Leveraging Large Language Models for Solving Rare MIP Challenges

Anonymous ACL submission

Abstract

Mixed Integer Programming (MIP) has been extensively applied to areas requiring mathematical solvers to address complex instances within tight time constraints. However, as the problem scale increases, the complexity of model formulation and finding feasible solutions escalates significantly. Beneficial from outstanding text generation capacity of Large Language Models (LLMs), building and solving industrial-level instances becomes insensitive to problem scale. While LLMs, like GPT-4, can handle some traditional medium-scale MIP problems, they struggle with uncommon or highly specialized MIP scenarios. Fine-tuning LLMs can yield some feasible solutions for medium-scale MIP instances, but these models typically fail to explore diverse solutions when constrained by a low and constant temperature. In this paper, we propose and evaluate a recursively dynamic temperature method integrated with a chain-of-thought approach to exploit a large feasible region. Our findings show that starting with a high temperature and gradually lowering it leads to better feasible solutions compared to other dynamic temperature strategies. Additionally, by comparing results generated by the LLM with those from Gurobi, we demonstrate that the LLM can produce solutions that complement traditional solvers by accelerating the pruning process and improving overall efficiency.

1 Introduction

Mixed Integer Programming (MIP) is a fundamental tool in many optimization domains, such as the Traveling Salesman Problem (TSP) (Laporte, 1992) and facility location planning (Klose and Drexl, 2005). MIP also plays a particularly critical role in time-sensitive applications like transportation and network scheduling (He et al., 2018), where finding a feasible solution within a short time frame is essential to maintaining system operability and avoiding downtime.

The traditional approach to solve MIP problems is the branch-and-bound (B&B) algorithm(Lawler and Wood, 1966). While this method guarantees to find the optimal solution for a given instance, the efficiency of mathematical solvers that use such method like Gurobi(Achterberg, 2019) diminishes as the problem scale increases (Jablonskỳ et al., 2015). Moreover, the complexity of model formulation grows significantly with the dimensionality of the problem. For example, the growth rate of the complexity of a 3D bin-packing problem (Martello et al., 2000) is considerably higher than that of a 2D bin-packing problem (Johnson, 1974).

To expedite the search for optimal solutions, mathematical solvers implement various techniques such as heuristics, cutting planes, parallelism, presolve(Gomory, 2010). However, despite these advanced methods, solvers still face challenges in efficiently handling large-scale MIP problems within tight time constraints. Large language models (LLMs), with their strong pattern recognition capabilities, can achieve similar objectives with only minimal data and modeling information. For instance, Yang et al. (Yang et al., 2023) pioneered the application of Chain-of-Thought (CoT) reasoning (Wei et al., 2022) in large language models such as GPT-3.5 (Brown, 2020) and GPT-4 (Achiam et al., 2023) to address problems like the TSP using only the coordinates of cities, without explicitly requiring distances between each pair of cities. This approach reduces the time complexity from $O(n^2)$, typically required for distance calculations in traditional mathematical solvers, offering a more efficient solution.

However, the previous work by Yang et al. (Yang et al., 2023) has several drawbacks. First, the instance data is generated from randomly sampled integers, which may reduce its validity as a demonstration of the LLMs capabilities in real-world MIP applications. Second, the TSP is a well-known and extensively studied problem, meaning LLMs

have been trained on similar data and the same TSP model numerous times. In our experiments, we observed that LLMs often struggle with complicated mathematic models and frequently fail to grasp the MIP modeling process. These factors raise concerns about the robustness of LLMs in real-world applications.

Our work focuses on how to integrate LLMs into real-world applications. To demonstrate the generalizability of LLMs in real-world scenarios, we developed a fine-grained simulator and utilized the operational dataset provided by DiDi in November 2016(Yao et al., 2018) to simulate the passenger-driver matching process in the ride-pooling market using MIP. Our work is divided into three main components: (1) We construct a carpooling MIP model based on real-world data while capturing and storing vehicle locations, order locations, MIP instances, and intermediate feasible solution statuses for future training purposes. (2) Leveraging the pattern recognition capabilities of LLMs and CoT reasoning, we generate prompts using only abstract information from the carpooling dataset, bypassing the need to compute MIP parameters, like the distance between the vehicle and the user's order, explicitly. We then perform supervised fine-tuning on LLaMA 3.1 (8B) (Dubey et al., 2024) to discover better feasible solutions, comparing these with the top three feasible solutions generated by traditional mathematical solvers. The results from LLM can be used to accelerate the pruning process in conventional mathematical solvers. (3) We employ recursive dynamic temperature adjustments to refine the quality of feasible solutions generated by the LLM. Through a strategy of starting at a higher temperature and gradually reducing it, we observe significant improvements in solution quality. By systematically evaluating performance under various temperature schedules, we identify the highly effective strategy for enhancing the effectiveness and consistency of the solutions produced.

2 Related Work

MIP plays a crucial role in combinatorial optimization, with applications in planning(Klose and Drexl, 2005), scheduling(Xiong et al., 2022), and routing(Braekers et al., 2016). Traditional methods like B&B (Lawler and Wood, 1966) have been widely used to solve MIP problems. However, these methods can be computationally intensive, leading to growing interest in enhancing MIP solvers with machine learning (ML) and LLMs.

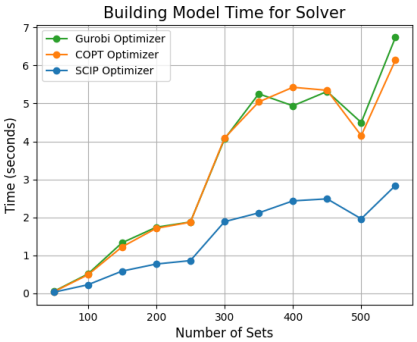

Figure 1: The time cost of building the model increases significantly as the problem scale grows. This trend illustrates the growing computational complexity associated with larger problem instances.

Recent works integrating ML with MIP can be categorized into two main approaches (Zhang et al., 2023): exact algorithms and heuristic algorithms. For exact methods like B&B, ML models have been used to optimize branching variable selection and node selection, significantly improving solution efficiency (Gasse et al., 2019) (Khalil et al., 2016). On the heuristic side, techniques like Large Neighborhood Search and Feasibility Pump have benefited from ML integration, leading to higher solution quality and computational efficiency (Song et al., 2020) (Qi et al., 2021). Additionally, Graph Neural Networks have been leveraged to represent MIP instances, enhancing decision-making processes like branching and node selection (Gasse et al., 2019). Reinforcement learning is also increasingly applied in both exact and heuristic methods to support adaptive decision-making within the B&B framework (Tang et al., 2020).

With the advent of LLMs and the rise of AI agents, more research has focused on translating natural language into operations research problems (Xiao et al., 2023; AhmadiTeshnizi et al., 2023; Wang et al., 2024). Although zero-shot learning typically performs poorly on complex problems, LLMs have significant potential, and their performance can be improved through techniques like the chain of thought (Wei et al., 2022), tree of thought (Yao et al., 2023), and self-consistency (Wang et al., 2022). Yang et al.'s work (Yang et al., 2023) utilizes models like PaLM (Chowdhery et al., 2023) and GPT-4 (Achiam et al., 2023) to tackle linear regression and the TSP with CoT reasoning, demonstrating success on small-scale problems. Our work fine-tunes the LLaMA 3.1 (8B) model (Dubey et al.,

2024) using both model information and real MIP instance data capable of generating feasible solutions, and proposes an adaptive temperature strategy that iteratively enhances LLM performance, leading to the generation of even more optimized feasible solutions.

3 Method

Given that LLMs have been trained on numerous traditional MIP problems, such as the TSP (Laporte, 1992), and considering the limitations in the generalizability of previous work due to the use of non-real-world data, we aim to assess the potential of LLMs in MIP under real-world conditions. To achieve this, we construct a carpooling MIP model and develop a simulator to replicate the vehicle dispatching process. The DiDi operational dataset (Yao et al., 2018), which consists of the location and time of orders from November 2016, serves as the foundation for generating real-world data in this study. The input is a long text containing information about the locations of orders and the positions of various categories of vehicles, while the output is feasible solutions for dispatching these vehicles to different users.

3.1 Problem Statement

There are two types of vehicles: (1) empty vehicles and (2) vehicles with one passenger. Additionally, we assume that each order is associated with a single customer and that a vehicle can accommodate at most two passengers at a time. Our goal is to minimize the total distance traveled for picking up customers, subject to the above constraints. The detailed method for calculating the distance in various scenarios is provided in Appendix 6.2.

The notation is as follows:

- x_{ij} is a decision variable indicating whether empty car i is assigned to user j.
- y_{ijk} is a decision variable indicating whether empty car i is assigned to pick up user j and then user k.
- z_{ij} is a decision variable indicating whether car i with one passenger willing to share is assigned to user j.
- d_{ij} is the distance between vehicle i and user j.
- d'_{jk} is the distance between user j and user k.
- d''_{ij} is the distance between vehicle i (with one order) and user j.
- m is the number of empty vehicles.
- n is the number of vehicles with one order.
- p is the number of orders.

1. Objective Function

The objective is to minimize the total distance between vehicles and passengers across all segments of the vehicle paths:

$$\min \sum_{i=1}^{n_1}\sum_{j=1}^{m} x_{ij}\cdot d_{ij} + \sum_{i=1}^{n_1}\sum_{j=1}^{m}\sum_{k=1,k\neq j}^{m} y_{ijk}\cdot(d_{ij}+d'_{jk}) + \sum_{i=1}^{n_2}\sum_{j=1}^{m} z_{ij}\cdot d''_{ij}$$

2. Constraints

Order Coverage: Each user is assigned to exactly one vehicle:

$$\sum_{i} x_{ij} + \sum_{i}\sum_{k,j\neq k}(y_{ijk}+y_{ikj}) + \sum_{i'} z_{i',j} = 1,\ \forall\, j$$

Vehicle Capacity for Empty Vehicles: Each empty vehicle picks up at most two people:

$$\sum_{j} x_{ij} + \sum_{j}\sum_{k\neq j} y_{ijk} \leq 1,\ \forall\, i$$

Vehicle Capacity for Shared Rides: Each vehicle with one passenger picks up at most one more person:

$$\sum_{j} z_{ij} \leq 1,\ \forall\, i$$

Binary Variable Constraints:

$$x_{ij}, y_{ijk}, z_{i,j} \in \{0,1\},\ \forall\, i,j,k$$

3. Analysis of the Constraint Matrix

As discussed in the previous constraints part, Figure 1 illustrates the relationship between the model-building time and the number of sets. The shape of the constraint matrix is given by:

$$(m+n+p,\ m\cdot p + m\cdot p^2 + n\cdot p)$$

The complexity of the constraint matrix increases significantly as the size of the set grows, resulting in a very high level of computational complexity. This makes it infeasible to solve the problem instantly if the size of the set increases significantly.

3.2 Data Collection and Prompt Generation

Prior to fine-tuning the LLMs, we first identify the positions of the orders and the various types of vehicles. The next step is to generate the labels, which consist of the intermediate feasible solution and the optimal solution for this MIP instance.

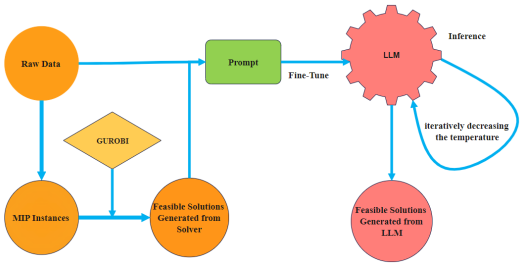

Figure 2: Workflow of training and inference, showing the recursive approach where the temperature starts high to explore a broad solution space and gradually decreases to refine and improve solution quality.

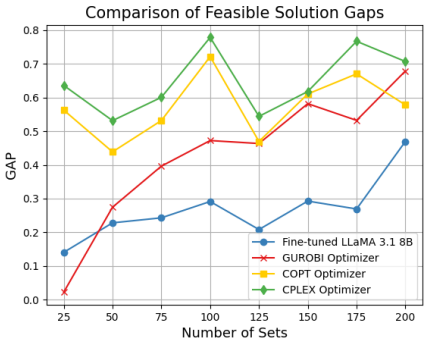

Figure 3: The gap, defined as the difference between the feasible and optimal solution, shows that a larger gap means worse performance. After fine-tuning, LLaMA 3.1 8B generates feasible solutions with a smaller gap than solver's first three solutions, especially as MIP instance scale grows.

3.2.1 Data Storage

When the simulator constructs and solves the MIP instances using the Gurobi (Achterberg, 2019) solver, we capture several critical pieces of information that include (1) the location of every vehicle and order, (2) the status of intermediate feasible solutions, and (3) the optimal solution.

3.2.2 Prompt Generation

Following the collection of raw and intermediate data, we compile the LaTeX code for the mathematical model-building process, the details of each vehicle and order scenario, and both feasible and optimal solutions into a text format, as presented in Appendix 6.1. This information is then used to generate the desired prompt for subsequent training and inference processes.

3.3 Recursive CoT with dynamic temperature

Since LLMs, after fine-tuning, exhibit a strong ability to grasp problem patterns, they often produce the same feasible solution at lower temperatures—even when the prompt suggests that this solution isn't optimal. Despite recursive adjustments from lower temperatures to higher temperatures intended to explore a broader solution space, LLMs may still become trapped in the previous bad solutions. Thus, a temperature strategy is needed to effectively explore diverse possibilities and enhance solution quality.

To address this, we employ a recursive approach with dynamic temperature to leverage CoT effectively. Our strategy involves initially generating feasible solutions with a high temperature to explore a broader solution space. We then iteratively refine these solutions by gradually lowering the temperature. This dynamic temperature adjustment begins with a high temperature to facilitate exploration and progressively decreases to a low temperature to focus on refinement. This balanced approach helps us improve solution diversity and quality, ultimately leading to better feasible solutions. Figure 2 illustrates the entire workflow of training and inference.

4 Experiment

Using LLMs to retrieve the exact optimal solution for medium and large-scale MIP instances is currently impractical. However, due to their strong pattern recognition capabilities, fine-tuned LLMs can provide satisfactory feasible solutions that serve as upper bounds for minimization problems.

We aim to explore the potential of LLMs in this context. If LLMs can provide satisfactory feasible solutions during the model-building process, or if traditional solvers like Gurobi face challenges in finding feasible solutions for large-scale instances, these LLM-generated solutions could serve as valuable upper bounds to accelerate the pruning process. By comparing the solutions generated by LLMs with the top three feasible solutions produced by traditional mathematical solvers, we can potentially leverage LLM-generated solutions to enhance pruning strategies and improve overall solver efficiency.

4.1 Comparision with Mathematical Solvers

We fine-tune the LLaMA-3.1-8B model for the following experiment and split 10% of the total instances in the generated dataset, which contains 12,500 MIP instances, as our test dataset.

We compare the best feasible solution obtained from the LLM using three recursive calls, where the temperature gradually decreases from 1 to 0.1

to 0.01, against the first three feasible solutions generated by GUROBI (Achterberg, 2019), CPLEX (Manual, 1987), and COPT (Ge et al., 2022) to evaluate which approach achieves a smaller gap. The gap is calculated using the formula:

$$\text{gap} = \frac{\text{current objective value} - \text{optimal value}}{\text{current objective value}}$$

Figure 3 illustrates the relationship between the scale of the MIP instance and the gap observed from LLMs, GUROBI, CPLEX, and COPT. The fine-tuned LLaMA-3.1-8B model is able to generate feasible solutions that are much closer to the optimal solution compared to the first three feasible solutions generated by traditional mathematical solvers.

4.2 Ablation

To demonstrate the effectiveness of the temperature adjustment strategy from higher to lower in enhancing the performance of our fine-tuned LLaMA-3.1-8B model, we conduct a comparison across several scenarios. These include cases (1) where the temperature initially rises from 0.01 to 1 and then falls back to 0.01 recursively, (2) where the temperature remains constant at 0.01 during recursive calls, (3) where a single temperature setting of 0.01 is used, and (4) where the temperature progressively rises from 0.01 to 0.1 to 1 with each recursion. This comparative analysis highlights the impact of each strategy on the model's ability to generate optimal feasible solutions.

The comparison results for medium-scale MIP instances are shown in Figure 4. The solution quality score represents the percentage of feasible solutions generated by LLMs that have a smaller gap compared to the Gurobi solver. The average score in the test dataset is illustrated in Table 1. The optimal approach involves using a high temperature to encourage LLMs to explore a broader range of possibilities, which helps prevent them from getting trapped in specific nodes. Subsequently, lowering the temperature exploits and refines the better results, allowing for improved solution quality.

5 Conclusion

Our study evaluates the potential of LLMs to address unknown MIP models and uses the carpooling dispatch case to demonstrate how LLMs can enhance efficiency in finding better feasible solutions. This, in turn, can expedite traditional mathematical solvers' processes by pruning unnecessary nodes. We also examine the effectiveness of various temperature management strategies for fine-tuned LLama-3.1-8B in solving medium-scale MIP instances. The comparison results reveal that different temperature management approaches significantly influence the quality of feasible solutions obtained by the model. Starting with a high temperature to explore more nodes and then exploiting better solutions by gradually lowering the temperature improves the quality of feasible solutions generated by LLMs.

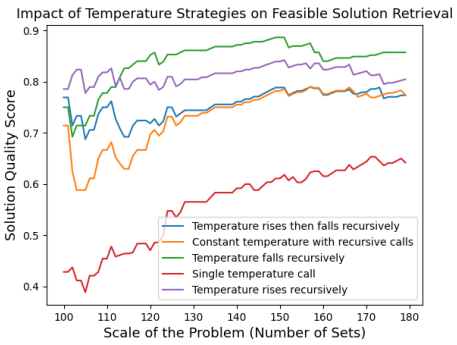

Figure 4: The solution quality score represents the percentage of feasible solutions generated by LLMs that have a smaller gap compared to the Gurobi solver, which shows a larger score means better performance.

Table 1: Average Scores for Different Temperature Strategies

Strategy	Average Score
Single Temperature Call	0.560
Constant Temperature	0.732
Temperature Rise Then Fall	0.756
Temperature Rise	0.813
Temperature Fall	**0.840**

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Tobias Achterberg. 2019. What's new in gurobi 9.0. *Webinar Talk url: https://www. gurobi. com/wp-content/uploads/2019/12/Gurobi-90-Overview-Webinar-Slides-1. pdf*, 5(9):97–113.

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. 2023. Optimus: Optimization modeling using mip solvers and large language models. *arXiv preprint arXiv:2310.06116*.

Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. 2016. The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering*, 99:300–313.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint ArXiv:2005.14165*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32.

Dongdong Ge, Qi Huangfu, Zizhuo Wang, Jian Wu, and Yinyu Ye. 2022. Cardinal optimizer (copt) user guide. *arXiv preprint arXiv:2208.14314*.

Ralph E Gomory. 2010. *Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem*. Springer.

Fang He, Jie Yang, and Meng Li. 2018. Vehicle scheduling under stochastic trip times: An approximate dynamic programming approach. *Transportation Research Part C: Emerging Technologies*, 96:144–159.

Josef Jablonskỳ et al. 2015. Benchmarks for current linear and mixed integer optimization solvers. *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 63(6):1923–1928.

David S Johnson. 1974. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):272–314.

Elias Khalil, Pierre Le Bodic, Le Song, George Nemhauser, and Bistra Dilkina. 2016. Learning to branch in mixed integer programming. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Andreas Klose and Andreas Drexl. 2005. Facility location models for distribution system design. *European journal of operational research*, 162(1):4–29.

Gilbert Laporte. 1992. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247.

Eugene L Lawler and David E Wood. 1966. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719.

CPLEX User's Manual. 1987. Ibm ilog cplex optimization studio. *Version*, 12(1987-2018):1.

Silvano Martello, David Pisinger, and Daniele Vigo. 2000. The three-dimensional bin packing problem. *Operations research*, 48(2):256–267.

Meng Qi, Mengxin Wang, and Zuo-Jun Shen. 2021. Smart feasibility pump: Reinforcement learning for (mixed) integer programming. *arXiv preprint arXiv:2102.09663*.

Jialin Song, Yisong Yue, Bistra Dilkina, et al. 2020. A general large neighborhood search framework for solving integer linear programs. *Advances in Neural Information Processing Systems*, 33:20012–20023.

Yunhao Tang, Shipra Agrawal, and Yuri Faenza. 2020. Reinforcement learning for integer programming: Learning to cut. In *International conference on machine learning*, pages 9367–9376. PMLR.

Teng Wang, Zhenqi He, Wing-Yin Yu, Xiaojin Fu, and Xiongwei Han. 2024. large language models are good multi-lingual learners : when llms meet cross-lingual prompts. *arXiv preprint arXiv:2409.11056*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

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

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, et al. 2023. Chain-of-experts: When llms meet complex operations research problems. In *The Twelfth International Conference on Learning Representations*.

Hegen Xiong, Shuangyuan Shi, Danni Ren, and Jinjin Hu. 2022. A survey of job shop scheduling problem: The types and models. *Computers & Operations Research*, 142:105731.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*.

Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Jiayi Zhang, Chang Liu, Xijun Li, Hui-Ling Zhen, Mingxuan Yuan, Yawen Li, and Junchi Yan. 2023. A survey for solving mixed integer programming via machine learning. *Neurocomputing*, 519:205–217.

6 Appendix

6.1 Prompt template

Prompt Template

Your task is to find the optimal solution for a carpool dispatch problem. If you cannot find the optimal solution, you should try to return a better feasible solution. The background is as follows:

Rules for vehicle-order pair:

- Each order is for one user only.
- Each car can accept up to two orders.
- There are three possible scenarios:
  1. Two people simultaneously hail a ride within a certain time range and decide to share the ride. The car will pick up Person A first, then Person B, and drop off Person A first, followed by Person B.
  2. A person willing to share a ride, Person A, is already in the car, and the car has started the trip. A new carpool request from Person B is received. The car will go from its current location to pick up Person B, drop off Person A, and finally drop off Person B.
  3. A person takes a ride from start to finish without any carpooling.

The objective is to minimize the total Manhattan distance between vehicles and users, considering all segments of the vehicle paths:

– LaTex code for building car pooling MIP model –

You are given x and y coordinates (which are transformed from latitude and longitude using the Mercator projection method) for every empty vehicle, one-person vehicle, and user. The format is as follows:

EMPTY VEHICLES: (0) (x0, y0), (1) (x1, y1) ... # Use "\n" if no vehicles.

ONE ORDER VEHICLES: (0) (x0, y0), (1) (x1, y1) ... # Use "\n" if no one-order vehicles.
USERS: (0) (x0, y0), (1) (x1, y1) ... # Use "\n" if no users.

After transforming the latitude and longitude into x and y, it's much easier to calculate the Manhattan distance between two places. The Manhattan distance between (x0, y0) and (x1, y1) = abs(x1-x0) + abs(y1-y0)

EMPTY VEHICLES: (0) (86.97, 35.86), (1) (85.23, 36.74), (2) (95.62, 28.43),
ONE ORDER VEHICLES: (0) (90.55, 35.17), (1) (101.43, 44.49), (2) (100.56, 44.77),
USERS: (0) (90.33, 35.82), (1) (97.04, 41.87), (2) (100.91, 42.75),

Below are some previous solutions. You can derive the optimal solution straightforwardly or derive the intermediate feasible solution step by step.

The term "gap" refers to the difference between the best-known solution and the best possible solution (optimal solution) within a given tolerance. The smaller the gap, the better the feasible solution. The format of previous solutions is:

x: (EMPTY_0, USER_5) (EMPTY_2, USER_3)... # Car whose index is 'EMPTY_0' is assigned to user whose index is 'USER_5', and so on.
y: (EMPTY_1, USER_1, USER_0) (EMPTY_3, USER_10, USER_9)... # Car whose index is 'EMPTY_1' picks up user whose index is 'USER_1', then user whose index is 'USER_0', and so on.
z: (ONE_REQUEST_0, USER_6), (ONE_REQUEST_1, USER_7)... # Car whose index is 'ONE_REQUEST_0' with one existing passenger picks up user whose index is 'USER_6', and so on.

The x line is "\n" if no one is assigned to a car alone.
The y line is "\n" if no two people are assigned to share a car.
The z line is "\n" if no one is assigned to a car with one existing passenger.
one of solutions starts:
x: (0, 1) (1, 0)

z: (1, 2)
gap: 1.0, objective value: 24.36
one of solutions ends
(—-more exemplars ——-)

6.2 The Method for Calculating Distance

Considering the high computational cost and impracticality of using Dijkstra's algorithm to calculate distances for each instance, we simplify the model by calculating Manhattan distances when formulating the MIP. However, in the simulation process, Dijkstra's algorithm is employed to simulate vehicle movement.