To the Globe (TTG): Towards Language-Driven Guaranteed Travel Planning

Da Ju$^{\triangle,*}$ **Song Jiang*** **Andrew Cohen*** **Aaron Foss**$^{+}$ **Sasha Mitts**$^{+}$ **Arman Zharmagambetov**
Brandon Amos **Xian Li** **Justine Kao** **Maryam Fazel-Zarandi** **Yuandong Tian**$^{\triangle,*}$
$^{\triangle}$ project lead * core contributor $^{+}$ equal contribution
Meta AI (FAIR)

Abstract

Travel planning is a challenging and time-consuming task that aims to find an itinerary which satisfies multiple, interdependent constraints regarding flights, accommodations, attractions, and other travel arrangements. In this paper, we propose *To the Globe* (TTG), a real-time demo system that takes natural language requests from users, translates it to symbolic form via a fine-tuned Large Language Model, and produces optimal travel itineraries with Mixed Integer Linear Programming solvers. The overall system takes ~ 5 seconds to reply to the user request with guaranteed itineraries. To train TTG, we develop a synthetic data pipeline that generates user requests, flight and hotel information in symbolic form without human annotations, based on the statistics of real-world datasets, and fine-tune an LLM to translate NL user requests to their symbolic form, which is sent to the symbolic solver to compute optimal itineraries. Our NL-symbolic translation achieves $\sim 91\%$ exact match in a backtranslation metric (i.e., whether the estimated symbolic form of generated natural language matches the groundtruth), and its returned itineraries have a ratio of 0.979 compared to the optimal cost of the ground truth user request. When evaluated by users, TTG achieves consistently high Net Promoter Scores (NPS [6]) of 35-40% on generated itinerary.

1 Introduction

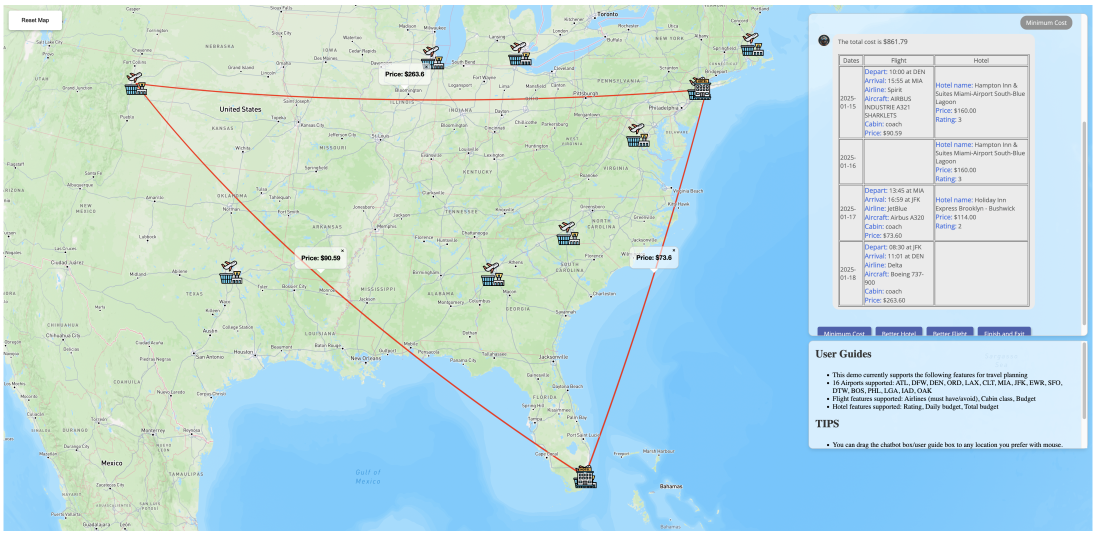
Travel planning is a routine activity that typically requires a significant amount of human time and effort to find an optimal itinerary satisfying many implicit and explicit constraints which interact and change over time. Ideally, a human would only need to provide natural language instructions (e.g., *"I want to go to Hawaii for three days with a budget of \$1,000."*) and an AI agent provides solutions which are optimal with respect to certain objectives (e.g., total expense) and feasible (i.e., satisfy all constraints). Moreover, the quality of the agent's decision should be reliable enough such that humans can fully delegate the task, or approve with a glimpse of check.

Designing such an AI system remains non-trivial. First, it involves *complex planning* with potentially vague natural language instructions, sophisticated objectives and constraints (e.g., hotels, flights, restaurants, attractions, budgets) and requiring multiple back-and-forth reasoning steps without a clear and predefined decision path. Despite impressive performance achieved by Large Language Models (LLMs), they are still weak at complex reasoning and planning [23, 28, 10], and may hallucinate [11] or be inconsistent [10], in particular during complicated reasoning. This raises concerns on whether its decision can be trusted [21]. Second, travel planning is a *time-dependent* task that requires constantly re-planning due to ever-changing situations. Even with exactly the same request, the optimal itinerary may be different given varying prices and availability. Third, such a decision is highly *personalized* depending on the private constraints and preferences. Users may speak a few brief words and expect the agent to give a solution that satisfies all of their implicit constraints, which can be quite subtle to capture from past conversations.

In this paper, we propose TTG, a demo system that takes natural language instructions from users and outputs optimal travel itineraries in seconds. To achieve this, our system leverages the power of LLMs and existing symbolic solvers, e.g., Mixed Integer Linear Programming (MILP). It first converts natural language instructions into a symbolic representation, which is solved by the symbolic solver, and then replies to the user with natural language outputs and a rich visualization. Compared to pure LLM-based systems (e.g., ClaudeAtlas[1], Expedia Romie[2]), TTG provides an up-to-date, pre-

[1] https://devpost.com/software/kickass-team

[2] https://partner.expediagroup.com/en-us/innovation/labs



The total cost is $861.79

Dates	Flight	Hotel
2025-01-15	Depart: 10:00 at DEN Arrival: 15:55 at MIA Airline: Spirit Aircraft: AIRBUS INDUSTRIE A321 SHARKLETS Cabin: coach Price: $90.59	Hotel name: Hampton Inn & Suites Miami-Airport South-Blue Lagoon Price: $160.00 Rating: 3
2025-01-16		Hotel name: Hampton Inn & Suites Miami-Airport South-Blue Lagoon Price: $160.00 Rating: 3
2025-01-17	Depart: 13:45 at MIA Arrival: 16:59 at JFK Airline: JetBlue Aircraft: Airbus A320 Cabin: coach Price: $73.60	Hotel name: Holiday Inn Express Brooklyn - Bushwick Price: $114.00 Rating: 2
2025-01-18	Depart: 08:30 at JFK Arrival: 11:01 at DEN Airline: Delta Aircraft: Boeing 737-900 Cabin: coach Price: $263.60	

User Guides

- This demo currently supports the following features for travel planning
- 16 Airports supported: ATL, DFW, DEN, ORD, LAX, CLT, MIA, JFK, EWR, SFO, DTW, BOS, PHL, LGA, IAD, OAK
- Flight features supported: Airlines (must have/avoid), Cabin class, Budget
- Hotel features supported: Rating, Daily budget, Total budget

TIPS

- You can drag the chatbot box/user guide box to any location you prefer with mouse.

Figure 1: Front-end interface of TTG. Users send their natural language requests to the demo system (TTG), and TTG replies with itineraries that satisfy user constraints and is optimal with respect to various criteria (e.g., minimal cost).

cise and executable itinerary with guarantees, in almost real time ($\sim$ 5 seconds per request).

2 Related Works

LLM for reasoning/planning. Teaching LLMs to do well in reasoning and planning tasks remains a challenging problem, even for SoTA LLMs [31, 15]. Previous works like CoT [24], ToT [29], ReAct [30], Reflexion [18], using synthetic data [17, 16], and multi-agent frameworks [26, 9] improve the reasoning power of LLMs in complicated problems but still cannot guarantee feasibility and optimality [28, 27]. More importantly, due to the black-box nature of LLMs, it remains an open problem on understanding failure modes in reasoning [25, 4], let alone generate guaranteed results that can be trusted by users.

Hybrid System of LLM and Solvers. Combining symbolic solvers with LLMs has been explored in many abstract planning (e.g, [14, 19, 20]) and real-world planning scenarios [22]. For travel planning, [8, 5] show that prompt engineering in pre-trained language models can be used to generate code (or symbolic specification) to invoke symbolic solvers such as formal verification tools like SMT [3] solvers, or A^* [16], to solve travel planning problems (e.g., [28]). In contrast, our TTG chooses to focus on real-world travel planning that may last for multiple days with realistic constraints (Table 1). TTG uses JSON format as symbolic specification because it has much simpler structures than generated codes, and can be guaranteed by constrained generation techniques using finite state

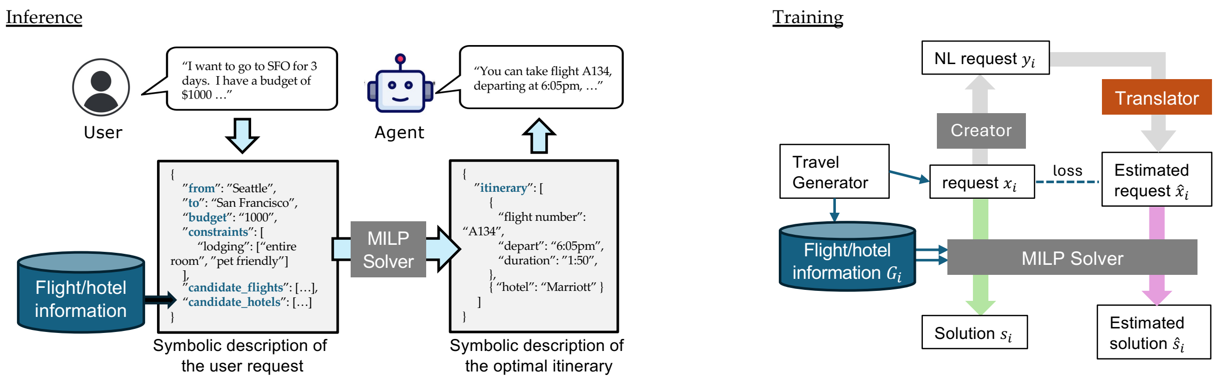
Item	Description
Airline Constraints	price range, (soft) departure and arrival constraints, cabin class, refundablity, non-stopness, plane type, airline preferences.
Hotel Constraints	price range, rating, brands.
Budget Constraints	Total budget, everyday budget.

Table 1: Factors considered in travel request generation.

machine (FSA) [7], which makes self-consistency-based verification, benchmarking, and training easier (see Sec. 5 for details). This also makes TTG independent of the specific solver (e.g., SCIP [2], Gurobi, etc.) and language used to solve the underlying MILP problem. Instead of prompt engineering, TTG also does model fine-tuning with thorough performance evaluation, including self-consistency and a thorough human study with $\sim$ 1.3k participants, which is not provided in previous works.

3 Overview of TTG

Fig. 1 shows the front-end interface of TTG. Users can obtain travel itineraries in a few seconds by sending natural language requests in the semi-transparent dialog box. Users can also visualize candidate itineraries on the rendered map of the globe, and select based on their preferences. We use a hybrid design leveraging both LLMs and symbolic solvers that can deal with natural language input and still guarantee the feasibility and optimality of the output itineraries, if user requests are translated correctly by the fine-tuned LLM.


Inference
User
"I want to go to SFO for 3 days. I have a budget of $1000 …"
Agent
"You can take flight A134, departing at 6:05pm, …"
Flight/hotel information
MILP Solver
Symbolic description of the user request
Symbolic description of the optimal itinerary
Training
NL request y_i
Translator
Creator
Travel Generator
request x_i
loss
Estimated request $\hat{x}_i$
Flight/hotel information G_i
MILP Solver
Solution s_i
Estimated solution $\hat{s}_i$

```
{
  "from": "Seattle",
  "to": "San Francisco",
  "budget": "1000",
  "constraints": [
     "lodging": ["entire
room", "pet friendly"]
  ],
  "candidate_flights": […],
  "candidate_hotels": […]
}
```

```
{
  "itinerary": [
     {
        "flight number":
"A134",
        "depart": "6:05pm",
        "duration": "1:50",
     },
     { "hotel": "Marriott" }
  ]
}
```

Figure 2: Overview of the workflow of `TTG`. **Inference:** our system first translates the user travel request in natural language (NL) into the symbolic description of a Mixed Integer Linear programming (MILP) solver using a fine-tuned Large Language Model (LLM), calls the solver to find its optimal solution that satisfies all constraints, and then returns the itinerary in natural language. **Training**: `TTG` has three components. A *Travel Generator* that generates flight/hotel information training data based on real-world data, and symbolic user request x_i. An *Instruction Translator* a pre-trained LLM fine-tuned to translate the NL user request y_i to its symbolic form $\hat{x}_i$, learned by self-consistency between the groundtruth request x_i and the estimated user request $\hat{x}_i$. A *Travel solver* that solves the estimated symbolic request $\hat{x}_i$ and yields the estimated solution $\hat{s}_i$.

Fig. 2 shows the overview of the `TTG` workflow. The components are: (1) A *Symbolic Travel Generator* which generates available flight and hotel information G_i using existing real-world data as well as symbolic user requests x_i (both in JSON format) where i is the sample index. (2) *Instruction Creator and Translator* that converts a user request x_i from JSON to a natural language (NL) request y_i, and a translator to convert the NL requests y_i back to its symbolic form in JSON $\hat{x}_i$ (Sec. 4.1). (3) *Travel Solver*, a Mixed Integer Linear Programming (MILP) solver that solves the underlying combinatorial optimization in its symbolic form, parameterized by (x_i, G_i), and gives the optimal solution s_i. That is, $s_i = \arg\min_{s'} f(s'; x_i, G_i)$, where $f > 0$ is the cost function to be minimized. During the user interaction, the solver only has access to an estimate of the user request $\hat{x}_i$, and the corresponding solution $\hat{s}_i = \arg\min_{s'} f(s'; \hat{x}_i, G_i)$.

4 Methodology

4.1 Symbolic Travel Generator

Since the existing TravelPlanner dataset [28] has a limited number of samples and does not provide symbolic grounding of user requests, we created our own Travel Generator that generates user requests and the corresponding flight and hotel information in symbolic form.

Travel Request. We consider a variety of variables when generating travel requests (see Table 1 for a complete list). We mostly consider round trips of 2 or 3 cities (1 or 2 stops) over multiple days (include <5% one-way for diversity). We randomly sample values for the constraints in Table 1 and prompt Llama-3 70B [1] to convert the symbolic representation into natural language. We generate 238k training samples and 29.8k test samples.

As is common in synthetic data generation with LLMs [12], there was some degree of inconsistency between the symbolic representation and generations, primarily in the ordering of departure and return dates. We again prompted Llama-3 70B to filter samples with this inconsistency as a few-shot task, removing approximately 27% of samples leaving 173.7k training and 21.8k test samples.

Generated Flight and Hotel information G_i. We use the flight price dataset[3] which contains existing real-world, one-way US domestic flight information from Expedia from Apr. 16, 2022 to Oct. 5, 2022 to build our travel generator. We replicate the data to cover a longer time frame. For hotels, we include public information and then add noise to prices, departure/arrival dates, and other attributes. We combine the two to create synthetic flight and hotel information G_i.

4.2 Travel Solver

We build a combinatorial solver to compute optimal solutions to the MILP formulation of the travel planning problem using SCIP [2]. We discretize the time into T slots, over the travel span (e.g., 3 days). A traveller is at location l at time t if and only if the corresponding variable $u_l(t) = 1$. The traveller must maintain location continuity and cannot teleport unless some event happens: $e(t) =$

[3]https://www.kaggle.com/datasets/dilwong/flightprices

$0 \Rightarrow u_l(t+1) = u_l(t)$. A traveller may be sleep at time slot t, which is represented as $m(t) = 1$. A hotel j (or a flight j) is booked if $h_j = 1$ (or $f_j = 1$). All the variables are binary.

To make sure the resulting solution is feasible, we impose the following three types of constraints.

Commonsense constraints. The traveller can only be present at a single location at time t, which means $\sum_l u_l(t) = 1$. The traveller needs a minimal L time slots per day, which can be represented as $\sum_{t\in[\text{day evening}]} m(t) \geq L$.

Flight constraints. If the traveller takes the flight j (i.e., $f_j = 1$) that departs from location src to location dst, then the following constraints should be satisfied:

$$f_j = 1 \Rightarrow \begin{cases} u_{\mathrm{src}}(t_{\mathrm{dep}}) = 1, u_{\mathrm{air}}(t_{\mathrm{dep}}+1) = 1 \\ u_{\mathrm{dst}}(t_{\mathrm{land}}) = 1, u_{\mathrm{air}}(t_{\mathrm{land}}-1) = 1 \\ e(t_{\mathrm{dep}}) = e(t_{\mathrm{land}}-1) = 1 \end{cases} \quad (1)$$

where, t_{dep} and t_{land} are the departure and landing time slots, and $e(t)$ is a binary variable suggesting whether there is an event happening at time slot t.

Hotel constraints. If the traveller decided to reside in hotel j (i.e., $h_j = 1$) at location l, then the following constraints need to be satisfied:

$$h_j = 1 \Rightarrow \begin{cases} u_l(t_{\mathrm{ckin}} : t_{\mathrm{ckout}}) \geq m(t_{\mathrm{ckin}} : t_{\mathrm{ckout}}) \\ m(t_{\mathrm{ckin}} : t_{\mathrm{ckout}}) \text{ allowed to be } 1 \end{cases} \quad (2)$$

where, t_{ckin} and t_{ckout} are the earliest and latest check-in and check-out times for hotel j.

Encoding ("implies" $\Rightarrow$) conditions. Note that MILP is able to encode conditional constraints (e.g., Eqn. 1 and Eqn. 2). Please check Appendix A for details.

5 Experiments

5.1 Automatic Evaluation by Self-consistency

Quality of Instruction Translator. We evaluate the quality of the generated symbolic form $\hat{x}_i$ from the Translator, by comparing with the original symbolic form x_i that is used to generate the natural language request y_i.

To compare the original symbolic user request x_i and reconstructed request $\hat{x}_i$ (both in JSON) from natural language request y_i, we use *exact match* (EM) accuracy that scores 0 if any of the entries in the two JSONs do not match. Additionally, since the Translator is generating output structured as JSON, we use vLLM logits_processors to ensure the model output is properly structured [13]. We refer to this as Constrained Decoding.

Decoding	EM Accuracy	Valid JSON
Constrained	92.0%	100.0%
Unconstrained	91.2%	99.1%

Table 2: Exact Match accuracy and validity of generations as JSON of TTG with Constrained and Unconstrained decoding on 21.8k test samples.

In Table 2, we report exact match accuracy and validity of the output as JSON for both Constrained and Unconstrained Decoding on the test set. With constrained decoding, the Translator achieves 92.0% exact match accuracy with output being valid JSON 100% of the time (because we forced it to be). Unconstrained decoding is surprisingly close to constrained decoding with an EM of 91.2% and produces valid JSON 99% of the time. We find that the filtering step discussed in Sec. 4.1 to be critical for unconstrained decoding to produce valid JSON at such a high degree. Constrained decoding is roughly 10% slower than unconstrained decoding but the 1% failure rate leads to a worse user experience, so we deploy constrained decoding in the demo.

Table 3 provides a breakdown of the errors and number of samples by the number of hotel constraints, airline constraints and cities. We point out that EM accuracy *decreases* as the number of airline constraints *increases* but is relatively robust across the number of hotel constraints and cities. We hypothesize the decreasing performance with airline constraints is due to data imbalance (i.e., there are only 173 samples with 8 constraints versus 9777 with 5 constraints) which can be addressed by changing the sampling parameters during data generation.

Hotel Constraints	2	3	4		
EM Accuracy	91.5%	92.5%	91.7%		
# samples	3345	10438	8001		
Airline Constraints	4	5	6	7	8
EM Accuracy	95.9%	92.8%	91.1%	77.0%	78.6%
# samples	4974	9777	5555	1299	173
Cities	2	3			
EM Accuracy	91.9%	93.0%			
# samples	18998	2786			

Table 3: Exact Match (EM) accuracy of TTG and the number of samples when sorting by the number of hotel constraints, airport constraints and cities. Accuracy decreases as the number of airline constraints increases but is relatively robust across the number of hotel constraints and cities.

Fig. 3 provides a breakdown of the sources of error of our model. The three major sources are

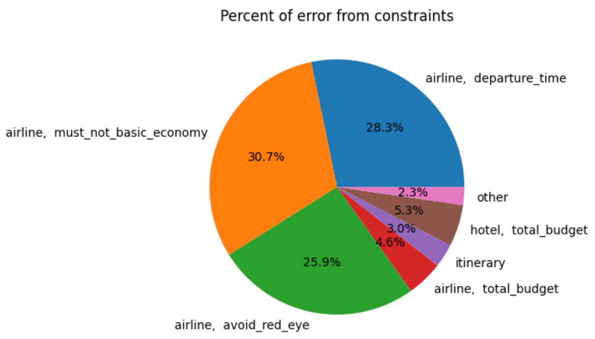


Figure 3: The breakdown of sources of error in EM accuracy. The three major sources of error are the airline constraints `must_not_basic_economy`, `departure_time`, and `avoid_red_eye`.

the airline constraints `must_not_basic_economy`, `departure_time`, and `avoid_red_eye`. A manual inspection reveals that Llama-3 is somewhat insensitive to these constraints and a common failure mode is that they simply do not appear in the generated NL requests. To further filter for these samples, as we did with issues with departure and return dates discussed above, is left for future work.

Quality of solutions. When there is no exact match, we instead evaluate the end-to-end performance by checking the feasibility and optimality of the response $\hat{s}_i$, by checking the *quality ratio* of the *cost* $f(\hat{s}_i; x_i, G_i)$ of generated solution $\hat{s}_i$ (as a function of estimated user request $\hat{x}_i$), to the minimal cost $f(s_i; x_i, G_i)$ if the solver is fed with a groundtruth user request x_i. Note, $\hat{s}_i$ is computed by solving the *estimated* symbolic user request $\hat{x}_i$ but we evaluate the cost with respect to the ground truth x_i.

$$score(i) = f(s_i; x_i, G_i)/f(\hat{s}_i; x_i, G_i) \qquad (3)$$

Since $f(\hat{s}_i; x_i, G_i) \geq f(s_i; x_i, G_i)$, we know $0 \leq score(i) \leq 1$ where a score of 0 corresponds violating one or more constraints and 1 corresponds to the optimal solution. A score between 0 and 1 corresponds to finding sub-optimal solutions to some constraints. We partition the 21.8k test samples into 8 subsets of 2.7k samples. The mean and standard deviation for `TTG` over the 8 subsets is 0.979 ± 0.002, which is very close to 1 (optimal). Within the samples where the generated constraints are not an exact match, the score is $0.726 \pm .0234$.

5.2 Efficiency of `TTG`

We also evaluate the performance of `TTG` by profiling the two major components: generation speed of the Translator and the speed of the MILP solver, tested on a AWS P4de node using one A100 for LLM inference and one CPU (Intel Xeon Platinum 8275CL@3GHz) core for the solver. As shown in Table 4, the primary bottleneck in our system is the model inference cost which takes 81.3% of the compute time. Overall, `TTG` is light-weight and provides responses in real-time.

Response phase	Time (s)
Instruction Translator	2.508±0.116
MILP Solver	
- Loading constraints	0.047±0.061
- Solving	0.527±0.457
- Total	0.575±0.507

Table 4: Time spent on each phase of `TTG`. We report the average and standard deviation over 100 examples.

5.3 Human evaluation

We performed an online survey and qualitative interviews to collect human judgment and feedback about our system's performance. The goal of the human evaluation study was two-fold: (1) validate performance and subjective perception of our system's outputs through a large pool of lay-people who routinely travel, and (2) identify factors that contribute to perceived itinerary quality to inform future work.

We screened from a broad pool of US-based participants who travel four or more times per year to complete a survey evaluating model performance. To maximize evaluation coverage, we randomly sampled 50 natural language travel queries from our generated test set, stratifying by number of stops (60% one-/40% two-stop) and encompassing a variety of trip durations and budgets. We then ran the queries through `TTG`, rendering the map and detailed travel itinerary per trip presented in tabular form (see Fig. 1) via a chat interface. We randomly assigned each of the 1385 participants to 5 of the sampled query-itinerary pairs and ask them to evaluate along three axes (see below). In addition, we also ask the participants to rank the factors that affect their travel decisions, and conduct in-depth interviews to find ways to improve `TTG` (see Appendix B for details).

5.3.1 On Satisfaction, Value and Efficiency

For each query-itinerary pair, participants answered three questions: (1) how much the query was satisfied, (2) the value and (3) the efficiency of the itinerary. Participants noted that they require extensive comparison on many hard (e.g., price) and soft (e.g., aesthetic) criteria as part of assessing optimality, often over many hours of research. Consequently, measuring whether a given itinerary was optimal via human evaluation was determined infeasible. Therefore, we use subjective metrics like (2) and (3) as proxy evaluations for the optimality of each itinerary, absent being able to assess optimality via human evaluation.

We evaluated the survey responses by computing a score constructed similarly to Net Prompter Scores (NPS [6]). This system used a five-point scale (percentage of supporters minus detractors where 5s are coded as promoters and 1-3s as detractors), as shown in Table 5. Our primary *'satisfies the request'* question received a 40.0. Our secondary *'value'* and *'efficiency'* questions scored 35.1 and 36.9, respectively. Overall, we consider these promising results, indicating user acceptance on all three evaluation metrics. We note that while this evaluation does not use the original NPS language, the method of analysis still enables us to understand the relative proportion of respondents who view our model favorably. Additionally, no material difference is seen between user evaluations of the one- and two-stop itinerary.

Question	Detractors %	Promoters %	Net %
...fully satisfies the...request	-13.3	+53.3	+40.0
...offers good value for the money...	-16.8	+52.0	+35.1
...is efficient...	-16.2	+53.1	+36.9

Table 5: Net Prompter-like Score (NPS) and its breakdown in survey questions. Please check the complete form of the questions (as well as other details) in Appendix B.

5.3.2 Preference ranking

Price and preferred travel times were ranked as the most important criteria in trip assessment, reinforcing the selection of these proxy criteria. We see these preferences manifesting in at least two large and distinct user clusters: the first group includes price sensitive travelers, looking for high value; the second cares more about departure times, service levels, and brands. Future work may include personalization; we expect closer alignment to user optimality by inferring user groups to re-weight criteria before computing itineraries.

5.3.3 In-depth Interviews

We then conducted in-depth user interviews with 8 participants who matched the recruitment criteria for the survey. These interviews followed a semi-structured, in-depth format. Participants were asked to reflect on recent travel, walking through their tools used, process of searching for and selecting flights and accommodations, including points of high and low friction and heuristics for prioritization. Finally participants assessed stimuli, which were generated via the same criteria as used to populate the survey.

Together, the survey and user interviews illuminated the following themes for future improvement. **Prioritization**. User requests demonstrate a hierarchy of importance, e.g., flight selection often precedes hotel bookings. **Flexibility**. Trip details should be changed with ease and enable comparison. **Personalization**. Users have a variety of preferences, e.g., cheap vs. cozy, business vs. casual, family vs. solo trips, etc. Many of them are implicit. Moreover, special needs like "*The room door opens to a hallway*" may not be available but can be part of user's ideal selection criteria. **Trust of AI agents**. Decisions made by the agent should be readily verifiable by users as feasible, optimal and fit to their personal use cases. For this, more convenient tools are needed to visualize copious information for confidence boost. While TTG moves towards these goals (e.g., guaranteed quality of solutions by solver), more works can to be done.

6 Conclusion and Future Work

In this work, we propose TTG, and end to end system which plans travel itineraries from user requests in natural language. TTG uses a hybrid architecture that combines an LLM with combinatorial solvers, dynamically formulating travel requests into a well-defined MILP problem, and translating the solution computed by the solver back to natural language. Overall, the system responds almost in real-time ($\sim$ 5 seconds), and outputs feasible and optimal guaranteed travel itineraries, given correctly understood user requests by the fine-tuned LLM, which happens $> 90\%$ of the time for queries up to 6 airline constraints and up to 4 hotel constraints. For this, a data generation pipeline is developed to provide synthetic symbolic and natural language pairs for model training.

We recognize that achieving true optimality requires a system that enables robust personalization, and human-driven filtering and selection. As a result, we anticipate the need for a human benchmark task that enables respondents to stipulate a travel goal in real time and compare between a few near-optimal results, both to measure system performance and to collect signal for improvement.

Future developments will therefore explore multi-round dialog and personalization to further improve user experience, and end-to-end trainable pipelines to make the system more adaptive.

References

[1] AI@Meta. Llama 3 model card. 2024.

[2] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, et al. The scip optimization suite 8.0. *arXiv preprint arXiv:2112.08872*, 2021.

[3] Nikolaj Bjørner, Anh-Dung Phan, and Lars Fleckenstein. νz-an optimizing smt solver. In *Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21*, pages 194–199. Springer, 2015.

[4] Xinyun Chen, Ryan A Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. *ICML*, 2024.

[5] 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.

[6] Nicholas I Fisher and Raymond E Kordupleski. Good and bad market research: A critical review of net promoter score. *Applied Stochastic Models in Business and Industry*, 35(1):138–151, 2019.

[7] Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured nlp tasks without finetuning. *arXiv preprint arXiv:2305.13971*, 2023.

[8] Yilun Hao, Yongchao Chen, Yang Zhang, and Chuchu Fan. Large language models can plan your travels rigorously with formal verification tools. *arXiv preprint arXiv:2404.11891*, 2024.

[9] Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 2023.

[10] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023.

[11] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[12] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Delong Chen, Wenliang Dai, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55:1 – 38, 2022.

[13] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[14] 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.

[15] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[16] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, 2016.

[17] Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *arXiv preprint arXiv:2406.14532*, 2024.

[18] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[19] Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20256–20264, 2024.

[20] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 foundation models for decision making workshop*, 2022.

[21] Gary Smith. Llms can't be trusted for financial advice. *Journal of Financial Planning*, 37(4), 2024.

[22] Yihong Tang, Zhaokai Wang, Ao Qu, Yihao Yan, Kebing Hou, Dingyi Zhuang, Xiaotong Guo, Jinhua Zhao, Zhan Zhao, and Wei Ma. Synergizing spatial optimization with large language models for open-domain urban itinerary planning. *arXiv preprint arXiv:2402.07204*, 2024.

[23] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[24] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[25] Sean Williams and James Huckle. Easy problems that llms get wrong. *arXiv preprint arXiv:2405.19616*, 2024.

[26] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. *arXiv preprint arXiv:2308.08155*, 2023.

[27] Chengxing Xie and Difan Zou. A human-like reasoning framework for multi-phases planning task with large language models. *arXiv preprint arXiv:2405.18208*, 2024.

[28] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents. *ICML*, 2024.

[29] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

[30] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[31] Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V Le, Ed H Chi, et al. Natural plan: Benchmarking llms on natural language planning. *arXiv preprint arXiv:2406.04520*, 2024.

A Using MILP solver to encode conditional constraints

Suppose $\{z_j\}$ are binary variables, then the conditional constraint "if all $z_j = 1$, then $x = y$" can be formulated as the the following:

$$x \leq y + M \sum_j (1 - z_j), \qquad y \leq x + M \sum_j (1 - z_j) \tag{4}$$

where M is a big constant. Intuitively, if all $z_j = 1$, then the above two constraints are equivalent to $x \leq y$ and $y \leq x$, which is $x = y$; if k of the binary variable $\{z_j\}$ are zero, then the above two constraints become $x \leq y + kM$ and $y \leq x + kM$, which becomes trivial for big M.

B Details of the User Study

1. Survey Design

Q1. [RANK]- What matters most to you when selecting a travel itinerary (airfare and hotels)? • Total Price • Value per dollar • Minimal Time in Transit • Simple or Few Steps • Travel/stay with preferred brands • Travel at preferred times • Travel at specific level of service (e.g. hotel stars, airfare class)

Q2-Q6. [SCALE]- For the following question, please reference the image shown. How much do you agree or disagree with the following statements? (5 Point Scale: Strongly Disagree - Strongly Agree) (Repeated 5 times)

• This travel itinerary fully satisfies the corresponding travel request. • This travel itinerary is efficient, given the corresponding travel request. • This travel itinerary offers good value for the money, given the corresponding travel request.

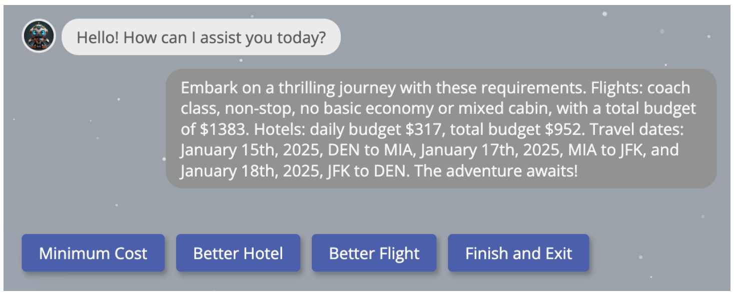
Q7. [OPEN END]- How could the format or quality of these itineraries be improved?

C Details of TTG Demo

We introduce the key features of our demo in detail, using the same example as shown in Fig. 1.

User request. The user request in our example is "*Embark on a thrilling journey with these requirements. Flights: coach class, non-stop, no basic economy or mixed cabin, with a total budget of $1383. Hotels: daily budget $317, total budget $952. Travel dates: January 15th, 2025, DEN to MIA, January 17th, 2025, MIA to JFK, and January 18th, 2025, JFK to DEN. The adventure awaits!*"

Itinerary Options. For a user travel request, TTG gives three itinerary options with three different considerations: 1) *Minimum Cost:* the total cost (flights+hotels) is minimized; 2) *Better Hotel:* More tolerant of hotel costs for a better hotel experience; and 3) *Better Flight:* More tolerant of flight costs for a better flight experience. These options are materialized by different objectives in the MILP travel solver. We show the user interface of three itinerary options in Fig. 4.

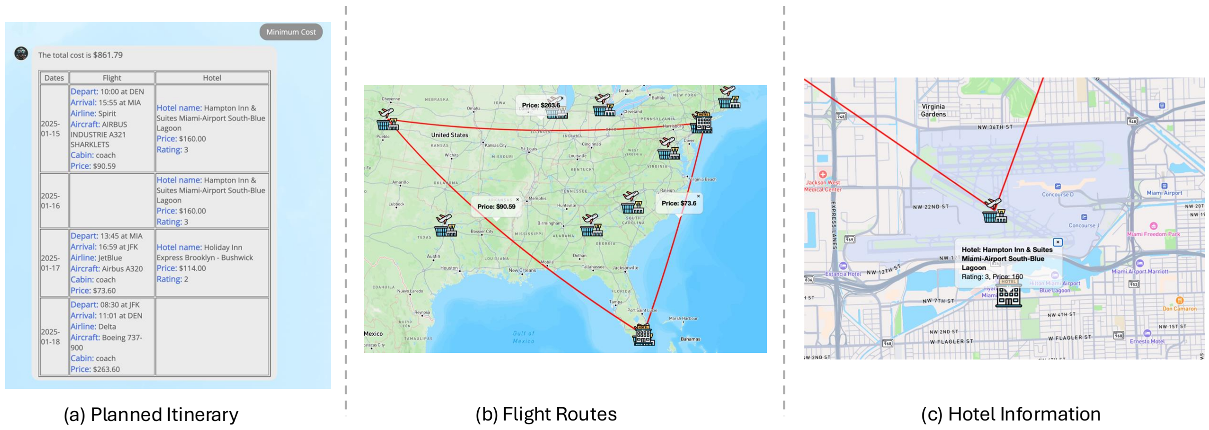

Hello! How can I assist you today?

Embark on a thrilling journey with these requirements. Flights: coach class, non-stop, no basic economy or mixed cabin, with a total budget of $1383. Hotels: daily budget $317, total budget $952. Travel dates: January 15th, 2025, DEN to MIA, January 17th, 2025, MIA to JFK, and January 18th, 2025, JFK to DEN. The adventure awaits!

Figure 4: Itinerary options in TTG demo.

Planned Itinerary. Fig. 5 (a) showcases the planned itinerary with minimum cost as objective. TTG presents this itinerary in a tabular format, detailing the total budget, flight specifics, and hotel information.

Flight Routes. As shown in the detailed view in Fig. 5 (b), TTG presents a sequence of flights according to the user's request (DEN to MIA, MIA to JFK, and JFK to DEK), with the corresponding prices of flights hovering above each each route.


Minimum Cost

The total cost is $861.79

Dates	Flight	Hotel
2025-01-15	Depart: 10:00 at DEN Arrival: 15:55 at MIA Airline: Spirit Aircraft: AIRBUS INDUSTRIE A321 SHARKLETS Cabin: coach Price: $90.59	Hotel name: Hampton Inn & Suites Miami-Airport South-Blue Lagoon Price: $160.00 Rating: 3
2025-01-16		Hotel name: Hampton Inn & Suites Miami-Airport South-Blue Lagoon Price: $160.00 Rating: 3
2025-01-17	Depart: 13:45 at MIA Arrival: 16:59 at JFK Airline: JetBlue Aircraft: Airbus A320 Cabin: coach Price: $73.60	Hotel name: Holiday Inn Express Brooklyn - Bushwick Price: $114.00 Rating: 2
2025-01-18	Depart: 08:30 at JFK Arrival: 11:01 at DEN Airline: Delta Aircraft: Boeing 737-900 Cabin: coach Price: $263.60	

(a) Planned Itinerary (b) Flight Routes (c) Hotel Information

Figure 5: Details of demo. (a) Planned itinerary is shown in tabular view; (b) Flights routes are shown on the map with prices on each travel segment; (c) Hotel infomation, including name, rating and price.

Hotel Information. Once clicking the hotel icon, TTG provides a zoomed-in view of the suggested hotels with their ratings and prices. For instance, as shown in Fig. 5 (c), TTG has booked the "Hampton Inn & Suites Miami-Airport South-Blue Lagoon" for the user's stay in Miami (MIA). This selection meets the user's daily hotel budget constraint of $317. Note that if a user specifies a minimum hotel rating, the MILP solver in TTG ensures this requirement is also met.

Packages Acknowledgement. Our TTG demo is built upon Mapbox [4] and BotUI [5].

[4] https://www.mapbox.com/
[5] https://github.com/botui/botui