

From Feasibility to Desirability: Plan, Learn, Adapt (PLA) Framework for Personalized On-Device Itinerary Generation

Himel Dev¹, Tanmoy Sen², Madhusudan Basak³, Bashima Islam³
¹529 Tech LLC, ²University of Virginia, ³University of Massachusetts Amherst
 himel@529-tech.com, ts5xm@virginia.edu, mbasak@umass.edu, bashima@umass.edu
 USA

Abstract

Generating personalized trip itineraries is a complex planning task and involves a tension between hard combinatorial feasibility and soft latent desirability. Classical optimization enforces constraints but fails to capture subjective traveler preferences. While learning-based approaches model preferences, they cannot guarantee feasibility. Mobile deployment imposes additional resource constraints on both. To address this, we propose Plan, Learn, Adapt (PLA), a three-stage framework for personalized on-device itinerary generation. The *Plan* stage builds a heterogeneous ensemble of lightweight planners that produces structurally diverse feasible candidates. From pairwise itinerary comparisons, *Learn* fits a compact Bradley-Terry reward model that captures emergent schedule properties such as pacing, geographic coherence, and day balance, which per-POI signals miss. Finally, *Adapt* applies feasibility-preserving local refinement within a device-aware compute budget; every intermediate state is feasible by construction. On 2,519 pairwise human comparisons across 100+ U.S. cities, the reward-guided ensemble achieves a 67.8% win rate (+11.2 points over the best single planner) with 100% feasibility, while three frontier LLMs (GPT-5, Claude Opus 4.5, Gemini 3 Pro) achieve 0% feasibility on the same constraints. The reward model generalizes across held-out cities, with a 67.6% mean leave-one-city-out accuracy. In production deployment within Fly-Enjoy, PLA increased itinerary completion rates by 91% with 109.9 ms average on-device latency.

Keywords

Trip Planning, Preference Learning, Combinatorial Optimization, On-Device Systems

1 Introduction

Multi-day trip itinerary generation is a canonical human-aligned combinatorial planning problem. The system must select, order, and time a set of points of interest (POIs) across multiple days under hard spatio-temporal constraints (opening hours, availability windows, daily time budgets) while also satisfying soft desirability criteria such as pacing, diversity, and alignment with stated interests.

This duality creates a tension between feasibility and desirability. Classical planners [10, 19] enforce constraints reliably but rely on hand-crafted objectives that are weak proxies for human satisfaction. In our earlier deployment, itineraries with similar heuristic scores were often judged very differently by users, reflecting preferences that no single composite score captures. Learning-based approaches [3, 6] capture preferences but, without explicit constraint handling, frequently produce invalid schedules. LLM-based

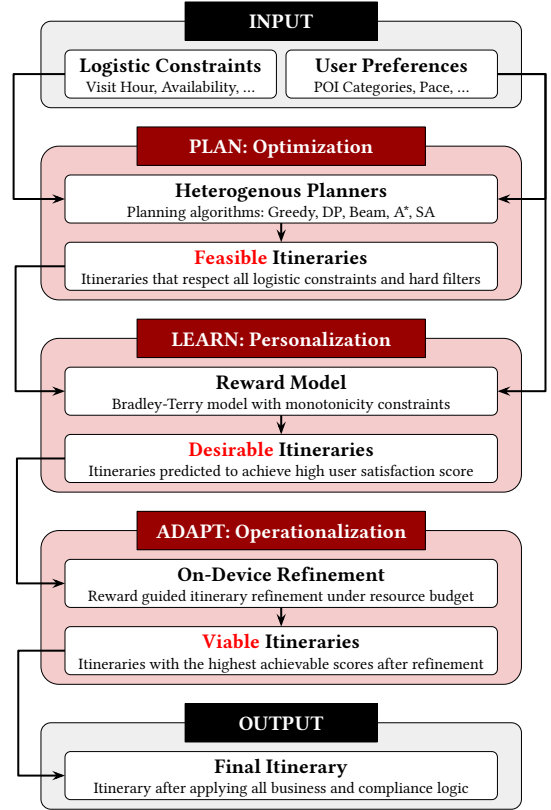


Figure 1: Overview of the PLA framework. PLAN produces a diverse set of feasible seed itineraries; LEARN fits a human-aligned reward model from pairwise preferences; ADAPT then refines the selected itinerary on-device under a resource budget while preserving feasibility throughout.

approaches [9, 15] face the same limitation. In our benchmark (Section 3.6), none of the 294 itineraries from GPT-5, Claude Opus 4.5, and Gemini 3 Pro satisfied all hard constraints simultaneously.

A third requirement is viability. Solutions must be deployable at scale on end-user devices. Our product, **FLYENJOY**¹, is an offline-first iOS travel app whose users often need itineraries in low or no connectivity settings (airports, in-flight, international roaming) with strict latency expectations. Cloud-dependent LLM and learning-based approaches are costly at scale and incompatible with offline, latency-sensitive, privacy-preserving use.

¹<https://apps.apple.com/app/6745104853>

These three requirements, feasibility, desirability, and viability, are unlikely to be met by any single paradigm. We propose PLAN, LEARN, ADAPT (PLA), a three-stage framework for personalized on-device itinerary generation (Figure 1). In Section 2.1, the PLAN stage produces structurally diverse feasible candidates through a heterogeneous planner ensemble. Section 2.2 then describes how LEARN fits a compact reward model from pairwise itinerary comparisons. Section 2.3 introduces ADAPT, which carries out reward-guided local refinement under a device-aware compute budget while preserving feasibility.

Our main contributions are as follows.

- **Heterogeneous Planner Ensemble:** No single planning algorithm dominates across trip contexts; we establish this empirically rather than assume it. Reward-guided selection over a heterogeneous planner ensemble yields a 67.8% win rate, +11.2 pp over the best individual planner (Section 2.1).
- **Itinerary-Level Preference Modeling:** Itinerary quality involves emergent properties (pacing, geographic coherence, day balance) that per-POI scores cannot capture. We collect 2,519 pairwise comparisons of complete multi-day itineraries across 100+ U.S. cities and train a compact Bradley–Terry reward model achieving 67.6% mean leave-one-city-out accuracy (Section 2.2).
- **On-Device Itinerary Refinement:** ADAPT improves desirability via reward-guided local edits while guaranteeing every intermediate state is feasible by construction (not merely the final output). A device-aware budget keeps on-device latency to 109.9 ms on average, enabling fully offline, interactive-speed execution (Section 2.3).

To support reproducibility, we publicly release a Python implementation of the core PLA components along with sample data.²

2 System Overview

Problem Setup. Given a trip context (including destination, duration, availability, budget, interests, etc.) and a set of candidate POIs (with locations, opening hours, and visit durations), the goal is to generate a multi-day itinerary that is both feasible and desirable. Feasibility imposes opening-hour, availability, travel-time, and time-budget constraints (Section 2.1). Desirability reflects latent user satisfaction at the itinerary level and is not directly observable. We learn a compact surrogate reward from pairwise human preferences over complete itineraries (Section 2.2), optimized under a device-aware compute budget (Section 2.3).

2.1 PLAN: Diverse Feasible Itinerary Generation

We implement five heterogeneous planners for multi-day itinerary generation under a shared feasibility layer. Each planner outputs a feasible itinerary (an ordered, time-stamped sequence of POI visits) that respects all hard constraints and optimizes a shared composite objective. Although the objective is shared, the planners’ distinct search paradigms yield *structurally diverse* candidates differing in POI selection, ordering, and time allocation. The planner suite is designed to supply a diverse candidate pool for downstream selection and refinement, since identifying a single best planner across all trip contexts is not feasible.

2.1.1 POI Data and User Inputs. Our system operates over city-specific POI datasets covering 100+ U.S. cities. Each POI p has opening hours, visit duration, cost tier, location (with a precomputed travel-time zone), categories, and a popularity score derived from user ratings. A trip context u specifies destination, duration, categories of interest, budget tier, availability windows, and pace (Relaxed, Balanced, or Packed), which sets the daily time budget. Together, these inputs define the feasible search space while remaining minimal enough for fast, on-device planning.

2.1.2 Feasibility Constraints. All planners enforce the following hard constraints that define a valid, executable itinerary:

- **Visit Hours:** Each visit must start after opening and end before closing, i.e., $start(p) \geq open(p)$ and $start(p) + dur(p) \leq close(p)$.
- **Availability Windows:** Every visit must lie entirely within the traveler’s available time on that day.
- **Travel-Time Consistency:** Sufficient time must exist between consecutive visits to travel between them. If arrival precedes opening, the traveler waits.
- **Time Budget:** Total active time (visits plus travel) on each day must not exceed the budget implied by the selected pace.
- **Cost Budget:** POIs with $cost(p)$ exceeding the user’s budget tier are excluded from consideration.
- **No Repeats:** Each POI is visited at most once per trip.

All planners share a *feasibility layer* that computes earliest-feasible arrival times (with waiting when arrival precedes opening) and enforces day- and trip-level constraints uniformly, so cross-planner differences reflect search strategy rather than inconsistent constraint handling.

2.1.3 Objective Components. Beyond feasibility, planners optimize for itinerary quality using a hand-crafted, interpretable objective. We score a candidate POI p at its earliest feasible start time t_s , given the current time cursor t and previously scheduled POI p_t :

- **Travel Efficiency:** We penalize long travel times to encourage geographically coherent itineraries: $travel(p_t, p) = T(zone(p_t), zone(p))$.
- **Waiting Time:** Arriving before a POI opens incurs idle time, which we penalize: $wait(p, t) = \max\{0, open(p) - (t + travel(p_t, p))\}$.
- **Scheduling Urgency:** POIs with narrow remaining windows should be prioritized to avoid missing them entirely. We define urgency based on the gap between visit end and closing time: $urgency(p, t_s) = \frac{1}{\epsilon + \max\{0, close(p) - (t_s + dur(p))\}}$, where ϵ is a small constant for numerical stability. POIs that would barely fit receive high urgency scores.
- **Category Diversity:** To avoid monotonous itineraries (e.g., five museums in a row), we reward visits that introduce under-represented categories at both day and trip levels: $div(p) = \lambda_d \sum_{c \in cat(p)} \frac{\gamma_c}{1 + count_d(c)} + \lambda_t \sum_{c \in cat(p)} \frac{\gamma_c}{1 + count_t(c)}$, where $count_d(c)$ and $count_t(c)$ are the number of visits to category c on the current day and trip respectively, γ_c is a category weight (higher for user-preferred categories), and λ_d, λ_t control the day vs. trip diversity trade-off.
- **POI Popularity:** Higher-quality POIs, as measured by $pop(p)$, contribute more value to the itinerary.

²<https://github.com/Official529Tech/pla-itinerary>

Table 1: Comparison of planning algorithms and trade-offs.

Algorithm	Paradigm	Key Trade-off
Greedy	Constructive	Solution quality vs. speed
DP	Dynamic prog.	Optimality vs. state space
Beam	Tree search	Solution quality vs. search breadth
A*	Tree search	Optimality vs. search cost
SA	Local search	Exploration vs. convergence speed

2.1.4 Shared Composite Score. The objective components are combined into a weighted composite score used by constructive planners (Greedy, Beam, A*) for step-level decisions:

$$\begin{aligned} \text{score}(p, t_s) = & w_{\text{pop}} \cdot \text{pop}(p) - w_{\text{travel}} \cdot \text{travel}(p_\ell, p) - w_{\text{wait}} \cdot \text{wait}(p, t) \\ & + w_{\text{urg}} \cdot \text{urgency}(p, t_s) + w_{\text{div}} \cdot \text{div}(p). \end{aligned} \quad (1)$$

All weights (w_{pop} , w_{travel} , w_{wait} , w_{urg} , w_{div}) are configurable hyperparameters shared across planners for fair comparison. Global optimization planners and local search use analogous objectives adapted to their formulations. The shared objective structure ensures that performance differences across planners stem from search strategy rather than from inconsistent quality definitions.

2.1.5 Planning Algorithms. We implement five planning algorithms across four paradigms: constructive heuristics (Greedy), dynamic programming (DP), tree search (Beam, A*), and local search (SA). Table 1 summarizes the key characteristics of each planning algorithm. All algorithms share the feasibility layer and composite objective (Eq. 1), differing only in search strategy.

- **Greedy:** Iteratively schedules the highest-scoring feasible POI at its earliest valid start time.
- **Dynamic Programming (DP):** Resource-bounded optimization over a time-expanded state space maximizing POI popularity and coverage.
- **Beam Search:** Anytime tree search over a bounded frontier of partial itineraries, pruning by composite score.
- **A* Search:** Best-first search with an admissible upper bound on remaining reward.
- **Simulated Annealing (SA).** Local search with probabilistic acceptance of structural edits over a feasible seed.

We also prototyped a Constraint Programming (CP-SAT) planner but excluded it from deployment because Google OR-Tools lacks a native Swift implementation.

2.2 LEARN: Human Preference Modeling

User satisfaction with an itinerary is latent and context-dependent. We therefore conduct a pairwise preference study over complete itineraries and train a compact reward model over interpretable itinerary-level features.

2.2.1 Preference Data Collection. We collect preferences via a web platform that presents side-by-side itinerary pairs from randomly chosen planners. Planner identities are hidden and left-right placement is randomized to prevent ordering bias. Guidelines emphasize holistic itinerary-level judgment; submissions with completion time well below median are filtered. This yields 2,519 pairwise comparisons across 100+ U.S. cities. We use a feature-based Bradley–Terry

Table 2: Itinerary-level features for reward modeling.

Category	Feature Example
POI Quality	Count, total/avg/min/max popularity
Category Diversity	Unique categories, entropy, alignment
Temporal Efficiency	Travel time, wait time, travel-to-activity ratio
Schedule Balance	Daily load, time utilization, max consecutive travel
Constraint Slack	Budget margin, closing-time slack

formulation for data efficiency: each comparison supervises ~ 20 interpretable features simultaneously (Section 2.2.2).

2.2.2 Reward Model Architecture. We learn a scoring function $R(x)$ that ranks itineraries by predicted human preference.

Feature Engineering. Each itinerary x is mapped to a fixed dimensional feature vector $\phi(x) \in \mathbb{R}^d$, and we learn $R_\theta(x) = f_\theta(\phi(x))$. The feature set ($d \approx 20$) captures structural, temporal, and preference-aligned properties. Table 2 summarizes the feature groups. All features are computed deterministically from POI metadata and the generated schedule (operating hours, visit durations, travel times, category counts) rather than from city-specific or cultural signals. The representation is therefore portable across destinations: the model learns that excessive travel time or imbalanced daily load reduce preference, properties that hold city to city. User-context interaction terms (e.g., `is_packed_x_num_pois`, `is_relaxed_x_avg_gap`) condition itinerary scores on the traveler’s stated preferences. A POI-dense schedule then ranks higher for a packed traveler than a relaxed one, providing context-conditioned personalization without per-user history.

Learning Objective. We adopt the Bradley–Terry framework [1, 12]: the probability of preferring x_a over x_b follows a logistic model $P(x_a \succ x_b) = \sigma(R_\theta(x_a) - R_\theta(x_b))$, where $\sigma(\cdot)$ is the sigmoid. Given a dataset $\mathcal{D} = \{(x_a^{(i)}, x_b^{(i)}, y^{(i)})\}_{i=1}^N$ where $y^{(i)} \in \{-1, +1\}$, we minimize the regularized negative log-likelihood and our loss is $\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=1}^N \log \sigma(y^{(i)} \cdot (R_\theta(x_a^{(i)}) - R_\theta(x_b^{(i)}))) + \lambda \|\theta\|_2^2$. Here, λ controls ℓ_2 regularization to prevent overfitting.

Model Implementation. We implement two parameterizations: a linear model $R_\theta(x) = w^\top \phi(x)$ with $\theta = w \in \mathbb{R}^d$ (high interpretability, $< 1\mu s$ inference), and a small gradient-boosted tree ensemble for nonlinear interactions.

Monotonicity Constraints. To align the reward function with domain knowledge, we enforce *monotonicity constraints*: negative (travel time, waiting time, and travel-to-activity ratio do not increase reward) and positive (POI popularity, category diversity, and preference alignment do not decrease reward). The linear model enforces these directly during optimization; tree-based models use post-hoc validation with re-weighting of non-monotonic features.

2.2.3 Reward-Guided Ensemble Selection. The reward model selects among planner candidates and serves as the training signal for downstream refinement. Rather than committing to a single planner, we treat the planners plus the reward model as an *implicit ensemble*. Concretely, given a trip context u , the PLAN stage generates a set of feasible itineraries $\mathcal{X}(u) = \{x_1, \dots, x_K\}$ using different planning algorithms. We then evaluate each candidate using the learned reward model and select the itinerary with the highest predicted user satisfaction: $x^* = \arg \max_{x \in \mathcal{X}(u)} R_\theta(x)$.

Diversity arises from algorithmic heterogeneity rather than parameter perturbations, distinguishing this from standard ensembles. A common human-aligned reward score enables direct comparison across planners with different internal objectives. Selection is lightweight, fully on-device, and adds negligible overhead to the planning stage.

2.3 ADAPT: On-Device Itinerary Refinement

PLAN produces diverse feasible itineraries; Learn provides a reward model $R_\theta(x)$ over itinerary-level features. Two *practical and operational* gaps remain: (i) seed planners are not optimized for the learned reward, and (ii) mobile devices impose strict runtime and memory limits. ADAPT addresses both via feasibility-preserving local edits under a resource-aware compute budget, with an anytime guarantee that returns the best feasible itinerary found so far.

2.3.1 Local Feasible Edit Operators. ADAPT explores a neighborhood $\mathcal{N}(x)$ of a feasible itinerary x , defined by atomic edit operators with a validation layer that guarantees feasibility. Each edit modifies a small portion of the itinerary (typically within a day), followed by local re-timing from the first affected position; proposals that violate constraints are discarded.

Atomic Operators. Let a visit be indexed by (d, i) , denoting position i on day d . We define the following six local moves:

- **Drop** (d, i) : remove the visit at position i on day d .
- **Insert** (d, p, i) : insert an unvisited POI p at position i on day d .
- **Replace** (d, i, p) : replace the visit (d, i) with an unvisited POI p .
- **SwapAdj** (d, i) : swap visits at positions i and $i+1$ on day d .
- **Relocate** (d, i, j) : move the visit from position i to j on day d .
- **MoveDay** (d, i, d', j) : move the visit (d, i) to (d', j) .

These operators form a compact move set that covers common user-driven edits (“I want to add this museum”, “swap lunch and park”, “remove this stop”) while enabling local search to correct structural defects such as excessive travel chaining, poor pacing, or category imbalance.

Feasibility Layer and Incremental Re-timing. After each edit, we reconstruct the schedule for the affected suffix via a forward pass. For day visits $(v_1, \dots, v_m)$ where v_j corresponds to POI p_j with duration dur_{p_j} and window $[open_{p_j}, close_{p_j}]$. With current cursor time t at the end of v_{j-1} , we compute $arr_j = t + T(p_{j-1}, p_j)$, $start_j = \max\{arr_j, open_{p_j}\}$, and $end_j = start_j + dur_{p_j}$. We accept the schedule iff (i) p_j is open at $start_j$, (ii) $(start_j, end_j)$ lies within the user’s availability interval, (iii) $end_j \leq day_start + B_{\text{schedule}}$ (the user’s daily time budget), and (iv) end_j does not exceed the day boundary. We then set $t \leftarrow end_j$ and continue. This procedure strictly preserves feasibility when it succeeds, provides natural waiting-time accounting via $start_j - arr_j$, and enables *incremental* evaluation because only a local suffix must be recomputed.

2.3.2 Reward-Guided Local Search. We refine a seed itinerary x_0 by iteratively applying local edits to improve $R_\theta(x)$. Feasibility is enforced by the re-timing layer, so every accepted itinerary is feasible. We use a simple hill-climbing core with stochastic neighborhood sampling; more complex acceptance rules (e.g., simulated annealing) are compatible but unnecessary under tight budgets. **Neighborhood Sampling.** Enumerating all feasible neighbors is expensive with many unvisited POIs. Instead, we sample a bounded

candidate set per iteration: (i) an operator type with fixed probabilities (favoring SwapAdj, Relocate, Replace); (ii) target indices uniformly from the current schedule; (iii) for Insert/Replace, candidate POIs from a filtered pool (budget tier, category interest, distance). Each candidate is re-timed incrementally; infeasible proposals are discarded.

Best-improvement under a fixed per-iteration cap. Over the sampled candidates $\tilde{\mathcal{N}}(x)$ we select $x^* = \arg \max_{y \in \tilde{\mathcal{N}}(x)} R_\theta(y)$ and accept it if $R_\theta(x^*) - R_\theta(x) \geq \delta$. The procedure is anytime: x_{best} is tracked globally and returned if the time budget expires.

2.3.3 Resource-Aware Budgeting. PLA must run fully offline with predictable latency. ADAPT therefore uses a *device-aware* budget that bounds refinement runtime. At runtime we query device signals: model class, core count, available memory, and low-power mode. We map these to a refinement time budget: $B_{\text{ms}} = \text{clip}(B_0 \cdot s_{\text{cpu}} \cdot s_{\text{mem}} \cdot s_{\text{power}}, B_{\text{min}}, B_{\text{max}})$, where B_0 is a nominal budget (200–500 ms) and scaling factors come from coarse bins over the device signals. Coarse binning avoids fragile device-specific calibration, and the anytime search ensures graceful degradation on resource-constrained devices.

3 Evaluation

We evaluate the PLA framework along five axes: (1) [PLAN] algorithmic diversity, (2) [LEARN] model accuracy, (3) [PLAN + LEARN] ensemble effectiveness, (4) [ADAPT] refinement uplift, and (5) [PLAN + LEARN + ADAPT] overall performance.

3.1 Dataset and Experimental Setup

Our evaluation uses 2,519 pairwise comparisons across 100+ U.S. cities. Each comparison shows side-by-side itineraries from two randomly chosen planners (full protocol in Section 2.2.1). Coverage spans 5 planners, 1–7 day trips, all schedule-intensity $\times$ budget-tier combinations, and varied interest profiles. We report 5-fold stratified cross-validation results with bootstrap 95% confidence intervals (1,000 resamples).

Inter-Annotator Agreement. Since each comparison is assigned to one annotator, we ran a dedicated IAA study where two annotators independently evaluated 50 shared itinerary pairs. The study yields 78.0% raw agreement (39/50), Cohen’s $\kappa = 0.56$, Krippendorff’s $\alpha = 0.56$, with balanced choices (27:23, 26:24). Across the full dataset we find no position bias (Plan 1 win rate 50.9%, $p = 0.38$) and low annotator heterogeneity (std 0.10 across 17 annotators). The moderate κ matches subjective preference tasks and motivates Bradley–Terry modeling, which treats disagreement as probabilistic signal. The 78.0% human-human rate bounds attainable model accuracy: the deployed 66.5% model (Section 3.3) captures 60% of the gap between random and human agreement.

3.2 [PLAN] Algorithmic Diversity

A key hypothesis of PLA is that heterogeneous planners produce structurally diverse feasible candidates. We validate this via POI overlap, head-to-head win rate, and feature diversity analyses.

POI Overlap Analysis. We measure itinerary differences via Jaccard similarity of POI sets (Figure 2). Mean Jaccard similarity is 0.697 across algorithm pairs, indicating algorithms share approximately

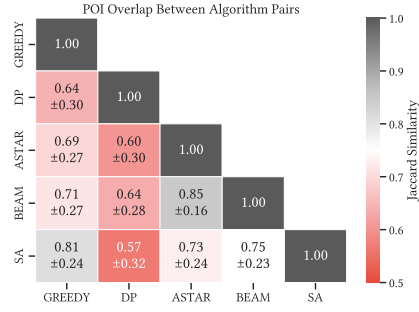


Figure 2: POI overlap (Jaccard similarity) between algorithm pairs. Mean overlap is 0.697; DP-SA show the most distinct selections.

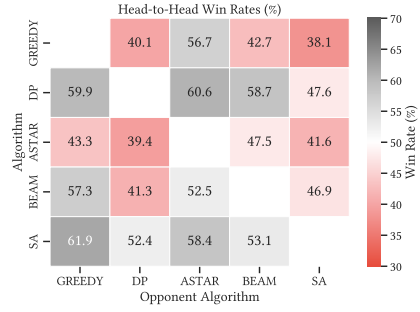


Figure 3: Head-to-head win rate matrix. No single algorithm dominates all matchups; the ten pairs exhibit complementary strengths.

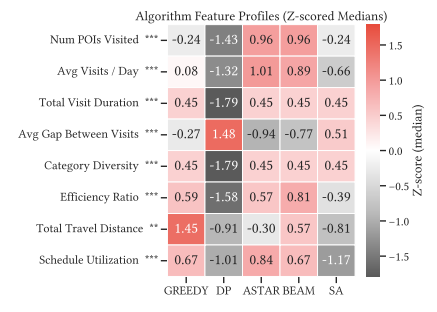


Figure 4: Z-scored median feature profiles per algorithm (deviation from cross-algorithm mean). All 8 features differ significantly ($p < 0.05$).

70% of POIs on average for the same trip context. DP-SA pairs are most distinct (0.565) and A*-Beam pairs most similar (0.847). Even so, 10.6% of comparisons fall below 30% POI overlap, indicating genuine exploration of different solution regions.

Win Rate Analysis. Figure 3 shows the head-to-head win rate matrix across all 10 algorithm pairs. All pairs are complementary: both algorithms win in at least some comparisons. DP beats A* most decisively (60.6%–39.4%); DP-SA is the most competitive (52.4%–47.6%). Complementarity is a prerequisite for ensemble selection: if one algorithm dominated, selection would collapse to that algorithm.

Feature Diversity Analysis. Beyond POI selection, we examine structural differences. Figure 4 shows Z-scored median feature profiles for each algorithm. A* and Beam produce the densest itineraries with the tightest visit spacing, DP the sparsest with the largest inter-visit gaps, while Greedy incurs the longest travel distances and SA the lowest schedule utilization. Kruskal-Wallis tests confirm *all 8 features* differ significantly across algorithms ($p < 0.05$), including number of POIs, visits per day, travel distance, and category diversity. Planners occupy different regions of feature space, producing structurally different itineraries.

3.3 [LEARN] Model Accuracy

We evaluate the reward model’s accuracy against heuristic baselines, compare architectures and input formulations, and characterize data requirements via learning curves.

Heuristic Baseline Comparison. In Table 3, learned Bradley-Terry model achieves $66.5\% \pm 2.3\%$ accuracy (5-fold CV), outperforming all single-feature heuristics. The improvement over the best heuristic, (num_pois at 52.2%) is +14.3% absolute. The model captures feature interactions that no single metric can express.

Model Architecture Comparison. Table 4 compares architectures under the Bradley-Terry formulation (5-fold CV). The deployed Logistic Regression (ElasticNet) achieves ROC-AUC 0.730 and F1 0.687. Random Forest reaches ROC-AUC 0.755 and F1 0.707, but the ROC-AUC and F1 gaps are within bootstrap variability and less relevant for ensemble selection, which depends on within-context ranking rather than global pair discrimination (Section 2.2.3). We deploy the linear model: it offers per-feature interpretability for production diagnosis, native enforcement of the monotonicity constraints (Section 2.2.2), and an order-of-magnitude smaller on-device footprint.

Model Formulation Comparison. Figure 5 shows F1 score by input formulation. The Bradley-Terry (difference-based) formulation achieves the highest F1, outperforming concatenated and augmented alternatives. The difference-based representation thus captures pairwise structure more effectively than formulations treating the two candidates independently.

Learning Curve Analysis. Figure 6 shows model ROC-AUC as training size grows. AUC improves from 0.628 ($n = 100$) to 0.719 ($n = 1,209$); doubling to 2,015 adds only +0.007, within bootstrap CIs. The 2,519-comparison dataset is in the saturated regime; additional gains would require richer features, not more data. The feature set is deliberately constrained for three on-device requirements: sub-millisecond inference, native monotonicity enforcement (Section 2.2.2), and per-feature interpretability. Richer representations (e.g., learned POI-sequence embeddings) are natural extensions when these constraints can be relaxed. Leave-one-city-out (LOCO) cross-validation assesses cross-city generalization. The model achieves 67.6% mean accuracy (8.6% std) across 100+ held-out cities, with per-city accuracy ranging from 51.5% to 90.0%. The variation reflects per-city discrimination difficulty: cities with less diverse POI distributions are harder to rank than data-rich cities.

3.4 [PLAN + LEARN] Ensemble Effectiveness

We evaluate whether reward-model-guided selection outperforms any single algorithm, examine the diversity of algorithm selection across contexts, and validate that predicted reward scores align with empirical human preferences.

Ensemble Model Performance. Table 5 compares the reward-model ensemble against individual algorithms via counterfactual analysis on the full 2,519 comparisons. The ensemble achieves a **67.8% win rate**, versus 56.6% for the best single algorithm (DP), a gain of +11.2% absolute. The 95% bootstrap CI [65.9%, 69.6%] does not overlap with DP’s interval [53.6%, 59.7%], confirming statistical significance (McNemar’s test $p < 0.001$).

Ensemble Selection Diversity. Figure 7 shows which algorithm is selected by the reward model. DP is selected most often (30.1%), followed by SA (22.5%), Beam (17.5%), Greedy (15.5%), and A* (14.4%). The selection entropy is 2.265 bits out of a maximum 2.322 bits (97.6% of maximum entropy), indicating highly diverse selection with no single algorithm dominating. The chi-square test confirms

Table 3: Pairwise prediction (5-fold CV).

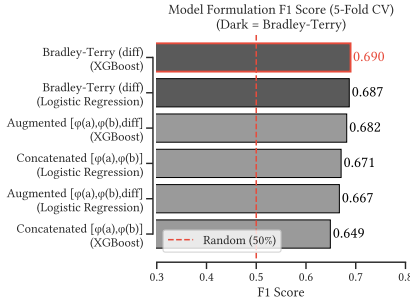
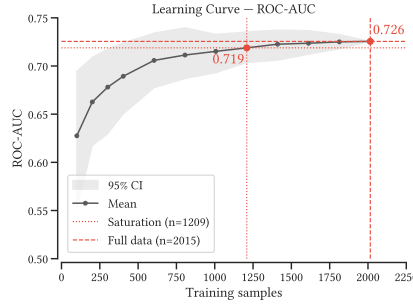
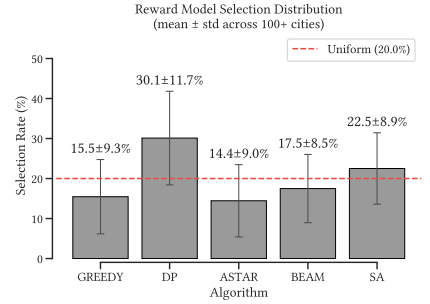
Method	Accuracy
Random	50.0%
Majority Class	51.0%
Heuristic: num_pois	52.2%
Heuristic: avg_visits_per_day	51.6%
Heuristic: total_travel_distance	46.6%
Bradley-Terry Model (deployed)	66.5%

Table 4: Model comparison (5-fold CV).

Model (Deployment Pref.)	ROC-AUC	F1
LR (L2)	0.728	0.687
SVM (RBF)	0.723	0.696
Random Forest	0.755	0.707
XGBoost	0.751	0.706
Ensemble (LR+XGBoost)	0.752	0.696
Ensemble (LR+RF+SVM)	0.750	0.704
LR (ElasticNet) [deployed]	0.730	0.687

Table 5: Counterfactual analysis.

Strategy	Win Rate	95% CI	n
Random Selection	50.0%	—	—
Greedy Only	44.6%	[41.5%, 47.8%]	1024
A* Only	42.9%	[39.7%, 46.1%]	1005
Beam Only	49.6%	[46.4%, 52.8%]	987
SA Only	56.4%	[53.4%, 59.5%]	998
DP Only	56.6%	[53.6%, 59.7%]	1024
Ensemble + R_θ	67.8%	[65.9%, 69.6%]	2519

**Figure 5: F1 score by input formulation. Bradley-Terry (difference-based, dark bars) outperforms concatenated and augmented alternatives.****Figure 6: Learning curve (ROC-AUC vs. training size). Performance plateaus near 1,209 samples, indicating data sufficiency under the current feature set.****Figure 7: Algorithm selection distribution across cities (±std). Near-maximum entropy (97.6%) indicates all algorithms contribute to the ensemble.**

significant non-uniformity ($p < 0.001$), validating that different algorithms excel in different contexts.

Ensemble Score Improvement. Figure 8 plots each algorithm's mean predicted reward against its empirical win rate, with 95% Wilson confidence interval error bars. The Spearman correlation between predicted score and empirical win rate is $\rho = 0.90$, so the reward model's ranking aligns with annotator preferences. The ensemble (shown in red) attains both a higher predicted score and a higher win rate than any individual algorithm; maximizing the reward model therefore yields itineraries that humans prefer.

3.5 [ADAPT] Refinement Uplift

We quantify the score uplift delivered by local refinement across all seed algorithms and characterize how improvement scales with the number of accepted moves. Reward score improvement serves as a proxy for human preference improvement: Section 3.4 shows that reward model scores predict human-labeled win rates at Spearman $\rho = 0.90$, indicating that score maximization reliably selects annotator-preferred itineraries.

Refinement Score Improvement. Table 6 shows reward before and after ADAPT refinement. All improvements are statistically significant ($p < 0.001$, Wilcoxon signed-rank test). Algorithms with lower initial reward scores benefit more: A* gains +11.7%, while the ensemble seed (already the highest-scoring before refinement) improves by +3.9%. ADAPT thus provides uplift regardless of seed quality, and ensemble selection followed by refinement produces the strongest final itinerary.

Refinement Iteration Improvement. Figure 9 shows ensemble improvement grouped by number of accepted moves, binned in intervals of five. Itineraries with more accepted moves show larger

score gains; each accepted move generally contributes to quality improvement. This upward trend is consistent with the hill-climbing acceptance criterion, which only incorporates score-improving proposals, though variance increases at higher moves.

3.6 [PLAN + LEARN + ADAPT] Performance

Combining all three stages, we benchmark PLA against LLM-based generators on feasibility, report on-device latency, and quantify real-world deployment impact. We do not compare against prior itinerary systems (TravelPlanner, PersonalTravelSolver) because they target a different operating point: open-ended natural-language input, single-day or unbounded-budget formulations, and cloud inference. Adapting them to our multi-day, on-device regime would measure adaptation, not the methods. The classical optimization baselines are covered by the per-planner comparisons in Section 3.4, while the LLMs are evaluated in the following comparison.

LLM Benchmark Comparison. We evaluated 294 itineraries (98 each from GPT-5, Claude Opus 4.5, Gemini 3 Pro) against our full hard-constraint set. Strict feasibility rate is **0/98 (0%)** for all three models, with a Wilson 95% CI upper bound of $\approx 3.8\%$ per model. Figure 10 shows total violation distributions: Claude Opus 4.5 has the fewest (median 1, mean 1.08), then Gemini 3 Pro (5, 4.44) and GPT-5 (9, 9.03). Table 7 shows that travel-time consistency dominates (an average of 1.04, 3.28, and 5.98 violations per itinerary for Claude Opus 4.5, Gemini 3 Pro, and GPT-5), with visit-duration second (0.03, 0.22, and 1.62, respectively); other categories rarely violate. The constraint regime is hard for autoregressive generation. On the other hand, PLA offers guaranteed feasibility at interactive on-device latency, versus approximate feasibility from a cloud API.

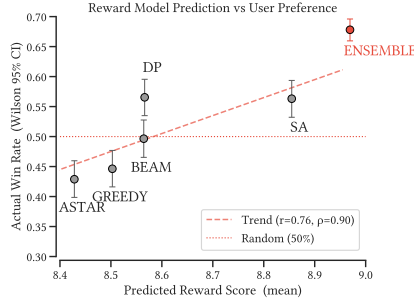


Figure 8: Predicted reward score vs. empirical win rate for algorithms. Correlation ($\rho = 0.90$) between predicted score and win rate shows that score maximization tracks human preference.

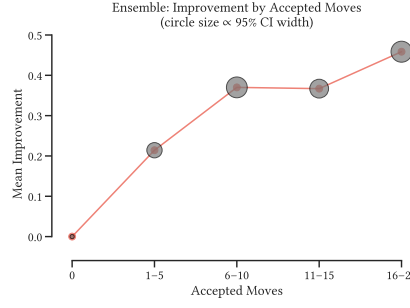


Figure 9: Ensemble improvement by number of accepted moves (binned). Each additional accepted move typically contributes to score gain, with higher variance at higher move counts.

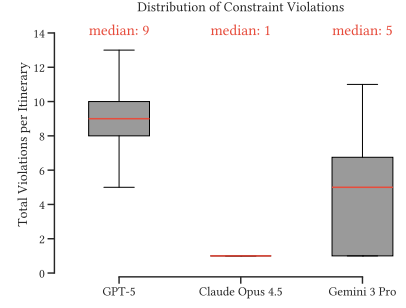


Figure 10: Distribution of constraint violations per itinerary for three frontier LLMs. All models achieve 0% strict feasibility; whereas PLA maintains 100% feasibility by construction.

Table 6: ADAPT refinement by seed. All score improvements are statistically significant ($p < 0.001$).

Seed	Before	After	Uplift	95% CI
Greedy	6.835	7.524	+10.1%	[0.563, 0.825]
DP	6.955	7.268	+4.5%	[0.213, 0.431]
A*	6.623	7.399	+11.7%	[0.671, 0.883]
Beam	6.805	7.390	+8.6%	[0.501, 0.676]
SA	6.983	7.491	+7.2%	[0.425, 0.597]
Ensemble	7.184	7.463	+3.9%	[0.191, 0.392]

Table 7: Mean constraint violations per itinerary by type (mean $\pm$ std) for GPT-5, Claude Opus 4.5, Gemini 3 Pro.

Violation	GPT-5	Opus 4.5	Gemini 3
Travel Time	5.98 ± 1.51	1.04 ± 0.32	3.28 ± 1.85
Visit Hours	1.36 ± 0.84	0.01 ± 0.10	0.79 ± 0.83
Duration	1.62 ± 1.84	0.03 ± 0.17	0.22 ± 0.49
Time Budget	0.00 ± 0.00	0.00 ± 0.00	0.13 ± 0.40
No Repeats	0.05 ± 0.22	0.00 ± 0.00	0.00 ± 0.00
Total	9.03 ± 3.36	1.08 ± 0.37	4.44 ± 2.91

Table 8: Post-deployment engagement changes in FLYENJOY relative to the single-heuristic baseline.

Metric	Type	Change
Itinerary Initiations	Usage	+83%
Completion Rate	Usage	+91%
Feature DAU	Engagement	+22%
DAU/WAU Ratio	Engagement	+20 pp
Time per Session	Efficiency	-8%
Time per User	Efficiency	-38%

On-Device Latency Analysis. To empirically validate viability, we measured end-to-end latency on 1,728 runs: 6 cities $\times$ 4 trip configurations $\times$ 24 algorithm configurations (Greedy/Beam Search, parallel/sequential fetching, Basic/Advanced/ML filtering, with/without optimizations) $\times$ 3 repetitions. Latency averages 109.9 ms (std: 69.5, p5 35.1, p95 253.1 ms). Both the mean and the p95 tail fall below the 300 ms interactive-response threshold. The device-aware budget bounds worst-case latency across heterogeneous devices.

Post-Deployment Quantification. We deployed PLA in FLYENJOY, a production iOS travel app. We evaluated real-world impact via anonymized logs in a pre-post observational analysis against the prior single-heuristic baseline; confounders (e.g., organic growth) cannot be fully excluded. Due to business sensitivity, we report only relative changes (Table 8). Itinerary initiations rose 83.33%, and completion rate 90.91%. Engagement also improved: itinerary-feature DAU rose +22% and DAU/WAU +20 pp (more repeat usage). Time per session (-8%) and per user (-38%) fell, reflecting fewer interactions per completed itinerary. Consistent gains across initiation, completion, repeat usage, and session efficiency suggest PLA reduces friction in addition to improving feasibility and preference alignment.

4 Related Work

Itinerary Planning via Optimization and Learning. Classical itinerary planning uses combinatorial optimization to balance POI utility under time-budget constraints [4, 10]. Extensions add real-world constraints [19] or learned preference models [16]. Reinforcement-learning approaches treat itinerary generation as sequential decision-making [3, 5, 6]. Optimization ensures feasibility, learning captures

preferences; neither alone balances both. We separate feasibility-preserving planning from itinerary-level preference learning.

Graph-Based, Multimodal, and LLM-Based Recommendation. Graph and sequence models address next-POI recommendation [11, 13, 17, 18], with multimodal extensions using geo-tagged images [8]. LLM-based generators [9, 15] use step-wise prediction or unconstrained generation and rarely guarantee global feasibility under temporal and resource constraints. Our work targets multi-day generation with feasibility guarantees and on-device execution. **Hybrid Preference Learning and Constraint-Aware Planning.** Hybrid approaches combine preference learning with constraint reasoning. Constraint-based recommenders enforce hard requirements with manually specified preferences [7], while preference models often assume unconstrained candidates [2]. Recent hybrid itinerary systems integrate preferences into optimization [14]. We differ by learning preferences at the itinerary level from pairwise judgments and optimizing them via feasibility-preserving refinement under a device-aware budget.

5 Conclusion

We presented PLA, a three-stage framework for personalized on-device itinerary generation. PLA composes complementary stages to address feasibility, desirability, and viability jointly. Across more than one hundred trip contexts, the planner ensemble reaches 97.6% normalized selection entropy, meaning every algorithm contributes and none dominates. Itinerary-level quality such as pacing, day balance, and geographic coherence emerges from the learned reward model rather than from any per-POI score, and feasibility-preserving refinement removes the need for the post-hoc repair

stage typical of LLM pipelines. In production deployment within FlyEnJoy, PLA increased itinerary completion rates by 91%, providing a blueprint for human-aligned planning under real deployment constraints.

GenAI Usage Disclosure

During the preparation of this work, the authors used LLM-based tools to assist with condensing and improving the clarity of the author-written text within selected sections. No new content, results, analyses or claims were created by AI tools. All AI-assisted edits were reviewed, verified and approved by the authors. The authors take full responsibility for the accuracy and integrity of all content in this paper.

References

- [1] Ralph Allan Bradley and Milton E Terry. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39, 3/4 (1952), 324–345.
- [2] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. In *The Adaptive Web*. 377–408.
- [3] Bo-Hao Chen, Jin Han, Shengxin Chen, Jia-Li Yin, and Zhaojiong Chen. 2022. Automatic itinerary planning using triple-agent deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems* 23, 10 (2022), 18864–18875.
- [4] Gang Chen, Sai Wu, Jingbo Zhou, and Anthony KH Tung. 2013. Automatic itinerary planning for traveling services. *IEEE transactions on knowledge and data engineering* 26, 3 (2013), 514–527.
- [5] Lei Chen, Jie Cao, Haicheng Tao, and Jia Wu. 2023. Trip reinforcement recommendation with graph-based representation learning. *ACM Transactions on Knowledge Discovery from Data* 17, 4 (2023), 1–20.
- [6] Shengxin Chen, Bo-Hao Chen, Zhaojiong Chen, and YunBing Wu. 2020. Itinerary planning via deep reinforcement learning. In *Proceedings of the 2020 international conference on multimedia retrieval*. 286–290.
- [7] Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Zanker. 2015. Constraint-based recommender systems. In *Recommender systems handbook*. Springer, 161–190.
- [8] Zhe Juan, Jing Zhang, and Ming Gao. 2024. A Multimodal Travel Route Recommendation System Leveraging Visual Transformers and Self-Attention Mechanisms. *Frontiers in Neurobotics* 18 (2024), 1439195. doi:10.3389/fnbot.2024.1439195
- [9] Peibo Li, Maarten de Rijke, Hao Xue, Shuang Ao, Yang Song, and Flora D Salim. 2024. Large language models for next point-of-interest recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1463–1472.
- [10] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2015. Personalized tour recommendation based on user interests and points of interest visit durations. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*. 1778–1784.
- [11] Yifang Qin, Yifan Wang, Fang Sun, Wei Ju, Xuyang Hou, Zhe Wang, Jia Cheng, Jun Lei, and Ming Zhang. 2023. DisenPOI: Disentangling sequential and geographical influence for point-of-interest recommendation. In *Proceedings of the sixteenth ACM international conference on web search and data mining*. 508–516.
- [12] Hao Sun, Yunyi Shen, and Jean-Francois Ton. 2024. Rethinking bradley-terry models in preference-based reward modeling: Foundations, theory, and alternatives. *arXiv preprint arXiv:2411.04991* (2024).
- [13] Ke Sun, Tieyun Qian, Tong Chen, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2020. Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 214–221.
- [14] Chieh-Yuan Tsai and Jing-Hao Wang. 2025. A Personalized Itinerary Recommender System: Considering Sequential Pattern Mining. *Electronics* 14, 10 (2025), 2077.
- [15] Katerina Volchek and Stanislav Ivanov. 2024. ChatGPT as a travel itinerary planner. In *ENTER e-Tourism Conference*. Springer, 365–370.
- [16] Jingyuan Wang, Ning Wu, Wayne Xin Zhao, Fanzhang Peng, and Xin Lin. 2019. Empowering A* search algorithms with neural networks for personalized route recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 539–547.
- [17] Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. 2023. Spatio-temporal hypergraph learning for next POI recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*. 403–412.
- [18] Song Yang, Jiamou Liu, and Kaiqi Zhao. 2022. GETNext: Trajectory flow map enhanced transformer for next POI recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on research and development in information retrieval*. 1144–1153.
- [19] Chenyi Zhang, Hongwei Liang, and Ke Wang. 2016. Trip recommendation meets real-world constraints: POI availability, diversity, and traveling time uncertainty. *ACM Transactions on Information Systems (TOIS)* 35, 1 (2016), 1–28.