

Hierarchical Optimization via LLM-Guided Objective Evolution for Mobility-on-Demand Systems

Yi Zhang¹, Yushen Long², Yun Ni³, Liping Huang¹, Xiaohong Wang¹, Jun Liu⁴

¹Agency for Science, Technology and Research, Singapore

²Morgan Stanley Asia Pte.

³Onto Innovation Inc.

⁴School of computing and communications, Lancaster University, UK

1. Introduction

➤ Motivation



LOW SUPPLY, HIGH DEMAND



HIGH SUPPLY, LOW DEMAND



KEY ISSUES IN MODERN RIDE-HAILING

1. Dynamic and Uneven Supply–Demand

Passenger requests and driver availability fluctuate sharply across time and regions, causing long wait times and idle cruising.

2. Real-World Uncertainty and Volatility

Sudden demand surges, cancellations, and traffic conditions make real-time coordination challenging.

3. Operational Inefficiency and Energy Waste

A large fraction of vehicle mileage occurs without passengers, leading to congestion and emissions.

A more **adaptive, data-efficient, and intelligent decision framework** is essential to effectively capture real-world dynamics and optimize system performance under uncertainty.

1. Introduction

➤ Existing work

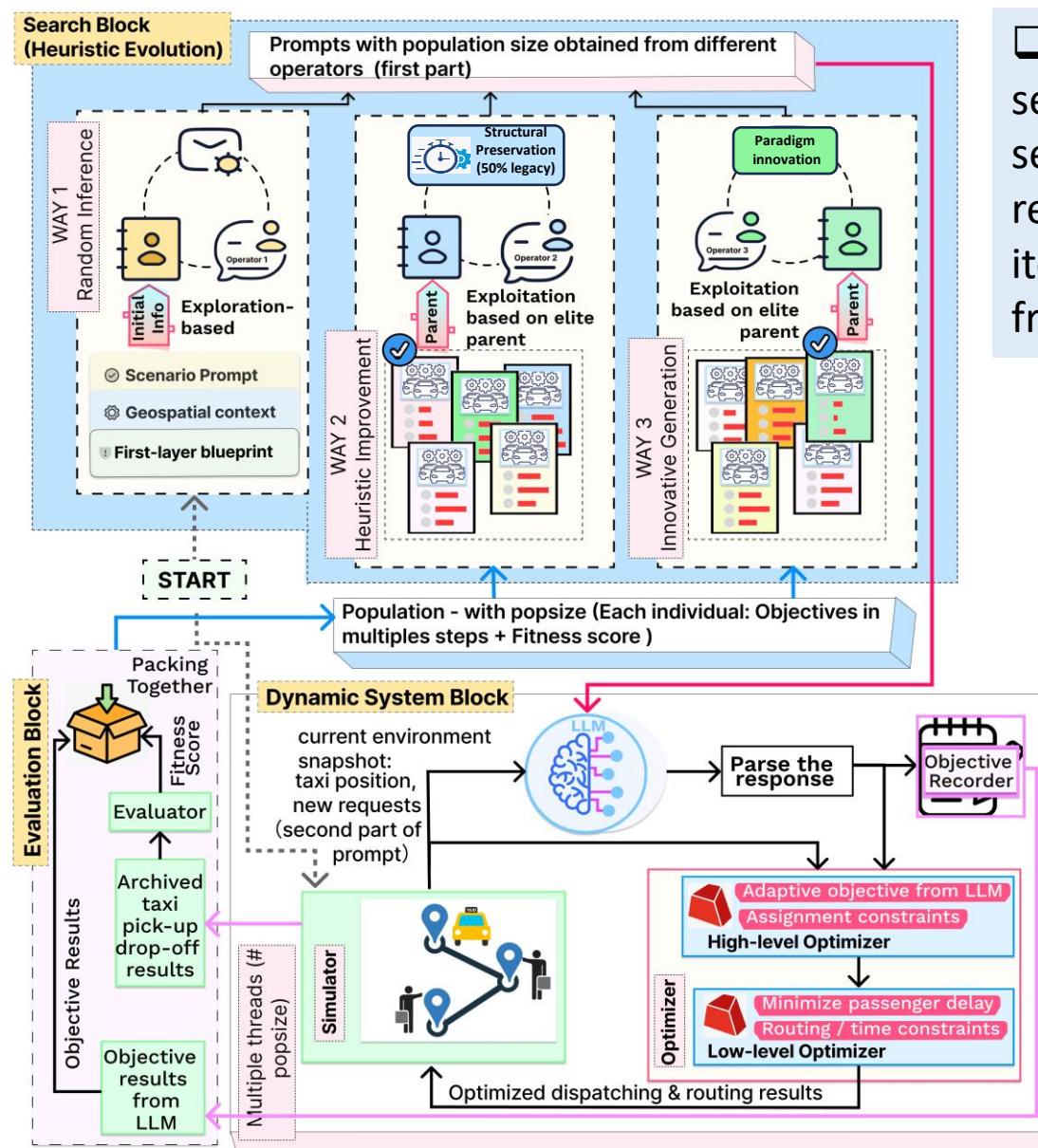
Methods	Strength/Advantages	Challenges/Limitations
Reinforcement Learning (RL)	<ul style="list-style-type: none">• Learns from historical and simulated interaction data.• Optimizes for long-term reward by considering expected future states (driver repositioning, future demand).• Deep RL enables complex policy learning and dynamic decision-making.	<ul style="list-style-type: none">• Requires large-scale interaction data for effective training.• Training is often unstable and sensitive to hyperparameters.• Hard to encode or enforce operational constraints (e.g., regulations, fairness).• Limited interpretability and generalization to new scenarios.
Online Optimization / Combinatorial Optimization	<ul style="list-style-type: none">• Provides mathematically rigorous formulation.• Enables explicit constraint enforcement.• Offers deterministic and explainable decision outcomes.	<ul style="list-style-type: none">• Scalability issues with large variable spaces (many drivers/orders).• Often decomposed into multiple stages, introducing sub-optimality.• High-level models lack awareness of low-level operational dynamics.
LLM for Operations Research (LLM-OR)	<ul style="list-style-type: none">• Automates model formulation, reducing reliance on domain experts.• Can generate heuristic or algorithmic code for optimization tasks.• Recent frameworks show promise in heuristic discovery.	<ul style="list-style-type: none">• Current studies mostly focus on static OR; lack real-time adaptability.• Generated solutions may lack feasibility guarantees or constraint compliance.• No prior integration with dynamic, online decision-making systems in ride-hailing context.

2. Proposed method

➤ Overall framework

- LLM as Meta-Objective Designer
- Optimizer as Constraint Enforcer
- Heuristics as Prompt Evolver

❑ **Evaluation block:** Computes the fitness score from simulator trajectories and pairs it with the LLM-inferred objectives to update the harmony search population.



❑ **Search block:** Uses harmony search algorithm to iteratively select and apply 3 prompt-refinement operators. Initial iterations prioritize heuristics from Operator 1.

❑ **Dynamic System block:** At each timestep, the LLM generates high-level objectives based on the refined prompt and simulator-reported states. The optimizer's decisions are executed in the simulator, updating system states. This closed-loop process continues until the simulation horizon concludes.

2. Proposed method

2.1 Dynamic Hierarchical Optimization Problem

➤ First-level assignment problem

$$\min J_{1st} \begin{cases} J_{1st}^{time} = \sum_{pv} y^{pv} |T^p - t_{S^v}| \\ J_{1st}^{dist} = \sum_{pv} y^{pv} (TR_{OpS^v} + TR_{DpS^v}) \\ J_{1st}^{util} = \sum_v \left(\sum_p y^{pv} \right)^2 \end{cases}$$

From Manual Design to LLM-Adaptation

$$\min \phi_t = LLM(\mathcal{S}_t, \mathcal{H}_{t-1})$$

$$s.t. \sum_v y^{pv} = 1$$

$s.t. \sum_v y^{pv} = 1$

➤ Second-level sequencing problem

$$\min J_{2nd} = \sum_v (DP_{(p,Op)} - T^p)$$

Initial Condition Setup

$$\sum_{i \in \mathcal{P}} x_{S,i} = 1 \quad \sum_{i \in \mathcal{P}} x_{i,E} = 1 \quad AR_S = \tilde{t}$$

Arrival Departure Constraints

$$AR_{(p,D^p)} \leq DP_{(p,D^p)}$$

$$AR_{(p',Op')} \leq DP_{(p',Op')}$$

$$DP_{(p,Op)} \geq T^p$$

System Dynamics

$$AR_{(p,D^p)} \leq DP_{(p,Op)} + TR_{(Op,D^p)} + M(1 - x_{(p,Op)(p,D^p)})$$

$$AR_{(p,D^p)} \geq DP_{(p,Op)} + TR_{(Op,D^p)} + M(1 - x_{(p,Op)(p,D^p)})$$

$$AR_{(p',Dp')} \leq DP_{(p,Op)} + TR_{(Dp,Op')} + M(1 - x_{(p,D^p)(p',Op')})$$

$$AR_{(p',Dp')} \geq DP_{(p,Op)} + TR_{(Dp,Op')} + M(1 - x_{(p,D^p)(p',Op')})$$

Conservation Law

$$\sum_{i \in \mathcal{P}} x_{ij} = \sum_{q \in \mathcal{P}} x_{jq}, \forall j \in \mathcal{D}$$

$$\sum_{j \in \mathcal{D}} x_{ji} = \sum_{q \in \mathcal{D}} x_{iq}, \forall i \in \mathcal{P}$$

2. Proposed method

2.2 LLM-Optimizer Interaction Protocol

➤ Scenario prompt setup

🕒 Scenario Prompt:

- Act as an **AI transportation optimization expert** designing the **primary objective function** for a two-level real-time mobility system

System Overview

- Simulator (executes commands) ↔ Dispatcher (calculates strategies) closed loop
- Periodic updates: Taxi states (idle/busy locations), passenger requests (pick-ups/drop-offs)

Optimization Structure

- First Level: Passenger-taxi assignment problem
- Second Level: TSP sequencing with arrival-time constraints, aiming to minimize the passenger waiting time

Your Task

- Design novel first-level objective function using:
 - Static map data
 - Taxi locations
 - Passenger request details

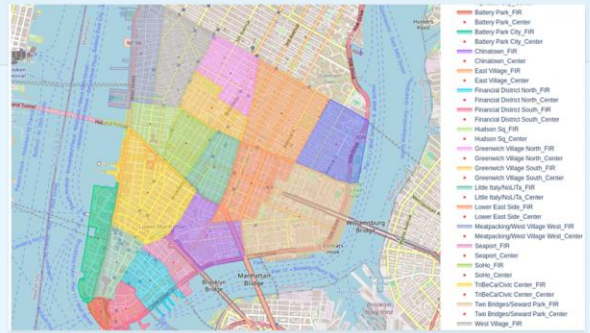
Key Requirements

- Enable timely service for all passengers
- Compatible with Gurobi-based MILP formulation
- Balance assignment efficiency with second-level sequencing feasibility

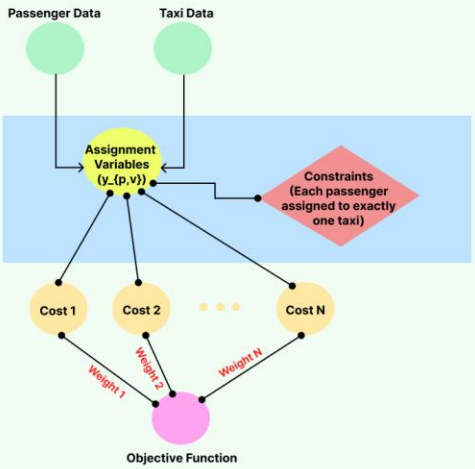
⚙️ Geospatial context

OD Matrix Explanation

- Matrix Dimensions
- Index Values (Origins)
- Column Values (Destinations)
- Cell Values: Travel time in seconds between origin (row) and destination (column)



📋 First-layer model blueprint




Structured Explanation

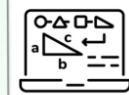
- core_components:**
 - Taxi: data class, include start_pos, arrival time, and taxi index
 - Passenger: data class, include origin, destination, arrival time, and passenger index
 - Assignment Model: Gurobi-based optimization engine - first-level model
- key_methods:**
 - setupVars: Creates binary assignment variables $y[v, p]$
 - setupCons: Ensures each passenger is assigned to exactly one taxi
 - setupObj: Configurable objective function

Scenario Prompt \mathbb{P}_{sce}

\mathbb{P}_{sce} equips the LLM with contextual knowledge regarding the problem, including:

 **System overview** provides basic system architecture

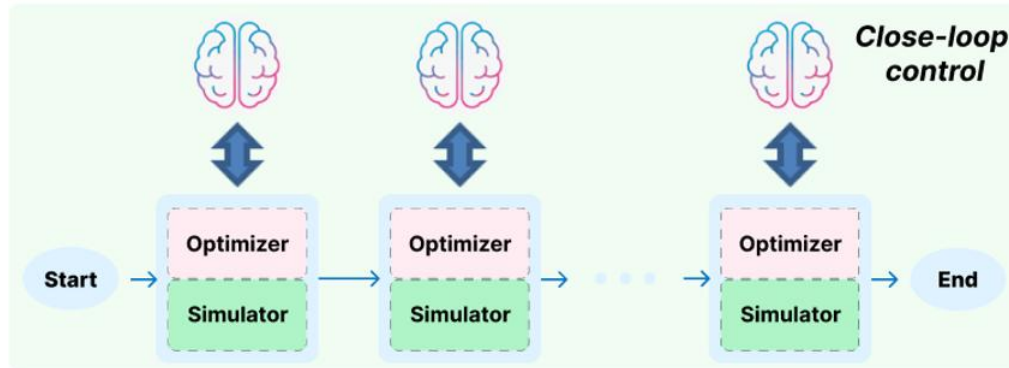
 **Geo-spatial context** encodes geospatial semantics through OD matrix

 **Mathematical model blueprint** ensures variable-aware objective formulation

2. Proposed method

2.2 LLM-Optimizer Interaction Protocol

➤ Dynamic environment feedback



- **Closed-loop control:**
 - Query made at each simulation step.
 - Incorporates **real-time environment feedback** from the simulator.
 - Allows the LLM to **update objectives dynamically** based on current states.
- **Dynamic states streaming** \mathbb{P}_{dyn}
 $P_{dyn}^t = \{veh_t, pass_t\}$

$veh_t \in \mathbb{R}^{|veh|*2}$: vehicle states (e.g., position, arrival time).
 $pass_t \in \mathbb{R}^{|pass|*3}$: passenger demand (origin, destination, request time).

➤ Solver format restriction

1. Structural requirements:

- Costs stored in list "costs (1-5 elements)"
- Weights in list weights (length matching costs)
- Final objective: $\text{sum}(w*c \text{ for } w, c \text{ in zip(weights, costs)})$

2. Gurobi expression rules:

- Use $gb.quicksum()/gb.max_()/gb.abs_()$ only when containing variables
- Use $sum()/max()/abs()$ when working with parameters
- Quadratic terms via variable multiplication
 $y[v,p] * y[v,q]$
- Never multiply Gurobi variable with Gurobi expression
 - Invalid: $y[v,p]*gb.max_(\text{expression_with_vars}, 0)$
- Never use Python *if/else* with Gurobi variables/expressions

2. Proposed method

2.2 LLM-Optimizer Interaction Protocol

➤ Evolutionary prompt optimization

We adapt the Harmony Search algorithm to evolve task-solving prompts for LLMs. The process balances exploration of new solutions with exploitation of existing ones following:

$$W_{next} = \mathbf{1}_{\{\alpha \leq HMCR\}} [\lambda W_2 + (1 - \lambda) W_3] + \mathbf{1}_{\{\alpha > HMCR\}} W_1$$

where $\alpha \sim U(0, 1), \lambda \sim \text{Bernoulli}(\text{PAR})$.

Fitness Evaluation: Prompts are evaluated in a simulation environment \mathcal{E} over a trajectory τ . The fitness of a prompt X is:

$$f(X) = \mathbb{E}_{\tau \sim \mathcal{D}} [\mathcal{R}(\mathcal{E}(X, \tau))]$$

where \mathcal{R} is the reward/cost function. The algorithm selects elite parents based on fitness to drive the next generation.

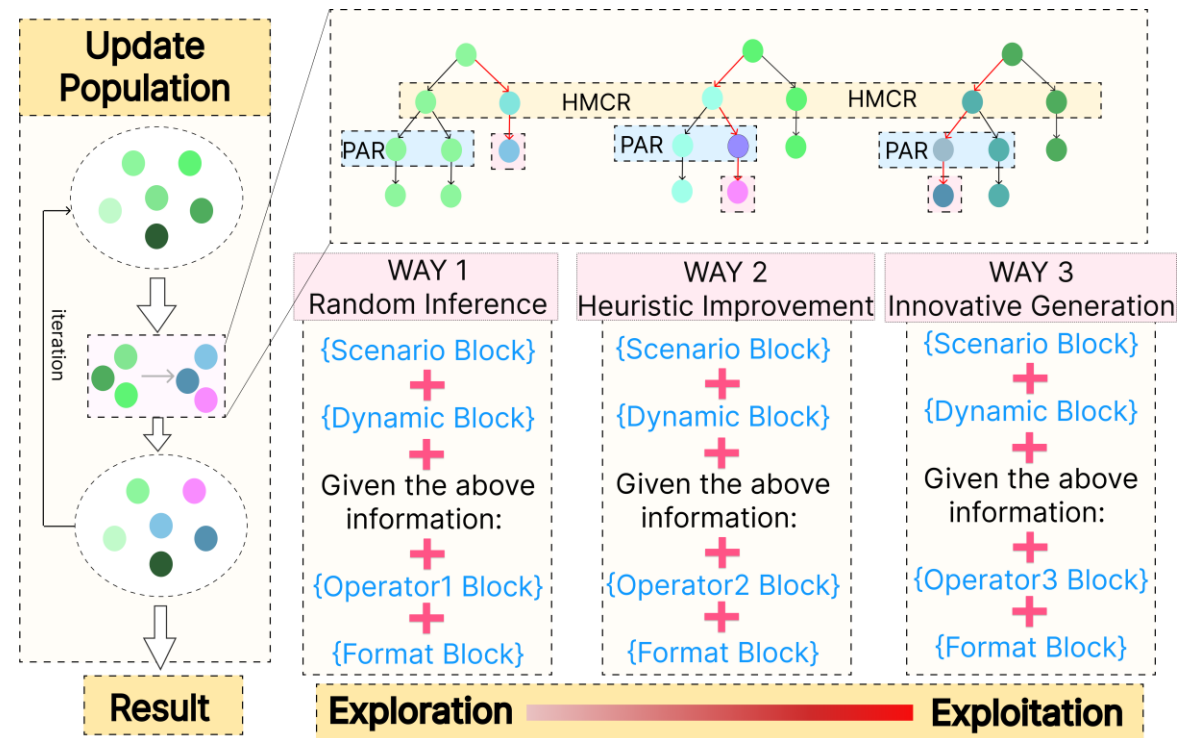


Table 1: Three operators for LLM-driven objective generation. Equations and example prompts illustrate the mechanisms of each operator.

Random Inference (W_1)	Heuristic Improvement (W_2)	Innovative Generation (W_3)
If exploration is chosen, the LLM generates a new objective without historical knowledge. $X_{new} = \text{LLM}(\emptyset, \mathcal{P}_{scc})$	The LLM generates an enhanced prompt by refining heuristics derived from the parent prompt. $X_{new} = \text{LLM}(X_{parent}, \mathcal{P}_{scc}, \mathcal{I}_1)$	The LLM formulates a novel prompt, drawing inspiration from the parent prompt. $X_{new} = \text{LLM}(X_{parent}, \mathcal{P}_{scc}, \mathcal{I}_2)$
Example prompt: "Please generate a new objective for first-level assignment model."	Example prompt: "Develop an improved objective function by [instruction block1]."	Example prompt: "Reinvent the objective function from the previous run by [instruction block2]."

3. Experiment

➤ Ablation Study

Prompt Composition	P50_C30_T300	P70_C60_T600	P100_C80_T900	P130_C80_T1200
$\mathcal{P}_{\text{sys}} \cup \mathcal{P}_{\text{geo}} \cup \mathcal{P}_{\text{data}}$	9.93 ± 0.00	5.86 ± 0.00	5.29 ± 0.00	6.46 ± 0.00
$\mathcal{P}_{\text{sys}} \cup \mathcal{P}_{\text{geo}} \cup \mathcal{P}_{\text{model}}$	8.72 ± 0.91	2.86 ± 0.29	3.07 ± 0.89	4.54 ± 0.36
$+\mathcal{P}_{\text{restriction}}$	8.16 ± 1.48	4.12 ± 0.83	2.59 ± 0.52	2.25 ± 0.05

Note: Mean \pm standard deviation across runs. Bold: best per scenario (P=Passengers, C=Taxis, T=Time(s)). Prompt variants: - $\mathcal{P}_{\text{sys}} \cup \mathcal{P}_{\text{geo}} \cup \mathcal{P}_{\text{data}}$: System+Geo+Data structure (Sec. 3.2.1). - $\mathcal{P}_{\text{sys}} \cup \mathcal{P}_{\text{geo}} \cup \mathcal{P}_{\text{model}}$: System+Geo+Model structure (Sec. 3.2.1). - $+\mathcal{P}_{\text{restriction}}$ denotes $\mathcal{P}_{\text{sys}} \cup \mathcal{P}_{\text{geo}} \cup \mathcal{P}_{\text{model}} \cup \mathcal{P}_{\text{restriction}}$: System+Geo+Model+Gurobi-compatible constraints (App. A.9.3). $\mathcal{P}_{\text{data}}$ defines variable formats, $\mathcal{P}_{\text{model}}$ specifies full MILP structure.

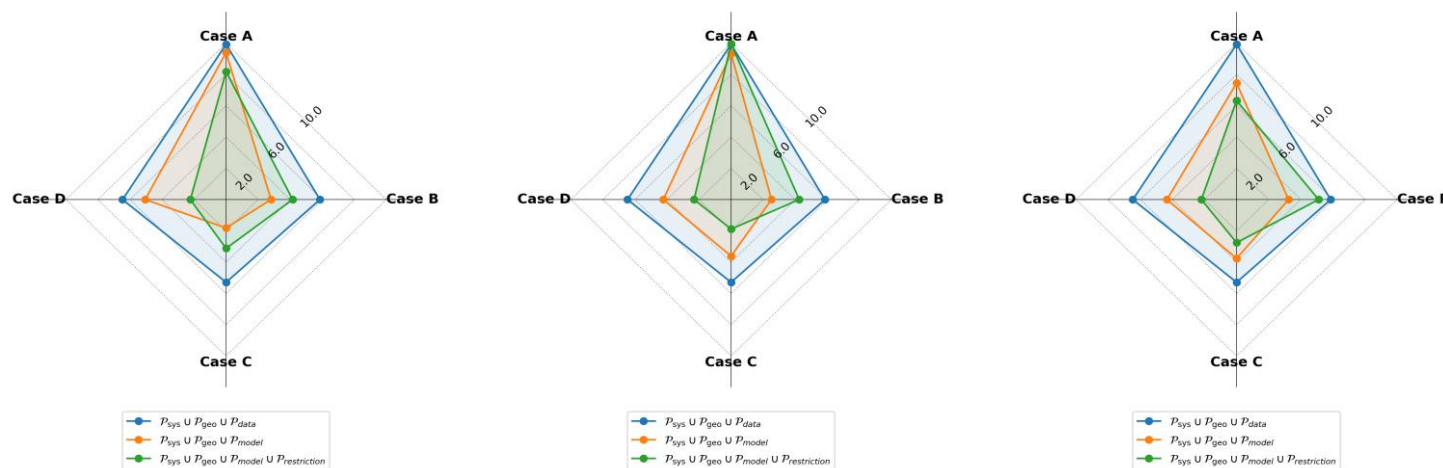


Fig. Average passenger waiting time (minutes) under different compositions and runs. Cases: A(P50_C30_T300), B (P70_C60_T600), C (P100_C80_T900), and D (P130_C80_T1200).

Effect of Structured Prompt Design on Performance

•Basic prompt:

- High passenger waiting time ($\approx 9.93 \pm 0.00$ min).
- No variation \rightarrow caused by invalid LLM objectives replaced by default static ones.

•With model blueprint:

- Waiting time drops significantly (e.g., 2.86 ± 0.29 min).
- Slightly higher variability due to diverse LLM-generated functions.

•With additional constraint block:

- Achieves lowest cost (e.g., 2.25 ± 0.05 min).
- Crucial for large, complex scenarios — reduces error rates and stabilizes outputs.

•Key takeaway:

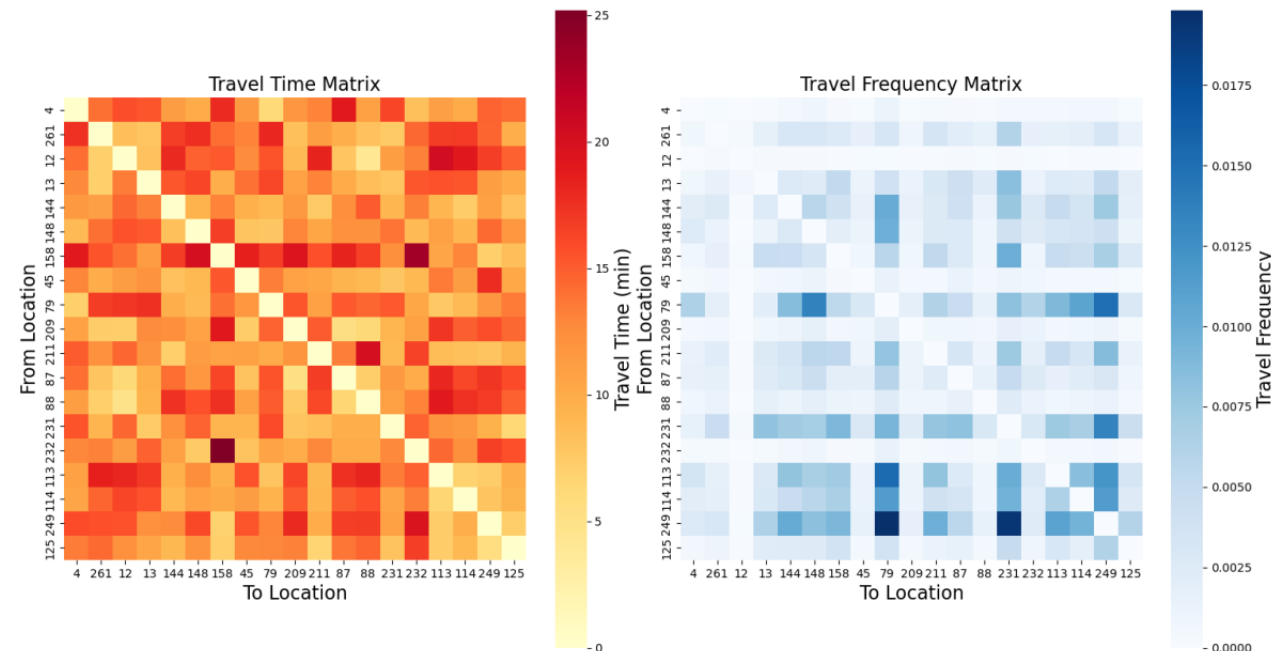
Structured task grounding (**model definitions**) + constraint formalization (**optimization restrictions**) \rightarrow more reliable and efficient LLM-driven optimization.

3. Experiment

➤ Performance comparison – Manhattan downtown



Figure 5: Zone-based Manhattan downtown map



- **Data source:** NYC Taxi & Limousine Commission trip data (Jan–Jun 2024).
- **Focus area:** Downtown Manhattan (trips originating and ending within the area)
- **Scenario generation:** Derived from **statistical patterns** (not direct historical replay). Synthetic travel tasks sampled from **OD freq. distribution**. Preserves **realistic spatial and temporal demand patterns**.
- **Spatial pattern:** **More balanced** across downtown regions, without strong OD concentration.
- **Scenario configuration:**
 - 9 synthetic scenarios varying:
 - Passenger volume (P): 35–200
 - Taxi fleet size (C): 60–100
 - Request window (T): 600 s / 900 s / 1200 s

3. Experiment

➤ Performance comparison – Manhattan downtown

Table 2: Average passenger waiting time (minutes) across optimization methods and scenarios on New York taxi dataset

Category	Method	Scenario								
		1	2	3	4	5	6	7	8	9
Manual Objectives	Distance [†] [2]	6.51	14.78	14.88	11.42	24.67	22.32	18.09	34.98	28.04
	Distance × Utilization [*]	5.02	7.38	7.94	6.96	9.07	9.89	7.06	9.24	11.59
	Temporal [‡] × Utilization [*] [24]	10.09	8.81	9.93	9.74	11.48	15.23	10.97	13.26	18.99
	Distance × Temporal × Utilization	3.32	4.99	5.77	4.37	6.06	6.72	5.19	7.07	9.27
RL Methods	Default [*] + RL-Seq [◊] [23]	3.10	4.37	4.90	4.92	6.63	6.59	7.48	7.51	10.35
	Full RL [6]	1.27	2.35	3.98	2.42	3.80	4.83	3.20	4.78	6.82
LLM Methods	FunSearch [15]	0.77	1.74	1.55	5.29	2.95	5.43	5.78	5.79	10.27
	EoH [16]	1.64	1.58	2.79	3.04	4.68	5.71	3.87	5.55	8.98
Hybrid (LLM+Optimizer)	Ours	1.55	1.37	2.50	1.89	2.59	4.14	3.10	2.25	4.01

Distance: Travel time between vehicle start position and passenger pickup&dropoff location.

Temporal: Gap between the vehicle non-idle time and passenger request time.

Utilization: Taxi service efficiency (vehicles/request).

Default: Use default objective (Distance × Temporal × Utilization) in first-level assignment optimization.

RL-Seq: Reinforcement learning method is adopted to solve second-level sequencing problem.

• Hybrid LLM + optimizer approach

- Integrates LLM's adaptability with optimizer's precision.
- Closed-loop feedback enables dynamic objective refinement.
- Achieves state-of-the-art performance without large-scale training data.

• Manual objective functions

- Combining distance, time, and utilization helps small-scale cases.
- Performance deteriorates as problem size grows (e.g., 9.27 min in Scenario 9).

• RL-based and LLM-only methods

- RL performs well in simple cases but suffers from reward sparsity and poor scalability.
- FunSearch and EoH both lack adaptive feedback and low-level optimization, leading to suboptimal large-scale results.

3. Experiment

➤ Performance comparison – Manhattan downtown

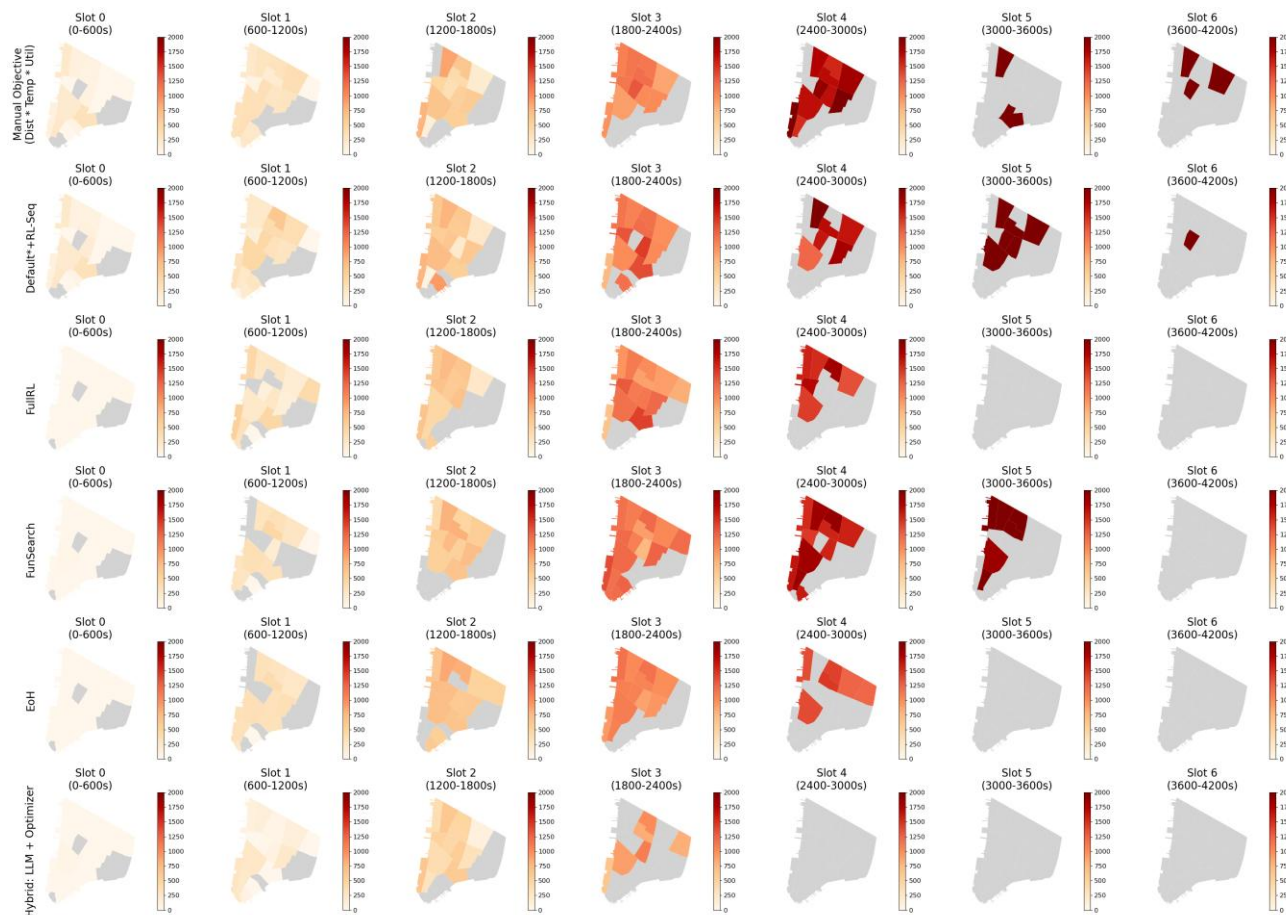


Figure 7: Spatial-temporal distribution of passenger waiting times across optimization methods in Scenario 9 (P200_C100_T1200) under New York taxi dataset. Each row represents a method: (1) Manual composite objective (Distance \times Temporal \times Utilization), (2) Default + RL-Seq, (3) Full RL, (4) FunSearch, (5) EoH, (6) Our LLM-Optimizer approach. Columns show 600-second time windows. Heatmap colors show average waiting times per zone.

Spatiotemporal analysis on Case 9

Spatiotemporal delays:

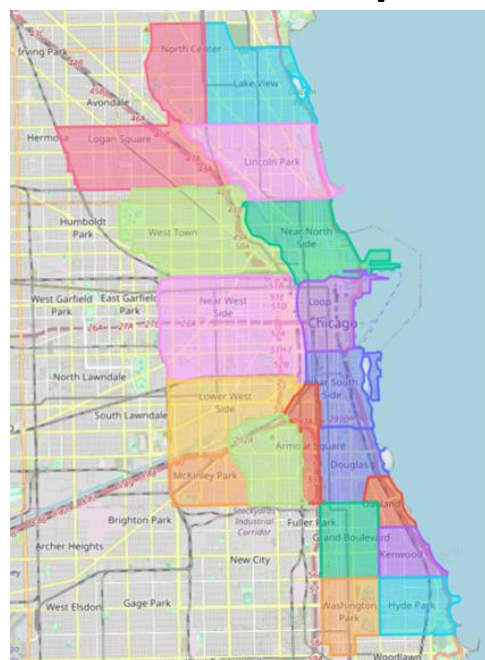
- Manual objectives show persistent hotspots (Zones 144, 79, 249) due to static design.
- Default+RL-seq method degrade over time (e.g., high-delay regions in Slot 5) and struggle with unseen demand patterns.

Framework advantage:

- LLM+optimizer completes all pickups within 4 time slots (vs. 5–7 for baselines).
- Maintains spatially consistent low delays, overcoming RL data dependency and rigidity of manual objectives.

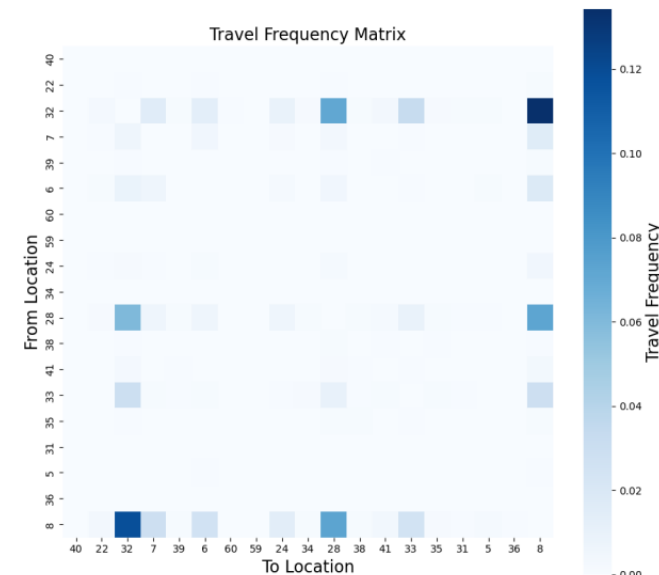
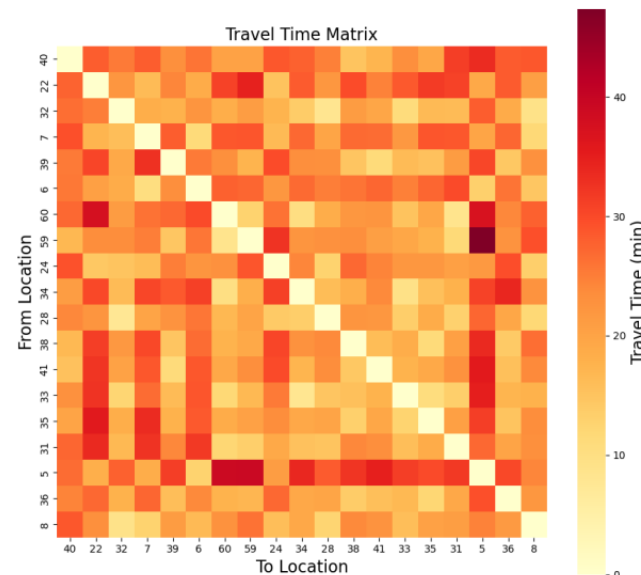
3. Experiment

➤ Performance comparison – Chicago central



Zones

- DOUGLAS(zone 35)
- OAKLAND(zone 36)
- GRAND BOULEVARD(zone 38)
- KENWOOD(zone 39)
- WASHINGTON PARK(zone 40)
- HYDE PARK(zone 41)
- LOGAN SQUARE(zone 22)
- WEST TOWN(zone 24)
- NEAR WEST SIDE(zone 28)
- LOWER WEST SIDE(zone 31)
- NEAR SOUTH SIDE(zone 33)
- ARMOUR SQUARE(zone 34)
- NEAR NORTH SIDE(zone 8)
- LOOP(zone 32)
- MCKINLEY PARK(zone 59)
- LAKE VIEW(zone 6)
- NORTH CENTER(zone 5)
- BRIDGEPORT(zone 60)
- LINCOLN PARK(zone 7)



- **Data source:** Chicago Data Portal taxi trips (Jan–Jun 2023).
- **Focus area:** Central district — Loop (zone 32) + 18 neighboring communities.
- **Spatial pattern:**
 - **Highly concentrated** OD pairs, especially (32, 8) and (32, 28).
 - Reflects strong commuter flows between CBD and nearby residential zones.
- **Scenario configuration:**
 - 9 synthetic scenarios varying:
 - Passenger volume (P): 35–200
 - Taxi fleet size (C): 60–100
 - Request window (T): 600 s / 900 s / 1200 s
 - Example: **P200_C100_T1200** → **200 requests, 100 taxis, 20 min window**

3. Experiment

➤ Performance comparison – Chicago central

Table 3: Average passenger waiting time (minutes) across optimization methods and scenarios on Chicago taxi dataset

Category	Method	Scenario								
		1	2	3	4	5	6	7	8	9
Manual Objectives	Distance [†] [2]	17.52	35.87	47.85	41.08	80.96	121.41	49.99	145.14	102.44
	Distance × Utilization [*]	11.60	15.34	18.35	15.70	18.50	22.77	19.07	22.74	30.38
	Temporal [‡] × Utilization [*] [24]	17.07	18.15	16.90	17.33	19.74	23.96	18.51	22.09	26.96
	Distance × Temporal × Utilization	9.54	13.79	18.19	11.81	16.71	20.74	17.21	20.68	25.01
RL Methods	Default [*] + RL-Seq [◇] [23]	10.94	10.87	15.37	19.27	18.68	21.57	24.02	24.86	24.63
	Full RL [6]	10.83	13.43	15.40	12.05	14.43	16.50	14.35	15.93	20.03
LLM Methods	FunSearch [15]	9.03	10.70	10.87	21.70	12.82	15.35	18.29	17.42	21.74
	EoH [16]	10.43	10.83	13.05	13.53	14.45	17.16	15.01	16.93	19.79
Hybrid (LLM+Optimizer)	Ours	8.65	9.58	11.30	8.40	12.32	14.96	10.79	14.51	15.37

Distance: Travel time between vehicle start position and passenger pickup&dropoff location.

Temporal: Gap between the vehicle non-idle time and passenger request time.

Utilization: Taxi service efficiency (vehicles/request).

Default: use default objective (Distance × Temporal × Utilization) in first-level assignment optimization.

RL-Seq: reinforcement learning method is adopted to solve second-level sequencing problem.

• Hybrid LLM + optimizer approach

- Robust across all scenarios, consistently reducing delays by >20% in key cases
- Closed-loop adaptation ensures strong performance under dynamic, high-demand conditions
- Example gains: Scenario 4 (8.40 vs 12.05 min), Scenario 7 (10.79 vs 14.35 min), Scenario 9 (15.37 vs 19.79 min)

• Manual objective functions

- Composite objectives (Distance × Temporal × Utilization) degrade at scale
- Single-objective variants fail under imbalanced demand → extreme delays (up to 145 min)

• RL-based & LLM-only methods

- Full RL better than manual baselines but still inferior to hybrid approach
- LLM-only methods (FunSearch, EoH) perform inconsistently: small-scale vs large-scale
- Lack of dynamic feedback and fine-grained optimization reduces solution quality

3. Experiment

➤ Performance comparison – Chicago central

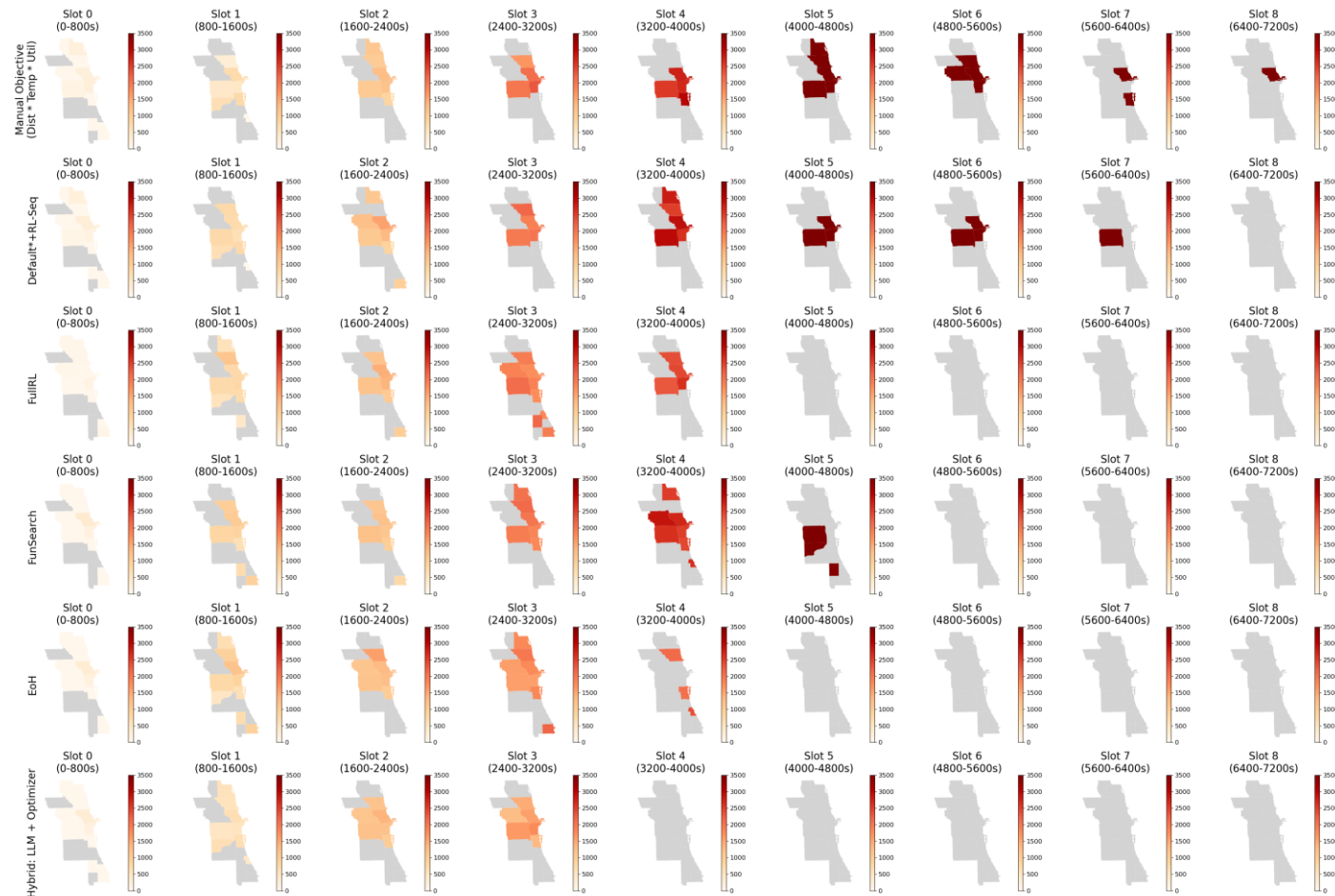


Figure 8: Spatial-temporal distribution of passenger waiting times across optimization methods in Scenario 9 (P200_C100_T1200) under Chicago taxi dataset. Each row represents a method: (1) Manual composite objective (Distance \times Temporal \times Utilization), (2) Default + RL-Seq, (3) Full RL, (4) FunSearch, (5) EoH, (6) Our LLM-Optimizer approach. Columns show 800-second time windows. Heatmap colors show average waiting times per zone.

Spatiotemporal analysis on Case 9

Spatiotemporal delays:

- Baselines require 8–9 time slots due to longer travel distances and highly imbalanced trips.
- RL and LLM-only methods improve (5–6 time slots) but still leave delays.

Framework advantage:

- Proposed method completes all pickups in 4 time slots.
- Demonstrates robustness to complex urban topologies and heterogeneous demand through LLM + optimizer integration.

4. Conclusion

➤ Contribution

- ✓ We propose a hybrid approach that combines LLM and optimizer in a dynamic hierarchical system, where LLM-generated objective in the high-level model serves as guiding heuristics for the low-level routing optimizer
- ✓ We introduce a harmony search algorithm with three novel operators (random inference, heuristic improvement, and innovative generation) to iteratively refine LLM-generated objectives using feedback from the mathematical solver.
- ✓ Extensive experiments on scenarios derived from the Manhattan New York and Central Chicago taxi datasets validate the effectiveness of our approach, demonstrating an average of 16% improvement over state-of-the-art baselines

➤ Future work

- ✓ Application layer scalability and simulation fidelity
- ✓ Reinforcement learning for LLM-Optimizer co-adaptation
- ✓ Quantum-enhanced hierarchical optimization



THANK YOU

www.a-star.edu.sg

