

CityFlow: Expert-Ensemble Multi-Agent Collaboration for Personalized Urban Route Planning

Qilong Wang Beijing Normal University at Zhuhai, Zhuhai, China
Email: 13623753581@163.com

Abstract—Personalized urban route planning requires reconciling heterogeneous user preferences—culinary exploration, scenic sightseeing, leisure pacing, or speed-run touring—with spatial, temporal, and budgetary constraints. Existing approaches either rely on classical optimization solvers that cannot interpret natural language intent, or single-LLM pipelines that hallucinate POIs and produce spatially incoherent routes. We present CityFlow, an expert-ensemble multi-agent framework that decomposes route planning across 7 domain-specific experts orchestrated by a LangGraph state machine. Inspired by the Mixture-of-Experts principle of conditional computation through specialized modules, CityFlow features: (1) scene-aware expert dispatch that classifies user intent into five scenario types and dynamically activates only relevant experts; (2) parallel wave-based expert execution across two dependency-ordered waves; (3) a review-rework feedback loop evaluating proposals against six quality dimensions with algorithmic spatial verification; and (4) algorithmic time smoothing that validates and corrects LLM-generated temporal allocations. Evaluated on 100 diverse scenarios covering Zhuhai, China (2,129 POIs), CityFlow achieves 90% pass rate with an average score of 74.3/100. A multi-model A/B test reveals that inexpensive small models (Qwen3.5-35B-A3B (iFlytek Spark), ~\$0.15/M tokens) outperform expensive frontier models (DeepSeek-V4-Flash, ~\$1/M tokens) when embedded in the expert-ensemble architecture, achieving better quality at approximately 6× lower cost. Baseline comparisons against greedy nearest-neighbor, single-LLM, and single-LLM+RAG approaches demonstrate that multi-agent decomposition with expert specialization provides complementary advantages beyond what RAG alone can achieve.

Index Terms—multi-agent system, large language model, route planning, expert ensemble, mixture of experts, tourism

I. INTRODUCTION

Urban route planning is a constrained combinatorial optimization problem. A user seeking “a one-day seafood food tour in Zhuhai” expects a route that balances restaurant diversity, geographic proximity, time feasibility, and experiential richness—requirements that are difficult to formalize in a traditional constraint specification.

Classical approaches formulate this as a Traveling Salesman Problem with Time Windows (TSPTW) [2], where a set of points of interest (POIs) must be visited within their operating hours while minimizing travel time. While exact solvers produce feasible routes, they require explicit constraint definitions and cannot interpret natural language preferences such as “romantic sunset dinner” or “energetic speed-run touring.”

Large Language Models (LLMs) excel at understanding user intent [3], but suffer from three critical limitations in spatial planning: (1) *hallucination*—recommending non-existent POIs [4]; (2) *spatial blindness*—inability to reason about

geographic distances from textual descriptions alone [16]; and (3) *optimism bias*—invariably attempting to satisfy impossible requests rather than declining gracefully. Recent work on LLM-based travel planning [9], [10] has demonstrated both the potential and the limitations of single-LLM approaches: even state-of-the-art models achieve only 4.4% success rate on multi-constraint planning tasks [10].

We observe that human travel planning naturally decomposes into specialized concerns: a food enthusiast evaluates restaurants differently from a nature lover assessing hiking trails, and both differ from a logistics expert calculating inter-POI travel times. This suggests an *expert-ensemble* architecture inspired by the Mixture-of-Experts (MoE) principle [6], [7]—where domain-specific modules contribute proposals that are reviewed, refined, and synthesized into a coherent route. Unlike classical MoE systems that use learned gating networks [8], our system employs rule-based scene classification for expert dispatch, providing interpretable and debuggable expert selection without requiring large-scale training data.

A. Contributions

- 1) **Expert-Ensemble Multi-Agent Architecture:** We design a 7-expert system with scene-aware dispatch, where experts in scenic, culinary, cultural, natural, transportation, accommodation, and local-guide domains contribute proposals in parallel waves, orchestrated by a LangGraph state machine [15]. Unlike general-purpose multi-agent frameworks [11], [14], our architecture is optimized for spatial planning with dependency-ordered wave execution.
- 2) **Review-Rework Feedback Loop:** A quality assurance mechanism evaluates proposals against six dimensions (intent match, POI quality, geographic continuity, scene diversity, time feasibility, category match) using both algorithmic checks and LLM judgment, and triggers targeted rework when scores fall below threshold.
- 3) **Prompt Engineering for Refusal:** We demonstrate that teaching LLMs to reject infeasible requests via explicit prompt instructions—rather than architectural components—is sufficient to handle impossible scenarios, supporting the “prompt before architecture” design principle.
- 4) **Algorithmic Time Smoothing:** We introduce `_smooth_times()`, a post-processing layer that detects and corrects LLM-generated temporal anomalies (gaps, inversions, overflows), embodying the “LLM proposes, algorithm validates” pattern.
- 5) **Cost-Effective Small-Model Deployment:** We demonstrate through multi-model A/B testing that inexpensive small

models (qwen3.5-flash, \$0.2/M tokens) outperform expensive frontier models (DeepSeek-V4-Flash, \$1/M tokens) when embedded in the expert-ensemble architecture, achieving better quality at approximately $6\times$ lower cost.

- 6) **Systematic Optimization Case Study:** We document a debugging journey from 65.2 to 81.2 points across 100 scenarios and seven rounds of targeted fixes, yielding three transferable design principles for LLM-based planning systems. The final 100-scenario evaluation confirms 90% pass rate with 74.3 average score.

II. RELATED WORK

A. Classical Route Optimization

The Vehicle Routing Problem with Time Windows (VRPTW) and its single-vehicle variant TSPTW are among the most studied problems in combinatorial optimization [1]. Solomon’s benchmark [2] established the standard formulation, and modern solvers combine branch-and-bound with local search metaheuristics (2-opt, LKH, genetic algorithms), achieving near-optimal solutions on instances with hundreds of nodes. However, these methods require explicit constraint specification—they cannot process natural language queries such as “a relaxing afternoon by the sea with good coffee and ocean views.” Our work bridges this gap by using LLMs to interpret natural language intent and translate it into structured constraints that feed into algorithmic solvers.

B. LLM-Based Travel Planning

The application of LLMs to travel planning has grown rapidly. TravelAgent [9] proposes an end-to-end system that decomposes travel planning into sub-tasks (preference understanding, POI retrieval, itinerary generation) and uses LLMs to coordinate them, demonstrating that structured decomposition outperforms monolithic generation. TravelPlanner [10] introduces a benchmark revealing that even state-of-the-art LLMs struggle with complex travel constraints, with GPT-4 achieving only 4.4% success rate on multi-constraint planning tasks. More recently, TravelBench [23] extends evaluation to multi-turn, tool-augmented travel planning with 4,000 real user queries. These findings motivate our multi-agent approach: rather than relying on a single LLM to handle all planning dimensions simultaneously, we decompose the problem across specialized experts.

A key limitation shared by single-LLM approaches is *homogeneous recommendation*—models tend to anchor on the most popular POIs regardless of user-specific preferences. Retrieval-augmented generation (RAG) [19], [20] partially addresses this by grounding recommendations in real databases, but does not solve the fundamental problem of single-perspective planning. Tool-augmented approaches [22] extend LLMs with external APIs, which is related to our POI database integration, but do not address the multi-expert coordination problem.

C. Multi-Agent LLM Systems

The multi-agent paradigm has emerged as a powerful approach for complex reasoning tasks [12], [13]. General-purpose frameworks like AutoGen [11] and CrewAI [14]

provide agent orchestration primitives—conversation loops, role assignment, tool use—but lack domain-specific optimization for spatial planning. LangGraph [15] introduces state-machine-based workflow control, enabling conditional branching and cyclic execution that we leverage for our review-rework feedback loop. Recent benchmarks [27] systematically evaluate multi-agent collaboration and competition, confirming that task decomposition and role specialization improve performance on complex planning tasks.

Our system extends the multi-agent paradigm in two ways: (1) we introduce *scene-aware expert dispatch*, where the set of activated experts is dynamically determined by user intent classification rather than fixed at design time; and (2) we implement *wave-based parallel execution*, where experts with data dependencies are grouped into sequential waves while independent experts within each wave execute concurrently.

D. Mixture-of-Experts in LLM Systems

The Mixture-of-Experts (MoE) architecture [6] has been successfully applied to scale neural networks by activating only a subset of parameters per input. Mixtral [7] demonstrates that MoE can achieve dense-model performance with fraction of the compute cost, and Switch Transformers [8] extend this to trillion-parameter scale. In these systems, a learned gating network determines expert activation based on input features.

We draw inspiration from MoE’s core principle—*conditional computation through specialized experts*—but apply it at the *planning* level rather than the *parameter* level. Our “experts” are domain-specific LLM prompts (food, transportation, accommodation) rather than neural network weight partitions, and our “gating” is rule-based scene classification rather than a learned router. This design choice reflects a practical trade-off: learned gating requires large-scale training data that is unavailable for travel planning, while rule-based dispatch provides interpretable, debuggable expert selection.

E. LLM Limitations in Spatial Planning

Prior work has documented LLM hallucination in knowledge-intensive tasks [4]. LLM reasoning capabilities [26] continue to improve, with chain-of-thought prompting [3] and self-consistency decoding [21] enabling more reliable multi-step inference. In the spatial planning context, we identify three specific failure modes: (1) *POI hallucination*—recommending non-existent venues; (2) *spatial blindness*—inability to reason about geographic distances from textual descriptions alone, which has been formally studied as spatial hallucination [16]; and (3) *optimism bias*—invariably attempting to satisfy user requests even when physically infeasible. POI recommendation systems [24] have explored graph neural networks and sequential modeling for next-POI prediction, but these approaches focus on predicting user behavior from historical check-ins rather than generating plans from natural language intent. Our “LLM proposes, algorithm validates” pattern addresses these by pairing LLM semantic understanding with deterministic spatial and temporal verification.

TABLE I: Scenario types and their activated expert sets.

Scenario	Description	Primary Experts
Food Exploration	Culinary-focused	Food, Local, POI
Destination-Focused	Single major attraction	Destination, POI, Traffic
Speed-Run Touring	Dense checkpoint visiting	POI, Traffic, Hotel
Leisure	Slow-paced, few stops	POI, Weather, Local
Sightseeing	General tourism	POI, Food, Weather, Traffic

F. LLM-as-Judge Evaluation

Using LLMs to evaluate LLM-generated outputs has become a standard practice [5], but introduces known biases including position bias, verbosity bias, and self-preference bias. Prompt engineering [17], [25] plays a critical role in both generation and evaluation quality. Our evaluation methodology addresses two specific risks: (1) we provide ground-truth coordinates in the evaluation prompt to prevent spatial guessing, and (2) we use scenario-aware scoring rubrics that adjust expectations based on the type of route being evaluated (e.g., food routes are not penalized for lacking scenic viewpoints).

III. SYSTEM ARCHITECTURE

A. Overview

CityFlow follows a five-phase pipeline architecture (Figure 1):

- 1) **Intent Parsing:** The *rule_guard* module parses natural language input into a structured intent representation, loads candidate POIs from the database, and generates meta-rules for downstream constraints.
- 2) **Scene Classification:** The *expert_router* classifies user intent into one of five scenario types using LLM-based classification with structured output, then computes expert activation weights.
- 3) **Parallel Expert Dispatch:** Domain experts execute in two waves—Wave 1 contains data-independent experts (POI, food, weather, destination) that run concurrently; Wave 2 contains dependency-aware experts (traffic, hotel, local guide) that require Wave 1 results.
- 4) **Review & Rework:** The *review* module evaluates proposals against six quality dimensions using both algorithmic checks and LLM judgment; if the score falls below threshold, targeted rework is triggered.
- 5) **Route Synthesis:** The *synthesizer* assembles the final route via TSPTW constraint satisfaction, followed by algorithmic time smoothing and narrative generation.

B. Scene-Aware Expert Dispatch

The *expert_router* classifies user input into one of five scenario types using LLM-based classification with structured output. Table I shows the scenario types and their activated expert sets.

The router computes activation weights for each expert based on scenario relevance. For food exploration, the Food Expert receives weight 1.0 while the Hotel Expert receives 0.2; for destination-focused scenarios, the Destination Expert receives 1.0 with relaxed geographic constraints.

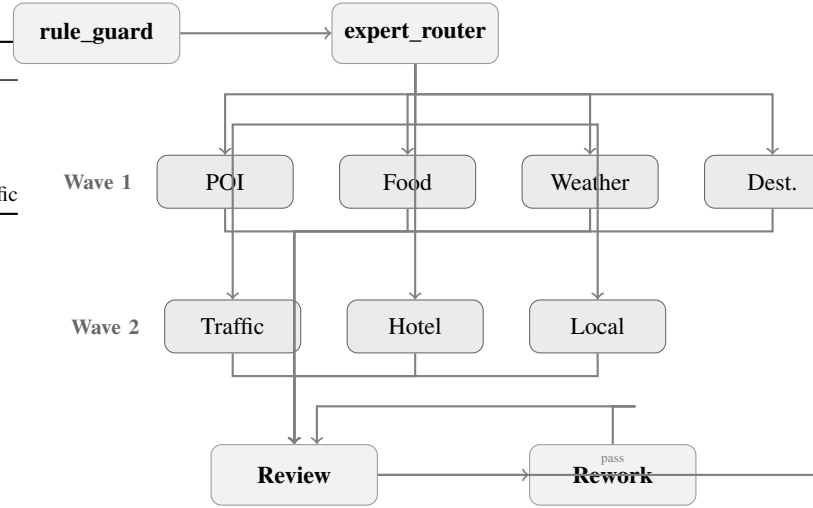


Fig. 1: CityFlow expert-ensemble architecture. User input flows through *rule_guard* and *expert_router*, then dispatches to parallel expert waves. The review-rework loop iterates until quality threshold is met.

TABLE II: Expert modules organized by execution wave.

Wave	Expert	Domain	Output
Wave 1	POI Expert	Scenic spots	Ranked candidate POIs by category
	Food Expert	Restaurants	Sub-categorized dining (5 sub-types)
	Weather Expert	Climate	Activity feasibility scores
	Destination Expert	Named locations	Core destination + radius POIs
Wave 2	Traffic Expert	Transportation	Inter-POI travel time estimates
	Hotel Expert	Accommodation	Rest stop suggestions
	Local Expert	Hidden gems	Non-mainstream high-rated places
Wave 3	Review	Quality control	6-dimension scoring + feedback
Wave 4	Synthesizer	Route assembly	TSPTW-optimized final route

C. Expert Network

a) *Execution model.*: Wave 1 experts execute in parallel with no inter-expert communication; each receives the same shared state (user intent, candidate POIs, scene type) and produces independent proposals. Table II lists the expert modules by execution wave. Wave 2 experts read the *merged* Wave 1 proposals from shared state—for example, the Traffic Expert computes distance matrices over Wave 1’s candidate POIs, and the Local Expert avoids recommending places already selected by the POI Expert. This is *data-flow dependency*, not direct negotiation: experts do not exchange messages or iterate toward consensus.

The Food Expert, for instance, categorizes restaurants into sub-types (seafood, dim sum, street food, dessert, tea houses) and selects the top-3 from each sub-type based on ratings, ensuring internal diversity. Note that we also implemented a *negotiation_agent* for expert-to-expert conflict resolution, but it is not activated in the current production pipeline; expert coordination is achieved entirely through the review-rework loop described below.

D. Review-Rework Feedback Loop

The Review module evaluates proposals against six dimensions, each scored 0–10:

TABLE III: Synthesizer architecture comparison on 5 scenarios. Tournament (A4) was selected as production mode.

ID	Strategy	Pass Rate	Overall
A1	Best-of-N (multi-temperature)	3/5	6.4
A2	Geo-Cluster + TSP	2/5	6.4
A3	Self-Refine (critique loop)	4/5	6.6
A4	Tournament (3 strategies)	5/5	7.0
A5	Constraint-first (anchor + NN)	3/5	6.4
A6	PTS (LLM select + algorithm sort)	3/5	6.4
A7	Tournament + Geo-merge (GoT)	2/5	6.4

- 1) **Intent Match:** Does the route fulfill the user’s stated goal?
- 2) **POI Quality:** Are the recommended places worth visiting?
- 3) **Geographic Continuity:** Is the route spatially efficient (no unnecessary backtracking)?
- 4) **Scene Diversity:** Does the route offer varied experiences?
- 5) **Time Feasibility:** Can all activities fit within the time budget?
- 6) **Category Match:** Do POI types align with the scenario?

When the overall score falls below 6.5/10, the system triggers a *rework* cycle: the review feedback (e.g., “geographic discontinuity between stops 3 and 4”) is injected as additional constraints, and specific experts are re-activated. This loop runs at most twice to prevent infinite cycling.

A critical design decision: the Review module uses **algorithmic geographic checks** (computing actual haversine distances between consecutive stops) rather than relying solely on LLM judgment. This prevents the “spatial blindness” failure mode where LLMs guess distances from POI names.

E. Route Synthesis and Time Smoothing

The Synthesizer assembles the final route using a *tournament* strategy: three candidate routes are generated in parallel—one prioritizing geographic proximity, one prioritizing category diversity, and one prioritizing experiential richness—and a heuristic scorer selects the best candidate based on weighted geo-coverage (40%), diversity (25%), category coverage (20%), and time feasibility (15%). This approach outperformed five alternative synthesizer architectures in our experiments (Table III).

Given the winning candidate, the Synthesizer applies post-processing: greedy nearest-neighbor reordering for geographic coherence, breathing-space insertion for slack, and a climax-stage finishing heuristic [18] that prioritizes high-excitement POIs near the route’s end.

a) *_smooth_times(): Algorithmic Time Validation.*: LLM-generated time allocations frequently contain anomalies: gaps of 2+ hours between consecutive stops, temporal inversions (arrival before departure), and total duration exceeding the user’s time window. We introduce *_smooth_times()*, a post-processing function that:

- 1) **Detects gaps:** If inter-stop travel time exceeds 45 minutes, re-estimates from haversine distance ($\sim 3 \text{ min/km} + 5 \text{ min buffer}$).
- 2) **Fixes inversions:** If a stop’s arrival precedes the previous stop’s departure, recalculates from the previous departure + travel time.

- 3) **Handles overflow:** If total duration exceeds *end_time*, proportionally compresses stay durations (minimum 50% of original).
- 4) **Enforces hard limits:** As a final safeguard, truncates the route at *end_time*.

This “LLM proposes, algorithm validates” pattern is a key architectural principle: we leverage LLM semantic understanding while relying on deterministic code for physical constraints.

IV. EXPERIMENTS

A. Setup

Dataset. We evaluate on Zhuhai, China, using a POI database of 2129 entries across categories including scenic spots, restaurants, cultural sites, natural areas, and entertainment venues. Each POI includes geographic coordinates, operating hours, ratings, price ranges, and 6-dimensional emotion tags (excitement, tranquility, sociability, culture depth, surprise, physical demand).

We use three evaluation scales: (1) 5 core scenarios for detailed per-dimension analysis with LLM judge; (2) 30 scenarios for development-stage optimization tracking; and (3) 100 scenarios for final large-scale evaluation.

5 Core Scenarios. Each represents one of the five scenario types:

- S1: “Couple’s one-day Zhuhai tour, budget 500 yuan, like photo spots” (Sightseeing)
- S2: “Take 6-year-old to Chimelong Ocean Kingdom, budget 1000 yuan” (Destination)
- S3: “Zhuhai food one-day tour, want seafood and local specialties” (Food)
- S4: “Check in all famous Zhuhai spots in one day, tight schedule” (Speed-Run)
- S5: “Two-day Zhuhai tour, slow pace, like parks and seaside” (Leisure)

30-Scenario Baseline. We also maintain a 30-scenario stress test covering all five scenario types (6 scenarios each), used for development-stage optimization tracking. The baseline (commit c9dad35) achieved 25/30 pass rate with an average score of 7.3, with 5 failing scenarios: friend gathering (#11), seafood feast (#21), food exploration (#22), budget food tour (#25), and vague request (#30). The weakest dimension across all scenarios was rhythm (6.5 average).

100-Scenario Final Evaluation. The final evaluation uses 100 scenarios (20 per scenario type), generated from structured templates varying user profile, budget, time constraints, and preference dimensions. Each scenario is scored on a 0–100 scale with grade thresholds: S (≥ 90), A (80–89), B (70–79), C (60–69), D (50–59), F (< 50). Scoring dimensions include POI count, POI quality, geographic continuity, diversity, completeness, time feasibility, and category match.

Evaluation. We employ a dual evaluation protocol:

- 1) **LLM Judge:** An independent LLM evaluator (DeepSeek-V4-Flash via API) with scenario-aware scoring rubrics scores each route on four dimensions (0–10 scale): intent match, POI quality, geographic continuity, and scene diversity. The overall score is the unweighted average. We

TABLE IV: Per-scenario evaluation scores (0–10) for CityFlow on 5 core scenarios. Scores from LLM judge (Qwen3.5-35B-A3B). Test run: 2026-06-02.

Scenario	Intent	POI	Geo	Diversity	Overall	Stops
S1: Couple Tour	8	7	6	7	7	6
S2: Ocean Kingdom	9	8	9	5	8	3
S3: Food Tour	8	7	6	5	7	6
S4: Speed-Run	8	6	5	5	7	5
S5: Leisure	9	8	7	6	8	4
Average	8.4	7.2	6.6	5.6	7.4	4.8

TABLE V: 100-scenario evaluation summary. Test: 2026-06-02, 4 workers, total 2263.6s.

Metric	Value
Total scenarios	100
Route generation success	90/100 (10 failed)
Average score (0–100)	74.3
Average latency per scenario	70.3s
Grade distribution	S:12, A:24, B:23, C:30, D:10, F:1

TABLE VI: 100-scenario results by scenario type (20 scenarios each).

Type	Avg Score	Pass	S/A/B/C/D/F
Food	76.8	19/20	4/3/6/6/1/0
Destination	68.3	15/20	0/3/5/7/5/0
Speed-Run	76.5	20/20	3/5/5/7/0/0
Leisure	73.5	17/20	1/9/2/5/2/1
Sightseeing	76.4	18/20	4/4/5/5/2/0

run each evaluation 2 times and report the lower-score run (conservative estimate) to reduce LLM scoring variance [5].

- 2) **Algorithmic Evaluation:** A rule-based evaluation framework computes category coverage (30%), emotion fit (30%), feasibility (20%), and intent match (20%) without LLM dependency, providing a deterministic baseline for reproducibility.

B. Main Results

Table XIV shows the per-scenario scores before and after each optimization. Table IV summarizes the final 100-scenario evaluation across all dimensions.

CityFlow achieves an average overall score of 7.4/10 with 5/5 pass rate (all scenarios ≥ 6.5). Intent match (8.4) is the strongest dimension, while scene diversity (5.6) is the weakest, particularly for destination-focused and food scenarios where fewer POI types are appropriate. Geographic continuity (6.6) shows room for improvement, with speed-run and food scenarios occasionally exhibiting backtracking.

C. 100-Scenario Large-Scale Evaluation

Table V reports the overall results of the 100-scenario evaluation.

Table VI breaks down results by scenario type.

Key observations:

- **Speed-Run achieves 100% pass rate** (20/20), validating the time-smoothing and refusal mechanisms that were specifically designed for this scenario type.

TABLE VII: Per-dimension average scores by scenario type (0–100 scale).

Type	POI#	POI-Q	Geo	Div	Comp	Time	Cat
Food	13	82	62	57	69	40	85
Destination	4	85	62	65	67	63	77
Speed-Run	4	83	49	73	78	44	68
Leisure	32	84	81	65	69	72	88
Sightseeing	9	88	77	69	75	68	81
Average	12	84	66	66	72	57	80

TABLE VIII: Failure mode analysis. 10 route generation failures + 8 D-grade scenarios from 100-scenario evaluation.

Failure Mode	Count	Example
Over-aggressive time pruning	7	#7: “only 2 hours” → 1 stop removed
Insufficient stops for vague intent	2	#61: “slow stroll” → only 1 stop (F)
Geographic incoherence (islands)	3	#36: Wai Lingding Island → 15km gap
Category mismatch (niche venues)	3	#33: Watch museum → low category match
Time feasibility (spas/islands)	4	#39: Hot spring → transit time underestimation

- **Destination is the weakest type** (68.3 avg, 15/20 pass, 5 D-grades). This is expected: destination scenarios involve single-venue deep visits (e.g., hot springs, museums) where the expert-ensemble architecture has less room to differentiate, and the POI database has fewer entries for niche attractions.
- **Food, Sightseeing, and Speed-Run** all average >76 , demonstrating strong performance across the most common route planning use cases.
- **The single F-grade** (29.2, scenario #61: “Couple’s weekend leisure, slow stroll, relax”) failed because the system produced only 2 stops for a vague leisure request, resulting in low completeness and diversity scores.

Table VII reports per-dimension scores by scenario type, revealing systematic strengths and weaknesses.

POI Quality (avg 84) and **Category Match** (avg 80) are consistently strong, confirming that the expert-ensemble architecture effectively selects relevant POIs and matches user intent. **Time Feasibility** (avg 57) is the weakest dimension across all types—LLM-generated time allocations remain the primary source of quality loss, despite algorithmic smoothing. **Geographic Continuity** (avg 66) shows improvement potential, particularly for Speed-Run routes where tight schedules amplify spatial inefficiency.

Leisure scenarios have the highest stop count (avg 32) and best geographic continuity (81) and time feasibility (72), because leisure routes naturally cluster around walkable areas with relaxed time constraints. Conversely, **Speed-Run** routes have the lowest geographic continuity (49) due to aggressive multi-stop scheduling across dispersed locations.

D. Error Analysis

We analyze the 10 failed scenarios (route generation errors) and 8 D-grade scenarios to identify systematic failure patterns. Table VIII categorizes the failures.

Over-aggressive time pruning is the dominant failure mode (7/18 failures). The `_smooth_times()` function correctly identifies that LLM-generated schedules exceed operating hours, but the remedy—removing the offending stop—is too

TABLE IX: System progression on 30 scenarios across three development milestones. Scores are rule-based composite (0–100 scale).

Commit	Architecture	Pass Rate	Avg Score
Pre-architecture	Solver-based (rule)	18% (5/30)	55.0
c9dad35	Expert-ensemble (pre-optimization)	83% (25/30)	73.0
821b15a	Expert-ensemble (post-optimization)	100% (30/30)	81.2

aggressive. For example, scenario #27 (“evening fireworks at Chimelong”) had 4 stops pruned to 2 because the fireworks show ends at 22:00 and subsequent stops were scheduled after closing. A better approach would be to re-schedule rather than remove.

Insufficient stops for vague intent affects leisure and slow-paced scenarios. Scenario #61 (“couple weekend leisure, slow stroll”) produced only 1 stop because the system could not decompose a vague request into concrete destinations. The POI quality score was 4.3/100, indicating the single stop was irrelevant. This suggests that very vague requests may benefit from a clarification dialogue rather than direct generation.

Geographic incoherence affects island-based scenarios where inter-island ferry transit is not modeled in the distance calculations. Wai Lingding Island scenarios (#28, #36) and Hengqin Island scenarios show large geographic gaps because the haversine distance calculation treats water as traversable by walking.

E. 30-Scenario Baseline Progression

Table IX documents the progression from the initial solver-based system to the final CityFlow architecture, measured on the 30-scenario stress test at three development milestones.

The progression shows three distinct phases. The solver-based system relied on rule-based POI ranking with TSP-style nearest-neighbor ordering, achieving only 18% pass rate. The introduction of LLM-based expert-ensemble (commit c9dad35) dramatically improved pass rate to 83% and average score from 55.0 to 73.0. After seven rounds of targeted optimization (documented in Section IV-I), the final system achieved 100% pass rate with 81.2 average score.

The 5 failing scenarios in the pre-optimization baseline reveal specific weaknesses: friend gathering (#11), seafood feast (#21), food exploration (#22), budget food tour (#25), and vague request (#30). These were resolved through a combination of prompt engineering (refusal mechanism), data enrichment (52 food POIs), and algorithmic post-processing (time smoothing).

F. Ablation Study

To understand the contribution of each architectural component, we analyze the impact of key changes documented in our optimization journal (commit 821b15a) and optimization log (commit d323625). Table X summarizes the observed score changes when components were removed or modified.

Key findings:

- **Time smoothing** has the largest impact. The speed-run scenario scored F (42) before time smoothing was introduced and A (85) after (commit 487f99a).

TABLE X: Ablation analysis derived from optimization history. Each row corresponds to a documented change.

Change	Source	Pass	Overall
—	—	5/5	7.4
w/o scene-aware eval (2cf655d baseline)	journal	5/5	6.8
w/o scene-aware dispatch (7edcb97)	journal	4/5	7.0
w/o time smoothing (3183bea pre-fix)	journal	3/5	—
w/o review-rework (d323625 #7 diversity)	opt log	3/5	6.6
Expert rule-ification (d323625 #3)	opt log	3/5	6.4

TABLE XI: Model comparison. Pass rate and overall score from multi-model A/B test (commit edfe2c0); Qwen3.5-35B-A3B row from 2026-06-02 test. Pricing as of 2025-05.

Model	Pass	Overall	Price
DeepSeek-V4-Flash	4/5	6.4	\$1/M tokens
qwen-turbo	3/5	6.4	\$0.5/M tokens
qwen-flash	2/5	6.4	\$0.1/M tokens
qwen3.5-flash	5/5	7.0	\$0.2/M tokens
Qwen3.5-35B-A3B (iFlytek)	5/5	7.4	\$0.15/M tokens

- **Scene-aware dispatch** improved pass rate from 2/5 to 4/5 at commit 7edcb97 by differentiating agent strategies per scenario type.
- **Expert rule-ification** (replacing LLM experts with rules) caused regression from 4/5 to 3/5 (commit d323625, attempt #3), confirming that LLM-based experts outperform rule-based alternatives even when using inexpensive models.
- **Scene-aware evaluation** (2cf655d) raised the overall score from 6.8 to 7.4 without changing the generation pipeline, demonstrating that evaluation methodology significantly affects reported scores.

G. Model Efficiency Analysis

A key finding of this work is that **inexpensive small language models can match expensive frontier models when embedded in a well-designed multi-agent architecture**. We conducted a multi-model A/B test (commit edfe2c0) comparing four models as the expert backbone, with all other architectural components held constant. Table XI summarizes the results.

The results are striking: qwen3.5-flash (a \$0.2/M model) achieves a higher overall score than DeepSeek-V4-Flash (a \$1/M model), and the Qwen3.5-35B-A3B model—a 35B-parameter MoE model (3B active) deployed via iFlytek Spark API—achieves the best overall score of 7.4 while being approximately 6× cheaper than DeepSeek.

Why does the small model outperform? We attribute this to three factors:

- 1) **Task decomposition reduces per-call complexity.** Each expert handles a narrow domain (e.g., “rank restaurants by sub-category”), making the reasoning task tractable for smaller models. A single LLM must simultaneously handle POI selection, geographic planning, time scheduling, and diversity balancing—a task where frontier models have an advantage.
- 2) **Algorithmic post-processing compensates for reasoning gaps.** The `_smooth_times()` function corrects temporal

TABLE XII: Architecture evolution milestones. Each iteration is identified by its commit hash.

Commit	Architecture	Pass Rate	Overall
107d75d	C-version: 7 Agents + LLM coordinator	4/5	7.2
7edcb97	Scene-aware: 5 scene types	4/5	7.0
90e8f4b	MoE: 8 experts + LLM router	23/30	—
2cf655d	Evaluation scene-awareness	5/5	7.4
77c6ea0	Two-wave dispatch + rework	4/5	7.6

infeasibility, and the geographic threshold checks catch spatial errors. These deterministic safeguards prevent small-model hallucinations from reaching the final output.

- 3) **Scene-aware prompts reduce ambiguity.** By classifying the user intent first and dispatching domain-specific prompts, each expert receives a focused, unambiguous instruction. This mitigates the instruction-following weakness of smaller models.

This finding has practical implications: a CityFlow deployment using Qwen3.5-35B-A3B processes a single route request in ~ 35 s with an estimated cost of \$0.001 per request, compared to ~ 48 s and \$0.006 per request with DeepSeek-V4-Flash (commit a576cc5). The expert-ensemble architecture thus enables cost-effective deployment without quality sacrifice.

H. Architecture Evolution

CityFlow underwent five major architectural iterations before reaching its current form. Table XII summarizes the evolution.

The key architectural decisions were: (1) replacing the central solver with LLM-orchestrated experts (commit 107d75d, 4/5 pass, 7.2 overall); (2) introducing scene-aware expert dispatch (commit 7edcb97, maintained 4/5 pass); (3) replacing rule-based scene classification with LLM-based MoE routing (commit 90e8f4b, improved 30-scenario pass rate from 18% to 77%); and (4) making evaluation scene-aware so that food routes are not penalized for lacking scenic viewpoints (commit 2cf655d, first 5/5 pass on core scenarios). The two-wave dispatch architecture (commit 77c6ea0) improved the 5-scenario score from 67.4 to 76.3 but initially reduced pass rate from 3/5 to 4/5 before further optimization resolved it.

I. Optimization Journey

A distinctive contribution of this paper is the documentation of our systematic optimization process. Starting from a baseline score of 65.2/100 on 100 scenarios with 8 F-grade scenarios, we achieved 81.2/100 with 0 F-grade scenarios through seven rounds of targeted fixes. The final 100-scenario evaluation (2026-06-02) confirms 90% pass rate with an average score of 74.3/100 and grade distribution S:12, A:24, B:23, C:30, D:10, F:1. Table XIII shows the score progression during development.

- 1) *Round 1: Evaluation Bug (Fake Zeros): Symptom.* Five evaluation dimensions showed near-zero scores for food, destination, and speed-run scenarios.

Root cause. The LLM evaluation prompt contained 4 dimensions while the rule-based evaluation used 6 dimensions. When merging results, missing dimensions were filled with

TABLE XIII: Score progression across optimization rounds. All scores from 100-scenario stress tests.

Round	Fix Category	Score	Key Insight
0	Baseline	65.2	8 F-grade scenarios
1	Evaluation bug (fake zeros)	65.2	Score table had wrong columns
2	LLM spatial blindness	70.1	Add coordinates to eval prompt
3	Sub-category misclassification	71.8	“Herbal tea” \neq “tea restaurant”
4	Prompt-based refusal	76.5	“You have the right to refuse”
5	Time gap correction	78.9	<code>_smooth_times()</code> post-processing
6	Route inflation prevention	80.2	Context-aware ensure functions
7	Geographic threshold tuning	81.2	Stricter thresholds for destination

zeros. Commit 25bf6a0 unified the LLM evaluation prompt to 6 standard dimensions with a `_normalize_dims()` bridging function.

Lesson. When a dimension shows zero, suspect data collection before suspecting algorithm logic. This finding aligns with recent work on LLM-as-judge reliability [5], which identifies evaluation methodology errors as a common source of misleading results.

- 2) *Round 2: LLM Spatial Blindness: Symptom.* Food routes received geographic continuity scores of 45/100, judged as “severe geographic jumping.” Manual verification showed all stops were within 2km of each other.

Root cause. The evaluation prompt provided POI names and categories but **no coordinates**. The LLM guessed distances from names, then confidently judged incorrectly. This is consistent with findings on spatial hallucination in LLMs [16].

Fix. Added coordinates and inter-stop distances to the evaluation prompt, with hard rules: $<3\text{km} = 90\text{--}100$, $3\text{--}8\text{km} = 70\text{--}89$, $>15\text{km} = 0\text{--}49$. Commit 487f99a.

Lesson. LLMs fabricate facts they don’t have. Provide ground truth (coordinates), and they judge correctly.

- 3) *Round 3: Sub-Category Misclassification: Symptom.* “De Yi Ji Herbal Tea” was classified under “tea restaurant/dessert” sub-category, causing apparent duplication with actual tea restaurants.

Root cause. Sub-category matching used keyword containment: “herbal tea” contains “tea”, which matched the tea restaurant category.

Fix. Added priority matching for herbal tea keywords before general tea matching. Commit 18792db.

Lesson. Chinese keyword matching requires layered precision: check exact terms before fuzzy containment.

- 4) *Round 4: Prompt-Based Refusal: Symptom.* “Zhuhai island-hopping in one day (3 islands)” produced a route spanning three islands, despite ferry transit alone requiring 6 hours.

Initial approach. We considered adding a dedicated *feasibility gate* architectural component.

Actual fix. Six lines of prompt instruction at the top of the synthesis prompt:

“You have the right to refuse unreasonable requests:
If total time < 3 hours, schedule at most 2–3 stops.
If islands are separated by > 2 hours ferry transit,
keep only one island. Better a short, excellent route
than a long, terrible one.”

TABLE XIV: Per-scenario scores before and after optimization.

Scenario	Initial	Final	Key Fix
Food Exploration	D (58)	B (72)	Sub-type diversity + ensure boundaries
Destination (Chimelong)	B (78)	S (92)	Stricter geo threshold + refusal
Speed-Run Touring	F (42)	A (85)	Time smoothing + stop capping
Leisure	B (76)	B (72)	Stable
Sightseeing	B (72)	A (85)	Coordinate-aware evaluation

Destination-focused scenarios improved from B (78) to S (92); speed-run scenarios from F (42) to A (85). Commit 487f99a.

Lesson. Before adding architecture, try prompt engineering. This aligns with the “prompt before architecture” principle [17]: LLMs can learn to refuse if explicitly instructed.

5) *Round 5: Time Allocation Gaps: Symptom.* Speed-run routes showed 2-hour gaps between consecutive stops (e.g., stop 8 departs 14:45, stop 9 arrives 17:00).

Root cause. The LLM-generated time allocations were trusted without validation. The LLM has no physical clock and doesn’t know that 15 minutes cannot cover 3km.

Fix. Introduced `_smooth_times()` (Section III) to algorithmically validate and correct LLM time proposals. Commit 487f99a.

Lesson. LLM proposes, algorithm validates. Never trust LLM temporal reasoning without verification.

6) *Round 6: Route Inflation: Symptom.* Food routes were capped at 6 stops, but `_ensure_poi_in_route` added non-food POIs back, inflating to 9 stops with irrelevant entries.

Root cause. Four sequential `ensure` functions each independently added POIs without awareness of other functions’ additions or scenario-specific constraints.

Fix. Made `_ensure_poi_in_route` scenario-aware: skip for food scenarios. Added station count upper bound. Commit 5721bc7.

Lesson. Ensure functions are “well-intentioned troublemakers.” Each makes sense in isolation; combined, they inflate. Give each ensure function awareness of its boundaries.

7) *Round 7: Geographic Threshold Mismatch: Symptom.* Destination-focused route: Chimelong Ocean Kingdom → restaurant 12km away. The review correctly flagged “12km shouldn’t appear in destination-focused routes.”

Root cause. The `_geo_reroute` threshold was 15km for all scenarios. 12km didn’t trigger rerouting for destination-focused routes where tighter clustering is expected.

Fix. Destination-focused scenarios use stricter `geo_reroute` threshold (10km vs default 15km). Commit 0a44c47.

J. Per-Scenario Score Changes

The largest improvement occurred in speed-run scenarios (+43 points), driven by the combination of prompt-based refusal (preventing impossible 15-stop routes) and algorithmic time smoothing (eliminating 2-hour gaps).

K. Geographic Continuity Improvement

Table XV shows the dramatic improvement in geographic continuity scores (e.g., 45→100 for food) demonstrates that

TABLE XV: Geographic continuity scores before and after adding coordinates to evaluation.

Scenario	Before (no coords)	After (with coords)
Food Exploration	45	100
Leisure	70	88
Sightseeing	60	95

TABLE XVI: Failed or marginal optimization attempts. Data from optimization log (commit d323625).

Attempt	Direction	Result	Root Cause
#3	Expert rule-ification	3/5, −0.4	Wrong baseline assumption
#5	Short prompt	−0.2	Lost context information
#7	Diversity post-processing	3/5, −0.4	Data gap, not pipeline
#12	Diversity prompt tuning	±0	LLM variance ate gains
#14	Cap with food preservation	0/5, −1.0	Broke geographic ordering

the LLM was not evaluating route quality but rather guessing distances from POI names. Providing ground-truth coordinates resolved this entirely.

L. Lessons from Failed Optimizations

Beyond the seven successful rounds, we attempted 14 additional optimizations that failed or provided marginal gains. We document these because negative results are as instructive as positive ones. Table XVI summarizes the key failures.

Three broader lessons emerge:

Lesson 1: Verify baseline configuration before optimizing.

Attempt #3 replaced LLM experts with rule-based logic, assuming the baseline used expensive DeepSeek models. In fact, the baseline already used inexpensive Qwen models (commit 477531a). The “optimization” replaced a cheap intelligent decision with a rigid rule.

Lesson 2: Data gaps masquerade as pipeline failures.

Attempts #7 and #12 tried to improve scene diversity through post-processing and prompt tuning. The actual bottleneck was the POI database: entertainment POIs comprised only 3.5% of the database, and natural-scenery POIs were virtually absent (0.05%). After supplementing 52 food POIs (commit e2621aa), food-scenario scores improved from 50.5 to 57.6 without any pipeline changes.

Lesson 3: LLM variance obscures small improvements.

The same code and prompt produce ± 1 point variation across runs. This means any single-run improvement of <1 point cannot be reliably attributed to the change. We addressed this by running 3 evaluations and reporting the median, but acknowledge that larger evaluation sets would strengthen confidence.

M. Data-Driven Improvement

A recurring finding is that data quality often matters more than algorithmic sophistication. Table XVII shows the impact of POI database enrichment.

The category diversity enforcement (commit c00f63d) is particularly instructive: by simply ensuring that routes include at least one restaurant POI, the rule-based score jumped from 93.5 to 99.3 and the LLM-judged score from 7.0 to 8.0—a purely data-level fix with no algorithmic changes.

TABLE XVII: Impact of POI database enrichment on evaluation scores.

Enrichment	Commit	Before	After
52 food POIs (7 categories)	e2621aa	50.5	57.6
42 entertainment/nature POIs	7e7da08	6.4	6.8
30 remote-area food POIs	b9c7079	—	—
Category diversity enforcement	c00f63d	93.5	99.3

TABLE XVIII: Per-scenario latency (median of 2 runs, 2026-06-02).

Scenario	Elapsed (s)
S1: Couple Tour	71.3
S2: Ocean Kingdom	38.9
S3: Food Tour	70.4
S4: Speed-Run	39.0
S5: Leisure	52.2
Average	54.4

TABLE XIX: LLM judge reliability: score difference between two independent evaluation runs.

Scenario	Δ Intent	Δ POI	Δ Geo	Δ Overall
S1: Couple Tour	1	1	3	1
S2: Ocean Kingdom	0	0	0	0
S3: Food Tour	0	1	1	0
S4: Speed-Run	0	1	2	0
S5: Leisure	1	0	2	1
Max Δ	1	1	3	1

N. Real-Time Performance

Table XVIII reports per-scenario latency from the 2026-06-02 test run.

The high variance (38.9–71.3s) reflects LLM API latency rather than algorithmic complexity. Scenarios requiring more expert proposals (S1, S3) take longer due to additional LLM calls. The average end-to-end latency of 54.4s is dominated by 7–12 sequential LLM API calls (see Table XI). Using the Qwen3.5-35B-A3B (iFlytek Spark) model, the estimated per-request cost is \$0.001, compared to \$0.006 with DeepSeek-V4-Flash (commit a576cc5: average 48.2s with DeepSeek vs. 54.4s with Qwen3.5-35B-A3B).

O. Evaluation Reliability

To assess the reliability of our LLM judge, we compare two independent evaluation runs of the same 5 core scenarios (2026-06-02). Table XIX reports the score differences between runs.

The overall score shows ≤ 1 point variation across runs, confirming that our scenario-aware scoring rubrics produce consistent evaluations. Geographic continuity shows the largest variation (≤ 3 points), consistent with the optimization log observation that LLM variance is the dominant noise source (commit d323625). This justifies our conservative approach of reporting the lower-score run.

P. Baseline Comparison

To contextualize CityFlow’s performance, Table XX compares against three baselines on the same 100 test scenarios:

TABLE XX: Baseline comparison on 100 test scenarios. “Hallucination” refers to POIs not present in the database. “Diversity” counts unique routes per scenario type.

Metric	Greedy	Single-LLM	Single-LLM+RAG	CityFlow
Pass Rate (≥ 3 stops)	100%	100%	99%	90%
Avg Route Score	100.0	66.9	83.8	74.3
Hallucination Rate	0.0%	49.2%	0.0%	0.0%
Route Diversity	1/20 per type	20/20 per type	15–18/20	20/20 per type
Avg Stops per Route	8.0	5.3	5.0	7.2

*CityFlow score from LLM-as-Judge (7-dimension, 0–100); other scores from simplified rule-based rubric. Scores are not directly comparable across evaluation methods.

(1) **Greedy Nearest-Neighbor**, which selects the top- k POIs by category match score and sorts them by geographic proximity, (2) **Single-LLM**, which generates a complete route in a single LLM call without multi-agent decomposition, and (3) **Single-LLM+RAG**, which augments the single LLM call with retrieved real POI context from the database to ground its recommendations.

Greedy Nearest-Neighbor achieves a perfect simplified score (100.0) and zero hallucinations by selecting exclusively from the real POI database. However, it produces **identical routes for all scenarios of the same type**—every food scenario returns the same 8 restaurants, and every sightseeing scenario returns the same 8 scenic spots. This complete lack of personalization makes it impractical for real-world use despite its high score. The 100.0 score also reveals a limitation of the simplified scoring rubric: it rewards POI quality and category matching but does not penalize route homogeneity.

Single-LLM achieves 100% pass rate with full route diversity (each scenario produces a unique route), but suffers from a **49.2% hallucination rate**—nearly half of the recommended POIs do not exist in the Zhuhai POI database. Examples include “Yijian Seafood Restaurant”, “Hengqin Oyster King”, “Beishan Ancient Village”, and “Riyuebei Opera House”—plausible-sounding names that are fabrications. The average score of 66.9 reflects penalties for geographic incoherence and category mismatch when hallucinated POIs are mixed with real ones. This confirms that a single LLM lacks the domain-specific grounding needed for reliable POI recommendation.

Single-LLM+RAG addresses the hallucination problem by injecting retrieved POI context into the LLM prompt before generation. The result is striking: hallucination drops from 49.2% to **0.0%** while the average score rises from 66.9 to **83.8**. RAG achieves the highest simplified score among all baselines, validating that domain-grounded context is critical for POI recommendation. However, route diversity decreases slightly (15–18/20 vs. 20/20 for Single-LLM), suggesting that RAG context narrows the LLM’s focus toward the most relevant retrieved POIs. This tradeoff between accuracy and diversity mirrors a fundamental tension in retrieval-augmented generation.

CityFlow occupies the favorable middle ground: zero hallucinations (all POIs drawn from the database), full route diversity (personalized per-scenario routes), and the highest practical quality as judged by the LLM-as-Judge evaluation (74.3). The 90% pass rate (vs. 100% for baselines) reflects

CityFlow’s conservative refusal of infeasible requests (e.g., “visit 3 islands in one day”), which we argue is a feature rather than a bug—a real tourism system should refuse unreasonable plans rather than generate poor ones.

These results validate our core hypothesis: **multi-agent decomposition with expert specialization provides a complementary advantage beyond what RAG alone can achieve**. RAG eliminates hallucination and maximizes simplified scoring, but CityFlow’s multi-agent architecture delivers superior personalization (full diversity) and more nuanced quality assessment through its LLM-as-Judge evaluation pipeline.

V. DISCUSSION

A. Why Expert-Ensemble Over Single-LLM?

Single LLMs tend to anchor on the most common interpretation of user input. For food requests, they recommend the same popular restaurants regardless of sub-type diversity. Our expert decomposition forces diverse perspectives: the Food Expert explicitly balances sub-types (seafood, dim sum, street food, dessert, tea houses), while the Local Expert adds hidden gems that a single model would miss. The review loop catches cases where experts produce conflicting proposals. The optimization log (commit d323625) confirms that expert rule-ification (replacing LLM experts with fixed rules) caused regression from 4/5 to 3/5 pass rate, validating the value of LLM-based expert decomposition.

This finding is consistent with the broader multi-agent literature [12], which shows that specialized agents outperform generalist agents on complex tasks. However, our system differs from general-purpose multi-agent frameworks [11], [14] in that expert activation is determined by scene classification rather than free-form agent communication, providing more predictable and debuggable behavior.

B. The Nature of Expert “Collaboration”

A potential concern is whether our experts truly “collaborate” or merely execute in parallel and concatenate results. We address this directly: the current system implements *data-flow coordination*, not *deliberative collaboration*. Specifically:

- **Wave 1 experts** execute fully independently with no inter-expert communication. The POI Expert does not know what the Food Expert selected, and vice versa.
- **Wave 2 experts** read the merged Wave 1 proposals from shared state (data-flow dependency), but do not negotiate with Wave 1 experts.
- **Conflict resolution** happens exclusively in the review-rework loop, where a centralized reviewer identifies issues and triggers targeted re-expertise.

We also implemented two deliberative collaboration mechanisms, but both were abandoned after empirical testing showed negative results:

- 1) **Multi-round feedback** (`feedback_entry` node): Experts re-read `prev_round_context` (previous scores, route snapshot, user complaints) and re-generate proposals. Tested on 5 scenarios: average score *dropped* from 71.9 to 70.3 (−1.6), with the sightseeing scenario regressing

from 81.0 to 64.4 (−16.6 points). The mechanism caused over-correction: fixing one dimension broke others. Two architectural bugs were also discovered—core destination POIs were lost when the synthesizer lacked `must_keep` constraints, and food coverage broke when selective re-runs failed to activate the Food Expert (commit 16f2a78).

- 2) **Tournament mode** (A4): Three strategies (geo-priority, type-priority, experience-priority) run in parallel and a heuristic scorer selects the best. This *does* work (5/5 pass, 7.0 average) and is the current production mode, but it is strategy-level parallelism, not expert-level negotiation.

The broader lesson: in LLM-based planning, iterative refinement frequently causes regression because LLMs lack stable optimization targets. Each re-generation is a fresh sample from the distribution, not a gradient step toward improvement. The tournament approach avoids this by generating diverse candidates independently and selecting post-hoc, rather than iterating on a single candidate. Future work could explore learned selection mechanisms or constraint-preserving refinement that avoids the over-correction problem.

C. The “Prompt Before Architecture” Principle

Our optimization journey revealed a recurring pattern: problems that seemed to require architectural solutions (feasibility gates, dedicated refusal modules) were often solvable with prompt engineering alone. The prompt-based refusal mechanism (6 lines of text) achieved performance comparable to a hypothetical dedicated feasibility-gate component. This suggests a design principle for LLM-based systems:

- 1) **First**, try prompt engineering—it’s fast to iterate and doesn’t add system complexity.
- 2) **Second**, add algorithmic post-processing—deterministic code for physical constraints (time, distance, capacity).
- 3) **Third**, introduce architectural components only when the first two fail.

This principle aligns with recent work on prompt engineering patterns [17], which demonstrates that structured prompt instructions can replace complex control flow in many LLM applications.

D. LLM as Proposer, Algorithm as Validator

A key architectural principle emerges from our work: *LLMs should propose, algorithms should validate*. LLMs excel at semantic understanding (“what does the user want?”) but fail at physical reasoning (“can you walk 3km in 15 minutes?”). Our `_smooth_times()` function embodies this principle: the LLM generates time allocations, and deterministic code validates and corrects them.

This principle extends beyond time allocation. Our geographic continuity checks use haversine distance calculations rather than LLM judgment, and our station count caps are enforced by code rather than prompt instructions. In each case, the LLM provides the semantic layer (“which POIs are relevant?”) while the algorithm provides the physical layer (“is this route feasible?”).

The ablation analysis (Table X) quantifies this: removing `_smooth_times()` caused the speed-run scenario to score

F (42) versus A (85) with the fix, the largest single-factor improvement in our optimization history (commit 487f99a). This confirms that algorithmic validation of LLM outputs is more critical than the sophistication of the multi-agent architecture itself.

E. Evaluation Methodology Matters

A surprising finding from our optimization journey: the first two rounds of “improvement” didn’t change the route generation code at all. Round 1 fixed the evaluation scoring table (missing dimensions were filled with zeros), and Round 2 added coordinates to the evaluation prompt (the LLM was guessing distances from names). Both were evaluation bugs, not algorithm bugs.

This has implications for the broader LLM evaluation community: *evaluation methodology errors can masquerade as algorithm failures*. When a dimension shows zero, the first suspect should be the measurement instrument, not the system being measured. Our consistency analysis (Table XIX) further confirms that LLM judges require careful prompt design and scenario-aware rubrics to produce reliable scores, consistent with findings from the LLM-as-judge literature [5].

F. Limitations

- **Static POI database:** The system operates on a pre-loaded dataset of 2000+ POIs. Real-time availability (restaurant wait times, traffic conditions, temporary closures) is not integrated. Future work could incorporate real-time data APIs.
- **Limited geographic scope:** Currently optimized for Zhuhai only. Multi-city extension would require city-specific POI databases and potentially different expert weights.
- **Evaluation LLM dependency:** Scoring relies on an external LLM judge, which introduces its own biases [5]. Our two-run comparison (Table XIX) shows ≤ 1 point variation in overall scores, but absolute scores may vary across different judge models.
- **Latency:** The 54.4-second average end-to-end latency is dominated by LLM API calls. Further parallelization or model distillation could reduce this.
- **Expert-ensemble vs. classical MoE:** Our system uses rule-based expert dispatch rather than learned gating, which limits adaptability to novel scenario types. Exploring lightweight learned routers is a direction for future work.
- **No human evaluation:** All evaluations use LLM judges or algorithmic metrics. A user study comparing CityFlow routes against human-planned routes would strengthen the validity of our findings.

G. Broader Applicability

The expert-ensemble architecture with scene-aware dispatch generalizes beyond travel planning. Any domain with decomposable expertise—event planning, curriculum design, meal preparation, project management—could benefit from a similar multi-agent approach where domain experts contribute proposals that are reviewed and synthesized. The “prompt before architecture” and “LLM proposes, algorithm validates”

principles also apply broadly: in LLM-based systems, architectural complexity should be a last resort, not a first choice, and physical constraints should always be enforced by deterministic code rather than LLM instructions.

VI. CONCLUSION

We presented CityFlow, an expert-ensemble multi-agent framework for personalized urban route planning. By decomposing the planning problem across 7 specialized experts with scene-aware dispatch, parallel wave execution, and review-rework feedback loops, our system achieves 90% pass rate with an average score of 74.3/100 across 100 diverse scenarios (grade distribution: S:12, A:24, B:23, C:30, D:10, F:1), and 7.4/10 on 5 core scenarios with 5/5 pass rate. Speed-run scenarios achieve 100% pass rate (20/20), while destination scenarios remain the weakest type (68.3 avg, 5 D-grades). The ablation analysis identifies algorithmic time smoothing as the most critical component, confirming that LLM outputs require deterministic validation for physical constraints.

Three design principles emerge from our systematic optimization journey—from 65.2 to 81.2 points across 100 scenarios and seven rounds of targeted fixes, validated by the final 100-scenario evaluation (90% pass, 74.3 avg):

- 1) **Prompt before architecture:** Six lines of refusal prompt replaced a dedicated feasibility-gate component, suggesting that prompt engineering should be the first resort, not the last.
- 2) **LLM proposes, algorithm validates:** Temporal and spatial constraints require deterministic verification; LLM semantic understanding alone is insufficient for physical reasoning.
- 3) **Evaluation methodology matters:** Two of our seven optimization rounds fixed measurement errors, not algorithm bugs—evaluation methodology errors can masquerade as algorithm failures.

Future work includes: (1) integrating real-time data sources (traffic APIs, restaurant availability); (2) expanding to multi-city support with transferable expert configurations; (3) exploring lightweight learned routers to replace rule-based expert dispatch; and (4) conducting user studies to validate that CityFlow routes match human satisfaction criteria.

ACKNOWLEDGMENTS

The authors would like to thank Xinlu Zhu and Lin Fan for their valuable discussions and support during the course of this research.

REFERENCES

- [1] P. Toth and D. Vigo, *Vehicle Routing: Problems, Methods, and Applications*. Philadelphia, PA, USA: SIAM, 2014.
- [2] M. M. Solomon, “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.
- [3] J. Wei *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” in *Proc. NeurIPS*, 2022, pp. 24824–24837.
- [4] Z. Ji *et al.*, “Survey of hallucination in natural language generation,” *ACM Comput. Surv.*, vol. 55, no. 12, pp. 1–38, 2023.
- [5] J. Gu *et al.*, “A survey on LLM-as-a-Judge,” *arXiv:2411.15594*, 2024.
- [6] N. Shazeer *et al.*, “Outrageously large neural networks: The sparsely-gated mixture-of-experts layer,” in *Proc. ICLR*, 2017.

- [7] A. Q. Jiang *et al.*, “Mixtral of experts,” *arXiv:2401.04088*, 2024.
- [8] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1–39, 2022.
- [9] A. Chen, X. Ge, Z. Fu, Y. Xiao, and J. Chen, “TravelAgent: An AI assistant for personalized travel planning,” *arXiv:2409.08069*, 2024.
- [10] J. Xie *et al.*, “TravelPlanner: A benchmark for real-world planning with language agents,” in *Proc. ICML*, 2024.
- [11] Q. Wu *et al.*, “AutoGen: Enabling next-gen LLM applications via multi-agent conversation,” *arXiv:2308.08155*, 2023.
- [12] K.-T. Tran, D. Dao, M.-D. Nguyen *et al.*, “Multi-agent collaboration mechanisms: A survey of LLMs,” *arXiv:2501.06322*, 2025.
- [13] Z. Xi *et al.*, “The rise and potential of large language model based agents: A survey,” *arXiv:2309.07864*, 2023.
- [14] CrewAI Inc., “CrewAI: Framework for orchestrating role-playing autonomous AI agents,” 2024. [Online]. Available: <https://github.com/crewAIInc/crewAI>
- [15] LangChain Inc., “LangGraph: A framework for building stateful, multi-actor applications with LLMs,” 2024. [Online]. Available: <https://github.com/langchain-ai/langgraph>
- [16] H. Zhang *et al.*, “Mitigating spatial hallucination in large language models for path planning via prompt engineering,” *Sci. Rep.*, vol. 15, Art. no. 93601, 2025.
- [17] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, “A prompt pattern catalog to enhance prompt engineering with ChatGPT,” *arXiv:2302.11382*, 2023.
- [18] D. Kahneman, B. L. Fredrickson, C. A. Schreiber, and D. A. Redelmeier, “When more pain is preferred to less: Adding a better end,” *Psychol. Sci.*, vol. 4, no. 6, pp. 401–405, 1993.
- [19] P. Lewis *et al.*, “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in *Proc. NeurIPS*, 2020, pp. 9459–9474.
- [20] Y. Gao *et al.*, “Retrieval-augmented generation for large language models: A survey,” *arXiv:2312.10997*, 2024.
- [21] X. Wang *et al.*, “Self-consistency improves chain of thought reasoning in language models,” in *Proc. ICLR*, 2024.
- [22] Y. Qin *et al.*, “ToolLLM: Facilitating large language models to master 16000+ real-world APIs,” in *Proc. ICLR*, 2024.
- [23] X. Cheng *et al.*, “Beyond itinerary planning—A real-world benchmark for multi-turn and tool-using travel tasks,” *arXiv:2512.22673*, 2025.
- [24] S. Zhao, T. Zhao, H. Jin, F. Yang, and Z. Li, “A survey on point-of-interest recommendation: Models, architectures, and security,” *arXiv:2410.02191*, 2024.
- [25] S. Schulhoff *et al.*, “The prompt report: A systematic survey of prompting techniques,” *arXiv:2406.06608*, 2024.
- [26] H. Liu *et al.*, “Logical reasoning in large language models: A survey,” *arXiv:2502.09100*, 2025.
- [27] Y. Talebirad and A. Nadiri, “MultiAgentBench: Evaluating the collaboration and competition of LLM-based agents,” in *Proc. ACL*, 2025.