing planning time ($p = .0027$) compared to plans generated via templated constraints, using the same MILP and A* backend.

II. Related Work

A. Scheduling and Motion Planning

Several frameworks exist to tackle the various interrelated problems of scheduling, task planning, task allocation, and motion planning. While these approaches excel at autonomously finding solutions to complex spatiotemporal planning problems, they typically require rigid and precise specifications, including constraints, heuristics, and cost functions, ultimately requiring expert knowledge and engineering support to extend to previously unseen environments or mission requirements.

In this work, we focus on high-level, discrete ordering and planning of tasks (scheduling) and low-level continuous planning of motions (motion planning). Though traditional approaches often treat scheduling and motion planning as separate problems, recent research in task and motion planning (TAMP) [13], [14] has shown the benefits of considering these problems jointly [15], [16]. Our approach, while not a full TAMP solution, takes advantage of the interconnected nature of scheduling and motion planning by providing users with a flexible natural language interface for generating specifications that can influence both aspects simultaneously.

B. LLM Frameworks

Recent Large Language Models (LLMs), pre-trained on web-scale text data, have demonstrated impressive capabilities in reasoning [17], planning [18], [19], code-writing [20], multi-agent coordination [21], and human-robot interaction, mapping natural language to robotic planning specifications. Early efforts directly used the LLM to generate plans, such as ReAct [9] and Inner Monologue [22]. These methods did not utilize underlying world models or planners, directly relying on in-context learning within the language model to generate plans. The next batch of works explored allowing LLMs to reason alongside underlying tools and functions, where rather than directly planning, the LLMs could generate code or call upon external tools to accomplish the task. Works such as Voyager [10], DEPS [11], LLM3 [23] and AutoTAMP [7] show promising performance on related planning problems. In this paper, we take inspiration from these approaches, leveraging the code-writing abilities of LLMs to parameterize a traditional scheduler and motion planner stack, thus enabling flexible specification by novice users of complex multi-constraint spatiotemporal planning problems.

While many works operate under the assumption that natural language interfacing with LLMs enhances usability, very few have conducted user studies to confirm this hypothesis. Prior related works, such as Abugurain et al. [24] focus on plan correctness and evaluation using offline datasets, rather than running a human-subjects study. A closer parallel to our work is SymPlan+ [12], which introduced an LLM-assisted symbolic planner for calendar scheduling tasks. That work included a user study demonstrating the LLM approach achieved comparable accuracy to symbolic scheduling. Our work extends beyond this, integrating multi-constraint motion planning into the scheduling problem, while additionally considering human-factors measures such as usability, workload, and planning time in our user study, with comprehensive statistical analysis.

III. Domain

We evaluate our approach in a simulated maritime route-planning environment consisting of locations $i \in \{1, \ldots, n\}$ (Figure 2), where the travel time between each pair of locations $< i, j >$ is denoted as $t_{i,j}$. The objective is to minimize the overall makespan while satisfying user-specified spatial and temporal constraints. We selected this domain because the geographic layout and variable weather effects are simple to convey to participants, while still reflecting the challenges of multi-constraint spatiotemporal planning.

Our study area includes four types of locations: cities, coral reefs, fishery sites, and oil rig sites, along with variable poor weather regions. The cities and reefs are static, placed at their true geographic coordinates in the Gulf of Mexico. The fishery and oil rig sites are randomly generated for each map and do not correspond to real-world locations. Each fishery and oil rig is labeled in $\{A, B, \ldots, E\}$ to disambiguate sites for route-planning (e.g., "fish A" or "oil B").

This challenging multi-objective planning domain blends spatial and temporal constraints with variable environmental effects. This maps not only to real-world naval operations, but also to similar domains and problem specifications, such as routing an autonomous delivery vehicle or ambulance, or specifying an integrated plan for robotic site inspection.

IV. Method

In this section, we describe our novel architecture, combining an underlying MILP scheduler and A* motion planner implementation with an LLM interface, capable of translating unstructured natural language provided by the user into constraints and cost functions for execution by the spatiotemporal planning stack.

A. Scheduling MILP Formulation

The role of the scheduling algorithm is to determine the order that certain locations should be visited to minimize the makespan z. We formulate a mixed-integer linear program (MILP) to model precedence constraints, travel times, visit requirements, and time windows, enabling us to encode diverse user-specified constraints. Precedence constraints, $\mathcal{P}$ ensure that one location is visited before another. Travel time constraints, $\mathcal{T}$ ensure that the agent has enough time to traverse between two locations. Finally, earliest start time and latest end time allow the scheduler to account for visiting certain locations during certain times of the day. The MILP

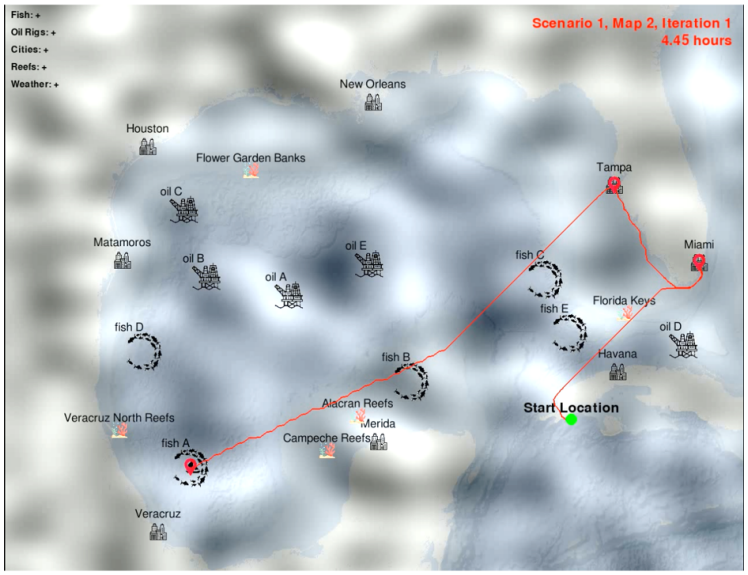

Fig. 2. Maritime route-planning domain: Our domain consists of four separate location types; fishing locations, oil locations, coral reefs, and cities. Our planning environment also models meteorological impacts, with darker regions of the map indicating poor weather.

formulation was inspired by Q-ITAGS [15]. We cover the specific parameters, variables, objective, and constraints used in our MILP formulation. The parameters define the input data for our scheduling problem, while the variables are what our MILP solver will determine. The constraints define the relationship between the given parameters and variables.

a) Parameters:

$\mathcal{T}$	Set of locations
t_{ij}	Travel time between locations i and j
$\mathcal{P}$	Set of precedence constraints (i, j) indicating location i must be visited before location j
V	Set of visit requirements (k, S) indicating at least k locations in subset S must be visited
E_i	Earliest start time for location i
L_i	Latest end time for location i

b) Variables:

s_i	Start time of task i
$v_i \in \{0, 1\}$	Binary variable indicating if task i is visited
z	Makespan

c) Objective:

$$\min z \tag{1}$$

d) Constraints:

$$z \geq s_i + d_i - (1 - v_i) \cdot \beta \qquad \forall i \in T \tag{2}$$
$$s_j \geq s_i + d_i + t_{ij} - \beta \cdot p_{ij} \qquad \forall (i, j) \in P \tag{3}$$
$$v_i \geq v_j \qquad \forall (i, j) \in P \tag{4}$$
$$\sum_{i \in S} v_i \geq k \qquad \forall (k, S) \in V \tag{5}$$
$$s_i \geq E_i - (1 - v_i) \cdot \beta \qquad \forall i \in T \tag{6}$$
$$s_i + d_i \leq L_i + (1 - v_i) \cdot \beta \qquad \forall i \in T \tag{7}$$
$$s_i \geq s_j + d_j + t_{ji} - \beta \cdot (1 - p_{ij}) \qquad \forall i \neq j \in T \tag{8}$$
$$s_i \geq 0 - (1 - v_i) \cdot \beta \qquad \forall i \in T \tag{9}$$

The model's constraints are defined in Equations (2)-(9). Equation (2) ensures the makespan z is at least as large as the end time of any visited location. Precedence constraints are enforced by Equation (3), which ensures that if location i must be visited before j, then j's start time is after i's end time plus travel time. Equation (4) maintains logical consistency in precedence relationships. Visit requirements are addressed in Equation (5), ensuring that for each requirement (k, S), at least k locations from subset S are visited. Time window constraints are handled by Equations (6) and (7), which ensure visits occur within allowed time frames. Equation (8) prevents overlap between visits to different locations. Finally, Equation (9) ensures non-negative start times for visited locations. In all relevant constraints, the term $(1 - v_i) \cdot \beta$ allows the constraint to be ignored for unvisited locations, where β is a sufficiently large constant.

B. Motion Planner

Once the scheduler has decided which locations should be visited in which order, these are passed as input to the motion planner, which traverses between points on the map. We discretize the map by placing a grid of nodes, spaced 10 pixels apart, on obstacle-free spaces. All destination locations (fishing points, oil rigs, cities, and coral reefs) connect to the nearest nodes. A^* [25] is chosen as the underlying motion planner due to its fast runtime and ability to support custom cost functions. Due to the large scale of the map, spanning hundreds of kilometers, fine-grained ship dynamics are abstracted from the domain.

C. Natural Language Interface

The natural language interface is tasked with parameterizing the scheduler and motion planner from user-specified language prompts. The interface is developed within the AutoGen conversational agent framework [26], featuring three

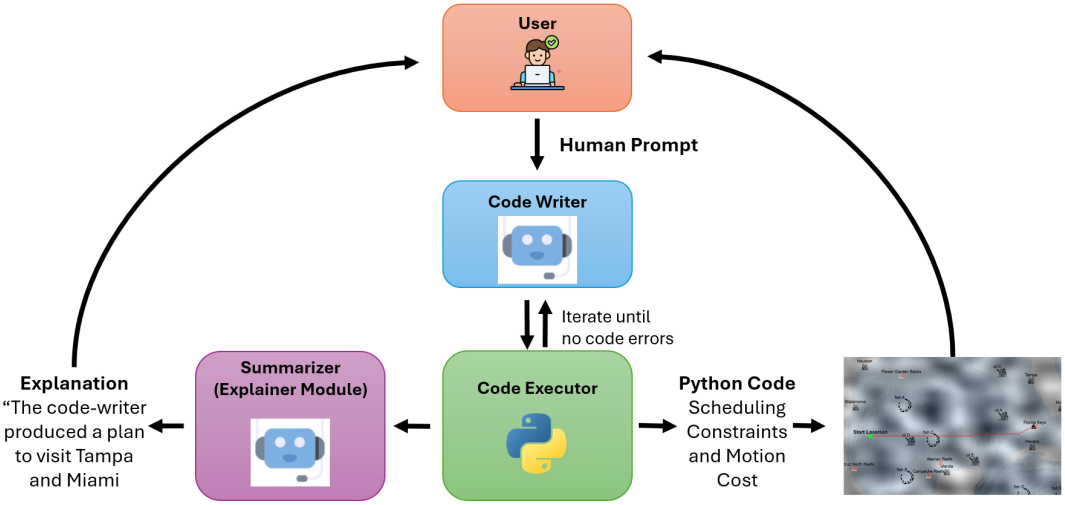

Fig. 3. Summary of Agentic Language Model Design: The user provides an initial prompt that the code writer converts into raw Python code. If any Python errors occur, they get sent back to the code writer to fix. Finally, when the code runs, the result of the plan is executed in the environment. A summarizer module takes the output of the code and explains to the user what the code does, such that they can debug it efficiently when an error occurs.

"agents" that work together: a code-writing agent, a code-executing agent, and a summarizing agent. This framework facilitates constraint translation, bridging unstructured, user-generated natural language with executable MILP and A* code. The interface uses GPT-4 as its underlying model due to its robust code-writing capabilities as of the time of implementation. Future work will see how performance extends to current state-of-the-art LLMs.

The code-writer agent (blue block in Fig. 3) is the primary LLM-driven agent tasked with generating the motion plan. We give the code-writer access to both the scheduler and motion planner, allowing the agent to leverage these tools rather than relying on the LLM itself to conduct the planning end-to-end. Therefore, the LLM only has to specify the precedence constraints and visit requirements to the scheduler. For the motion planner, we allow the LLM to generate any cost function, so long as it is written in Python. This way, the LLM has flexibility in the types of paths it can generate, while responding to the user's commands. Finally, the code-writer can call functions from the sim environment to retrieve the positions of all locations and features on the map (including weather and coastline information).

The code-executor agent (green block in Fig. 3) acts as an intermediary between the code-writer and the underlying simulator. It executes the code and reports any errors that occur. If any errors are detected, the code writer is automatically sent the Python stack trace and re-prompted to regenerate the code, without additional input from the user. While there is no guarantee that the resulting code will execute successfully, we found empirically that almost all errors were self-corrected by the code-writing agent. If errors did occur, they were typically resolved within a few iterations of code-writing and code-executing, increasing planning time due to the additional calls to the language model. Users of the system were notified that an error occurred and saw in real-time that the model was rewriting its code.

Lastly, the summarizer agent (purple block in Fig. 3), serves to generate explanations of the code-writing agent's code to users. If the resulting plan does not meet the criteria, or the original code-writing agent hallucinated, this explanation can be leveraged by the user to adjust their natural language prompt.

V. Baseline Input Modalities

Our study evaluates two additional input modalities for multi-constraint spatiotemporal navigation and task planning. We selected ablations of our LLM-planning architecture to demonstrate the spectrum of planning approaches, from direct human control to structured interfaces. The "Templated Constraint Interface" modality ablates the LLM interface from our approach, requiring users to manually input mission requirements from a pre-set menu UI, while retaining automated planning. The "Manual Control" modality further ablates automated planning via MILP and A*, requiring users to manually route their ship. The key advantage of our approach, the LLM-based "Natural Language Interface" modality, lies in its ability to dynamically generate code for novel constraints without pre-implementation, a capability not feasible with traditional approaches. This allows users to specify new requirements without engineering support, addressing a fundamental limitation of conventional UI systems where each constraint requires explicit implementation.

a) Manual Control: As a baseline modality, users manually route the ship without access to either the scheduler or motion planner. This represents the status quo in most real-world naval planning domains. In this modality, users navigate the ship by left-clicking on the map, prompting the ship to travel in a direct line toward the selected destination. No intelligent path planning is enabled, resulting in potential obstructions by land masses directly in the ship's path. Users must manually circumvent these obstacles while controlling the ship.

b) Templated Constraint Interface: The second baseline modality (Figure 4) implements a classical interface for inputting relevant constraints to specify the planning problem, comprising three sections: *Precedence Requirements*, *Visit Requirements*, and *Motion Requirements*. Precedence and visit requirements integrate into the MILP scheduler, whereas motion requirements call pre-built cost functions within the A* motion planner. This traditional UI effectively represents a basic form of templated language through its pre-built options for constraint types and locations. Although this UI offers more automation than manual control, its capability to generate diverse plans is inherently limited. It is infeasible to define a priori a comprehensive list of templates to accommodate all conceivable plans. New spatial relationships or custom motion constraints would require explicit engineering implementation and continual support.

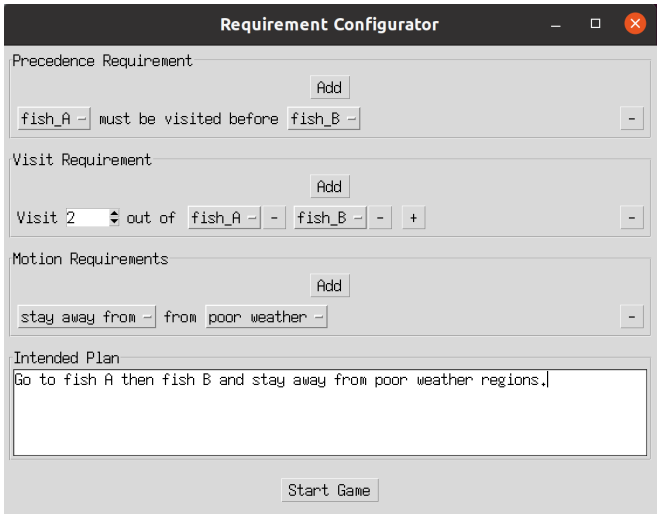

Fig. 4. The templated constraint interface is organized into three sections for specifying precedence, visit, and motion requirements.

VI. User Study Design

In this section, we describe our within-subjects user study to characterize the effects of input modality by comparing three conditions: (1) **"Manual Control,"** where users directly click to move the ship; (2) the templated constraint interface **("Templated"),** where users specify precedence, visit, and motion constraints through menus, which are then solved by the MILP scheduler and A* planner; and (3) our natural language interface **("LLM"),** where users issue free-form text commands, translated via LLM into code parameterizing the same MILP scheduler and A* planner.

a) Procedure: The study employed a 1 x 3 within-subjects design, where each participant engaged with all three input modalities in a randomized sequence. The experiment comprised five scenarios, each associated with distinct scheduling and motion planning specification that participants were required to replicate in the game. Instead of verbal instructions, participants were shown unlabeled video examples of target specifications to prevent exact replication from a written prompt, especially in the natural language condition. This approach encouraged participants to independently formulate their specifications. Each of the five scenarios had three different variants, resulting in a total of 15 unique scenarios. The five scenario types included:

Scenario 1: *Only Visit Constraints* – Visit specified locations in any order (e.g., Visit Oil Rig A and Tampa in any order.)

Scenario 2: *Visit + Precedence Constraints* – Visit a set of locations in a specific sequence (e.g., Visit Oil A, B, and C, in that order.)

Scenario 3: *Nearest Location* – Visit the closest location from the starting point.

Scenario 4: *Furthest Location + Motion Constraints* – Visit furthest location while satisfying motion constraints (e.g., Visit the furthest coral reef while staying as close to the coast as possible.)

Scenario 5: *Directional Constraints* – Visit all locations on a specific side of the starting point (e.g., Visit all cities east of the starting location.)

Scenarios 1–3 are readily encoded with the templated UI using predefined menus. In contrast, Scenarios 4 and 5 introduce constraints, e.g. "furthest location" and "locations on a certain side of the starting point," that the templated UI cannot directly represent. These scenarios reflect real-world deployments where supporting new mission concepts would require ad hoc, manual engineering to implement additional cost functions and constraint logic. Without these extensions, users are often forced to either specify lengthy sets of compositional constraints that approximate the desired specification, fall back on manual control, or accept solutions that fail to meet their intent. As such, we included these scenarios to evaluate how well the natural language interface handles novel constraints that cannot be captured in advance.

After watching the videos produced by the ground truth specification, participants interacted with the assigned input modality to recreate the specification. They repeated the specification for three maps in each scenario and were allowed a maximum of five attempts per map. Upon completing all five scenarios, participants completed a NASA TLX workload assessment [27] and a System Usability Scale survey [28]. The ordering of scenarios within each condition was additionally counterbalanced between participants.

VII. Hypotheses

We propose four hypotheses, **H1-H4**, to evaluate the effects of input modality on planning time, path optimality, usability, and workload.

H1 – The **LLM interface** will reduce participant planning time, compared to templated constraint specification.

H2 – The **LLM interface** will improve the optimality of user-generated paths, compared to manual control.

H3 – The **LLM interface** will be rated as the most usable among all the conditions.

H4 – The **LLM interface** will be rated as having the lowest workload among all the conditions.

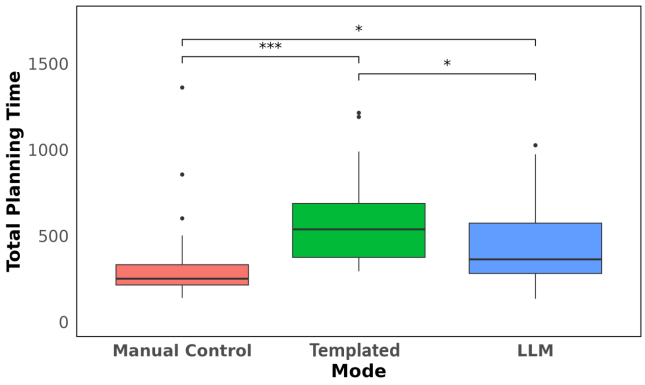

Fig. 5. Planning Time: The LLM condition significantly reduced total planning time compared to the Templated condition. Participants in the Manual Control condition spent the least time planning overall. Significance levels are indicated as $^{*}P \leq .05$,** $P \leq .01$,*** $P \leq .001$. Outliers are determined by values beyond $1.5\times$ the interquartile range from the quartile boundaries.

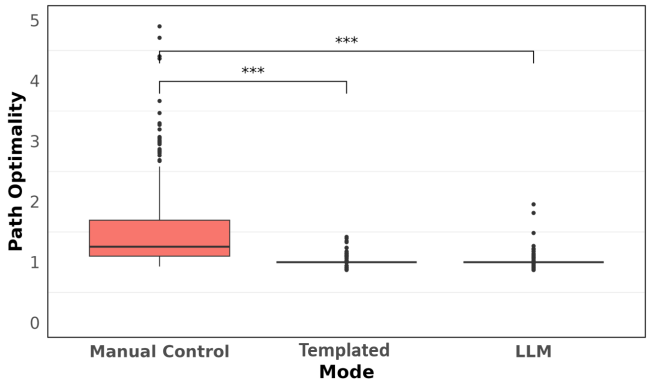

Fig. 6. Path Optimality: Participants in the Manual Control condition produced significantly less optimal paths than in both the Templated and LLM conditions. Significance levels are indicated as $^{*}P \leq .05$,** $P \leq .01$,*** $P \leq .001$. Outliers are determined by values beyond $1.5\times$ the interquartile range from the quartile boundaries.

VIII. Results

We recruited 30 participants under an IRB approved protocol (ages $19-33$, $M = 25.0$; $SD = 2.665$; Female 36.6%). We report the results of our study and establish statistical significance at the $\alpha = .05$ level. We conducted multiple linear regressions with mixed effects to test our hypotheses and linear mixed effect analyses for each dependent variable. We included input modality as the primary fixed effect and controlled for age, race, gender, and education to account for potential demographic influences on user performance and interaction. This approach allows us to isolate the effect of input modality while controlling for known sources of variation, ensuring our results are more generalizable and robust across diverse user groups. Random intercepts were used for each participant. We assessed the assumptions of homoscedasticity and normality of residuals to ensure the validity of the linear mixed effects model applied. Post-hoc tests were applied to investigate specific differences between input modalities using Tukey's Honestly Significant Difference (HSD), adjusting for multiple comparisons.

A. Objective Metrics

1) Planning Time: We found a significant main effect for the total planning time used (Figure 5) between input modalities ($F(2, 56) = 19.68, p < .0001$). A post-hoc test showed that Manual Control ($M = 326.9$) requires less time than both Templated ($M = 638.9$) ($p < .0001$), and LLM ($M = 464.8$) ($p = .021$), and LLM requires less time than Templated ($p = .0027$). We also found an ordering effect ($F(2, 56) = 25.49, p < 0.0001$), where conditions presented first took longer to plan than both the second ($p < .0001$) and third rounds ($p < .0001$). There was no statistical difference in the total planning time taken between the second and third rounds. No interaction effects were found. Despite the apparent learning effect, as condition ordering was randomized between participants, the main effect still holds, supporting hypothesis **H1**.

2) Path Optimality: We define path optimality as the ratio of path time for user-generated paths over path time for the target path shown in the video specification (with a ratio of 1.0 being perfectly optimal). The optimality of the paths produced by the users had a significant effect (Figure 6) between input modalities through the Kruskal-Wallis test ($\chi^2(2) = 727.57, p < .0001$). A post-hoc Dunn's test with Bonferroni adjustment found that Manual Control ($M = 1.62$) was worse than the Templated ($M = 1.01$) ($p < .0001$) and LLM ($M = 1.01$) ($p < .0001$) conditions. These results support hypothesis **H2**.

Combined, these objective results show that participants are able to achieve similar path optimality using our LLM-planning architecture to traditional templated constraint specification, while requiring significantly less planning time.

3) Post-Hoc Error Analysis: To ensure plan validity and filter out errors during analysis, we applied post-hoc filtering to exclude incorrect plans. One potential source of error stemmed from participants misinterpreting the intended plans, particularly when their assumptions conflicted with the rules in the demonstration videos. For instance, if a video instructed participants to visit all cities east of the starting location, but a participant instead visited western cities, their plan was deemed invalid. Of the 450 scenarios executed (30 participants $\times$ 5 scenarios $\times$ 3 modes), 24 were invalid, constituting approximately 5.3% of all scenarios. The distribution of errors across conditions was relatively even, with 7 errors in the Manual Control condition, 9 in the Templated condition, and 8 in the LLM condition.

A second source of error arose from either LLM hallucinations or user input mistakes. These errors are measured by the number of iterations required for the user to submit a plan, as they either correct their own mistake (in the Manual Control or Templated conditions) or request the LLM to correct a mistake (due to hallucinations or their own mistake). In the LLM condition, hallucinations typically involved misinterpreting conjunctions (e.g., interpreting "*visit A and B*" as "*visit A or B*") or exhibiting incorrect spatial reasoning. In

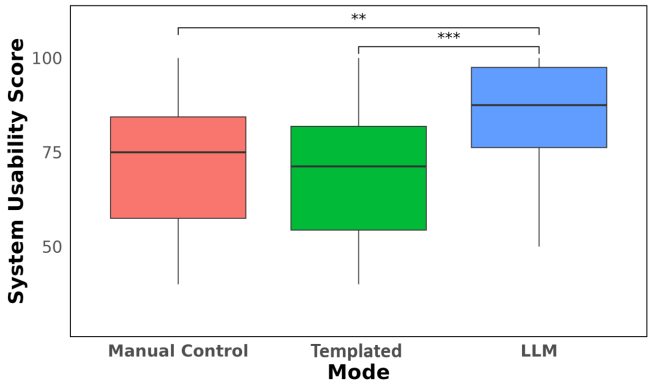

Fig. 7. System Usability: The LLM condition was rated as significantly more usable than the Templated and Manual Control conditions. Significance levels are indicated as $^{*}P \leq .05$,** $P \leq .01$,*** $P \leq .001$. Outliers are determined by values beyond $1.5\times$ the interquartile range from the quartile boundaries.

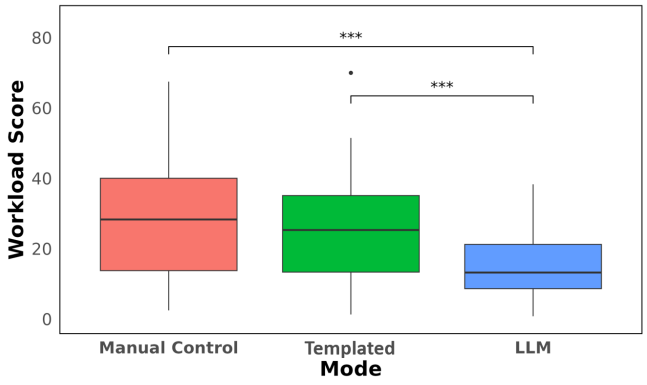

Fig. 8. Workload: The LLM condition was rated as requiring significantly less workload compared to the Templated and Manual Control conditions. Significance levels are indicated as $^{*}P \leq .05$,** $P \leq .01$,*** $P \leq .001$. Outliers are determined by values beyond $1.5\times$ the inter-quartile range from the quartile boundaries.

the Templated condition, errors often occurred when users mistakenly selected incorrect constraints or locations from dropdown menus. We then measured the average number of iterations required before users accepted a plan. The Templated condition required the most iterations (3.13 ± 3.15), followed by the Manual Control (2.67 ± 1.95) and LLM conditions (1.90 ± 1.45). Our method therefore generated trajectories that aligned with the users expectation despite occasional hallucinations.

B. Subjective Metrics

1) Usability: As shown in Figure 7, we found a significant main effect for perceived usability across the input modalities ($F(2, 56) = 11.06, p < .0001$). Participants rated the LLM condition ($M = 84.0$) as significantly more usable than both the Templated ($M = 69.4$) ($p < .0001$) and Manual Control conditions ($M = 73.5$) ($p = .005$). These results support hypothesis **H3**.

2) Workload: As shown in Figure 8, results showed a significant main effect for perceived workload as a function of the input modality ($F(2, 56) = 15.959, p < .0001$). Participants rated the LLM condition ($M = 15.8$) as requiring significantly less workload than the Templated ($M = 26.4$) ($p = .0005$) and Manual Control conditions ($M = 30.1$) ($p < .0001$). These results support hypothesis **H4**.

Combined, these subjective results indicate an overall user preference for leveraging the autonomy afforded by the MILP scheduler and A* path planner, while interfacing with an intuitive natural language frontend.

IX. Discussion

The results of our study demonstrate the potential of natural language interfaces powered by large language models (LLMs) to significantly improve human-autonomy interaction in complex planning tasks. The LLM-based interface not only enhanced usability and reduced workload but also maintained path optimality while decreasing planning time. These findings have several important implications for the field of human-robot interaction and autonomous systems.

First, the reduced workload and increased usability of the LLM interface suggest that natural language interaction can lower the cognitive barriers associated with complex planning tasks. This is particularly important in high-stress or time-sensitive scenarios. By offloading the technical details of schedule and motion plan parameterization to an LLM, users can focus on high-level strategies and decision-making.

We observed a significant reduction in planning time for the LLM condition compared to the Templated condition. This efficiency gain could translate to faster response times in real-world applications, potentially improving outcomes in time-critical scenarios. While the total planning time was shorter for Manual Control compared to both the Templated and LLM conditions, users compensated by spending more time executing the path itself, leading to delays during the actual motion of the ship. Often, users would reach destinations and leave the ship stopped to plan their path to the next location, instead of following a seamless, continuous motion plan. Furthermore, scenarios requiring users to navigate close to coastlines frequently cost additional time, as users would get stuck on land regions, spending extra recovery time maneuvering away from obstacles.

Our study supported all four hypotheses, **H1-H4.** Overall, these results suggest that our technique integrating an LLM-powered natural language interface with an MILP scheduler and A* path planner can improve usability, lower cognitive effort, and lower planning time compared to alternative input modalities, all without sacrificing plan quality, potentially making advanced spatiotemporal planning tools more practical for a broader range of users.

X. Limitations and Future Work

While our results are promising, it's crucial to address the issue of LLM hallucinations. In our experiment, we observed occasional instances where the LLM generated incorrect plans. These hallucinations occurred in approximately 5% of plan specification attempts. One of the most common sources of hallucinations was the LLM misinterpreting logical conjunctions in user-specified constraints. For example, if

a user specified, "Go to Fish A, B, C", the LLM sometimes interpreted this as "Go to *either* Fish A, B, or C" rather than "Go to Fish A, B, *and* C" which the user intended. Note that these LLM-introduced hallucinations are confined to the plan specification layer, with successfully generated spatiotemporal plans abiding by formal guarantees. Still, there is a need for robust error-checking mechanisms and user verification in real-world deployment of such systems.

Future work could explore avenues for addressing hallucinations. Some users indicated preference for the templated interface, as it offered full visibility into the parameters driving the spatiotemporal planning stack. A hybrid approach could combine the strengths of the traditional templated UI with the strengths of natural language interactions. Perhaps the language model could specify common constraints, but also dynamically add additional UI elements for capabilities the users ask for during interaction.

XI. Conclusion

We developed a novel LLM-planning architecture for multi-constraint spatiotemporal planning, combining an LLM-powered natural language interface with a MILP scheduler and A* planner. This interface is capable of interpreting unstructured natural language specifications, translating them into actionable plans, and providing live explanations to users, providing an intuitive and flexible interaction modality for complex and dynamic planning tasks. Through a user study, we showed that our LLM-planning approach significantly reduces perceived workload, improves usability, and increases planning efficiency, highlighting the potential usefulness of such architectures for users specifying real-world spatiotemporal planning tasks, in domains spanning from robotics to logistics to maritime planning.

References

[1] M. Gombolay, A. Bair, C. Huang, and J. Shah, "Computational design of mixed-initiative human–robot teaming that considers human factors: situational awareness, workload, and workflow preferences," *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 597–617, 2017. [Online]. Available: https://doi.org/10.1177/0278364916688255

[2] R. Paleja, A. Silva, L. Chen, and M. Gombolay, "Interpretable and personalized apprenticeship scheduling: Learning interpretable scheduling policies from heterogeneous user demonstrations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6417–6428, 2020.

[3] J. M. Framinan, R. Leisten, and R. R. García, "Manufacturing scheduling systems," *An integrated view on Models, Methods and Tools*, pp. 51–63, 2014.

[4] A. Yamashita, T. Arai, J. Ota, and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 223–237, 2003.

[5] E. Kondili, C. C. Pantelides, and R. W. Sargent, "A general algorithm for short-term scheduling of batch operations—i. milp formulation," *Computers & Chemical Engineering*, vol. 17, no. 2, pp. 211–227, 1993.

[6] J.-C. Latombe, *Robot motion planning*. Springer Science & Business Media, 2012, vol. 124.

[7] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "Autotamp: Autoregressive task and motion planning with llms as translators and checkers," 2024. [Online]. Available: https://arxiv.org/abs/2306.06531

[8] L. Erickson and S. LaValle, "A simple, but np-hard, motion planning problem," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, no. 1, 2013, pp. 1388–1393.

[9] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[10] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

[11] Z. Wang, S. Cai, G. Chen, A. Liu, X. Ma, and Y. Liang, "Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents," 2024. [Online]. Available: https://arxiv.org/abs/2302.01560

[12] Y. Li, N. Kamra, R. Desai, and A. Halevy, "Human-centered planning," 2023. [Online]. Available: https://arxiv.org/abs/2311.04403

[13] M. Mansouri, F. Pecora, and P. Schüller, "Combining task and motion planning: Challenges and guidelines," *Frontiers in Robotics and AI*, vol. 8, p. 637888, 2021.

[14] C. R. Garrett, T. Lozano-Perez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 104–136, 2018.

[15] G. Neville, J. Liu, S. Chernova, and H. Ravichandar, "Q-itags: Quality-optimized spatio-temporal heterogeneous task allocation with a time budget," 2024. [Online]. Available: https://arxiv.org/abs/2404.07902

[16] A. Messing, G. Neville, S. Chernova, S. Hutchinson, and H. Ravichandar, "Grstaps: Graphically recursive simultaneous task allocation, planning, and scheduling," *The International Journal of Robotics Research*, vol. 41, no. 2, pp. 232–256, 2022. [Online]. Available: https://doi.org/10.1177/02783649211052066

[17] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[18] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[19] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 523–11 530.

[20] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *arXiv preprint arXiv:2209.07753*, 2022.

[21] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," 2024. [Online]. Available: https://arxiv.org/abs/2402.01680

[22] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[23] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, "Llm3:large language model-based task and motion planning with motion failure reasoning," 2024. [Online]. Available: https://arxiv.org/abs/2403.11552

[24] M. Abugurain and S. Park, "Integrating disambiguation and user preferences into large language models for robot motion planning," 2024. [Online]. Available: https://arxiv.org/abs/2404.14547

[25] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[26] Q. Wu, G. Bansal, J. Zhang, Y. Wu, S. Zhang, E. Zhu, B. Li, L. Jiang, X. Zhang, and C. Wang, "Autogen: Enabling next-gen llm applications via multi-agent conversation framework," 2023.

[27] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Human Mental Workload*, ser. Advances in Psychology, P. A. Hancock and N. Meshkati, Eds. North-Holland, 1988, vol. 52, pp. 139–183. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166411508623869

[28] J. Brooke *et al.*, "Sus-a quick and dirty usability scale," *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.