Project on Railway Timetabling

April 20, 2025

Train timetable defines the departure/arrival time of each train $j \in J$ at the origin, destination and intermediate stations. For example, in the Beijing-Shanghai line, each train departs from Beijing and heads to Shanghai (called down direction) or departs from Shanghai and heads to Beijing (called up direction). In the timetable, the total running time of train j is defined by the elapsed time from origin station to destination station. There are so called ideal time schedule for some trains. However, they may be modified to meet practical constraints such as track capacity, interval times, etc.

Now we consider the macro-scope rail timetabling problem, in which we do not consider the internal operations within the station and assume that the station has only one track for up and down direction. At the same time, in order to take care of the inter-station operation requirements, we define several different types of time intervals to avoid any two trains being too close to each other. These assumptions simplifies the complexity of modelling. In meso-scope or micro-scope models, it's usually necessary to consider the inner-station structure to construct a feasible and practical train timetabling plan, which involves more complicated decision variables and constraints (more details in [Zhang et al., 2020b]).

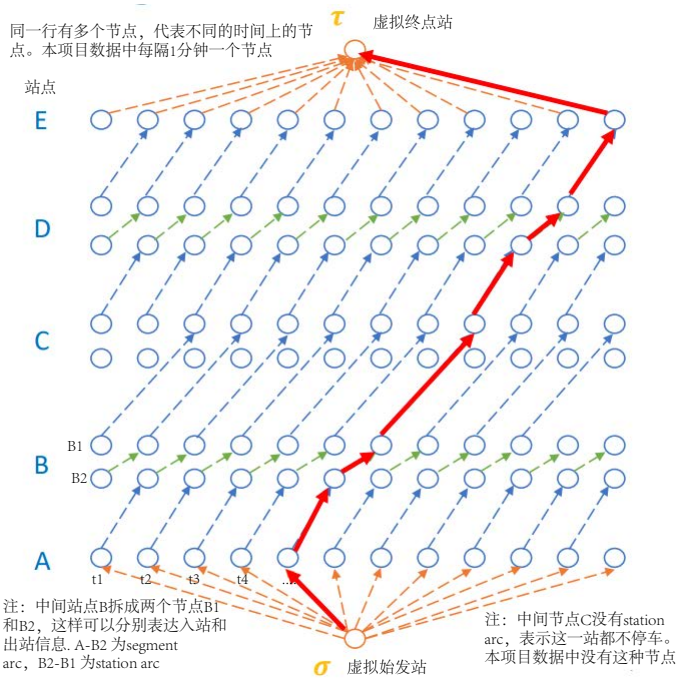
In this note, we create a space-time network model to solve discretized-time train timetabling problem based on the model [Caprara et al., 2002], we create a space-time network model, and apply the Lagrangian relaxation method to solve the problem. Results for a toy example involving 7 states and 16 trains are presented to show the effectiveness of our model and method.

1 Space-Time Network Model

We use a directed, acyclic and multiplicative graph $G = (V, E)$ to characterize the train timetabling problem. The node set V has the form $\{\sigma, \tau\} \cup U \cup W$, where σ, τ denote artificial origin and destination nodes, respectively. In addition to the artificial nodes, we further assume U denotes the set of arrival nodes and W denotes the set of departure nodes. Each normal node $v \in U \cup V$ is denoted by a binary vector $v = (s(v), t(v))$ where $s(v)$ denotes the station and $t(v)$ denotes the discrete time point.

The arcs E in graph G can be divided into the following categories:

1. starting arcs (σ, v), where $s(v)$ denotes the starting station of certain train.
2. station arcs $(u, w), u \in U, w \in W, s(u) = s(w)$ denotes that a certain train enters the station $s(u)$ at time $t(u)$ and leave the same station at time $t(w)$.
3. segment arcs $(w, u), u \in U, w \in W$ denotes the route of some train, i.e., a train leaves the station $s(w)$ at time $t(w)$ and arrive at another station $s(u)$ at time $t(u)$.



4. ending arcs (u, τ), where $s(u)$ is the terminal of some train.

Since the route and timetable plan of each train $j \in J$ are fairly different, the available nodes and arcs of each train are also different, so we define a subgraph $G_j \subseteq G$ for each train j. Any schedule of each train can be viewed as a path in the subgraph G_j (also in the original graph G). Finding the conflict-free paths for all the trains is defined as the train timetable problem.

1.1 A Binary Integer Programming Model

First, we introduce our model parameters, decision variables, objective functions, and various types of constraints.

Model Parameters

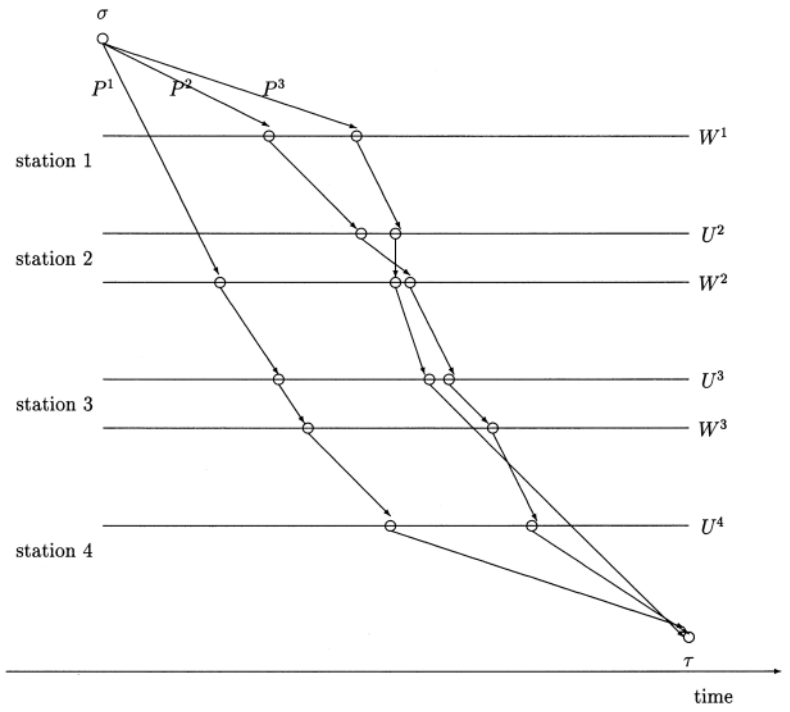
- p_e : the "profit" of using a certain arc e;
- σ, τ : the artificial origin and destination node;
- J : the set of trains;
- $\delta_j^-(v)$: set of in arcs of node v in E^j;
- $\delta_j^+(v)$: set of out arcs of node v in E^j;
- E^j : set of available arcs of train j;
- E : set of all arcs in graph G;
- V^j : set of available nodes of train j;
- V : set of all nodes in graph G;
- $\mathcal{T}(v)$: set of trains may passing through node v;
- $\mathcal{N}(v)$: set of nodes conflicted with node v.

Decision Variables

- $x_e = \{0, 1\}$: whether or not use the arc $e \in E$;
- y_v : whether or not use the node v;
- z_{jv} : whether or not node v is occupied by train$j$.

Objective Function The objective defines as $\sum_{j\in J}\sum_{e\in E^j} p_e x_e$, which represents the sum of the "profits" of all occupied edges in a certain timetable. Although this objective is a simple linear function, we can greatly enrich the practical meaning by interpreting different definition of "profit" of each edge. For example, if we set all $p_{(\sigma,v)}$ to 1 and all others to 0, the objective means we maximize the number of trains in the timetable; if we set p_e to the opposite of the block section running time, the objective means that we minimize the total running time of all trains; on top of that, some artificial adjustments are made to some p_e, such as assigning smaller values to those arcs which may be more congested, then the objective function indicates minimizing the total running time as well as considering congestion to some degree. This idea is especially crucial in the subsequent Lagrangian relaxation method, which in essence is to control the degree of congestion of arcs through adjusting the "profit" p_e of each arc in the space-time network.

An example of train paths in graph G (with $s=4,\ t=3,\ f_1=2, l_1=4, f_2=1, l_2=4,\ f_3=1, l_3=3$).



Model Constraints Any feasible solution of the problem should satisfy the following constraints:

- For each train j, it can choose at most one starting/ending arcs. Some starting/ending occupied means that there exists some train j in the timetable (to occupy this arc):

$$\sum_{e\in\delta_j^+(\sigma)} x_e \le 1,\ \sum_{e\in\delta_j^-(\tau)} x_e \le 1,\quad j\in J.$$

- Non-artificial nodes must have equal in and out degrees. Actually, the degree should be in $\{0,1\}$.

$$\sum_{e\in\delta_j^-(v)} x_e = \sum_{e\in\delta_j^+(v)} x_e,\quad j\in J,\quad v\in V\backslash\{\sigma,\tau\}.$$

- Logic constraints: whether the node v is occupied by train j and whether the node v is occupied:

$$z_{jv} = \sum_{e \in \delta_j^-(v)} x_e, \quad j \in J, \quad v \in V^j, \qquad \text{and} \qquad y_v = \sum_{j \in \mathcal{T}(v)} z_{jv}, \quad v \in V^j.$$

- Headway constraints between trains, which means that only one of the conflicting nodes can be occupied. Headway constraints indicates that the trains departing/entering same station should satisfy certain time lower limit to avoid collision. For any node $v \in U \cup W$, the neighbourhood $\mathcal{N}(v) \subseteq V$ defines a clique constraint:

$$\sum_{v' \in \mathcal{N}(v)} y_{v'} \leq 1, \quad v \in V.$$

For any train $j \in J$, the sets or parameters with superscript or subscript notation corresponds to relevant object to j. The entire $0-1$ integer programming model is given by

$$\max_x \sum_{j \in J} \sum_{e \in E^j} p_e x_e \tag{1}$$

$$\textbf{s.t.} \sum_{e \in \delta_j^+(\sigma)} x_e \leq 1, \quad j \in J \tag{2}$$

$$\sum_{e \in \delta_j^-(v)} x_e = \sum_{e \in \delta_j^+(v)} x_e, \quad j \in J, \quad v \in V \backslash \{\sigma, \tau\}, \tag{3}$$

$$\sum_{e \in \delta_j^-(\tau)} x_e \leq 1, \quad j \in J \tag{4}$$

$$z_{jv} = \sum_{e \in \delta_j^-(v)} x_e, \quad j \in J, \quad v \in V^j \tag{5}$$

$$y_v = \sum_{j \in \mathcal{T}(v)} z_{jv}, \quad v \in V^j \tag{6}$$

$$\sum_{v' \in \mathcal{N}(v)} y_{v'} \leq 1, \quad v \in V \tag{7}$$

$$x_e \in \{0, 1\} \quad e \in E, \tag{8}$$

where (2), (3), (4) denotes the arcs of train j should form a valid path in G, (5), (6) represent the logical relationship of x, y, z, (7) represents headway constraints.

This model is a pure binary programming problem with many variables and constraints, and may take a long time to solve directly by a mathematical optimization solver (e.g. Gurobi or COPT).

1.2 The Lagrangian Relaxation Method

Note that constraint (3) is a flow conservation constraint, which means the in and out degree of v must be balanced. Constraints (2) and (4) denote whether a certain train is in the timetable or not. If we only consider constraints (2), (3), (4) and (8), then the model is separable respect to each train and the model for train j is:

$$\max_{x} \sum_{e \in E^j} p_e x_e \tag{9}$$

$$\textbf{s.t.} \sum_{e \in \delta_j^+(\sigma)} x_e \leq 1, \tag{10}$$

$$\sum_{e \in \delta_j^-(v)} x_e = \sum_{e \in \delta_j^+(v)} x_e, \qquad v \in V \backslash \{\sigma, \tau\}, \tag{11}$$

$$\sum_{e \in \delta_j^-(\tau)} x_e \leq 1, \tag{12}$$

$$x_e \in \{0, 1\} \quad e \in E, \tag{13}$$

The above problem is a shortest path problem which can be solved efficiently in a polynomial time.

Compared to the constraints (2)-(4), the constraints (5)-(7) are all coupling constraints involved with multiple trains. Let $\{\lambda_{v'}\}$ be the Lagrangian multiplier associated to the constraints (7). At the k-th iteration, the Lagrangian relaxation method is to relax the constraints (7) and solves the subproblem

$$x^{k+1} = \arg\max_{x} \sum_{j \in J} \sum_{e \in E^j} p_e x_e - \sum_{v \in V} \lambda_v^k (\sum_{v' \in \mathcal{N}(v)} y_{v'} - 1) \tag{14}$$

$$\textbf{s.t.} \sum_{e \in \delta_j^+(\sigma)} x_e \leq 1, \quad j \in J \tag{15}$$

$$\sum_{e \in \delta_j^-(v)} x_e = \sum_{e \in \delta_j^+(v)} x_e, \quad j \in J, \quad v \in V \backslash \{\sigma, \tau\}, \tag{16}$$

$$\sum_{e \in \delta_j^-(\tau)} x_e \leq 1, \quad j \in J \tag{17}$$

$$z_{jv} = \sum_{e \in \delta_j^-(v)} x_e, \quad j \in J, \quad v \in V^j \tag{18}$$

$$y_v = \sum_{j \in \mathcal{T}(v)} z_{jv}, \quad v \in V^j \tag{19}$$

$$x_e \in \{0, 1\} \quad e \in E. \tag{20}$$

Although the constraints (18) and (19) are still kept in the model but they are eventually eliminated. Thus, the model (14)-(20) only has variables x_e, and both the objective function and constraints can be decomposed into shortest path sub-problems for each train. Then, the Lagrangian multiplier is updated as

$$\lambda_v^{k+1} = \max\{0, \lambda_v^k + \eta(\sum_{v' \in \mathcal{N}(v)} y_{v'} - 1)\}.$$

One important drawback of Lagrangian relaxation is the violation of the relaxing constraints. It is often necessary to obtain a feasible solution by some primal heuristic algorithm.

Primal heuristic algorithm The primal heuristic algorithm module is important for the success of the Lagrangian relaxation method. Since the Lagrangian relaxation method can only obtain pairwise solutions and cannot guarantee to satisfy the relaxation constraints, the heuristic algorithm directly determines the quality of the final output feasible solution. One commonly used heuristic method is Algorithm 1. It is based on the dual solution of Lagrangian relaxation, and constructs a primal solution by scheduling

the congested train first. Note that this heuristic works well when the timetable is not very crowed, otherwise the problem needs to be solved with the aid of a mathematical optimization solver (e.g., gurobi and COPT).

Algorithm 1 Ranking based SPP Primal Heuristic

Require: reordering trains by dual objective function (including multipliers) in descending order.
priority_list ← sort by desending order of dual obj(more congested train first)
while *priority_list* Not Empty **do**
Step 1. $j \leftarrow$ first train in *priority_list*
Step 2. run the SPP algorithm in the origin graph, and remove all conflicting nodes and arcs. If the algorithm succeed, then keep the train in the timetable, otherwise skip the train.
end while
Output all trains with feasible paths, which defines a timetable.

2 A Toy Example

In this section, we present a toy example involving 7 states and 16 trains. The object is to schedual a timetable for these trains withing 160 minutes. The time interval is 1 minute, i.e., there are 161 time nodes at each train station in the space-time network model.

Station Data Each column of Table 1 indicates the name of the station and the distance between each station and starting station A.

station	**mile**
A	0
B	50
C	100
D	170
E	210
F	250
G	300

Table 1: information of the train stations

Trains Data Each column of Table 2 indicates, from left to right, the train number, the train speed, and the status of the train at this station. Specifically, 0 means that the train will pass through the station directly and 1 means that this train must stop at the station. These information will determine the available arcs E^j of train j.

trainNO	speed	A	B	C	D	E	F	G
G1	350	1	0	0	0	0	1	1
G3	350	1	1	1	0	1	0	1
G5	350	1	1	0	1	0	1	1
G7	350	1	1	0	0	1	0	1
G9	350	1	1	0	0	0	1	1
G11	350	1	1	0	0	1	0	1
G13	300	1	1	1	1	1	1	1
G15	300	1	1	1	1	1	1	1
G17	300	1	1	1	1	1	1	1
G19	300	1	1	1	1	1	1	1
G21	300	1	1	1	1	1	1	1
G23	300	1	1	1	1	1	1	1
G25	300	1	1	1	1	1	1	1
G27	300	1	1	1	1	1	1	1
G29	300	1	1	1	1	1	1	1
G31	300	1	1	1	1	1	1	1

Table 2: status at train stations

Block Section Data Each column of Table 3 indicates, from left to right, the name of the blocked interval, the running time (minutes) of the train at the speed of 300 km/h, and the running time (minutes) of the train at the speed of 350 km/h.

station	runtime(300)	runtime(350)
A-B	10	9
B-C	20	18
C-D	14	12
D-E	8	7
E-F	8	7
F-G	10	8

Table 3: Running time between stations

Other Parameter All kinds of headway time lower bound are set as 5 minutes. If a train stop at a station, it needs to stop at least 2 mins and at most 15 mins.

Simulation Results We run the Lagrangian relaxation method on the sample data with stopping criteria as $ub - lb \leq 0.1ub$. Figure 1 shows the bound updated through iterations and Figure 2 shows the output timetable.

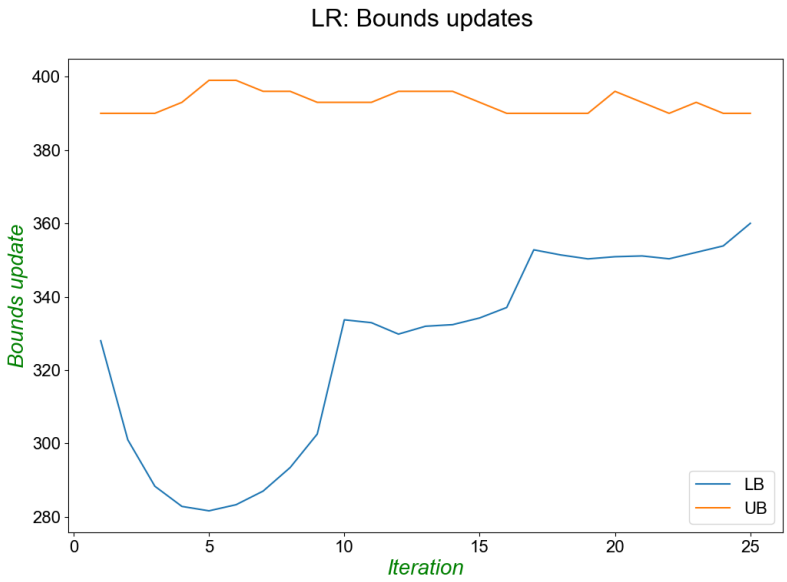


Figure 1: Bound update through iterations

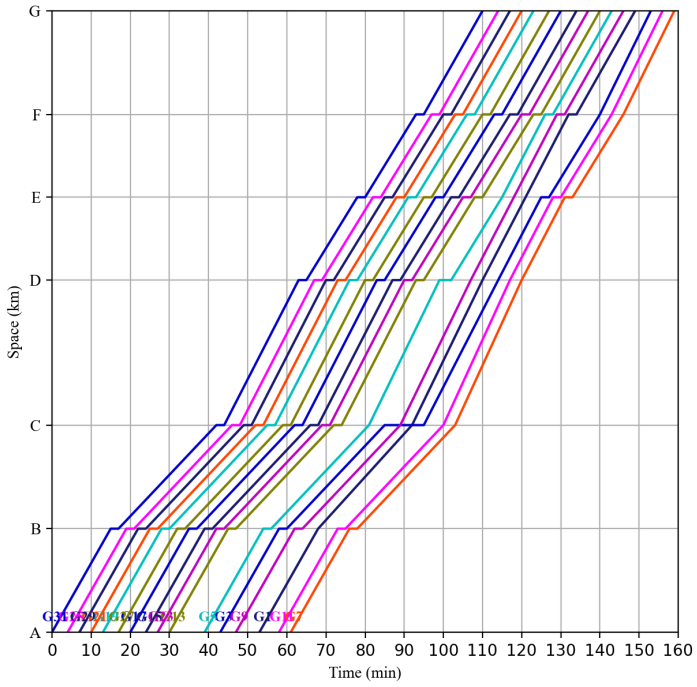


Figure 2: timetable generated by the Lagrangian relaxation method

3 Automated Optimization Modeling Using LLMs

In optimization practice, the most challenging aspect is often not the execution of algorithms, but the mathematical modeling of the problem. Successful optimization modeling requires deep domain knowledge, extensive experience, and a thorough understanding of the problem domain. With the development of modern commercial solvers, many standard optimization problems can be efficiently solved by calling these mature tools, but the modeling process still requires highly specialized skills, which limits the widespread application of optimization techniques.

In recent years, Large Language Models (LLMs) have demonstrated remarkable capabilities in understanding natural language and performing complex reasoning tasks, offering new possibilities for automating optimization modeling[Xiao et al., 2024, Astorga et al., 2024, Huang et al., 2025, Lu et al., 2025]. By leveraging LLMs to transform natural language problem descriptions into formalized mathematical models or even generating code that calls solvers directly, the barrier to applying optimization techniques can be significantly reduced, enabling non-specialists to utilize optimization methods for solving real-world problems.

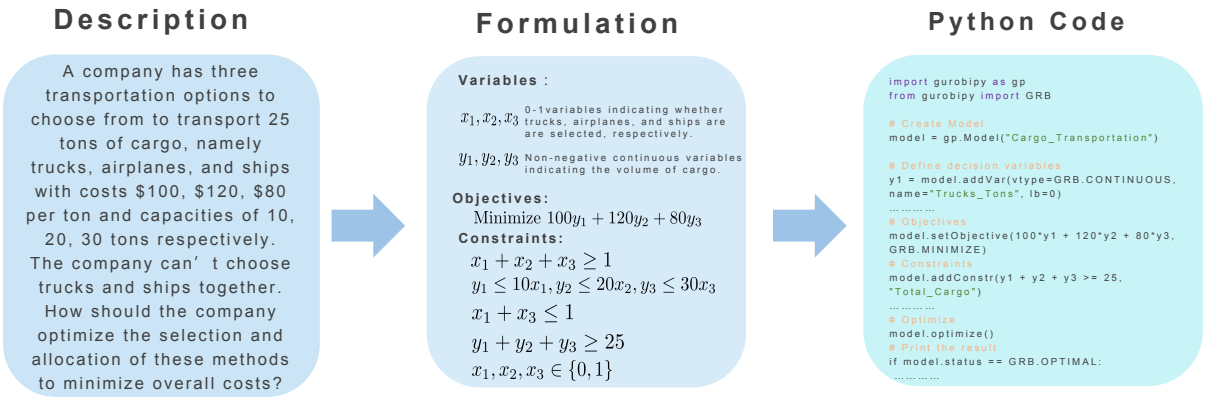


Figure 3: An example of Automated Optimization Modeling Using LLMs

Figure 3 illustrates how LLMs can facilitate the transformation from a natural language problem description to a formal optimization model and executable code. In this example, a transportation logistics problem is presented in plain language, describing a company's need to optimize cargo distribution across three available modes of transportation (trucks, airplanes, and ships) with different costs and capacity constraints.

The LLM first interprets this description to extract the essential elements of the optimization problem, formulating a mathematical model with clearly defined decision variables (binary variables indicating transportation mode selection and continuous variables for cargo volume), an objective function (minimizing the total transportation cost), and constraints (capacity limitations, incompatibility between certain modes, and total cargo requirements). This formalization process demonstrates the LLM's ability to recognize the underlying optimization structure from natural language.

The LLM then generates executable Python code that implements this mathematical formulation using the Gurobi optimization package. The generated code includes all necessary imports, model initialization, variable definitions, objective specification, constraint implementation, and solver execution commands. This end-to-end pipeline—from problem description to ready-to-run code—exemplifies how LLMs can democratize access to optimization techniques by bridging the gap between domain-specific problems and their mathematical solutions, allowing users without specialized optimization knowledge to leverage powerful solver technologies.

4 Questions

Submission requirement:

1. Prepare a report including
   - detailed answers to each question
   - numerical results and their interpretation
2. The programming language can be either matlab, Python or c/c++.
3. 6月22日晚12点前将书面报告(包括latex源文件，程序等等）打包, 发email给助教(pkuopt@163.com). 提交的文件请全部打包，文件名为“train-name1-name2.zip”.
   提交word 的同学需要提供word 原文件并将其转换成pdf 文件。
4. 请勿大量将代码粘在报告中，涉及到实际结果需要打表或者作图，不要截图或者直接从命令行拷贝结果。
5. If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

Project Questions:

1. Read the description of the problem (1)-(8) carefully. Create the space-time network model and write a code to construct the data of the toy example in section 2. Then solve the problem and its LP relaxation by using either Gurobi or Mosek or COPT. Report the number of the variables and constraints as well the CPU time. Plot the timetable similar to Figure 2.
2. Consider reformulating the problem using a job-shop scheduling approach instead of the space-time network model. Compare these two modeling approaches in terms of their formulation differences, solution times, model size, and practical advantages/disadvantages for train timetabling problems.A few references are [Cebi et al., 2020, Sharma and Jain, 2016, ho Zeno. Yu, 2021]
3. Design a LLM-assisted optimization modeling pipeline to solve the problem (1)-(8).
   - Read and understand the papers and code in [AhmadiTeshnizi et al., 2023][1]. Prepare a description of the problem in natural language, then ask the large language model (LLM) using its web interface to generate a corresponding mathematical model and executable code based on your description.
   - (Optional) Design a modular pipeline system by calling the API interface of LLM, referencing the OptiMUS architecture. Your pipeline should include at least the following modules:
     - Model generation module: Using LLM to generate mathematical model formulations
     - Code generation module: Converting mathematical models into executable code
     - Error correction module: Using different Large Language Models or user feedback to validate and improve generated models and code
     - RAG (Retrieval-Augmented Generation) module: Retrieving appropriate modeling techniques and examples from relevant literature (Optional)

     Complete the following tasks:

     (a) Apply your pipeline to separately formulate the train timetabling problem as: (1) a space-time network model, and (2) a job-shop scheduling model. Generate code for both formulations using Gurobi and compare their effectiveness.

[1] https://github.com/teshnizi/OptiMUS

(b) Test your solution code on toy examples and provide detailed analysis

(c) Evaluate the performance differences of various Large Language Model APIs on this task

4. Implement the Lagrangian relaxation method for solving problem (1)-(8). Write down more detailed description the Lagrangian relaxation method if your implementation is different from 1.2. Report the CPU time and the violation of constraints:

$$\text{feas} := \sum_{v \in V} \max \left\{ 0, \sum_{v' \in \mathcal{N}(v)} y_{v'} - 1 \right\}. \tag{21}$$

Plot the timetable similar to Figure 2.
Requirements: feas should be zero. Otherwise, this solution is not meaningful.

5. Write down and implement either the augmented Lagrangian method or the alternating direction method of multipliers for solving problem (1)-(8). Report the CPU time and the violation of constraints defined in (21). Plot the timetable similar to Figure 2.
Requirements: feas should be zero. Otherwise, this solution is not meaningful.
Hints: The objective function of the subproblem with respect to the variable x is a general quadratic function. A possible strategy is to linearize the objective function and add a proximal term. Since $x_e^2 = x_e$ when $x_e \in \{0, 1\}$, the resulted subproblem is still linear and can be solved the same as the shortest path problem (9)-(13).

6. (Optional) Construct a more realistic dataset based on (1)-(8). Following the implementation of OptMATH [Lu et al., 2025], design a concise pipeline to generate 100 high-quality data samples, where each sample is a triplet consisting of (natural language problem description, mathematical formulation, implementation code).

7. Propose a prototye reinforcement learning (RL) algorithm to solve the train time table problem. A few references are [Lemos et al., 2019, Kool et al., 2018, Cappart et al., 2021, Zhang et al., 2020a, Zhou et al., 2020, Mazyavkina et al., 2021, Joshi et al., 2022]. Unlike the standard job-shop scheduling problem, the train timetable problem requires determining not only the departure times but also the dwelling times at each station. These dwelling times directly influence the headway constraints, rendering traditional priority-based rules insufficient.
Requirements:

- Formulate a Markov decision process (MDP), clearly defining the state space, action space, transition function, and reward function.
- Explain how headway constraints are handled. Two possible strategies include: masking invalid actions to prevent constraint violations, or incorporating the headway constraints into the reward function to guide the learning process.
- Design the problem features and policy network architecture. The features should capture both static problem data and dynamic decision-making context. The policy network should take these features as input and output a distribution over the action space at each decision step.
- Specify the RL training algorithm to be used.
- (Optional) Implement the proposed algorithm. Train the model using the dataset constructed in Question 6. Compare its performance on the toy example against the traditional solvers developed in Question 4 and 5.

Hints: When formulating the MDP, two distinct scheduling paradigms can be considered:

- Priority-Based Sequential Scheduling. Trains are scheduled sequentially based on a learned priority order, with both departure times and dwelling times determined for each train in turn.
- Synchronized Time-Step Scheduling. At each decision step, first select a train currently dwelling at a station, then decide whether to extend its dwell time or dispatch it. Note that decision steps are aligned with real-time progression in this situation.

References

[AhmadiTeshnizi et al., 2023] AhmadiTeshnizi, A., Gao, W., and Udell, M. (2023). OptiMUS: Optimization modeling using MIP solvers and large language models.

[Astorga et al., 2024] Astorga, N., Liu, T., Xiao, Y., and Schaar, M. v. d. (2024). Autoformulation of mathematical optimization models using LLMs.

[Cappart et al., 2021] Cappart, Q., Chételat, D., Khalil, E., Lodi, A., Morris, C., and Veličković, P. (2021). Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*.

[Caprara et al., 2002] Caprara, A., Fischetti, M., and Toth, P. (2002). Modeling and Solving the Train Timetabling Problem. *Operations Research*, 50(5):851–861.

[Cebi et al., 2020] Cebi, C., Ata, E., and Sahingoz, O. K. (2020). Job shop scheduling problem and solution algorithms: A review. *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7.

[Gasse et al., 2019] Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. *arXiv preprint arXiv:1906.01629*.

[ho Zeno. Yu, 2021] ho Zeno. Yu, Y. (2021). A research review on job shop scheduling problem. *E3S Web of Conferences*.

[Huang et al., 2025] Huang, C., Tang, Z., Hu, S., Jiang, R., Zheng, X., Ge, D., Wang, B., and Wang, Z. (2025). ORLM: A customizable framework in training large models for automated optimization modeling.

[Joshi et al., 2022] Joshi, C. K., Cappart, Q., Rousseau, L.-M., and Laurent, T. (2022). Learning the travelling salesperson problem requires rethinking generalization. *arXiv preprint arXiv:2006.07054*.

[Karalias and Loukas, 2020] Karalias, N. and Loukas, A. (2020). Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. *Advances in Neural Information Processing Systems*, 33:6659–6672.

[Kool et al., 2018] Kool, W., Van Hoof, H., and Welling, M. (2018). Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*.

[Lemos et al., 2019] Lemos, H., Prates, M., Avelar, P., and Lamb, L. (2019). Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 879–885. IEEE.

[Li et al., 2018] Li, Z., Chen, Q., and Koltun, V. (2018). Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31.

[Lu et al., 2025] Lu, H., Xie, Z., Wu, Y., Ren, C., Chen, Y., and Wen, Z. (2025). OptMATH: A scalable bidirectional data synthesis framework for optimization modeling.

[Mazyavkina et al., 2021] Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400.

[Schuetz et al., 2022] Schuetz, M. J., Brubaker, J. K., and Katzgraber, H. G. (2022). Combinatorial optimization with physics-inspired graph neural networks. *Nature Machine Intelligence*, 4(4):367–377.

[Sharma and Jain, 2016] Sharma, P. and Jain, A. (2016). A review on job shop scheduling with setup times. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 230:517 – 533.

[Xiao et al., 2024] Xiao, Z., Zhang, D., Wu, Y., Xu, L., Wang, Y., Han, X., Fu, X., Zhong, T., Zeng, J., Song, M., and Chen, G. (2024). Chain-of-experts: When llms meet complex operation research problems. *arXiv preprint*.

[Zhang et al., 2020a] Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., and Chi, X. (2020a). Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1621–1632.

[Zhang et al., 2020b] Zhang, Q., Lusby, R. M., Shang, P., and Zhu, X. (2020b). Simultaneously reoptimizing timetables and platform schedules under planned track maintenance for a high-speed railway network. *Transportation Research Part C: Emerging Technologies*, 121:102823.

[Zhou et al., 2020] Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., and Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.