Homework for "Algorithms for Big-Data Analysis"

Beijing International Center for Mathematical Research
Peking University

February 17, 2024

Note: Please write up your solutions independently. If you get significant help from others, write down the source of references. A formal mathematical proof for all your claims is required.

1. This exercise shows that an efficient procedure for updating a tableau can be derived from the SMW formula in numerical linear algebra.

   (a) Let C be an $m \times m$ invertible matrix and let $u, v \in \mathbb{R}^m$ be two vectors. Show that

   $$(C + uv^\top)^{-1} = C^{-1} - \frac{C^{-1}uv^\top C^{-1}}{1 + v^\top C^{-1}u}.$$

   (b) Assuming that C^{-1} is available, explain how to obtain $(C + uv^\top)^{-1}$ using only $O(m^2)$ arithmetic operations.

   (c) Let B and $\bar{B}$ be basis matrices before and after an iteration of the simplex method. Let $A_{B(l)}$ and $A_{\bar{B}(l)}$ be the exiting and entering column, respectively. Show that

   $$\bar{B} - B = (A_{\bar{B}(l)} - A_{B(l)})e_l^\top,$$

   where e_l is the lth unit vector.

   (d) Note that $e_i^\top B^{-1}$ is the ith row of B^{-1} and $e_l^\top B^{-1}$ is the pivot row. Show that

   $$e_i^\top \bar{B}^{-1} = e_i^\top B^{-1} - g_i e_l^\top B^{-1}, \quad i = 1, \ldots, m,$$

   for suitable scalars g_i. Provide a formula for g_i. Interpret the above equation in terms of the mechanics for pivoting in the revised simplex method.

2. Let x be an element of the standard form polyhedron $P = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$. Prove that a vector $d \in \mathbb{R}^n$ is a feasible direction at x if and only if $Ad = 0$ and $d_i \geq 0$ for every i such that $x_i = 0$.

Homework for “Algorithms for Big-Data Analysis”

Beijing International Center for Mathematical Research
Peking University

March 20, 2024

Note: Please write up your solutions independently. If you get significant help from others, write down the source of references. A formal mathematical proof for all your claims is required.

1. 给定矩阵$A \in \mathbb{R}^{m\times n}$，向量$b \in \mathbb{R}^n$。考虑优化问题：

$$\min_{x\in\mathbb{R}^n} \ \|x\|_1, \text{ s.t. } Ax = b, \tag{1}$$

其中$\|x\|_1 = \sum_{i=1}^n |x_i|$。

(a) 写出问题(1)的对偶问题。

(b) 写出问题(1)的最优性(KKT)条件。

(c) 假设z 是问题(1)的最优解。如何进一步加强(b)里给出的最优性条件，使得z是唯一最优解并给出证明。

(d) 假设z是方程组$Ax = b$非零元最少的解。证明z也是(1) 的最优解的充要条件是：对所有满足$Ah = 0$的向量h有

$$\sum_{i\in T} \text{sign}(z_i)h_i \le \sum_{i\in T^c} |h_i|,$$

其中$T = \{i \mid z_i \neq 0\}$，$T^c$是$T$的补集，以及

$$\text{sign}(z_i) = \begin{cases} 1, & \text{if } z_i > 0, \\ 0, & \text{if } z_i = 0, \\ -1, & \text{if } z_i < 0. \end{cases}$$

Project on "Optimal Transport"

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

June 18, 2023

Consider the optimal transport problem:

$$
\begin{aligned}
\min_{\pi\in\mathbb{R}^{m\times n}} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\pi_{ij} \\
s.t. \quad & \sum_{j=1}^{n}\pi_{ij} = \alpha_i, \quad \forall i = 1,\ldots,m, \\
& \sum_{i=1}^{m}\pi_{ij} = \beta_j, \quad \forall j = 1,\ldots,n, \\
& \pi_{ij} \geq 0,
\end{aligned} \tag{1}
$$

where c, α and β are given, and $\sum_{i=1}^{m}\alpha_i = \sum_{j=1}^{n}\beta_j = 1$, $\alpha \geq 0$ and $\beta \geq 0$.

1. Solve (1) by calling mosek and gurobi directly in Matlab or python. The package "CVX" is not allowed to use here. Compare the performance between the simplex methods, the interior point methods and the network simplex method (if it is available).
2. Write down and implement the Sinkhorn method.
3. (Optional) Write down and implement a first-order method, for example, the alternating direction method of multipliers.
4. Test problems:
   - Generate the data c, α and β using the following code and images:
     `http://faculty.bicmr.pku.edu.cn/~wenzw/bigdata/gen_ot_data.m`
     `http://faculty.bicmr.pku.edu.cn/~wenzw/bigdata/source.png`
     `http://faculty.bicmr.pku.edu.cn/~wenzw/bigdata/dest.png`
   - Choose two other images based on your own preference and generate the data.
5. Requirement:
   (a) Compare the efficiency (cpu time) and accuracy (checking optimality condition) of different methods.
   (b) Prepare a report including
      - detailed answers to each question

- numerical results and their interpretation

(c) Pack all of your codes in one file named as “projot-name-ID.zip” and send it to TA: pkuopt@163.com

(d) If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

Homework for “Algorithms for Big-Data Analysis”

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

March 19, 2025

1 Submission Requirement

1. Prepare a report including
   - detailed answers to each question
   - numerical results and their iterpretation
2. The programming language can be either matlab, Python or c/c++.
3. Pack all of your codes named as “proj1mk-name-ID.zip” send it to TA: pkuopt@163.com
   作业提交需要统一打包成压缩文件，命名格式为：proj1mk-学号-姓名，文件类型随意。文件名中不要出现空格，最好不要出现中文。
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. 提交word 的同学需要提供word 原文件并将其转换成pdf 文件。
6. If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

2 Algorithms for ℓ_1 minimization

Consider the problem

$$\min_x \quad \mu\|x\|_1 + \frac{1}{2}\|Ax-b\|_2^2, \tag{2.1}$$

where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$ are given. Test data are as follows:

```
n = 1024;
m = 512;
A = randn(m,n);
u = sprandn(n,1,0.1);
b = A*u;
```

```
    mu = 1e-2;
```

See `http://bicmr.pku.edu.cn/~wenzw/bigdata/Test_BP.m`

1. First write down an equivalent model of (2.1) which can be solved by calling mosek and gurobi directly, then implement the code.
   Mosek: `http://www.mosek.com/`
   Gurobi: `http://www.gurobi.com/`

2. Write down and implement the proximal gradient method for (2.1).
   Reference: Neal Parikh, Stephen Boyd, Proximal Algorithms, Foundations and Trends in Optimization, `https://web.stanford.edu/~boyd/papers/pdf/prox_algs.pdf`

3. Write down and implement the alternating direction method of multipliers (ADMM) for the primal problem (2.1) or its dual problem.
   Reference: Junfeng Yang, Yin Zhang, *Alternating direction algorithms for l1-problems in Compressed Sensing*, SIAM Journal on Scientific Computing, `https://epubs.siam.org/doi/abs/10.1137/090777761`

4. Reformulate problem (2.1) as a saddle-point problem and implement the primal-dual hybrid gradient algorithm (PDHG).
   Reference: Antonin Chambolle, Thomas Pock, A first-order primal-dual algorithm for convex problems with applications to imaging, Journal of Mathematical Imaging and Vision, `https://link.springer.com/article/10.1007/s10851-010-0251-1`

5. GPU acceleration for algorithms in questions 2, 3, and 4.

   (a) Identify and analyze which specific computational steps in the proximal gradient method, ADMM, and PDHG can be parallelized and accelerated using GPU capabilities. For each algorithm, consider operations such as matrix-vector multiplications, proximal operations, gradient computations, and dual updates. Discuss the potential speedup and challenges associated with GPU implementation for these steps.
   References: David Applegate, Mateo Diaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, Warren Schudy, PDLP: A Practical First-Order Method for Large-Scale Linear Programming, `https://arxiv.org/abs/2501.07018`

   (b) (Optional) Implement the GPU-accelerated versions of the algorithms identified in part (a) using a programming language of your choice (e.g., MATLAB, Python with CUDA/PyTorch, or Julia with CUDA.jl). Provide a performance comparison between the CPU and GPU implementations, highlighting any observed improvements or limitations.
   References:
   cuPDLP.jl, `https://github.com/jinwen-yang/cuPDLP.jl`
   cuPDLP-C, `https://github.com/COPT-Public/cuPDLP-C`
   `https://ww2.mathworks.cn/help/parallel-computing/gpu-computing-in-matlab.html`
   `https://pytorch.org/docs/stable/cuda.html`
   `https://cuda.juliagpu.org/stable/`

6. Algorithm unrolling for the proximal gradient method in question 2:

   (a) Write down the unrolled form of the proximal gradient method. Clearly specify the learnable parameters, number of the unrolled layers. Define the training loss.

Reference: Vishal Monga, Yuelong Li, Yonina C. Eldar, Algorithm Unrolling: Interpretable, Efficient Deep Learning for Signal and Image Processing, IEEE Signal Processing Magazine, `https://ieeexplore.ieee.org/document/9363511`

(b) (Optional) Generate the training dataset for the unrolled proximal gradient method. Train the unrolled algorithm using `PyTorch` and compare the trained one with the proximal gradient method.
Reference: Open-L2O, `https://github.com/VITA-Group/Open-L2O`

7. Requirements for questions 1-4:

   (a) The interface of each method should be written in the following format

   ```
   [x, out] = method_name(x0, A, b, mu, opts);
   ```

   Here, x0 is a given input initial solution, A and b are given data, opts is a struct which stores the options of the algorithm, out is a struct which saves all other output information.

   (b) Compare the efficiency (cpu time) and accuracy (checking optimality condition) in the format as `http://bicmr.pku.edu.cn/~wenzw/bigdata/Test_BP.m`

3 Algorithms For Low-rank Recovery

Consider the model

$$\min_{X\in\mathbb{R}^{m\times n}} \mu\|X\|_* + \sum_{(i,j)\in\Omega}(X_{ij}-M_{ij})^2, \tag{3.1}$$

where the nuclear norm $\|X\|_* = \sum_i \sigma_i(X)$.

1. Write down and implement a proximal gradient method for solving (3.1).
2. Write down and implement an alternating direction method of multipliers (ADMM) for solving (3.1).
3. The data M and Ω are specified in the following script:
   `http://bicmr.pku.edu.cn/~wenzw/bigdata/Test_MC.m`
   Test your method for $\mu = 10^{-1}, 10^{-2}, 10^{-3}$.
4. (Optional) Design a method for solving the following problem:

$$\min_{X\in\mathbb{R}^{m\times n}} \mu\|X\|_* + \sum_{(i,j)\in\Omega}|X_{ij}-M_{ij}|. \tag{3.2}$$

Homework for "Algorithms for Big-Data Analysis"

Beijing International Center for Mathematical Research
Peking University

February 17, 2024

Note: Please write up your solutions independently. If you get significant help from others, write down the source of references. A formal mathematical proof for all your claims is required.

1. Consider the integer programming problem:

$$\begin{aligned} \max \quad & x_1 + 2x_2 \\ \text{s.t.} \quad & -3x_1 + 4x_2 \le 4 \\ & 3x_1 + 2x_2 \le 11 \\ & 2x_1 - x_2 \le 5 \\ & x_1, x_2 \ge 0 \\ & x_1, x_2 \text{ integer.} \end{aligned}$$

   (a) What is the optimal cost of the linear programming relaxation? What is the optimal cost of the integer programming problem?

   (b) What is the convex hull of the set of all solutions to the integer programming problem?

   (c) Illustrate how the Gomory cutting plane algorithm would work. Give the first cut.

   (d) Solve the problem by branch and bound. Solve the linear programming relaxations graphically.

   (e) Suppose you dualize the constraint $-3x_1 + 4x_2 \le 4$. What is the optimal value Z_D of the Lagrangian dual?

   (f) Suppose you dualize the constraint $2x_1 - x_2 \le 5$. What is the optimal value Z_D of the Lagrangian dual?

Homework for "Algorithms For Big Data Analysis"

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

February 17, 2024

1 Theoretical exercises

1. 令$f_i(x) \in \mathbb{R}^n \to \mathbb{R}, \forall i = 1, \ldots, N$, 是闭凸函数，存在次梯度。假设随机次梯度方差是一致有界，即存在M，对任意$x \in \mathbb{R}^n$以及随机下标s_k，有

$$E_{s_k}[\|g\|^2] \leq M^2 < +\infty, \quad \forall g \in \partial f_{s_k}(x).$$

考虑求解优化问题$\min_{x\in\mathbb{R}^n} \ f(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$ 的随机梯度下降算法(SGD)：

$$x^{k+1} = x^k - \alpha_k g_k, \quad g_k \in \partial f_{s_k}(x^k),$$

其中s_k是从$\{1, \cdots, N\}$中随机等可能地抽取的一个样本，$\alpha_k > 0$为步长。

(a) 令x^*是优化问题的最优解。证明对所有的$K \geq 1$，下面不等式成立：

$$\sum_{k=1}^{K} \alpha_k E[f(x^k) - f(x^*)] \leq \frac{1}{2}E[\|x^1 - x^*\|^2] + \frac{1}{2}\sum_{k=1}^{K} \alpha_k^2 M^2.$$

(b) 令$A_K = \sum_{i=1}^{K} \alpha_i$，定义$\bar{x}_K = \frac{1}{A_K}\sum_{k=1}^{K} \alpha_k x^k$，证明存在常数$D$，使得下面不等式成立：

$$E[f(\bar{x}_K) - f(x^*)] \leq \frac{D + \sum_{k=1}^{K} \alpha_k^2 M^2}{2\sum_{k=1}^{K} \alpha_k}.$$

2 Coding exercises

2.1 Submission Requirement

1. Prepare a report including
   - detailed answers to each question
   - numerical results and their iterpretation
2. The programming language can be either matlab, Python or c/c++.

3. Pack all of your codes named as "sto-ID-name.zip" and upload the file to send it to TA: pkuopt@163.com
   作业提交需要统一打包成压缩文件，命名格式为：sto-学号-姓名，文件类型随意。文件名中不要出现空格，最好不要出现中文。

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. 提交word 的同学需要提供word 原文件并将其转换成pdf 文件。

6. If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

2.2 Variants of Stochastic Gradients Algorithms

Consider the nonconvex problem

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(w) + \lambda\|w\|_2^2, \tag{2.1}$$

where $f_i(w) = 1 - \tanh(y_i w^\top x_i)$, $\lambda > 0$ and (x_i, y_i) is the i-th data pair.

1. Write down and implement two of the following algorithms: Adagrad, adam, SVRG

2. You are encouraged to read the implementation in Pytorch, tensorflow as well as other packages. However, you should implement the codes by yourself.

3. Download the datasets covtype and gisette from
   `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`.
   If the testing set is not available, please split the data set into a training set and testing set randomly according to a ratio $7:3$ (Also check references on cross validation).

4. Test a few choices of λ (for example, $10, 1, 0.1, 0.001$. This value probably depends on the data sets). Generate figures on training error versus epoch, testing error versus epoch, training error versus time, testing error versus time, etc.

5. Extra-credit: propose, implement and test one of the following algorithms
   (a) stochastic quasi-Newton method
   (b) any other better idea

Homework for “Algorithms For Big Data Analysis”

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

February 17, 2024

1 Submission Requirement

1. Prepare a report including
    - detailed answers to each question
    - numerical results and their iterpretation
2. The programming language can be either matlab, Python or c/c++.
3. Pack all of your codes named as ”svd-ID-name.zip” and send it to TA: pkuopt@163.com
   作业提交需要统一打包成压缩文件，命名格式为：svd-学号-姓名，文件类型随意。文件名中不要出现空格，最好不要出现中文。
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. 提交word 的同学需要提供word 原文件并将其转换成pdf 文件。
6. If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

2 Randomized Singular Value Decomposition Algorithms

Given a matrix $A \in \mathbb{R}^{m\times n}$, compute p-largest singular values and their corresponding left and right singular vectors.

1. Write down and implement one of the algorithms in (extra credit for choosing both algorithms)
    - LinearTimeSVD Algorithm on page 166 of “Petros Drineas, Ravi Kannan, and Michael W. Mahoney, Fast Monte Carlo Algorithms for Matrices II: Computing a Low-Rank Approximation to a Matrix, SIAM J. Comput., 36(1), 158183”
    - Prototype for Randomized SVD on page 227 of “N. Halko, P. G. Martinsson, and J. A. Tropp, Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions,SIAM Rev., 53(2), 217288. ”

2. Compute $r \in \{5, 10, 15, 20\}$ largest singular values and their corresponding singular vectors on the following two examples.

   - A random matrix A generated as follows:

     ```
     m = 2048;
     n = 512;
     p = 20;
     A = randn(m,p)*randn(p,n);
     ```

   - Pick one of your favorite images. The smallest dimension of the image should be at least 1000. Suppose that the file name of the image is “peppers.png”. The following matlab codes construct a matrix A.

     ```
     A1 = imread('peppers.png'); %read the image peppers.png
     imshow(A1); %display the image
     A = rgb2gray(A1); %Convert to grayscale
     A = double(A); %convert the type of data to double
     ```

3. Extra-credit: Accelerate the speed for solving the following matrix completion problem using the randomized SVD techniques:

$$\min_{X \in R^{m \times n}} \quad \frac{1}{2} \sum_{(i,j) \in \Omega} (X_{ij} - M_{ij})^2 + \mu \|X\|_*.$$

Homework for “Algorithms For Big Data Analysis”

Zaiwen Wen
Beijing International Center for Mathematical Research
Peking University

February 28, 2022

1 Submission Requirement

1. Prepare a report including
   - detailed answers to each question
   - numerical results and their iterpretation
2. The programming language can be either matlab, Python or c/c++.
3. Pack all of your codes named as “phase-ID-name.zip” and send it to TA: pkuopt@163.com
   作业提交需要统一打包成压缩文件，命名格式为：phase-学号-姓名，文件类型随意。文件名中不要出现空格，最好不要出现中文。
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. 提交word 的同学需要提供word 原文件并将其转换成pdf 文件。
6. If you get significant help from others on one routine, write down the source of references at the beginning of this routine.

2 Algorithms for phase retrieval

One popular formulation of the phase retrieval problem is solving a system of quadratic equations in the form

$$y_r = |\langle a_r, z\rangle|^2, \;\; r = 1, 2, ..., m, \tag{2.1}$$

where $z \in \mathcal{C}^n$ is the decision variable, $a_r \in \mathcal{C}^n$ are known sampling vectors, $\langle a_r, z\rangle$ is the inner product between a_r and z in $\mathcal{C}^n$, $|a|$ is the magnitude of $a \in \mathcal{C}$, and $y_r \in \mathcal{R}$ are the observed measurements.

Choose either a) or b). Write down and implement the algorithm.

a) (i) Write down an SDP relaxation for problem (2.1) from one of the following references.

   - E. J. Candes, Y. Eldar, T. Strohmer and V. Voroninski. Phase retrieval via matrix completion. SIAM J. on Imaging Sciences 6(1), 199–225.

- Irene Waldspurger, Alexandre dAspremont and Stephane Mallat, Phase recovery, MaxCut and complex semidefinite programming, Mathematical Programming, Ser. A (2015) 149:4781

(ii) Design an alternating direction method of multipliers (ADMM) to solve this SDP.

b) Consider the nonlinear least squares problem:

$$\min_z \ f(z) = \frac{1}{m}\sum \left| |\langle a_r, z\rangle|^2 - y_r \right|. \tag{2.2}$$

Write down a subgradient method to solve (2.2).

- Reference:
  E. J. Candes, X. Li and M. Soltanolkotabi. Phase retrieval via Wirtinger flow: theory and algorithms. IEEE Transactions on Information Theory 61(4), 1985–2007.

c) Test problems:

- The 1D test problems in
  `https://viterbi-web.usc.edu/~soltanol/WFcode.html`
- A real image from Matlab:

  `imread('ngc6543a.jpg');`

d) DO NOT copy the codes online directly!

Homework for “Algorithms for Big-Data Analysis”

Beijing International Center for Mathematical Research
Peking University

February 17, 2024

Note: Please write up your solutions independently. If you get significant help from others, write down the source of references. A formal mathematical proof for all your claims is required.

1. 考虑有限情形的MDP(S, A, P, R, γ)，其中S是有限个离散状态的集合，A是有限个离散动作的集合，R是奖励函数，$\gamma \in (0,1)$是折扣因子。给定时刻t的状态s和动作a，下一时刻转移到状态s'的概率是$P(s' \mid s, a) = P(s_{t+1} = s' \mid s_t = s, a_t = a)$。令$V(s)$为价值函数，定义Bellman 算子$B$：

$$BV(s) = \max_a \left\{ R(s,a) + \gamma \sum_{s' \in S} P(s' \mid s, a) V(s') \right\}, \quad \forall s \in S.$$

   (a) 证明算子B是压缩映射，即：

$$\|BV - BV'\|_\infty \le \gamma \|V - V'\|_\infty,$$

   其中$\|V - V'\|_\infty = \max_s |V(s) - V'(s)|$。

   (b) 从V_0开始执行迭代算法：$V_{k+1} = BV_k$。对于任意$k > 0$，证明：

$$\|V_{n+k} - V_n\|_\infty \le \frac{\gamma^n}{1-\gamma} \|V_1 - V_0\|_\infty.$$

2. 令考虑有限步MDP (S, A, s_1, P, R, H)，其中S为状态集合，A为动作集合，s_1为初始状态，P为转移概率矩阵，R为奖励矩阵，H为终止时间，γ为折扣因子。定义$\pi(s,a)$ 是在状态s根据随机策略π 执行动作a的概率。定义$\tau = (s_1, a_1, s_2, \ldots, a_{H-1}, s_H)$ 是从状态s_1出发，执行策略π产生的轨道(状态-动作对)，即$a_t \sim \pi(s_t, \cdot)$。

   (a) 写出轨道τ的概率表达式$D^\pi(\tau)$。

   (b) 定义$\rho(\pi)$是有限步总奖励的均值，即：

$$\rho(\pi) = \mathbb{E}\left[\sum_{t=1}^{H} \gamma^{t-1} r_t | \pi, s_1 \right].$$

   令$R(\tau)$是在轨道τ获得的总奖励。写出$\rho(\pi)$关于$R(\tau)$的表达式。

(c) 假设$\pi_\theta(s,a)$是参数化之后的策略，其中θ是参数。证明

$$\nabla_\theta \rho\left(\pi_\theta\right)=\mathbb{E}_\tau\left[R(\tau) \nabla_\theta \log \left(D^{\pi_\theta}(\tau)\right)\right]=\mathbb{E}_\tau\left[R(\tau) \sum_{t=1}^{H-1} \nabla_\theta \log \left(\pi_\theta\left(s_t, a_t\right)\right)\right].$$

(d) 给定状态s_t和参数θ，假设$b_t(s_t)$条件独立于π_θ产生的抽样。证明：

$$\mathbb{E}_\tau\left[\sum_{t=1}^{H-1} b_t\left(s_t\right) \frac{\partial}{\partial \theta_j} \log \left(\pi_\theta\left(s_t, a_t\right)\right) | \theta, s_1\right]=0.$$

(e) 给出满足(d)里条件的一种$b_t(s_t)$。

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]).

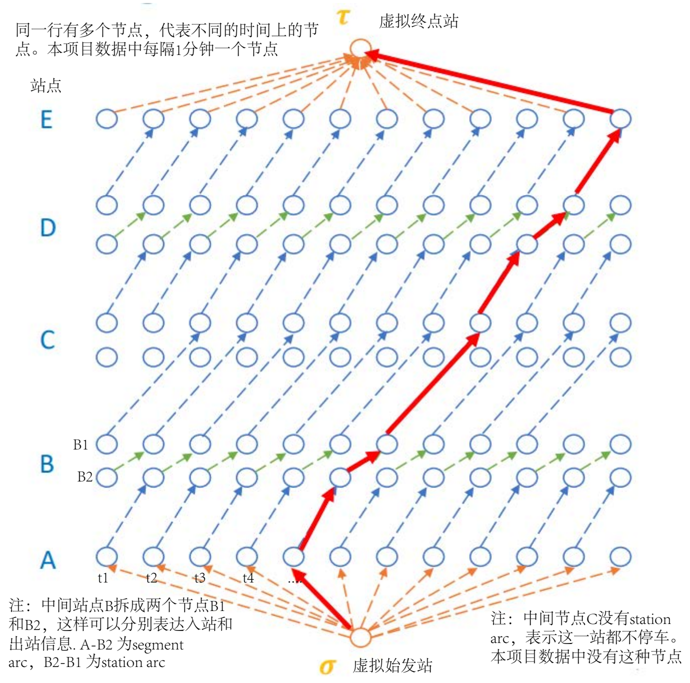
In this note, we create a space-time network model to solve discretized-time train timetabling problem based on the model [Caprara et al., 2002], we create a space-time network model, and apply the Lagrangian relaxation method to solve the problem. Results for a toy example involving 7 states and 16 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 $t(u)$.


同一行有多个节点，代表不同的时间上的节点。本项目数据中每隔1分钟一个节点
τ
虚拟终点站
站点
E
D
C
B1
B
B2
A
t1
t2
t3
t4
....
注：中间站点B拆成两个节点B1和B2，这样可以分别表达入站和出站信息. A-B2 为segment arc，B2-B1 为station arc
σ
虚拟始发站
注：中间节点C没有station arc，表示这一站都不停车。本项目数据中没有这种节点

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

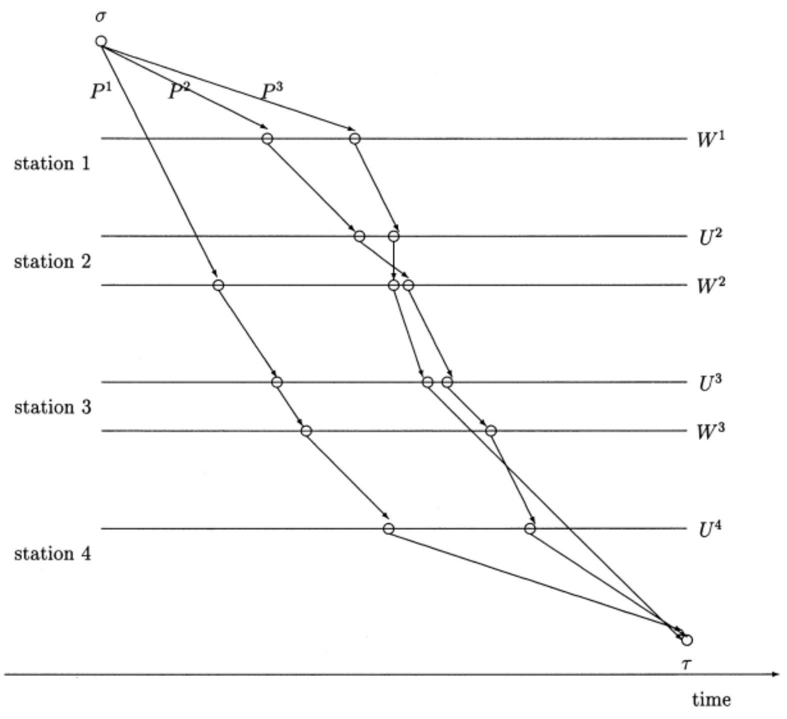
- p_e : the "profit" of using a certain arc e;
- σ, τ : the artificial origin and destination node;
- J : the set of trains;
- $\delta_j^-(v)$: set of in arcs of node v in E^j;
- $\delta_j^+(v)$: set of out arcs of node v in E^j;
- E^j : set of available arcs of train j;
- E : set of all arcs in graph 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:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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. $j$ ← 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.

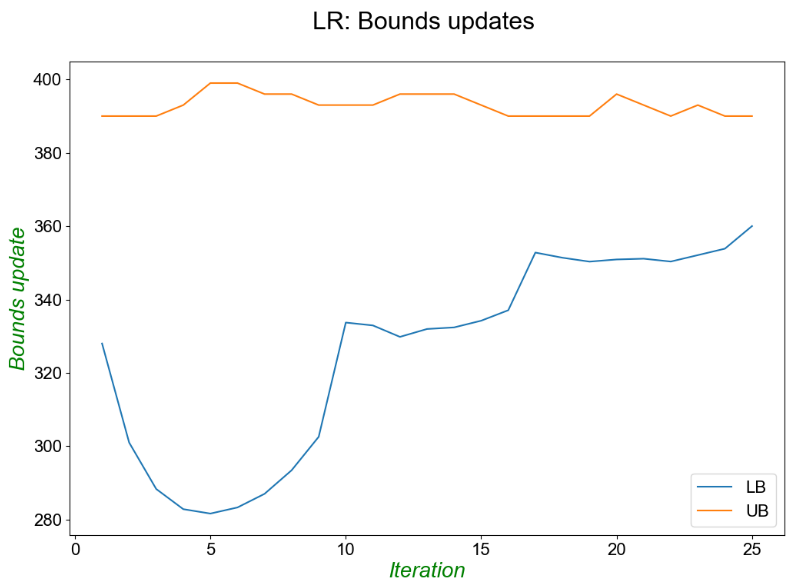


Figure 1: Bound update through iterations

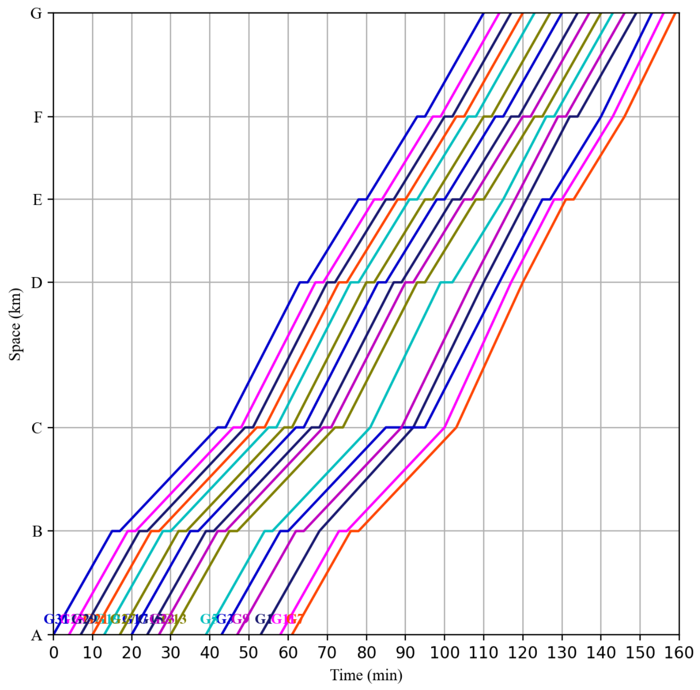
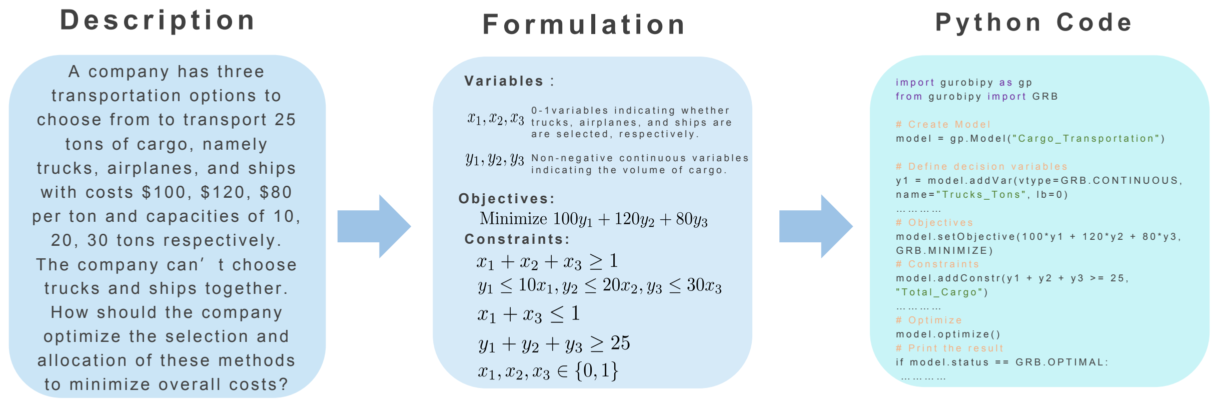


Figure 2: timetable generated by the Lagrangian relaxation method

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.



Description

A company has three transportation options to choose from to transport 25 tons of cargo, namely trucks, airplanes, and ships with costs $100, $120, $80 per ton and capacities of 10, 20, 30 tons respectively. The company can' t choose trucks and ships together. How should the company optimize the selection and allocation of these methods to minimize overall costs?

Formulation

Variables :

x_1, x_2, x_3 0-1variables indicating whether trucks, airplanes, and ships are are selected, respectively.

y_1, y_2, y_3 Non-negative continuous variables indicating the volume of cargo.

Objectives:

Minimize $100y_1 + 120y_2 + 80y_3$

Constraints:

$$x_1 + x_2 + x_3 \geq 1$$
$$y_1 \leq 10x_1, y_2 \leq 20x_2, y_3 \leq 30x_3$$
$$x_1 + x_3 \leq 1$$
$$y_1 + y_2 + y_3 \geq 25$$
$$x_1, x_2, x_3 \in \{0, 1\}$$

Python Code

```
import gurobipy as gp
from gurobipy import GRB

# Create Model
model = gp.Model("Cargo_Transportation")

# Define decision variables
y1 = model.addVar(vtype=GRB.CONTINUOUS,
name="Trucks_Tons", lb=0)
… … … …
# Objectives
model.setObjective(100*y1 + 120*y2 + 80*y3,
GRB.MINIMIZE)
# Constraints
model.addConstr(y1 + y2 + y3 >= 25,
"Total_Cargo")
… … … …
# Optimize
model.optimize()
# Print the result
if model.status == GRB.OPTIMAL:
 … … … …
```

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 re-optimizing 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.

A Customized Augmented Lagrangian Method for Block-Structured Integer Programming

Rui Wang, Chuwen Zhang, Shanwen Pu, Jianjun Gao, Zaiwen Wen

Abstract—Integer programming with block structures has received considerable attention recently and is widely used in many practical applications such as train timetabling and vehicle routing problems. It is known to be NP-hard due to the presence of integer variables. We define a novel augmented Lagrangian function by directly penalizing the inequality constraints and establish the strong duality between the primal problem and the augmented Lagrangian dual problem. Then, a customized augmented Lagrangian method is proposed to address the block-structures. In particular, the minimization of the augmented Lagrangian function is decomposed into multiple subproblems by decoupling the linking constraints and these subproblems can be efficiently solved using the block coordinate descent method. We also establish the convergence property of the proposed method. To make the algorithm more practical, we further introduce several refinement techniques to identify high-quality feasible solutions. Numerical experiments on a few interesting scenarios show that our proposed algorithm often achieves a satisfactory solution and is quite effective.

Index Terms—Integer programming, augmented Lagrangian method, block coordinate descent, convergence

1 Introduction

In this paper, we consider a block-structured integer programming problem:

$$\begin{aligned} \min\ & \boldsymbol{c}^\top \mathbf{x} && \text{(1a)}\\ \text{s.t. } & \boldsymbol{A}\mathbf{x} \le \boldsymbol{b}, && \text{(1b)}\\ & \mathbf{x}_j \in \mathcal{X}_j,\ j = 1,2,...,p, && \text{(1c)} \end{aligned}$$

where $\mathbf{x}_j \in \mathbb{R}^{n_j}$ is the j-th block variable of $\mathbf{x} \in \mathbb{R}^n$, i.e., $\mathbf{x} = (\mathbf{x}_1; ...; \mathbf{x}_p)$ for $p \ge 1$ with $n = \sum_{j=1}^{p} n_j$. In (1), $\boldsymbol{A} \in \mathbb{R}^{m\times n}, \boldsymbol{b} \in \mathbb{R}^m, \boldsymbol{c} \in \mathbb{R}^n$ and the constraint $\mathcal{X}_j$ is the set of $0,1$ vectors in a polyhedron, i.e.,

$$\mathcal{X}_j := \{\mathbf{x}_j \in \{0,1\}^{n_j} : \boldsymbol{B}_j\mathbf{x}_j \le \boldsymbol{d}_j\},\ j = 1,2,...,p.$$

The constraints (1c) can be reformulated as $\mathbf{x} \in \mathcal{X} := \{\mathbf{x} \in \{0,1\}^n : \boldsymbol{B}\mathbf{x} \le \boldsymbol{d}\}$ where the block diagonal matrix $\boldsymbol{B} \in \mathbb{R}^{q\times n}$ is formed by the small submatrices $\boldsymbol{B}_j$ as the main diagonal and $\boldsymbol{d} = (\boldsymbol{d}_1; ...; \boldsymbol{d}_p)$. Correspondingly, $\boldsymbol{c}$ and $\boldsymbol{A}$ can be rewritten as $\boldsymbol{c} = (\boldsymbol{c}_1; \boldsymbol{c}_2; ...; \boldsymbol{c}_p)$ with $\boldsymbol{c}_j \in \mathbb{R}^{n_j}$ and $\boldsymbol{A} = (\boldsymbol{A}_1\ \boldsymbol{A}_2\ ...\ \boldsymbol{A}_p)$ with $\boldsymbol{A}_j \in \mathbb{R}^{m\times n_j}$.

Assume that these constraints (1c) are "nice" in the sense that an integer program with just these constraints is easy. Therefore, if the coupling constraints (1b) are ignored, the remaining problem which is only composed of the constraints (1c) is easier to solve than the original problem (1). For convenience, we assume $\mathcal{X}$ is not empty and (1) is feasible. Denote by f^{IP} the optimal value of the problem (1). This block structure is closely related to an important model known as "n-fold integer programming (IP)" studied extensively in computer vision [1], [2], machine learning [3], [4] and theoretical computer science [5], [6], etc. The theoretical foundations of n-fold IPs have significant implications for efficient algorithm development in various fields. For example, an algorithmic theory of integer programming based on n-fold IP was proposed in [5]. Recent advancements have been provided in [6]. Furthermore, the progress in theory and application of integer programming with a block structure was summarized in [7], [8]. While existing works on n-fold IP mainly focus on asymptotic analyses, this work aims to develop an efficient augmented Lagrangian approach tailored to the block structure for practical efficiency with a convergence guarantee.

- *Rui Wang and Zaiwen Wen are with the Beijing International Center for Mathematical Research, Peking University, Beijing 100871, China (email: ruiwang@bicmr.pku.edu.cn; wenzw@pku.edu.cn).*
- *Chuwen Zhang, Shanwen Pu and Jianjun Gao are with the School of Information Management and Engineering, Shanghai University of Finance and Economics, Shanghai 200433, China (email: chuwzhang@gmail.com; 2019212802@live.sufe.edu.cn;gao.jianjun@shufe.edu.cn).*

This research is supported in part by the National Natural Science Foundation of China (NSFC) grants 12331010, 72150001, 72225009, 72394360, 72394365.

1.1 Related Work

The branch and bound algorithm for general integer programming (IP) was first introduced by Land and Doig [9]. Gomory [10] developed a cutting plane algorithm for integer programming problems. These two approaches are at the heart of current state-of-the-art software for integer programming. Unfortunately, these methods often suffer from high computational burdens due to the discrete constraint, thus they are not good choices for solving some large-scale practical problems. Therefore, it is necessary to develop efficient approaches to obtain feasible and desirable solutions, even if they may not be globally optimal. Considering the block structure of the integer linear programming (1), a natural idea is to decompose a large global problem into smaller local subproblems. There are three typical decomposition methods for solving such problem: Benders [11], Dantzig-Wolfe (DW) [12] and Lagrangian decompositions [13]. The Benders decomposition method is used to deal with mixed integer programming problems by decomposing them into a master problem and a primal subproblem. It generates cuts that are added to the master problem by solving the dual of the primal subproblem. This method is not suitable for (1), since our primal subproblem is an integer programming problem, which is still NP-hard. The DW decomposition method can solve block structured integer programming problem (1) based on resolution theorem. To improve the tractability of large-scale problems, the DW decomposition relies on column generation. However, it might be harder or even intractable to solve the master problem by column generation if further constraints are applied to $\mathcal{X}_j$ [14, Chapter 8.2]. The Lagrangian decomposition introduces Lagrange multipliers and constructs a sequence of simpler subproblems. Although the method itself has a limitation due to its inherent solution symmetry for certain practical problems, the Lagrangian duality bound is useful in the branch-and-bound procedure. A Lagrangian heuristic algorithm has also been proposed in [15] for solving a real-world train timetabling problem, which decomposed the problem into smaller subproblems using Lagrangian relaxation and developed a heuristic algorithm based on subgradient optimization to generate feasible solutions.

Many techniques from continuous optimization including the alternating direction method of multipliers (ADMM) have been applied to solve integer programming recently. The authors in [16] proposed an ℓ_p-box ADMM for solving a binary integer programming, where the binary constraint is replaced by the intersection of a box and an ℓ_p-norm sphere. The authors in [17] proposed augmented Lagrangian method (ALM) and ADMM based on the ℓ_1 augmented Lagrangian function for two-block mixed integer linear programming (MILP). For the multi-block MILP problem, an extended ADMM was proposed in [18] to employ a release-and-fix approach to solve the subproblems. There are also some heuristic methods based on ADMM [19], [20], [21], [22] that have been successfully applied to various practical integer programming problems.

There are some other widespread approaches on the relaxation of the binary constraint including linear programming (LP) relaxation [23], [24] and semidefinite relaxation [25], [26]. For the multi-block MILP problem, several inexact methods have been proposed including a distributed algorithm relying on primal decomposition [27], a dual decomposition method [28], and a decomposition-based outer approximation method [29]. By introducing continuous variables to replace the discrete variables, the exact penalty methods [30], [31], [32] have been studied for solving the nonlinear IP problems. Then the problem is transformed into an equivalent nonlinear continuous optimization problem.

1.2 Contributions

In this paper, we propose a customized ALM for solving (1). Our main contributions are listed below.

(i) We define a novel augmented Lagrangian (AL) function that differs from the classical AL function and establish strong duality theory for the augmented Lagrangian relaxation of the block-structured integer programming (1), which motivates us to utilize the ALM for solving the problem (1).

(ii) Based on the special structure of (1), we propose two block coordinate descent (BCD)-type methods that are well-suited for solving the resulting subproblems in our ALM framework. These methods utilize classical update and proximal linear update techniques, denoted as ALM-C and ALM-P, respectively. We also analyze their convergence properties under proper conditions.

(iii) To address challenges in finding the global optimal solution for practical problems such as train timetabling, we introduce refinement strategies and propose a customized ALM to enhance the quality of solutions generated by the ALM. Our numerical experiments demonstrate the effectiveness of the proposed method in solving large-scale instances.

Note that the ADMM-based method in [21] solves each subproblem only once per iteration for a fixed Lagrangian multiplier. On the other hand, our ALM method involves multiple iterations using the BCD method to minimize each AL function until certain rules are satisfied. Once the AL subproblem is solved exactly, the strong duality guarantees that the ALM can converge to a global minimizer of the problem (1). Therefore, achieving high accuracy in minimizing the AL function allows our ALM to achieve superior solutions with fewer iterations, resulting in significantly reduced computational time compared to the ADMM. This claim is supported by our numerical tests. Additionally, we introduce Assumptions 3.1 and 3.2, derived from the structural characteristics of the practical problem in [20] and [21], and subsequently analyze the theoretical properties of the ALM-C method under these assumptions. However, our ALM-P method has wider applicability and does not require these specific assumptions. Moreover, the ALM can be regarded as a dual ascent algorithm with respect to dual variables, hence ensuring its convergence. In contrast, the convergence analysis of the ADMM for solving this kind of problem remains unclear.

The existing ALM-based methods for solving integer programming in [33] mainly focused on the duality of the augmented Lagrangian dual for MILP with equality constraints, but numerical experiments were not available. Moreover, our approach differs significantly in that we handle inequality constraints directly, without introducing slack variables. This approach has two key benefits: it reduces the number of variables, thus decreasing computational burden in high-dimensional cases, and it allows for more customized algorithmic design based on the inherent structure of the problem. In [17], the ℓ_1 norm was considered as an augmented term in the AL function for MILP such that the minimization of the AL function can be decomposed into multiple low-dimensional ℓ_1-penalty subproblems due to the separable blocks. These subproblems were then solved in parallel using the Gurobi solver. In our work, we take a different approach by using a quadratic term in the AL function which allows the augmented Lagrangian subproblem to be reduced to a linear programming under certain conditions. Furthermore, we update the blocks of variables sequentially one at a time.

1.3 Notation and Organization

Let $\mathbb{N}_m := \{1, 2, ..., m\}$, $\mathbb{N}_p := \{1, 2, ..., p\}$ and $\mathbb{R}^m_+ := \{x \in \mathbb{R}^m : x_i \geq 0 \text{ for all } i\}$. The superscript "$\top$" means "transpose". Denote $\boldsymbol{a}_i$ by the i-th row of the matrix $\boldsymbol{A}$ and b_i by the i-th element of the vector $\boldsymbol{b}$. For convenience, we let $\boldsymbol{A}_{\mathcal{I},j}$ denote the submatrix consisting of columns of $\boldsymbol{A}_j$ indexed by $\mathcal{I}$ and $\boldsymbol{b}_{\mathcal{I}}$ denote the subvector consisting of entries of $\boldsymbol{b} \in \mathbb{R}^m$ indexed by $\mathcal{I}$. A neighborhood of a point $\mathbf{x}^*$ is a set $\mathcal{N}(\mathbf{x}^*, 1)$ consisting of all points $\mathbf{x}$ such that $\|\mathbf{x} - \mathbf{x}^*\|^2 \leq 1$. Let $\mathbf{1}$ be a row vector of all ones.

This paper is structured as follows. An AL function is defined and the strong duality of the problem (1) is discussed in section 2. We propose a customized ALM incorporating BCD methods and refinement strategies to improve the quality of the ALM solutions in section 3. We establish the convergence results of both the BCD methods to minimize the AL function and the ALM applied to the whole problem (1) in section 4. The proposed method is applied to two practical problems in section 5. Concluding remarks are made in the last section.

2 The AL Strong Duality

The Lagrangian relaxation (LR) of (1) with respect to the constraint (1b) has the following form:

$$\min_{\mathbf{x}\in\mathcal{X}} \ \mathcal{L}(\mathbf{x}, \lambda) := \sum_{j=1}^{p} \boldsymbol{c}_j^\top \mathbf{x}_j + \lambda^\top \left(\sum_{j=1}^{p} \boldsymbol{A}_j \mathbf{x}_j - \boldsymbol{b} \right), \tag{2}$$

where $\lambda \in \mathbb{R}_+^m$ is a Lagrange multiplier associated with the constraint (1b). We can observe that (2) is much easier to be solved than the original problem (1) since (2) can be decomposed into p-block low-dimensional independent subproblems. However, there may exist a non-zero duality gap when certain constraints are relaxed by using classical Lagrangian dual [33]. To reduce the duality gap, we add a quadratic penalty function to the Lagrangian function in (2) and solve the augmented Lagrangian dual problem which is defined as follows.

Definition 2.1 (AL Dual). We define an AL function by

$$\begin{aligned} L(\mathbf{x}, \lambda, \rho) \ = \ & \sum_{j=1}^{p} \boldsymbol{c}_j^\top \mathbf{x}_j + \lambda^\top \left(\sum_{j=1}^{p} \boldsymbol{A}_j \mathbf{x}_j - \boldsymbol{b} \right) \\ & + \frac{\rho}{2} \left\| \left(\sum_{j=1}^{p} \boldsymbol{A}_j \mathbf{x}_j - \boldsymbol{b} \right)_+ \right\|^2, \end{aligned} \tag{3}$$

where $\lambda \in \mathbb{R}_+^m, \rho > 0$. The corresponding AL relaxation of (1) is given by

$$d(\lambda, \rho) := \min_{\mathbf{x}\in\mathcal{X}} \ L(\mathbf{x}, \lambda, \rho). \tag{4}$$

We call the following maximization problem the AL dual problem:

$$f_\rho^{\mathrm{LD}} := \max_{\lambda\in\mathbb{R}_+^m} d(\lambda, \rho). \tag{5}$$

Note that the classical AL function of (1) is given by

$$\hat{L}(\mathbf{x}, \lambda, \rho) = \boldsymbol{c}^\top \mathbf{x} + \frac{\rho}{2} \left\| \left(\boldsymbol{A}\mathbf{x} - \boldsymbol{b} + \frac{\lambda}{\rho} \right)_+ \right\|^2 - \frac{\|\lambda\|^2}{2\rho}. \tag{6}$$

We prefer using the form of the AL function (3) rather than the classical version (6) due to the absence of λ/ρ in the quadratic term of the max function. This makes it possible to convert the AL function into a linear function under certain conditions, making the problem (4) easier to solve, which will be explained in the next section. We also verify that the quadratic term in (3) is an exact penalty and strong duality holds between the AL dual problem (5) and the primal problem (1).

Lemma 2.1 (Strong Duality). Suppose the problem (1) is feasible and its optimal value is bounded. If a minimum achievable non-zero slack exists, i.e., there is a δ such that for any $i \in \mathbb{N}_m$,

$$0 < \delta \leq \min_{\mathbf{x}\in\mathcal{X}} \{(\boldsymbol{a}_i\mathbf{x} - b_i)^2 : \boldsymbol{a}_i\mathbf{x} > b_i\}, \tag{7}$$

then there exists a finite $\rho^* \in (0, +\infty)$ such that

$$f_{\rho^*}^{\mathrm{LD}} = \min_{\boldsymbol{A}\mathbf{x}\leq\boldsymbol{b}, \mathbf{x}\in\mathcal{X}} \boldsymbol{c}^\top \mathbf{x}.$$

Proof For any $\lambda \in \mathbb{R}_+^m$ and $\rho > 0$, since $\{\mathbf{x} \in \mathcal{X} : \boldsymbol{A}\mathbf{x} - \boldsymbol{b} \leq 0\} \subseteq \mathcal{X}$, we have

$$d(\lambda, \rho) \leq \min_{\substack{\mathbf{x}\in\mathcal{X} \\ \boldsymbol{A}\mathbf{x}-\boldsymbol{b}\leq 0}} L(\mathbf{x}, \lambda, \rho) \leq \min_{\substack{\mathbf{x}\in\mathcal{X} \\ \boldsymbol{A}\mathbf{x}-\boldsymbol{b}\leq 0}} \boldsymbol{c}^\top \mathbf{x} = f^{\mathrm{IP}}. \tag{8}$$

Then $f_\rho^{\mathrm{LD}} \leq f^{\mathrm{IP}}$.

Now it suffices to find a finite ρ^* such that $f_{\rho^*}^{\mathrm{LD}} \geq f^{\mathrm{IP}}$. We first let $\mathbf{x}^0$ be any arbitrary feasible solution of (1), that is, $\mathbf{x}^0 \in \mathcal{X}$ and $\boldsymbol{A}\mathbf{x}^0 \leq \boldsymbol{b}$. Denote by f^{LP} the linear programming (LP) relaxation of f^{IP}. Since the value of the LP relaxation of (1) is bounded [34], i.e., $-\infty < f^{\mathrm{LP}} \leq \boldsymbol{c}^\top \mathbf{x}^0 < +\infty$, we set $\rho^* = 2(\boldsymbol{c}^\top \mathbf{x}^0 - f^{\mathrm{LP}})/\delta$, then $0 < \rho^* < +\infty$. Moreover,

$$f^{\mathrm{LD}} \geq \max_{\lambda\in\mathbb{R}_+^m} d(\lambda, \rho^*) \geq d(\lambda^*, \rho^*) = \min_{\mathbf{x}\in\mathcal{X}} L(\mathbf{x}, \lambda^*, \rho^*), \tag{9}$$

where $\lambda^* \in \mathbb{R}_+^m$ is a given parameter. Let $\mathbb{I} := \{i \in \mathbb{N}_m : \boldsymbol{a}_i\mathbf{x} - b_i > 0, \mathbf{x} \in \mathcal{X}\}$, we consider following two cases:

Case 1: $\mathbb{I} = \emptyset$. In this case we have $\boldsymbol{A}\mathbf{x} \leq \boldsymbol{b}$ for all $\mathbf{x} \in \mathcal{X}$. By letting $\bar{\lambda} = 0$, we can obtain that

$$\begin{aligned} L(\mathbf{x}, \bar{\lambda}, \rho^*) &= \boldsymbol{c}^\top \mathbf{x} + \bar{\lambda}^\top (\boldsymbol{A}\mathbf{x} - \boldsymbol{b}) + \frac{\rho^*}{2} \| (\boldsymbol{A}\mathbf{x} - \boldsymbol{b})_+ \|^2 \\ &= \boldsymbol{c}^\top \mathbf{x} \geq f^{\mathrm{IP}}. \end{aligned} \tag{10}$$

Case 2: $\mathbb{I} \neq \emptyset$. Denote by λ^{LP} a positive optimal vector of dual variables for $\boldsymbol{A}\mathbf{x} \leq \boldsymbol{b}$ in the LP relaxation of (1). In this case we get

$$\begin{aligned} L(\mathbf{x}, \lambda^{\mathrm{LP}}, \rho^*) =& \boldsymbol{c}^\top \mathbf{x} + (\lambda^{\mathrm{LP}})^\top (\boldsymbol{A}\mathbf{x} - \boldsymbol{b}) + \frac{\rho^*}{2} \sum_{i\in\mathbb{I}} (\boldsymbol{a}_i\mathbf{x} - b_i)^2 \\ & + \frac{\rho^*}{2} \sum\nolimits_{i\notin\mathbb{I}} ((\boldsymbol{a}_i\mathbf{x} - b_i)_+)^2 \\ =& \boldsymbol{c}^\top \mathbf{x} + (\lambda^{\mathrm{LP}})^\top (\boldsymbol{A}\mathbf{x} - \boldsymbol{b}) + \frac{\rho^*}{2} \sum_{i\in\mathbb{I}} (\boldsymbol{a}_i\mathbf{x} - b_i)^2. \end{aligned}$$

Since

$$\frac{\rho^*}{2} \sum_{i\in\mathbb{I}} (\boldsymbol{a}_i\mathbf{x} - b_i)^2 \geq \frac{\rho^*}{2} \min_{i\in\mathbb{I}} (\boldsymbol{a}_i\mathbf{x} - b_i)^2 \overset{(7)}{\geq} \frac{\rho^*}{2} \delta = \boldsymbol{c}^\top \mathbf{x}^0 - f^{\mathrm{LP}},$$

it yields

$$\begin{aligned} L(\mathbf{x}, \lambda^{\mathrm{LP}}, \rho^*) &\geq \boldsymbol{c}^\top \mathbf{x} + (\lambda^{\mathrm{LP}})^\top (\boldsymbol{A}\mathbf{x} - \boldsymbol{b}) + (\boldsymbol{c}^\top \mathbf{x}^0 - f^{\mathrm{LP}}) \\ &\geq f^{\mathrm{LP}} + (\boldsymbol{c}^\top \mathbf{x}^0 - f^{\mathrm{LP}}) = \boldsymbol{c}^\top \mathbf{x}^0 \geq f^{\mathrm{IP}}, \end{aligned} \tag{11}$$

where the second inequality holds due to the definition of λ^{LP}. Thus the inequalities (10) and (11) by letting λ^* be λ^{LR} and $\bar{\lambda}$ imply that

$$d(\lambda^*,\rho^*) = \min_{\mathbf{x}\in\mathcal{X}}\ L(\mathbf{x},\lambda^*,\rho^*) \geq f^{\mathrm{IP}}.$$

This together with (9) and (8) yields that

$$f^{\mathrm{LD}}_{\rho^*} = d(\lambda^*,\rho^*) = f^{\mathrm{IP}}.$$

Hence we complete the proof. □

One can observe from the proof that there exists a finite value ρ^* such that for all $\rho \geq \rho^*$, the strong duality still holds. Therefore, given a sufficiently large penalty parameter ρ, we can achieve a satisfactory feasibility of (1). Specifically, as ρ increases beyond ρ^*, the augmented Lagrangian method penalizes constraint violations (1b) more heavily, thereby making the solution closer to the feasible region of the problem. The strong duality allows us to obtain a globally optimal solution to the problem (1) by solving the augmented Lagrangian dual problem (5).

To utilize the concave structure of the dual function $d(\lambda,\rho)$, we apply the projected subgradient method for solving (5) since the dual function is not differentiable. We first give the definition of subgradient and subdifferential.

Definition 2.2. Let $h:\mathbb{R}^m\to\mathbb{R}$ be a convex function. The vector $s\in\mathbb{R}^m$ is called a subgradient of h at $\bar{x}\in\mathbb{R}^m$ if

$$h(x)-h(\bar{x}) \geq s^\top(x-\bar{x}),\ \ \forall x\in\mathbb{R}^m.$$

The subdifferential of h at $\bar{x}$ is the set of all subgradients of h at $\bar{x}$ which is given by

$$\partial h(x) = \{s\in\mathbb{R}^m : h(x)-h(\bar{x}) \geq s^\top(x-\bar{x}),\ \ \forall x\in\mathbb{R}^m\}.$$

Since $d(\lambda,\rho)$ is concave, we adjust Definition 2.2 to correspond to the set $-\partial(-d(\lambda,\rho))$, allowing us to apply properties of the subdifferential of a convex function to a concave function.

Proposition 2.1. Consider the dual function $d(\lambda,\rho)$: $\mathbb{R}^{m+1}\to\mathbb{R}$. Then the subdifferentials of d at λ and ρ satisfy

$$\boldsymbol{A}\mathbf{x}-\boldsymbol{b}\in\partial_\lambda d(\lambda,\rho),\quad \frac{1}{2}\|(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+\|^2\in\partial_\rho d(\lambda,\rho),\ \ (12)$$

where $\mathbf{x}$ is a solution of (4) with input (λ,ρ).

Proof Let $\mathbf{x}$ be a solution of (4) with input (λ,ρ). Then for any pair $(\hat{\lambda},\hat{\rho})\in\mathbb{R}^m_+\times\mathbb{R}_+$, it holds that

$$\begin{aligned} d(\hat{\lambda},\hat{\rho}) &\leq \boldsymbol{c}^\top\mathbf{x}+\hat{\lambda}^\top(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})+\tfrac{\hat{\rho}}{2}\|(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+\|^2\\ &= \boldsymbol{c}^\top\mathbf{x}+\lambda^\top(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})+\tfrac{\rho}{2}\|(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+\|^2\\ &\quad+(\hat{\lambda}-\lambda)^\top(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})+\tfrac{\hat{\rho}-\rho}{2}\|(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+\|^2\\ &= d(\lambda,\rho)+(\hat{\lambda}-\lambda)^\top(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})+\tfrac{\hat{\rho}-\rho}{2}\|(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+\|^2,\end{aligned}$$

where the last equality holds due to the optimality of $\mathbf{x}$. Then by the definition of subgradient and subdifferential, we arrive at (12). □

3 Augmented Lagrangian Method

In this section, we introduce an augmented Lagrangian method framework. This method is composed of two steps in solving the augmented Lagrangian dual problem (5). We first use the primal information $\mathbf{x}$ to construct the subgradient of d at (λ,ρ). Since λ and ρ satisfy $\lambda\geq 0$ and $\rho>0$, then we apply the projected subgradient method to update parameters λ and ρ. Starting from $\lambda^0\in\mathbb{R}^m_+$ and $\rho^0>0$, we can update λ and ρ at $(k+1)$-th iteration by

$$\lambda^{k+1} = \left(\lambda^k+\alpha^k(\boldsymbol{A}\mathbf{x}^{k+1}-\boldsymbol{b})\right)_+, \qquad (13a)$$

$$\begin{aligned}\rho^{k+1} &= \left(\rho^k+\frac{\alpha^k}{2}\left\|(\boldsymbol{A}\mathbf{x}^{k+1}-\boldsymbol{b})_+\right\|^2\right)_+\\ &= \rho^k+\frac{\alpha^k}{2}\left\|(\boldsymbol{A}\mathbf{x}^{k+1}-\boldsymbol{b})_+\right\|^2, \qquad (13b)\end{aligned}$$

where $\mathbf{x}^{k+1}$ is an optimal solution of the augmented Lagrangian relaxation problem (4) at λ^k and ρ^k, that is,

$$\mathbf{x}^{k+1}\in\arg\min_{\mathbf{x}\in\mathcal{X}}\ L(\mathbf{x},\lambda^k,\rho^k). \qquad (14)$$

Therefore, the iterative processes (14), (13a) and (13b) consisting of primal and dual variables make up the ALM framework.

Solving the $\mathbf{x}$-subproblem (14) is an important step in ALM. While the subproblem (14) has no closed form solution in general, we can apply an iterative method to solve it exactly or inexactly. Consequently, the ALM for solving the problem (1) consists of outer and inner iterations. The subscript k is used to denote the outer iteration number, and the subscript t is used to denote the inner iteration number.

3.1 A BCD method for subproblem (14)

In this subsection, we present a BCD method for solving the $\mathbf{x}$-subproblem (14) in the ALM framework. The BCD method minimizes the function L by iterating cyclically in order $\mathbf{x}_1,\ldots,\mathbf{x}_p$, fixing the previous iteration during each iteration. Denote $\mathbf{x}^t=(\mathbf{x}^t_1;\mathbf{x}^t_2;\ldots;\mathbf{x}^t_p)$ where $\mathbf{x}^t_j$ is the value of $\mathbf{x}_j$ at its t-th update. Let

$$L^t_j(\mathbf{x}_j,\lambda,\rho) = L(\mathbf{x}^{t+1}_1,\ldots,\mathbf{x}^{t+1}_{j-1},\mathbf{x}_j,\mathbf{x}^t_{j+1},\ldots,\mathbf{x}^t_p,\lambda,\rho),$$

and

$$\mathbf{x}^t(j) = \left(\mathbf{x}^{t+1}_1,\mathbf{x}^{t+1}_2,\ldots,\mathbf{x}^{t+1}_{j-1},\mathbf{x}^t_j,\mathbf{x}^t_{j+1},\ldots,\mathbf{x}^t_p\right).$$

Therefore $\mathbf{x}^t(1)=(\mathbf{x}^t_1;\mathbf{x}^t_2;\ldots;\mathbf{x}^t_p)=\mathbf{x}^t$ and $\mathbf{x}^t(p+1)=(\mathbf{x}^{t+1}_1;\mathbf{x}^{t+1}_2;\ldots;\mathbf{x}^{t+1}_p)=\mathbf{x}^{t+1}$. Given fixed parameters $\lambda\geq 0$ and $\rho>0$, we can also calculate the gradient of $L(\mathbf{x}^t(j),\lambda,\rho)$ at $\mathbf{x}_j$ by

$$\begin{aligned} g_j(\mathbf{x}^t) &:= \nabla_{\mathbf{x}_j}L(\mathbf{x}^t(j),\lambda,\rho)\\ &= \boldsymbol{c}_j+\boldsymbol{A}^\top_j\lambda+\rho\boldsymbol{A}^\top_j\left(\boldsymbol{A}\mathbf{x}^t(j)-\boldsymbol{b}\right)_+.\end{aligned}$$

At each step, we consider two types of updates for every $\mathbf{x}_j\in\mathcal{X}_j$:

Classical: $\mathbf{x}^{t+1}_j\in\arg\min_{\mathbf{x}_j\in\mathcal{X}_j}\ L^t_j(\mathbf{x}_j,\lambda,\rho), \qquad (15a)$

Proximal linear:

$$\mathbf{x}^{t+1}_j\in\arg\min_{\mathbf{x}_j\in\mathcal{X}_j}\left\{\langle\mathbf{x}_j-\mathbf{x}^t_j,g_j(\mathbf{x}^t)\rangle+\frac{1}{2\tau}\|\mathbf{x}_j-\mathbf{x}^t_j\|^2\right\}, \quad (15b)$$

where $\tau > 0$ is a step size. In fact, if one defines the projection operator by

$$P_{\mathcal{X}}(v) \in \arg\min\{\|u - v\| : u \in \mathcal{X}\},$$

then we obtain the following equivalent form of (15b):

$$\mathbf{x}_j^{t+1} = P_{\mathcal{X}_j}\left(\mathbf{x}_j^t - \tau g_j(\mathbf{x}^t)\right).$$

In general, the classical subproblem (15a) is fundamentally harder to solve because of the quadratic term in the objective function. However, we derive a simplified form of this subproblem under certain conditions, which will be discussed in Subsection 3.1.2. By contrast, the prox-linear subproblem (15b) is relatively easy to solve because the objective function is linear with respect to $\mathbf{x}_j$. Now we summarize the ALM for solving (1) in Algorithm 1, which allows each $\mathbf{x}_j$ to be updated by (15a) or (15b). We assume that each block j is updated by the same scheme in (15a) and (15b) for all iteration t.

Algorithm 1: ALM with BCD

```
Input: Initial point $\mathbf{x}^0, \lambda^0, \rho^0$.
Output: A feasible solution $\mathbf{x}^{k+1}$.
1 for $k = 0, 1, ..., k_{\max}$ do
2     for $t = 0, 1, ..., t_{\max}$ do
3         for $j = 1, 2, ..., p$ do
4             Compute $\mathbf{x}_j^{(t+1)}$ by (15a) or (15b);
5         if $\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)}$, then let $\mathbf{x}^{k+1} = \mathbf{x}^{(t+1)}$ and
           break;
6     if $\|(\boldsymbol{A}\mathbf{x}^{k+1} - \boldsymbol{b})_+\|_2 = 0$, then Teminate;
7     Update the Lagrangian multipliers $\lambda^{k+1}$ by (13a)
      and the penalty coefficient $\rho^{k+1}$ by (13b).
```

Subsequently, we show more detailed information about these two updates (15a) and (15b).

3.1.1 *Proximal linear update of BCD*

Before considering the proximal linear subproblem (15b), we first give a definition of a linear operator, which is essential in the BCD method.

Definition 3.1. The linear operator associated to a vector $v \in \mathbb{R}^n$ is defined by

$$\mathcal{T}_\Omega(v) = \arg\min_{\mathbf{u}\in\Omega} v^\top \mathbf{u},$$

where Ω is a nonempty and closed set.

Benefiting from the good characteristics of $\mathbf{x}$ being a binary variable, we have

$$\|\mathbf{x}_j\|^2 = \mathbf{1}^\top \mathbf{x}_j, \quad \text{for } \forall j \in \mathbb{N}_p, \tag{16}$$

then the objective function in (15b) is linear. Therefore, we can also rewrite (15b) as

$$\begin{aligned}\mathbf{x}_j^{t+1} &\in \arg\min_{\mathbf{x}_j\in\mathcal{X}_j} \left\{\mathbf{x}_j^\top g_j(\mathbf{x}^t) + \frac{1}{2\tau}\mathbf{1}^\top\mathbf{x}_j - \frac{1}{\tau}\mathbf{x}_j^\top \mathbf{x}_j^t\right\} \\ &= \mathcal{T}_{\mathcal{X}_j}\left(\tau g_j(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}_j^t\right).\end{aligned} \tag{17}$$

Due to the discrete property of the feasible set $\mathcal{X}_j$, it is crucial to choose the step size τ appropriately. If τ is too large, the BCD method will not converge. If τ is too small, the BCD method will be stuck at some points. The reason why this happens will be explained in the convergence analysis.

3.1.2 *Classical update of BCD*

We start by giving the following assumption of model (1), which is very common in many applications such as train timetabling, vehicle routing, and allocation problems.

Assumption 3.1. The entries of the matrix $\boldsymbol{A}$ are either 1 or 0. The vector $\boldsymbol{b}$ equals $\mathbf{1}$.

Under this assumption, we consider the subproblem (15a). For each $j \in \mathbb{N}_p$, if the condition $\boldsymbol{A}_j\mathbf{x}_j^{t+1} \le \mathbf{1}$ always holds for each iteration of the BCD method, then

$$\begin{aligned}\mathbf{x}_j^{t+1} &\in \arg\min_{\mathbf{x}_j\in\mathcal{X}_j} L_j^t(\mathbf{x}_j, \lambda, \rho) \\ &= \mathcal{T}_{\mathcal{X}_j}\left(\boldsymbol{c}_j + \boldsymbol{A}_j^\top\lambda + \rho\boldsymbol{A}_j^\top\left(\sum_{l\neq j}^{p}\boldsymbol{A}_l\mathbf{x}_l^t(j) - \frac{\mathbf{1}}{2}\right)_+\right).\end{aligned} \tag{18}$$

To prove the above derivation, we introduce following two notations at the t-th update:

$$\begin{aligned}\mathcal{I} &:= \{i \in \mathbb{N}_m : \textstyle\sum_{l\neq j}^p \boldsymbol{A}_{i,l}\mathbf{x}_l^t(j) = 0\}, \\ \bar{\mathcal{I}} &:= \{i \in \mathbb{N}_m : \textstyle\sum_{l\neq j}^p \boldsymbol{A}_{i,l}\mathbf{x}_l^t(j) \ge 1\},\end{aligned} \tag{19}$$

where $\mathbf{x}_l^t(j) = \begin{cases}\mathbf{x}_l^{t+1}, & \text{if } l < j, \\ \mathbf{x}_l^t, & \text{if } l > j.\end{cases}$ Obviously, $\mathcal{I}\cap\bar{\mathcal{I}} = \emptyset$ and $\mathcal{I}\cup\bar{\mathcal{I}} = \mathbb{N}_m$. Therefore, we have

$$\sum\nolimits_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\bar{\mathcal{I}}}}{2} = \left(\sum\nolimits_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\bar{\mathcal{I}}}}{2}\right)_+, \tag{20}$$

$$\sum\nolimits_{l\neq j}^p \boldsymbol{A}_{\mathcal{I},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\mathcal{I}}}{2} = 0, \tag{21}$$

and

$$\begin{aligned}&\left(\boldsymbol{A}_{\mathcal{I},j}\mathbf{x}_j + \sum_{l\neq j}^p \boldsymbol{A}_{\mathcal{I},l}\mathbf{x}_l^t(j) - \mathbf{1}_{\mathcal{I}}\right)_+ = (\boldsymbol{A}_{\mathcal{I},j}\mathbf{x}_j - \mathbf{1}_{\mathcal{I}})_+ = \mathbf{0}, \\ &\boldsymbol{A}_{\bar{\mathcal{I}},j}\mathbf{x}_j + \textstyle\sum_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \mathbf{1}_{\bar{\mathcal{I}}} \ge \mathbf{0}.\end{aligned} \tag{22}$$

Since the element in $\boldsymbol{A}_j\mathbf{x}_j$ is either 1 or 0, we have

$$\|\boldsymbol{A}_j\mathbf{x}_j\|^2 = \mathbf{1}^\top(\boldsymbol{A}_j\mathbf{x}_j), \quad \forall\, j \in \mathbb{N}_p. \tag{23}$$

Denote $C = \left\|\sum_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \mathbf{1}_{\bar{\mathcal{I}}}\right\|^2$. Then

$$\begin{aligned}&\left\|\left(\boldsymbol{A}_j\mathbf{x}_j + \textstyle\sum_{l\neq j}^p \boldsymbol{A}_l\mathbf{x}_l^t(j) - \mathbf{1}\right)_+\right\|^2 \\ \overset{(22)}{=}\ & \|\boldsymbol{A}_{\bar{\mathcal{I}},j}\mathbf{x}_j\|^2 + 2(\boldsymbol{A}_{\bar{\mathcal{I}},j}\mathbf{x}_j)^\top\left(\textstyle\sum_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \mathbf{1}_{\bar{\mathcal{I}}}\right) + C \\ \overset{(23)}{=}\ & 2(\boldsymbol{A}_{\bar{\mathcal{I}},j}\mathbf{x}_j)^\top\left(\textstyle\sum_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\bar{\mathcal{I}}}}{2}\right) + C \\ \overset{(20)}{=}\ & 2(\boldsymbol{A}_{\bar{\mathcal{I}},j}\mathbf{x}_j)^\top\left(\textstyle\sum_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\bar{\mathcal{I}}}}{2}\right)_+ + C \\ \overset{(21)}{=}\ & 2(\boldsymbol{A}_{\bar{\mathcal{I}},j}\mathbf{x}_j)^\top\left(\textstyle\sum_{l\neq j}^p \boldsymbol{A}_{\bar{\mathcal{I}},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\bar{\mathcal{I}}}}{2}\right)_+ \\ & + 2(\boldsymbol{A}_{\mathcal{I},j}\mathbf{x}_j)^\top\left(\textstyle\sum_{l\neq j}^p \boldsymbol{A}_{\mathcal{I},l}\mathbf{x}_l^t(j) - \frac{\mathbf{1}_{\mathcal{I}}}{2}\right)_+ + C \\ =\ & 2(\boldsymbol{A}_j\mathbf{x}_j)^\top\left(\textstyle\sum_{l\neq j}^p \boldsymbol{A}_l\mathbf{x}_l^t(j) - \frac{\mathbf{1}}{2}\right)_+ + C.\end{aligned}$$

Then the iterative scheme for $\mathbf{x}$-update is given by

$$\begin{aligned}
\mathbf{x}_j^{t+1} &= \arg\min_{\mathbf{x}_j\in\mathcal{X}_j}\ L_j^t(\mathbf{x}_j,\lambda,\rho)\\
&= \arg\min_{\mathbf{x}_j\in\mathcal{X}_j}\ \Big\{\boldsymbol{c}_j^\top\mathbf{x}_j+\lambda^\top(\boldsymbol{A}_j\mathbf{x}_j-\mathbf{1})\\
&\quad+\tfrac{\rho}{2}\Big\|\Big(\boldsymbol{A}_j\mathbf{x}_j+\textstyle\sum_{l\neq j}^p\boldsymbol{A}_l\mathbf{x}_l^t(j)-\mathbf{1}\Big)_+\Big\|^2\Big\}\\
&\overset{(24)}{=} \arg\min_{\mathbf{x}_j\in\mathcal{X}_j}\Big\{\boldsymbol{c}_j^\top\mathbf{x}_j+\lambda^\top(\boldsymbol{A}_j\mathbf{x}_j-\mathbf{1})\\
&\quad+\rho(\boldsymbol{A}_j\mathbf{x}_j)^\top\Big(\textstyle\sum_{l\neq j}^p\boldsymbol{A}_l\mathbf{x}_l^t(j)-\frac{1}{2}\Big)_+\Big\}\\
&= \mathcal{T}_{\mathcal{X}_j}\Big(\boldsymbol{c}_j+\boldsymbol{A}_j^\top\lambda+\rho\boldsymbol{A}_j^\top\Big(\textstyle\sum_{l\neq j}^p\boldsymbol{A}_l\mathbf{x}_l^t(j)-\frac{1}{2}\Big)_+\Big).
\end{aligned}$$

We can observe that the condition $\boldsymbol{A}\mathbf{x}\le\mathbf{1}$ induces the decomposition of minimizing the augmented Lagrangian function (3) into a set of subproblems with a linear objective function, which makes (15a) easier to solve. Therefore, the key point in the derivation of the linearization process is utilizing the fact that the entries in $\boldsymbol{A}_j\mathbf{x}_j$ are either 1 or 0. We next verify this condition is always true in each iteration of the BCD method under the following assumption.

Assumption 3.2. Denote by $T_{\boldsymbol{A}_j}$ the indices of columns of the matrix $\boldsymbol{A}_j$ containing only zeros and c_{j_s} the s-th element of $\boldsymbol{c}_j$. At least one of the following holds.

(i) For all $j\in\mathbb{N}_p$, there exists $\mathbf{x}_j\in\mathcal{X}_j$ such that $\boldsymbol{A}_j\mathbf{x}_j\le\mathbf{1}$ and $\{s\in\mathbb{N}_{n_j}:c_{j_s}\neq0\}\subseteq T_{\boldsymbol{A}_j}$.

(ii) For all $j\in\mathbb{N}_p$, there is at most one nonzero element in the vector $\boldsymbol{c}_j$ and $\{\mathbf{x}_j\in\mathbb{R}^{n_j}:\boldsymbol{A}_j\mathbf{x}_j\le\mathbf{1}\}\subseteq\{\mathbf{x}_j\in\mathbb{R}^{n_j}:\mathbf{x}_j\in\mathcal{X}_j\}$.

Remark: (a) Assumption 3.2 (i) requires that if an element $\mathbf{x}_{j_s}$ satisfies $\mathbf{x}_{j_s}=1$, then at least one of c_{j_s} and the s-th column vector of $\boldsymbol{A}_j$ is zero. Assumption 3.2 (ii) requires that if the block constraint set $\mathcal{X}_j$ is large enough such that $\boldsymbol{A}_j\mathbf{x}_j\le\mathbf{1}$ holds, then the weight $\boldsymbol{c}_j$ corresponding to each block variable has only one nonzero element. (b) Assumption 3.2 usually holds when $\boldsymbol{A}$ and $\boldsymbol{c}$ are sparse. This is particularly true in the train timetabling problem with the objective of scheduling more trains, as will be shown in Section 5.

Lemma 3.1. Suppose Assumption 3.2 hold. From any starting point $\mathbf{x}^0$, the solution generated by the BCD method with the classical update at each iteration satisfies $\boldsymbol{A}_j\mathbf{x}_j^{t+1}\le\mathbf{1}$ for all $j\in\mathbb{N}_p$ and $t=0,1,2,\ldots$.

Proof Let $\mathbf{x}_j^{t+1}$ be the solution generated by the BCD method after t-th classical update. We argue by contradiction and suppose that there exists $i\in\mathbb{N}_m$ such that $\boldsymbol{A}_{i,j}\mathbf{x}_j^{t+1}>1$. Then for any $\bar{\mathbf{x}}_j\in\mathcal{X}_j$ satisfying $\boldsymbol{A}_j\bar{\mathbf{x}}_j\le1$, we have

$$L_j^t(\mathbf{x}_j^{t+1},\lambda,\rho)<L_j^t(\bar{\mathbf{x}}_j,\lambda,\rho),\quad\forall j\in\mathbb{N}_p,$$

which implies that for any $j\in\mathbb{N}_p$,

$$\left(\boldsymbol{c}_j+\boldsymbol{A}_j^\top\lambda+\rho\boldsymbol{A}_j^\top\left(\sum_{l\neq j}^p\boldsymbol{A}_l\mathbf{x}_l^t(j)-\frac{1}{2}\right)_+\right)^\top(\mathbf{x}_j^{t+1}-\bar{\mathbf{x}}_j)<0,$$

then

$$\begin{aligned}
&\boldsymbol{c}_j^\top(\mathbf{x}_j^{t+1}-\bar{\mathbf{x}}_j)+\lambda^\top(\boldsymbol{A}_j\mathbf{x}_j^{t+1}-\boldsymbol{A}_j\bar{\mathbf{x}}_j)+\\
&\rho\Big(\textstyle\sum_{l\neq j}^p\boldsymbol{A}_l\mathbf{x}_l^t(j)-\frac{1}{2}\Big)_+^\top(\boldsymbol{A}_j\mathbf{x}_j^{t+1}-\boldsymbol{A}_j\bar{\mathbf{x}}_j)<0.
\end{aligned}\tag{24}$$

We consider following two cases corresponding to Assumption 3.2:

Case 1: Assumption 3.2 (i) implies $c_{j_s}=0$ for any $s\in S_j:=\{s\in\mathbb{N}_{n_j}:\mathbf{x}_{j_s}^{t+1}=1,\boldsymbol{A}_{i,j_s}=1\}$. Taking $\bar{\mathbf{x}}_j=(\mathbf{x}_{j_1}^{t+1},\ldots,\mathbf{x}_{j_s}^{t+1}-1,\ldots,\mathbf{x}_{j_{n_j}}^{t+1})$ with $s\in S_j$ such that $\boldsymbol{A}_j\bar{\mathbf{x}}_j\le\mathbf{1}$ gives us

$$\boldsymbol{c}_j^\top(\mathbf{x}_j^{t+1}-\bar{\mathbf{x}}_j)=\sum\nolimits_s c_{j_s}=0\text{ and }\boldsymbol{A}_{i,j}\mathbf{x}_j^{t+1}-\boldsymbol{A}_{i,j}\bar{\mathbf{x}}_j\ge1.\tag{25}$$

This contradicts (24) since $\lambda\ge0,\rho>0$.

Case 2: Assumption 3.2 (ii) implies that there exists

$$\bar{\mathbf{x}}_j=(\mathbf{x}_{j_1}^{t+1},\ldots,\bar{\mathbf{x}}_{j_{s'}},\ldots,\mathbf{x}_{j_s}^{t+1}-1,\ldots,\mathbf{x}_{j_{n_j}}^{t+1})\text{ with }\bar{\mathbf{x}}_{j_{s'}}=1\tag{26}$$

for any $s\in S_j$ and $s'\neq s$ such that $\boldsymbol{A}_j\bar{\mathbf{x}}_j\le\mathbf{1}$. If $c_{j_{s'}}\ge0$, we can apply the same procedure in Case 1 and arrive at the contradiction. If $c_{j_{s'}}<0$, we take $\bar{\mathbf{x}}_j$ in (26), then $\mathbf{x}_{j_{s'}}^{t+1}-\bar{\mathbf{x}}_{j_{s'}}\le0$. Hence,

$$\boldsymbol{c}_j^\top(\mathbf{x}_j^{t+1}-\bar{\mathbf{x}}_j)=c_{j_{s'}}(\mathbf{x}_{j_{s'}}^{t+1}-\bar{\mathbf{x}}_{j_{s'}})\ge0,\ \ \boldsymbol{A}_{i,j}\mathbf{x}_j^{t+1}-\boldsymbol{A}_{i,j}\bar{\mathbf{x}}_j\ge1,$$

which is a contradiction that completes the proof. □

Overall, we can reformulate the subproblem (15a) into (18) based on the special structure of the model (1) under certain conditions, and thus make it easier to solve. It is worth noting that if Assumption 3.2 is not satisfied, we can consider (18) as a linear approximation of (15a). This linearization technique is widely used, including in works [20] and [21]. However, they lacked theoretical guarantees. Our main result shows that under specified assumptions, using the update (18) in the ALM-C method is equivalent to solving (15a) exactly, theoretically ensuring the effectiveness of our method.

3.2 Finding a good feasible solution by set packing

Since the BCD method may not always yield a feasible solution [35], we can adopt several refinement strategies that are very useful to find a feasible solution to the problem (1) in practice. Very often, (1) represents a problem where we optimize under limited resources, and this provides us a view of set packing problems. Since we produce candidate solutions all along, a natural idea is to utilize past iterates to construct a feasible solution. A simple strategy could be constructing a solution pool for each j that includes the past BCD iterates $V_j^k=\{\mathbf{x}_j^1,\mathbf{x}_j^2,\ldots,\mathbf{x}_j^k\},j=1,\ldots,p$. Intuitively, the solution pool may include feasible or "nice" solutions in some sense.

To illustrate the above approach, we first introduce the iterative sequential technique.

3.2.1 A sweeping technique

The most simple technique is probably to select a subset of blocks, one by one, until the infeasibility is detected. We present this sequential method in Algorithm 2, which can be understood by simply sweeping the blocks and selecting a candidate solution from V_j^k if feasibility is still guaranteed, otherwise simply skip the current block and continue.

Algorithm 2: A sweeping technique

```
Input: The set of past BCD solutions $V_j^k, j = 1, ..., p$.
Output: A feasible solution $\hat{\mathbf{x}}^k$
1 while the termination is not satisfied do
2     for $j = 1, 2, ..., p$ do
3         Select $\mathbf{v}_j \in V_j^k$ ;
4         if $\boldsymbol{A}_j \mathbf{v}_j + \sum_{l=1}^{j-1} \boldsymbol{A}_l \hat{\mathbf{x}}_l^k - \boldsymbol{b} \leq 0$ then
5             let $\hat{\mathbf{x}}_j^k = \mathbf{v}_j$,
6         else
7             let $\hat{\mathbf{x}}_j^k = 0$;
```

3.2.2 A packing technique

Let us further explore the idea of selecting solutions in a systematic way. Formally, for each block $j \in \mathbb{N}_p$, we introduce a set of binary variables μ_j that defines the current selection:

$$\hat{\mathbf{x}}_j^k = X_j^k \mu_j, \ \mu_j \in \{0, 1\}^k,$$

where $X_j^k = [\mathbf{x}_j^1; \mathbf{x}_j^2; ...; \mathbf{x}_j^k]$. We consider the following problem:

$$\min_{\mu_j \in \{0,1\}^k} \quad \sum\nolimits_{j=1}^{p} c_j^\top X_j^k \mu_j \tag{27a}$$

$$\text{s.t.} \quad \mu_j^\top \mathbf{1} \leq 1, \ \forall j = 1, ..., p, \tag{27b}$$

$$\sum\nolimits_{j=1}^{p} \boldsymbol{A}_j X_j^k \mu_j \leq \boldsymbol{b}. \tag{27c}$$

In view of (27b), one may recognize the above problem as a restricted master problem appearing in column generation algorithms where (27c) stands for a set of knapsack constraints.

Specifically, the coupling constraints (27c) in our model are cliques, representing complete subgraphs where each pair of distinct nodes is connected. This allows us to reformulate the model as a maximum independent set problem. Therefore, we can find a feasible solution $\mathbf{x}^*$ which satisfies $\boldsymbol{A}\mathbf{x}^* \leq \boldsymbol{b}$ by solving the problem (27). Since this problem is still hard, we only solve the relaxation problem of the maximal independent set. Now we go into details. Since the knapsack (27c) means the candidate solutions may conflict, we can construct a conflict graph $F = (V, E)$. In this graph, V represents the set of nodes, where each node corresponds to a solution generated as the algorithm proceeds, and E is the set of edges that connect two conflicting solutions, meaning they violate the coupling constraints. In this view, we only have to maintain the graph F and find a maximal independent set $\mathcal{K}$. Therefore, the output feasible point $\mathbf{x}_j^*$ corresponds to $\mathbf{v}_j \in \mathcal{K}$ for each $j \in \mathbb{N}_p$. If $\mathbf{v}_j \notin \mathcal{K}$, then $\mathbf{x}_j^* = 0$. We summarize the maximal independent set technique in Algorithm 3. We also note that Algorithm 2 can be seen as a special case of Algorithm 3.

Based on the above-mentioned techniques, we improve the ALM and propose a customized ALM in Algorithm 4.

Although Algorithms 2 and 3 can help us find a feasible solution to problem (1), the quality of output by them remains unjustified. To evaluate and improve the quality of the solution, the simple way is to estimate the upper and lower bounds of the objective function value, and then

Algorithm 3: A packing (maximal independent set) technique

```
Input: The BCD solution $\mathbf{x}^k$, last conflict graph
       $F^{k-1} = (V^{k-1}, E^{k-1})$
Output: A feasible solution $\mathbf{x}_*^k$
1 Step 1: Conflict graph update
2 for $j = 1, 2, ..., p$ do
3     Collect new candidate paths $\tilde{V}_j^k$ for block $j$;
4     (1.1 node-update) $V^k \leftarrow V^{k-1} + \tilde{V}_j^k$;
5     (1.2 self-check) $E^k \leftarrow E^{k-1} + \{(p, p') \mid \forall p \neq p', p \in V_j^k, p' \in V_j^k\}$;
6     (1.3 edge-completion) $E^k \leftarrow E^k + \Big\{(p, p') \mid$
      if $p, p'$are compatible for $\forall p \in \tilde{V}_j^k, p' \in V^k \backslash V_j^k\Big\}$,
      here 'compatible' means that these two nodes
      (block variables) satisfy the binding constraints;
7 Step 2: Maximal independent set (MIS) for a
  feasible solution
8 Select a candidate solution set $K \subset V^k$;
9 for $v \in K$ do
10    Compute a maximal independent set $\mathcal{K}(v)$ with
      respect to $v$ in $O(|E^k|)$ iterations;
11 Compute $\mathbf{x}^* = \arg\max_v \mathcal{K}(v)$.
```

Algorithm 4: A customized ALM

```
Input: $\mathbf{x}^0, \lambda^0, \rho^0 > 0$ and the best objective function
       value $f^* = +\infty$. Set $k = 0$.
Output: A local (global) optimal solution $\mathbf{x}^*$.
1 while the termination is not satisfied do
2     Step 2: Construct solution using Alg. 1
3     Update the BCD solution $\mathbf{x}^{k+1}$ by the procedure
      in lines 2-5 of Alg. 1;
4     Step 3: Generate a feasible solution
5     if the BCD solution is not feasible then
6         transform the BCD solution to a feasible
          solution $\bar{\mathbf{x}}^{k+1}$ by calling a refinement
          method in Alg. 2 or 3;
7     else
8         let $\bar{\mathbf{x}}^{k+1} = \mathbf{x}^{k+1}$;
9     Step 4: Update the best solution
10    if $\mathbf{c}^\top \bar{\mathbf{x}}^{k+1} \leq f^*$ then
11        Set $f^* = \mathbf{c}^\top \bar{\mathbf{x}}^{k+1}$ and $\mathbf{x}^* = \bar{\mathbf{x}}^{k+1}$;
12    Update the Lagrangian multipliers $\lambda^{k+1}$ by (13a)
      and the penalty coefficient $\rho^{k+1}$ by (13b). Let
      $k = k + 1$.
```

calculate the gap between them. The smaller the gap, the better the current solution. Obviously, our method can provide an upper bound of the objective function value. As for the generation of the lower bound, we can use the following method. (i) LP relaxation: by directly relaxing the binary integer variables to $[0,1]$ continuous variables, we solve a linear programming problem exactly to obtain the lower bound of the objective function value. (ii) LR relaxation: since the Lagrangian dual problem (2) is separable, we can solve the decomposed subproblems exactly to obtain the lower bound of the objective function value.

In general, the Lagrangian dual bound is at least as tight as the linear programming bound obtained from the usual linear programming relaxation [14]. Note that the relaxation must be solved to optimality to yield a valid bound. Therefore, we can use a combination of LR method and Alg. 4. To be specific, LR aims at generating the lower bound of the objective function value in (1), the solution is usually infeasible. Steps 2 and 3 in Alg. 4 are used for generating feasible solutions of (1). The smaller the gap between the lower bound and the upper bound, the closer the feasible solution is to the global optimal solution of the problem. It can be seen from the iterative procedure that after many iterations, this algorithm can generate many feasible solutions, and the lower bound of the model is constantly improving. Finally, we select the best solution. The combination of these two methods takes advantage of the ALM and the LR method, it not only finds a good feasible solution but also evaluates the quality of the solution.

Compared with the ADMM-based method in [21], we utilize BCD method to perform multiple iterations to solve the subproblem (14) until the solutions remain unchanged, which can improve the solution accuracy of the subproblem, thereby reducing the total number of iterations. Moreover, we adopt different refinement techniques to further enhance the solution quality.

4 Convergence Analysis

In this section, we present the convergence analysis of the block coordinate descent method for solving the augmented Lagrangian relaxation problem (4) and the augmented Lagrangian method for solving the dual problem (5). Unless otherwise stated, the convergence results presented in this section do not rely on Assumptions 3.1 and 3.2.

4.1 Convergence of BCD

We begin this section with the property of augmented Lagrangian function (3) that is fundamental in the convergence analyis.

Proposition 4.1. The gradient of $L(\mathbf{x},\lambda,\rho)$ at $\mathbf{x}$ is Lipschitz continuous with constant κ on $\mathcal{X}$, namely,

$$\|\nabla L(\mathbf{x},\lambda,\rho)-\nabla L(\bar{\mathbf{x}},\lambda,\rho)\|\leq\kappa\|\mathbf{x}-\bar{\mathbf{x}}\|$$

for all $\mathbf{x},\bar{\mathbf{x}}\in\mathcal{X}$, where $\kappa=\rho\|\boldsymbol{A}\|_2^2$. Furthermore,

$$L(\bar{\mathbf{x}},\lambda,\rho)\leq L(\mathbf{x},\lambda,\rho)+\langle\bar{\mathbf{x}}-\mathbf{x},\nabla L(\mathbf{x},\lambda,\rho)\rangle+\frac{\kappa}{2}\|\bar{\mathbf{x}}-\mathbf{x}\|^2 \tag{28}$$

for all $\mathbf{x},\bar{\mathbf{x}}\in\mathcal{X}$.

Proof For any $\mathbf{x},\bar{\mathbf{x}}\in\mathcal{X}$,

$$\begin{aligned}&\|\nabla L(\mathbf{x},\lambda,\rho)-\nabla L(\bar{\mathbf{x}},\lambda,\rho)\|\\=&\left\|\rho\boldsymbol{A}^\top(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+-\rho\boldsymbol{A}^\top(\boldsymbol{A}\bar{\mathbf{x}}-\boldsymbol{b})_+\right\|\\\leq&\rho\|\boldsymbol{A}\|_2\left\|(\boldsymbol{A}\mathbf{x}-\boldsymbol{b})_+-(\boldsymbol{A}\bar{\mathbf{x}}-\boldsymbol{b})_+\right\|\\\leq&\rho\|\boldsymbol{A}\|_2\left\|\boldsymbol{A}\mathbf{x}-\boldsymbol{A}\bar{\mathbf{x}}\right\|\leq\rho\|\boldsymbol{A}\|_2^2\left\|\mathbf{x}-\bar{\mathbf{x}}\right\|.\end{aligned}$$

Then we have

$$\begin{aligned}&\langle\nabla L(\mathbf{x},\lambda,\rho)-\nabla L(\bar{\mathbf{x}},\lambda,\rho),\mathbf{x}-\bar{\mathbf{x}}\rangle\\\leq&\|\nabla L(\mathbf{x},\lambda,\rho)-\nabla L(\bar{\mathbf{x}},\lambda,\rho)\|\|\mathbf{x}-\bar{\mathbf{x}}\|\leq\kappa\|\mathbf{x}-\bar{\mathbf{x}}\|^2.\end{aligned}$$

It follows from the convexity of the function $f(\mathbf{x})=\frac{\kappa}{2}\|\mathbf{x}\|^2-L(\mathbf{x},\lambda,\rho)$ that

$$f(\bar{\mathbf{x}})\geq f(\mathbf{x})+\nabla f(\mathbf{x})^\top(\mathbf{x}-\bar{\mathbf{x}}),$$

which implies the estimate (28). □

Considering two different updates in the BCD method, we first summarize the convergence property of the BCD method for solving (15a), which has been presented in [36]. Before that, we give the definition of the blockwise optimal solution, which is also called coordinatewise minimum point in [37]. Given parameters λ and ρ, a feasible solution $\mathbf{x}^*$ is called a blockwise optimal solution of the problem (4) if for each $j\in\mathbb{N}_p$, we have for all $\mathbf{x}=(\mathbf{x}_1^*,\ldots,\mathbf{x}_{j-1}^*,\mathbf{x}_j,\mathbf{x}_{j+1}^*,\ldots,\mathbf{x}_p^*)\in\mathcal{X}$, $L(\mathbf{x}^*,\lambda,\rho)\leq L(\mathbf{x},\lambda,\rho)$.

Lemma 4.1. Suppose Assumptions 3.1 and 3.2 hold. If the starting point satisfies $\mathbf{x}^0\in\mathcal{X}$, then the BCD method for solving (15a) is always executable and terminates after a finite number of iterations with a blockwise optimal solution of the problem (4).

Proof If Assumptions 3.1 and 3.2 hold, Lemma 3.1 tells us that the subproblem (15a) can be solved exactly. (i) Since the constraint set of each subproblem (15a) is bounded, then all subproblems have an optimal solution. Therefore, the BCD method for solving (15a) is executable. (ii) The result in (i) illustrates that the sequence $\{\mathbf{x}^t\}$ generated by the BCD method exists. Thus the sequence of objective function values $\{L(\mathbf{x}^t,\lambda,\rho)\}$ can only take finitely many different values. This together with the monotonically decreasing property of the function $L(\mathbf{x},\lambda,\rho)$ yields that $L(\mathbf{x}^t,\lambda,\rho)$ must become a constant. Then the BCD method is executable. (iii) By the definition of the blockwise optimal solution and the fact that the subproblem (15a) can be solved exactly, we arrive at the conclusion. □

A similar conclusion can be found in [36], but it does not clarify how to solve the subproblem. Note that each (global) optimal solution of the model (4) is blockwise optimal, but not vice versa. Subsequently, we analyze the convergence of (15b). The following lemma ensures decreasing the function value of L after each iteration if $\mathbf{x}^{t+1}\neq\mathbf{x}^t$ and the step size is chosen properly.

Lemma 4.2. Let $\{\mathbf{x}^t\}_{t\in\mathbb{N}}$ be a sequence generated by (15b), we obtain

$$(\frac{1}{2\tau}-\frac{\kappa}{2})\|\mathbf{x}^{t+1}-\mathbf{x}^t\|^2\leq L(\mathbf{x}^t,\lambda,\rho)-L(\mathbf{x}^{t+1},\lambda,\rho). \tag{29}$$

Proof Since

$$\mathbf{x}_j^{t+1}=\mathcal{T}_{\mathcal{X}_j}\left(\tau g_j(\mathbf{x}^t)^\top\mathbf{x}_j+\frac{\mathbf{1}}{2}-\mathbf{x}_j^t\right)$$

and $\mathbf{x}_j^t \in \mathcal{X}_j$ for all $j \in \{1,2,...,p\}$, we have

$$\left(\tau g_j(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}_j^t\right)^\top \mathbf{x}_j^{t+1} \leq \left(\tau g_j(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}_j^t\right)^\top \mathbf{x}_j^t,$$

which implies that

$$2\tau\langle \mathbf{x}_j^{t+1} - \mathbf{x}_j^t, g_j(\mathbf{x}^t)\rangle + \|\mathbf{x}_j^{t+1} - \mathbf{x}_j^t\|^2 \leq 0. \quad (30)$$

Proposition 4.1 tells us that

$$\begin{aligned} &L(\mathbf{x}^{t+1},\lambda,\rho) - L(\mathbf{x}^t,\lambda,\rho) \\ \leq &\sum\nolimits_{j=1}^p \langle \mathbf{x}_j^{t+1} - \mathbf{x}_j^t, g_j(\mathbf{x}^t)\rangle + \frac{\kappa}{2}\sum\nolimits_{j=1}^p \|\mathbf{x}_j^{t+1} - \mathbf{x}_j^t\|^2. \quad (31)\end{aligned}$$

Combining inequalities (30) and (31) yields (29). The proof is completed. □

This lemma tells us that a small step size satisfying $\tau < 1/\kappa$ leads to a decrease in the function value L when $\mathbf{x}^{t+1} \neq \mathbf{x}^t$. However, the following lemma states that when the step size is too small, the iteration returns the same result. We let $g(\mathbf{x})$ denote the gradient of $L(\mathbf{x},\lambda,\rho)$ at $\mathbf{x}$.

Lemma 4.3. If the step size satisfies $0 < \tau < \frac{1}{2\|g(\mathbf{x}^t)\|}$ when $g(\mathbf{x}^t) \neq \mathbf{0}$, then it holds that

$$\mathbf{x}^t = \mathcal{T}_{\mathcal{X}}\left(\tau g(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}^t\right).$$

Proof If $0 < \tau < \frac{1}{2\|g(\mathbf{x}^t)\|}$, for all $\mathbf{x}^t \neq \mathbf{x} \in \mathcal{X}$, we have

$$-2\tau g(\mathbf{x}^t)^\top(\mathbf{x}^t - \mathbf{x}) \leq 2\tau\|g(\mathbf{x}^t)\|\|\mathbf{x}^t - \mathbf{x}\| \leq \|\mathbf{x}^t - \mathbf{x}\| < \|\mathbf{x}^t - \mathbf{x}\|^2,$$

where the last inequality holds due to $\mathbf{x}^t, \mathbf{x} \in \{0,1\}^n$. It yields that

$$\left(\tau g(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}^t\right)^\top \mathbf{x}^t \leq \left(\tau g(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}^t\right)^\top \mathbf{x}.$$

By the definition of the operator $\mathcal{T}_{\mathcal{X}}(\cdot)$, we arrive at the conclusion. □

Based on Lemma 4.3, the implementation of the BCD method heavily relies on the choice of step size τ. Hence, we give a definition of a τ-stationary point.

Definition 4.1. For the AL relaxation problem (4), if a point $\mathbf{x}^*$ satisfies

$$\mathbf{x}^* = \mathcal{T}_{\mathcal{X}}\left(\tau g(\mathbf{x}^*) + \frac{\mathbf{1}}{2} - \mathbf{x}^*\right)$$

with $\tau > 0$, then it is called a τ-stationary point.

For the augmented Lagrangian relaxation problem (4), given parameters λ and ρ, we say that $\mathbf{x}^*$ is a δ-local minimizer if there is an integer $\delta > 0$ such that

$$L(\mathbf{x},\lambda,\rho) \geq L(\mathbf{x}^*,\lambda,\rho), \quad \text{for all } \mathbf{x} \in \mathcal{N}(\mathbf{x}^*,\delta) \cap \mathcal{X}.$$

Note that a n-local minimizer is a global minimizer due to the fact $\|\mathbf{x} - \mathbf{x}^*\| \leq n$ for all $\mathbf{x}, \mathbf{x}^* \in \mathcal{X}$. The following important result reveals a relationship between a τ-stationary point and a global minimizer of the problem (4).

Theorem 4.1. We have the following relationships between the τ-stationary point and the local minimizer of the problem (4).

(i) If $\mathbf{x}^*$ is a local minimizer, then $\mathbf{x}^*$ is a τ-stationary point for any step size $0 < \tau < 1/\kappa$.

(ii) If $\mathbf{x}^*$ is a τ-stationary point with $\tau > \delta/2$, then $\mathbf{x}^*$ is a δ-local minimizer of the problem (4) under the assumption that the entries in $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \lambda$ and ρ are integral.

Proof (i) If $\mathbf{x}^*$ is a local minimizer, then for any $\mathbf{x} \in \mathcal{X}$ we have

$$\begin{aligned} L(\mathbf{x}^*,\lambda,\rho) \leq &L(\mathbf{x},\lambda,\rho) \\ \leq &L(\mathbf{x}^*,\lambda,\rho) + \langle \nabla L(\mathbf{x}^*,\lambda,\rho), \mathbf{x} - \mathbf{x}^*\rangle + \frac{\kappa}{2}\|\mathbf{x} - \mathbf{x}^*\|^2.\end{aligned}$$

Therefore,

$$\langle \nabla L(\mathbf{x}^*,\lambda,\rho), \mathbf{x} - \mathbf{x}^*\rangle \geq -\frac{\kappa}{2}\|\mathbf{x} - \mathbf{x}^*\|^2 \geq -\frac{1}{2\tau}\|\mathbf{x} - \mathbf{x}^*\|^2,$$

which implies that $\mathbf{x}^*$ is a τ-stationary point.

(ii) Since $\mathbf{x}^*$ is a stationary point, then for any $\mathbf{x} \in \mathcal{X}$ we have

$$\left(\tau g(\mathbf{x}^*) + \frac{\mathbf{1}}{2} - \mathbf{x}^*\right)^\top \mathbf{x}^* \leq \left(\tau g(\mathbf{x}^*) + \frac{\mathbf{1}}{2} - \mathbf{x}^*\right)^\top \mathbf{x}. \quad (32)$$

For any $\mathbf{x} \in \mathcal{X} \cap \mathcal{N}(\mathbf{x}^*,\delta)$, we define the index sets $\mathbb{J} := \{l \in \mathbb{N}_n : x_l^* = 0, x_l = 1\}$ and $\bar{\mathbb{J}} := \{l \in \mathbb{N}_n : x_l^* = 1, x_l = 0\}$. Then

$$\begin{aligned} &\left(\tau g(\mathbf{x}^*) + \frac{\mathbf{1}}{2} - \mathbf{x}^*\right)^\top (\mathbf{x} - \mathbf{x}^*) \\ = &\sum\nolimits_{l\in\mathbb{J}}\left(\tau g_l(\mathbf{x}^*) + \frac{1}{2}\right) - \sum\nolimits_{l\in\bar{\mathbb{J}}}\left(\tau g_l(\mathbf{x}^*) - \frac{1}{2}\right) \\ \leq &\tau\left(\sum\nolimits_{l\in\mathbb{J}} g_l(\mathbf{x}^*) - \sum\nolimits_{l\in\bar{\mathbb{J}}} g_l(\mathbf{x}^*)\right) + \frac{\delta}{2},\end{aligned}$$

which together with (32) and $\tau > \frac{\delta}{2}$ yields that

$$\sum\nolimits_{l\in\mathbb{J}} g_l(\mathbf{x}^*) - \sum\nolimits_{l\in\bar{\mathbb{J}}} g_l(\mathbf{x}^*) \geq -\frac{\delta}{2\tau} > -1.$$

Since the entries in $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \lambda$ and ρ are integral, then $g_l(\mathbf{x}^*)$ is an integer for every $l \in \mathbb{N}_n$. This implies that

$$\sum\nolimits_{l\in\mathbb{J}} g_l(\mathbf{x}^*) - \sum\nolimits_{l\in\bar{\mathbb{J}}} g_l(\mathbf{x}^*) \geq 0.$$

Then for any $\mathbf{x} \in \mathcal{X} \cap \mathcal{N}(\mathbf{x}^*,\delta)$, it follows from the convexity of L that

$$\begin{aligned} &L(\mathbf{x},\lambda,\rho) - L(\mathbf{x}^*,\lambda,\rho) \\ \geq &\langle \nabla L(\mathbf{x}^*,\lambda,\rho), \mathbf{x} - \mathbf{x}^*\rangle \\ = &\sum\nolimits_{l\in\mathbb{J}} g_l(\mathbf{x}^*)(x_l - x_l^*) + \sum\nolimits_{l\in\bar{\mathbb{J}}} g_l(\mathbf{x}^*)(x_l - x_l^*) \\ = &\sum\nolimits_{l\in\mathbb{J}} g_l(\mathbf{x}^*) - \sum\nolimits_{l\in\bar{\mathbb{J}}} g_l(\mathbf{x}^*) \geq 0.\end{aligned}$$

Therefore, $\mathbf{x}^*$ is a δ-local minimizer of the problem (4). □

We can observe from Theorem 4.1 that every blockwise optimal solution of (4) is a τ-stationary point. Conversely, if the entries in $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \lambda$ and ρ are integral and $\tau > \frac{1}{2}\max_j\{n_j\}$, then every τ-stationary point of (4) is blockwise optimal. These results on relationships between τ-stationary point, blockwise optimal solution and local (global) minimizer are summarized in Figure 1 intuitively.

We finally present the main result on the convergence of the BCD method for solving (15b).

Theorem 4.2. (Convergence properties) Let $\{\mathbf{x}^t\}_{t\in\mathbb{N}}$ be a sequence generated by (15b). If the step size satisfies $0 < \tau < \frac{1}{2\kappa}$, then we have

$$\begin{tikzcd}
\tau\text{-stationary point} \arrow[rr, shift left, "\text{Condition (a), } \tau > \frac{1}{2}\max_j\{n_j\}"] \arrow[dr, shift left, "\text{Condition (a), } \tau > \delta/2"] & & \text{blockwise optimal solution} \arrow[ll, shift left] \arrow[dl, shift right, "\text{Assumption 3.1}"'] \\
& \delta\text{-local minimizer} \arrow[ul, shift left, "0 < \tau < 1/\kappa"] \arrow[ur, shift right, "\delta = n"'] &
\end{tikzcd}$$

Fig. 1: The relationships among τ-stationary point, blockwise optimal solution and local minimizer. Condition(a): the entries in $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}, \lambda$ and ρ are integral.

(i) The sequence $\{L(\mathbf{x}^t, \lambda, \rho)\}_{t\in\mathbb{N}}$ is nonincreasing and

$$\frac{\kappa}{2}\|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \leq L(\mathbf{x}^t, \lambda, \rho) - L(\mathbf{x}^{t+1}, \lambda, \rho). \quad (33)$$

(ii) The sequence $\{\mathbf{x}^t\}$ converges to a τ-stationary point after at most $\lceil \frac{2C\sqrt{n}+\kappa n}{\kappa} \rceil$ iterations, where $C = \max_{\mathbf{x}\in\mathcal{X}} \|\nabla L(\mathbf{x}, \lambda, \rho)\|$.

Proof (i) It is obvious from (29) that if $0 < \tau < \frac{1}{2\kappa}$, then (33) holds.

(ii) Given the parameters $\lambda \in \mathbb{R}^m_+$ and $\rho > 0$, we can observe that the model (4) can be equivalently written as

$$\min F(\mathbf{x}) := L(\mathbf{x}, \lambda, \rho) + \delta_{\mathcal{X}}(\mathbf{x}),$$

where $\delta_{\mathcal{X}}(\mathbf{x})$ is the indicator function of the set $\mathcal{X}$, i.e., $\delta_{\mathcal{X}}(\mathbf{x}) = +\infty$, if $\mathbf{x} \in \mathcal{X}$ and 0 otherwise. To verify that the sequence $\{\mathbf{x}^t\}$ converges to a τ-stationary point, we examine that the conditions in [38, Theorem 1] hold.

Firstly, since the set $\mathcal{X} = \{\mathbf{x} \in \{0,1\}^n : \boldsymbol{B}\mathbf{x} \leq \boldsymbol{d}\}$ is semi-algebraic, then $\delta_{\mathcal{X}}(\mathbf{x})$ is a Kurdyka-Łojasiewicz (KL) function [38], [39]. It is obvious to see that $L(\mathbf{x}, \lambda, \rho)$ is also a KL function, and hence the function $F(\mathbf{x})$ satisfies the KL property, which is a crucial condition ensuring convergence.

Secondly, the Lipschitz constant $\kappa > 0$ in Proposition 4.1 is bounded if ρ has an upper bound, and the problem (4) is inf-bounded.

Thirdly, for any $\mathbf{x}, \bar{\mathbf{x}} \in \mathcal{X}$ with $\bar{\mathbf{x}} = (\mathbf{x}_1, ..., \bar{\mathbf{x}}_j, ..., \mathbf{x}_p)$,

$$\begin{aligned}
&\|\nabla_{\mathbf{x}_j} L(\mathbf{x}, \lambda, \rho) - \nabla_{\mathbf{x}_j} L(\bar{\mathbf{x}}, \lambda, \rho)\| \\
= &\left\|\rho \boldsymbol{A}_j^\top (\boldsymbol{A}\mathbf{x} - \boldsymbol{b})_+ - \rho \boldsymbol{A}_j^\top (\boldsymbol{A}\bar{\mathbf{x}} - \boldsymbol{b})_+\right\| \\
\leq &\rho\|\boldsymbol{A}_j\|_2 \|(\boldsymbol{A}\mathbf{x} - \boldsymbol{b})_+ - (\boldsymbol{A}\bar{\mathbf{x}} - \boldsymbol{b})_+\| \\
\leq &\rho\|\boldsymbol{A}_j\|_2 \|\boldsymbol{A}\mathbf{x} - \boldsymbol{A}\bar{\mathbf{x}}\| \leq \rho\|\boldsymbol{A}_j\|_2^2 \|\mathbf{x}_j - \bar{\mathbf{x}}_j\|.
\end{aligned}$$

Therefore, the result follows from [38, Theorem 1].

Suppose $\mathbf{x}^*$ is a τ-stationary point, then for every $\mathbf{x} \in \mathcal{X}$, we have

$$\begin{aligned}
&L(\mathbf{x}, \lambda, \rho) - L(\mathbf{x}^*, \lambda, \rho) \\
\leq &\langle \nabla L(\mathbf{x}^*, \lambda, \rho), \mathbf{x} - \mathbf{x}^* \rangle + \frac{\kappa}{2}\|\mathbf{x} - \mathbf{x}^*\|^2 \\
\leq &C\|\mathbf{x} - \mathbf{x}^*\| + \frac{\kappa}{2}\|\mathbf{x} - \mathbf{x}^*\|^2 \leq C\sqrt{n} + \frac{\kappa n}{2}. \quad (34)
\end{aligned}$$

If each iteration always finds a new point until arriving at $\mathbf{x}^*$, we get

$$\begin{aligned}
\frac{\kappa T}{2} &\leq \sum_{t=0}^{T} \left(L(\mathbf{x}^t, \lambda, \rho) - L(\mathbf{x}^{t+1}, \lambda, \rho)\right) \\
&\leq L(\mathbf{x}^0, \lambda, \rho) - L(\mathbf{x}^*, \lambda, \rho), \quad (35)
\end{aligned}$$

where the first inequality holds due to the conclusion in (i) and the fact $\|\mathbf{x}^{t+1} - \mathbf{x}^t\| \geq 1$. Thus combining (34) with (35) yields that $T \leq \lceil \frac{2C\sqrt{n}+\kappa n}{\kappa} \rceil$. It implies that we only need at most $\lceil \frac{2C\sqrt{n}+\kappa n}{\kappa} \rceil$ steps to arrive at a τ-stationary point. $\square$

If each iteration of the BCD method always finds a new point, then according to Theorem 4.2 (i), we have $\sum_{t=0}^{\infty} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 < +\infty$. By Theorem 1 in [38], we can conclude that $\lim_{t\to\infty} \mathbf{x}^t = \mathbf{x}^*$, where $\mathbf{x}^*$ is a limit point of the sequence $\{\mathbf{x}^t\}_{t\in\mathbb{N}}$. If we choose an initial point that is already close to an optimal solution and an appropriate step size τ, based on specific convergence criteria detailed in Theorem 4.2, the BCD method can find a global solution of problem (4).

4.2 Convergence of ALM

Assume the BCD method returns a global minimizer $\mathbf{x}^*$ to the augmented Lagrangian relaxation problem (4) in each inner loop. In this subsection, we focus on the convergence property of the projected subgradient method for solving the dual problem (5). For convenience, we introduce the following constants for subsequent analysis:

$$\begin{aligned}
S &:= \arg\max_{\lambda\in\mathbb{R}^m_+, \rho>0} d(\lambda, \rho), \\
\theta &:= \min_{(\lambda,\rho)\in S} \|\lambda^0 - \lambda\|^2 + (\rho^0 - \rho)^2.
\end{aligned}$$

Let d_g^k denote the subgradient of $d(\lambda^k, \rho^k)$. We estimate the distance between the dual function value in each iteration and the optimal value of the problem (1) in the following theorem.

Theorem 4.3. If we take the step size $\alpha^k = \frac{\beta_k}{\|d_g^k\|}$ with $\beta_k = \sqrt{\frac{\theta}{K}}$, then,

$$f^{\text{IP}} - \max_{k\in\{1,2,...,K\}} d(\lambda^k, \rho^k) \leq \frac{\zeta}{2}\sqrt{\frac{5\theta}{K}},$$

where $\zeta = \max_{\mathbf{x}\in\mathcal{X}} \|\boldsymbol{A}\mathbf{x} - \boldsymbol{b}\|^4$. Furthermore, if the positive sequence $\{\beta_k\}_{k\in\mathbb{N}}$ is bounded and $\sum_{k\in\mathbb{N}} \beta_k^2 < +\infty$. Then (λ^k, ρ^k) converges to some $(\lambda^*, \rho^*) \in S$.

Proof Let $\{(\lambda^k, \rho^k)\}_{k\in\mathbb{N}}$ be the sequence generated by Algorithm 1. We derive from the Lemma 3 in [17] that

$$f^{\text{IP}} - \max_{k\in\{1,2,...,K\}} d(\lambda^k, \rho^k) \leq \frac{\theta + \sum_{k=1}^{K}(\alpha^k\|d_g^k\|)^2}{2\sum_{k=1}^{K} \alpha^k}. \quad (36)$$

For all $k \in \mathbb{N}$, we have

$$\|d_g^k\|^2 = \|\boldsymbol{A}\mathbf{x}^k - \boldsymbol{b}\|^2 + \frac{1}{4}\|(\boldsymbol{A}\mathbf{x}^k - \boldsymbol{b})_+\|^4 \leq \frac{5}{4}\|\boldsymbol{A}\mathbf{x}^k - \boldsymbol{b}\|^4 \leq \frac{5}{4}\zeta.$$

Then the right-hand side of (36) satisfies

$$\frac{\theta + \sum_{k=1}^{K}(\alpha^k\|d_g^k\|)^2}{2\sum_{k=1}^{K} \alpha^k} = \frac{\sqrt{5}\zeta(\theta + \sum_{k=1}^{K} \beta_k^2)}{4\sum_{k=1}^{K} \beta_k} = \frac{\zeta}{2}\sqrt{\frac{5\theta}{K}}.$$

Thus we arrive at the first result.

The second claim is then proved in [40, Theorem 7.4], where the proof for the subgradient method is readily extended to the projected subgradient method. □

Although the theoretical analysis of ALM-C relies on Assumption 3.1, our tests show that ALM-C is also effective in more general settings. Furthermore, we present the ALM-P method as a versatile alternative not relying on this assumption.

5 Applications

In this section, we show the performance of the proposed algorithms on two practical problems. One is the train timetabling problem and the other is the vehicle routing problem. To show the performance of compared methods, we define three termination criteria: a maximum number of iterations, a time limit, and an optimality gap based on the difference between the objective value generated by our methods and the best known value. All instances are tested on a MacBook Pro 2019 with 8GB of memory and Intel Core i5 (Turbo Boost up to 4.1 GHz) with 128 MB eDRAM.

5.1 Capacitated vehicle routing problem

We consider the capacitated vehicle routing problem with time windows (CVRPTW). The problem is defined on a complete directed graph $G = (V, E)$, where $V = \{0, 1, ..., n\}$ is the node set and E is the edge set. Node 0 represents the depot where the vehicles are based, and nodes 1 to n represent the customers that need to be served. Each edge (s, t) in E has an associated travel time T_{st}. Each customer s has a demand c_s and a service time window $[a_s, b_s]$. We let d_{st} be the distance from node s to node t and M be a large constant. The objective is to construct a set of least-cost vehicle routes starting and ending at the depot, such that each customer is visited exactly once within their time window, and the total demand of customers served in each route does not exceed vehicle capacity C.

To formulate this problem as an integer program, we define the following decision variables: (i) x_{st}^j: binary variable equal to 1 if edge (s, t) is used by vehicle j, 0 otherwise. (ii) w_s^j: continuous variable indicating the start of service time at customer s by vehicle j. Then the block structured integer linear programming formulation is:

$$\min \sum_{j\in\mathbb{N}_p} \sum_{(s,t)\in E} d_{st} x_{st}^j \tag{37a}$$

$$\text{s.t.} \sum_{j\in\mathbb{N}_p} \sum_{t\in V:t\neq s} x_{st}^j = 1, \quad s \in V\backslash 0 \tag{37b}$$

$$\sum_{t\in V\backslash s} x_{st}^j = \sum_{t\in V\backslash s} x_{ts}^j, \quad i \in V,\ j \in \mathbb{N}_p \tag{37c}$$

$$\sum_{t\in V\backslash 0} x_{0t}^j = 1, \quad j \in \mathbb{N}_p \tag{37d}$$

$$\sum_{s\in V} \sum_{t\in V\backslash s} c_s x_{st}^j \le C, \quad j \in \mathbb{N}_p \tag{37e}$$

$$w_s^j + T_{st} - M(1 - x_{st}^j) \le w_t^j, (s,t) \in E, j \in \mathbb{N}_p \tag{37f}$$

$$a_s \le w_s^j \le b_s, \quad s \in V,\ j \in \mathbb{N}_p \tag{37g}$$

$$x_{st}^j \in 0, 1, \quad (s,t) \in E,\ j \in \mathbb{N}_p \tag{37h}$$

The block structure lies in the routing variables x_{st}^j for each vehicle j, which are constrained by the flow balance, capacity and time window constraints. By relaxing the coupling constraints (37b), the problem decomposes into separate routing subproblems per vehicle.

5.1.1 Parameter Setting

We let ALM-C and ALM-P denote Algorithm 4 using the updates (15a) and (15b) in step 2, respectively. We compare our proposed methods with the Gurobi solver (version 11.0.0) and OR-tools on all the instances from the Solomon dataset [41]. Since the ADMM in [20] lacks adaptability for all instances, then we do not compare with it. To illustrate the scale of these instances, we present a subset of representative examples in Table 1. The notations $|F|, |V|$ and $|E|$ represent the number of vehicles, the number of customers and the number of edges. n and n_w denote the number of variables x and w, respectively. To evaluate the robustness of the compared methods, we conduct experiments on the C1-type instances with various problem sizes, as detailed in Table 2. The results of all remaining instances from the Solomon dataset are presented in Table 3.

We adopt a "vehicle and route-based" formulation to model the CVRPTW problem, which is different from the space-time network flow formulation used in [20]. Therefore, the subproblem is a route problem with capacity and time window constraints. We utilize the Gurobi solver to solve the subproblem for convenience. Note that the formulation does not affect Gurobi's performance. To ensure a fair comparison and maximize Gurobi's utilization, we have implemented efficient callback functions based on Danzig's formulation to fine-tune its performance. We set a time limit of 2000 seconds for the compared methods, except for or-tools where the time limit is set to 500 seconds, and use the symbol "–" to signify that the solver failed to find a feasible solution within the allotted time limit.

TABLE 1: A subset of representative examples of the Solomon datasets

No.	$\|F\|$	$\|V\|$	$\|E\|$	m	q	n	n_w
R101-50	12	50	2,550	50	30,636	30,600	612
R201-50	6	50	2,550	50	15,318	15,300	306
RC101-50	8	50	2,550	50	20,424	20,400	408
RC201-50	5	50	2,550	50	12,765	12,750	255
C101-100	10	100	10,100	100	101,030	101,000	1,010
C201-100	3	100	10,100	100	30,309	30,300	303

5.1.2 Performances of the Proposed Algorithm

In the subsequent tables, the "f^*" and "f" columns correspond to the best-known objective values and the objective values of feasible solutions generated by these compared methods, respectively. The "Time" column denotes the CPU time (in seconds) that the methods taken by the algorithms to meet the stopping criteria. The optimality gap is defined by $\text{gap}^1 = |f - f^*|/|f^*|$. Due to the lack of an inherent termination criterion in OR-tools, it runs for the entire limited time and outputs the corresponding feasible solution. Therefore, we do not report its solution time. Table 2 shows the stability of the compared methods under various problem sizes on the C1-type instances. We can observe that our methods outperform the OR-Tools and Gurobi, and are more stable than the OR-Tools. From Table 3, we can find that the ALM-C algorithm outperforms the OR-Tools and Gurobi in most instances, achieving significantly lower optimality gaps and competitive computation times. The ALM-P algorithm also exhibits promising results, often outperforming Gurobi in efficiency and solution quality. Overall, both the proposed ALM-C and ALM-P algorithms demonstrate their superiority and robustness in solving the CVRPTW problem, providing high-quality solutions with good computational efficiency when compared to existing solvers and heuristic approaches.

To demonstrate the advantages of our methods, we show the convergence curve of the primal bound and dual bound of Gurobi, comparing it to the solution found by our solver with the instance "C109.50" as an example. As we can observe in Figure 2, our methods achieve near-optimal solutions in around 15 seconds, significantly faster than Gurobi which takes approximately 300 seconds. We notice that as the objective values increase, the corresponding constraint violation decreases simultaneously, facilitating fast convergence.

5.2 Train timetabling problem

We consider following space-time network model for the train timetabling problem (TTP) on a macro level, which is based on the model in [42]. We use a directed, acyclic and multiplicative graph $G = (V, E)$ to characterize the train timetabling problem, where V and E denote the set of all nodes and the set of all arcs. For each train $j \in \mathbb{N}_p$, the sets or parameters with superscript or subscript notation corresponds to relevant object to j. For each arc $e \in E_j$, we introduce a binary variable x_e equal to 1 if the arc e is selected. For each node $v \in V$, let $\delta_j^+(v)$ and $\delta_j^-(v)$ be the sets of arcs in E_j leaving and entering node v, respectively. Then the integer programming model of TTP is given by

$$
\begin{aligned}
\max\ & \sum\nolimits_{j\in\mathbb{N}_p}\sum\nolimits_{e\in E^j} p_e x_e && && \text{(38a)}\\
\text{s.t.}\ & \sum\nolimits_{e\in\delta_j^+(\sigma)} x_e \le 1, && j\in\mathbb{N}_p && \text{(38b)}\\
& \sum_{e\in\delta_j^-(v)} x_e = \sum_{e\in\delta_j^+(v)} x_e, && j\in\mathbb{N}_p,\quad v\in V\backslash\{\sigma,\tau\} && \text{(38c)}\\
& \sum\nolimits_{e\in\delta_j^-(\tau)} x_e \le 1, && j\in\mathbb{N}_p && \text{(38d)}\\
& \sum\nolimits_{v'\in\mathcal{N}(v)}\sum\nolimits_{j\in\mathcal{T}(v')}\sum\nolimits_{e\in\delta_j^-(v')} x_e \le 1, && v\in V && \text{(38e)}\\
& \sum\nolimits_{e\in C} x_e \le 1, && C\in\mathcal{C} && \text{(38f)}\\
& x_e\in\{0,1\}, && e\in E, && \text{(38g)}
\end{aligned}
$$

where p_e is the "profit" of using a certain arc e. σ and τ denote artificial origin and destination nodes, respectively. $\mathcal{T}(v)$ and $\mathcal{N}(v)$ denote the set of trains may passing through node v and the set of nodes conflicted with node v, respectively. $\mathcal{C}$ denotes the (exponentially large) family of maximal subsets C of pairwise incompatible arcs. In this model, (38b), (38c), (38d) imply the arcs of train j should form a valid path in G, (38e) represents headway constraints, (38f) forbids the simultaneous selection of incompatible arcs, imposing the track capacity constraints.

Let $\mathbf{x}_j = \{x_e \mid e \in E_j\}$. Then we can rewrite this space-time network model in the general form as (1). Our goal is to show that the proposed Algorithm 4 is fully capable to provide implementable time tables for the Jinghu railway. Specifically, our algorithms are tested on the time-tabling problem for Beijing-Shanghai high-speed railway (or Jinghu high-speed railway in Mandarin). As one of the busiest railways in the world, the Beijing-Shanghai high-speed railway transported over 210 million passengers in 2019*. In our test case, the problem consists of 29 stations and 292 trains in both directions (up and down), including two major levels of speed: 300 km/h and 350 km/h. Several numerical experiments are carried out based on the data of Beijing-Shanghai high-speed railway to demonstrate the feasibility and effectiveness of the proposed strategy.

We compare our proposed methods with the Gurobi solver and ADMM [21] on small and real-world instances, as presented in Tables 4 and 5. The notations $|F|, |S|$ and $|T|$ represent the number of trains, the number of stations, and the time window. In most large-scale cases, the Gurobi solver takes a much longer time to solve, so we set a time limit of two hours. Since (38a) maximizes the positive revenue, we have a negative cost if reversing the objective to a minimization problem, then both subproblems (15a) and (15b) are solved by the Bellman-Ford algorithm, which is an efficient tool for solving the shortest path problem.

5.2.1 *Performances on small instances*

We first validate our algorithm on a smaller sub-network using a subset of stations of the Jinghu railway. We set the revenue of each arc proportional to the distance between two stations, which corresponds to an intuition that longer rides should bear higher incomes. For our proposed methods, we set uniformly $\rho^0 = 20$ and $\sigma = 1.2$ for all instances. We set a time limit of 100 seconds for our algorithms. Since the optimal value is unknown, we report the upper bounds (UB) obtained by the Gurobi solver as a reference for comparison in Table 4. We can observe that our ALM-C performs competitively with the Gurobi solver in terms of both the optimal value and computation time.

5.2.2 *Performances on real-world instances*

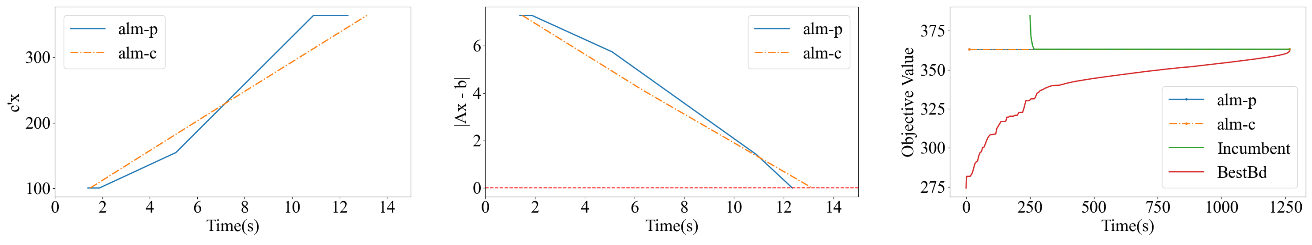
To verify the efficiency of the ALM-C and ALM-P on large-scale data with all stations involved, we consider the following five examples of ascending problem size in Table 5. Similarly, we try to maximize the total revenue of our schedule. In practice, the revenue of an arc may be ambiguous. Besides, since Jinghu high-speed railway always has a high level of utilization, introducing new trains is always beneficial since it further covers unmet demand. In this view, we introduce an alternative objective function to simply maximize the number of scheduled trains. Perhaps not surprisingly, this simplified objective can dramatically speed up our methods for practical interest. This is due to the fact that the coefficient p_e in the objective function is sparse, resulting in faster identification of the optimal solution under this model.

For our proposed methods, we set $\rho^0 = 10^{-3}$ and $\sigma = 2$ for the instances No. 1-2, $\rho^0 = 10^{-2}$ and $\sigma = 1.1$ for the instances No. 3 and No. 5, $\rho^0 = 10^{-3}$ and $\sigma = 1.1$

*. For details, see https://en.wikipedia.org/wiki/Beijing-Shanghai_high-speed_railway.

TABLE 2: Performance comparison on Solomon's C1 instances with varying problem size and perturbation. A time limit of 500 seconds is set for OR-Tools.

ins.	$\lvert V\rvert$	$\lvert F\rvert$	f^*	OR-Tools		Gurobi			ALM-P			ALM-C		
				f	gap[1]	f	gap[1]	Time	f	gap[1]	Time	f	gap[1]	Time
c101	25	3	191.3	632.6	230.7%	191.8	0.3%	0.0	191.8	0.3%	2.4	191.8	0.3%	2.7
	50	5	362.4	514.4	41.9%	363.2	0.2%	1.1	363.2	0.2%	5.9	363.2	0.2%	5.0
	100	10	827.3	828.9	0.2%	828.9	0.2%	8.9	828.9	0.2%	57.8	828.9	0.2%	55.0
c102	25	3	190.3	786.1	313.1%	190.7	0.2%	1.9	190.7	0.2%	3.7	190.7	0.2%	4.6
	50	5	361.4	667.9	84.8%	362.2	0.2%	25.5	362.2	0.2%	16.0	362.2	0.2%	15.3
	100	10	827.3	828.9	0.2%	-		2000	828.9	0.2%	167.4	828.9	0.2%	196.4
c103	25	3	190.3	786.1	313.1%	190.7	0.2%	7.0	190.7	0.2%	15.9	190.7	0.2%	11.0
	50	5	361.4	667.9	84.8%	362.2	0.2%	1584.8	365.3	1.1%	40.0	362.2	0.2%	41.5
	100	10	826.3	-	-	-		2000	839.0	1.5%	362.6	828.1	0.2%	259.9
c104	25	3	186.9	875.6	368.5%	187.4	0.3%	45.1	188.6	0.9%	33.5	188.6	0.9%	17.9
	50	5	358.0	606.2	69.3%	360.1	0.6%	2000	362.2	1.2%	57.1	358.9	0.3%	319.8
	100	10	822.9	1202.3	46.1%	-		2000	890.9	8.3%	667.5	904.9	10.0%	363.3
c105	25	3	191.3	609.6	218.6%	191.8	0.3%	0.1	191.8	0.3%	3.3	191.8	0.3%	3.4
	50	5	362.4	482.4	33.1%	363.2	0.2%	0.3	363.2	0.2%	6.4	363.2	0.2%	6.0
	100	10	827.3	828.9	0.2%	828.9	0.2%	15.8	828.9	0.2%	53.3	828.9	0.2%	34.2
c106	25	3	191.3	639.6	234.3%	191.8	0.3%	0.1	191.8	0.3%	3.6	191.8	0.3%	3.8
	50	5	362.4	430.4	18.8%	363.2	0.2%	1.4	363.2	0.2%	5.8	363.2	0.2%	5.9
	100	10	827.3	828.9	0.2%	828.9	0.2%	650.6	828.9	0.2%	69.6	828.9	0.2%	47.0
c107	25	3	191.3	565.6	195.6%	191.8	0.3%	0.1	191.8	0.3%	3.2	191.8	0.3%	4.3
	50	5	362.4	457.4	26.2%	363.2	0.2%	0.8	363.2	0.2%	6.1	363.2	0.2%	6.6
	100	10	827.3	828.9	0.2%	828.9	0.2%	14.6	828.9	0.2%	65.2	828.9	0.2%	35.2
c108	25	3	191.3	564.6	195.1%	191.8	0.3%	0.9	191.8	0.3%	4.5	191.8	0.3%	4.4
	50	5	362.4	-	-	363.2	0.2%	18.6	363.2	0.2%	8.4	363.2	0.2%	14.2
	100	10	827.3	-	-	-		2000	828.9	0.2%	98.0	828.9	0.2%	265.4
c109	25	3	191.3	475.6	148.6%	191.8	0.3%	9.6	191.8	0.3%	6.3	191.8	0.3%	9.7
	50	5	362.4	363.2	0.2%	363.2	0.2%	1867.4	363.2	0.2%	12.3	363.2	0.2%	13.1
	100	10	827.3	828.9	0.2%	-		2000	828.9	0.2%	234.5	828.9	0.2%	304.3



(a) Objective Value (b) Constraint Violation (c) Primal and Dual Bound of Gurobi

Fig. 2: Comparison of Gurobi and our methods

for the instance No. 4. We set $\mathbf{x}^0$ and λ^0 to be zero for all instances. Both the number of variables and the number of constraints of the model (38) are very large for this practical TTP instance, whose dimensions are up to tens of millions. Tables 6 and 7 show the performance results of our proposed methods and the Gurobi solver, where $\text{gap}^2 := (\text{UB} - f)/f$.

The results clearly demonstrate the high effectiveness of our methods compared to Gurobi, especially when dealing with large-scale data. For the instance No. 1, both ALM-C and ALM-P can schedule 27 trains in a few seconds, which are at least 1000 times faster than the Gurobi solver and meanwhile obtain satisfactory accuracy performance. In particular, when the scale of data is up to tens of millions in instance No. 5, our ALM-C and ALM-P successfully schedule all 292 trains. In contrast, the Gurobi solver produces a relatively small number of trains, only 30 trains, and takes much longer. The results of other instances also illustrate the effectiveness of this technique. Therefore, we can conclude that the customized ALM provides a fast and global optimal solution for this practical problem.

6 Conclusion

In this paper, we study general integer programming with block structure. Benefiting from its special structure, we extend the augmented Lagrangian method, originally designed for continuous problems, to effectively solve the

TABLE 3: Complete results on other Solomon's instances, all instances are associated with 50 customers. A time limit of 500 seconds is set for OR-Tools.

ins.	$\|J\|$	f^*	OR-Tools		Gurobi			ALM-P			ALM-C		
			f	ε	f	ε	t	f	ε	t	f	ε	t
c201.50	3	360.2	1622	350.3%	361.8	0.4%	0.1	361.8	0.4%	4.2	361.8	0.4%	3.4
c202.50	3	360.2	1622	350.3%	361.8	0.4%	5.7	361.8	0.4%	19.1	366.8	1.8%	7.1
c203.50	3	359.8	1713.4	376.2%	361.4	0.4%	113.5	361.4	0.4%	43.7	361.4	0.4%	76.8
c204.50	2	350.1	1435.3	310.0%	351.7	0.5%	2000	366.9	4.8%	55.7	351.7	0.5%	166.3
c205.50	3	359.8	1729	380.5%	361.4	0.4%	1.1	361.4	0.4%	7.9	361.4	0.4%	7.0
c206.50	3	359.8	1741.8	384.1%	361.4	0.4%	1.5	361.4	0.4%	14.4	361.4	0.4%	19.1
c207.50	3	359.6	1622.3	351.1%	361.2	0.4%	11.9	361.2	0.5%	25.0	361.2	0.4%	125.7
c208.50	2	350.5	1629.2	364.8%	352.1	0.5%	8.0	355.9	1.5%	14.2	352.1	0.5%	31.5
r101.50	12	1044	1541.9	47.7%	1046.7	0.3%	2.1	1046.7	0.3%	38.5	1046.7	0.3%	47.3
r102.50	11	909	1374.8	51.2%	911.4	0.3%	2000	939.5	3.4%	248.9	925.3	1.8%	317.2
r103.50	9	772.9	1069.1	38.3%	-	-	2000	806.4	4.3%	289.8	803.8	4.0%	256.1
r104.50	6	625.4	743.5	18.9%	-	-	2000	654.4	4.6%	472.1	686.6	9.8%	649.9
r105.50	9	899.3	1101.7	22.5%	901.9	0.3%	268.5	906.13	0.8%	61.1	901.9	0.3%	60.2
r106.50	5	793	937.8	18.3%	-	-	2000	-	-	-	-	-	-
r107.50	7	711.1	790.5	11.2%	-	-	2000	816.47	14.8%	120.5	741.5	4.3%	336.2
r108.50	6	617.7	698.1	13.0%	-	-	2000	645.7	4.5%	215.1	643.9	4.2%	466.5
r109.50	8	786.8	873.2	11.0%	805.6	2.4%	2000	796.3	1.2%	77.2	788.7	0.2%	324.9
r110.50	7	697	887.1	27.3%	-	-	2000	754.9	8.3%	560.6	768.1	10.2%	225.4
r111.50	7	707.2	784.5	10.9%	-	-	2000	750.0	6.0%	297.8	801.9	13.4%	220.8
r112.50	6	630.2	730.6	15.9%	-	-	2000	665.7	5.6%	476.5	698.0	10.8%	300.2
r201.50	6	791.9	1196.9	51.1%	794.3	0.3%	5.2	803.5	1.5%	27.0	801.3	1.2%	30.4
r202.50	5	698.5	1173.2	68.0%	723.0	3.5%	2000	735.7	5.3%	437.0	726.3	4.0%	337.2
r203.50	5	605.3	1173.2	93.8%	608.0	0.4%	2000	639.4	5.6%	168.6	653.0	7.9%	380.9
r204.50	2	506.4	1127.9	122.7%	512.4	1.2%	2000	524.0	3.5%	75.8	583.8	15.3%	401.6
r205.50	4	690.1	1132.3	64.1%	700.2	1.5%	2000	732.4	6.1%	74.5	731.7	6.0%	87.8
r206.50	4	632.4	1066.9	68.7%	657.2	3.9%	2000	682.1	7.9%	188.4	677.2	7.1%	266.6
r207.50	3	361.6	1046.8	189.5%	-	-	2000	362.6	0.3%	13.1	364.1	0.7%	25.2
r208.50	1	328.2	1019.6	210.7%	-	-	2000	329.3	0.3%	11.9	329.3	0.3%	13.3
r209.50	4	600.6	1043.6	73.8%	639.6	6.5%	2000	608.5	1.3%	217.9	609.4	1.5%	58.5
r210.50	4	645.6	1119.5	73.4%	661.0	2.4%	2000	710.2	10.0%	288.6	669.2	3.7%	253.4
r211.50	3	535.5	958.7	79.0%	-	-	2000	590.4	10.3%	168.8	555.1	3.7%	339.8
rc101.50	8	944	1108.3	17.4%	945.6	0.2%	2000	945.6	0.2%	191.2	945.6	0.2%	78.8
rc102.50	7	822.5	940.7	14.4%	-	-	2000	823.1	0.1%	510.3	823.1	0.1%	242.9
rc103.50	6	710.9	834.9	17.4%	-	-	2000	736.4	3.6%	134.7	751.3	5.7%	335.2
rc104.50	5	545.8	641.4	17.5%	-	-	2000	546.5	0.1%	101.6	546.5	0.1%	91.5
rc105.50	8	855.3	1112.7	30.1%	-	-	2000	873.2	2.1%	99.0	902.7	5.5%	134.4
rc106.50	6	723.2	793	9.7%	-	-	2000	733.7	1.5%	79.3	728.1	0.7%	152.4
rc107.50	6	642.7	752.3	17.1%	-	-	2000	650.3	1.2%	235.2	644.0	0.2%	183.7
rc108.50	6	598.1	690.3	15.4%	-	-	2000	600.7	0.4%	256.1	599.2	0.2%	276.4
rc201.50	5	684.8	1904.7	178.1%	686.3	0.2%	12.5	687.7	0.4%	31.9	686.3	0.2%	14.8
rc202.50	5	613.6	1172.6	91.1%	615.0	0.2%	2000	615.6	0.3%	62.2	615.0	0.2%	85.9
rc203.50	4	555.3	1163.1	109.5%	556.5	0.2%	2000	590.4	6.3%	256.7	558.5	0.6%	265.8
rc204.50	3	444.2	1090.3	145.5%	509.4	14.7%	2000	451.8	1.7%	218.4	462.3	4.1%	153.7
rc205.50	5	630.2	1222.7	94.0%	632.0	0.3%	2000	632.0	0.3%	89.8	632.0	0.3%	56.0
rc206.50	5	610	1088.5	78.4%	611.7	0.3%	2000	611.7	0.3%	36.8	611.7	0.3%	42.9
rc207.50	4	558.6	998.3	78.7%	-	-	2000	591.7	5.9%	315.5	608.5	8.9%	281.1
rc208.50	2	269.1	936.9	248.2%	-	-	2000	269.6	0.2%	65.0	269.6	0.2%	109.1

problem (1). By introducing a novel augmented Lagrangian function, we establish the strong duality and optimality for the problem (1). Furthermore, we provide the convergence results of the proposed methods for both the augmented Lagrangian relaxation and dual problems. To obtain high-quality feasible solutions, we develop a customized ALM combined with refinement techniques to iteratively improve the primal and dual solution quality simultaneously. The numerical experiments demonstrate that the customized ALM is time-saving and performs well for finding optimal solutions to a wide variety of practical problems.

References

[1] P. Wang, C. Shen, A. van den Hengel, and P. H. Torr, "Large-scale binary quadratic optimization using semidefinite relaxation and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 470–485, 2016.

[2] B. S. Y. Lam and A. W.-C. Liew, "A fast binary quadratic programming solver based on stochastic neighborhood search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 32–49, 2020.

[3] X. Lin, Z. J. Hou, H. Ren, and F. Pan, "Approximate mixed-integer programming solution with machine learning technique and linear programming relaxation," in *2019 3rd International Conference on Smart Grid and Smart Cities (ICSGSC)*. IEEE, 2019, pp. 101–107.

TABLE 4: Performance on four small examples on sub-networks of the Jinghu railway for maximizing the total revenue

$\|F\|$	$\|S\|$	$\|T\|$	Gurobi			ADMM		ALM-P		ALM-C	
			UB	f	Time	f	Time	f	Time	f^*	Time
20	15	200	20952	**20952**	**5.0**	**20952**	14.2	**20952**	13.1	**20952**	10.0
25	15	200	23396	**23396**	11.5	23396	31.3	21085	22.1	**23396**	**11.4**
30	15	300	25707	**25707**	14.4	24552	14.1	**25707**	23.0	**25707**	**9.2**
40	15	300	34305	**34305**	24.3	31817	100.0	32061	100.0	**34305**	**17.0**

TABLE 5: Five examples of the large practical railway network

No.	$\|F\|$	$\|S\|$	$\|T\|$	m	q	n
1	50	29	300	49,837	231,722	308,177
2	50	29	600	113,737	1,418,182	1,396,572
3	100	29	720	145,135	1,788,088	4,393,230
4	150	29	960	199,211	3,655,151	9,753,590
5	292	29	1080	228,584	13,625,558	26,891,567

[4] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Mathematical Programming*, vol. 183, no. 1, pp. 3–39, 2020.

[5] F. Eisenbrand, C. Hunkenschröder, K.-M. Klein, M. Kouteckỳ, A. Levin, and S. Onn, "An algorithmic theory of integer programming," *arXiv preprint arXiv:1904.01361*, 2019.

[6] J. Cslovjecsek, M. Kouteckỳ, A. Lassota, M. Pilipczuk, and A. Polak, "Parameterized algorithms for block-structured integer programs with large entries," in *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, 2024, pp. 740–751.

[7] J. A. De Loera, R. Hemmecke, S. Onn, and R. Weismantel, "N-fold integer programming," *Discrete Optimization*, vol. 5, no. 2, pp. 231–241, 2008.

[8] L. Chen, *On Block-Structured Integer Programming and Its Applications*. Cham: Springer International Publishing, 2019, pp. 153–177.

[9] A. H. Land and A. G. Doig, "An automatic method for solving discrete programming problems," in *50 Years of Integer Programming 1958-2008*. Springer, 2010, pp. 105–132.

[10] R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs," *Bulletin of the American Mathematical Society*, vol. 64, pp. 275–278, 1958.

[11] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.

[12] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations Research*, vol. 8, no. 1, pp. 101–111, 1960.

[13] A. M. Geoffrion, "Lagrangean relaxation for integer programming," in *Approaches to Integer Programming*. Springer, 1974, pp. 82–114.

[14] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer programming*. Springer, 2014, vol. 271.

[15] A. Caprara, M. Monaci, P. Toth, and P. L. Guida, "A lagrangian heuristic algorithm for a real-world train timetabling problem," *Discrete Applied Mathematics*, vol. 154, no. 5, pp. 738–753, 2006.

[16] B. Wu and B. Ghanem, "ℓ_p-box ADMM: A versatile framework for integer programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1695–1708, 2018.

[17] K. Sun, M. Sun, and W. Yin, "Decomposition methods for global solutions of mixed-integer linear programs," *arXiv preprint arXiv:2102.11980*, 2021.

[18] M. Feizollahi, "Large-scale unit commitment: Decentralized mixed integer programming approaches," Ph.D. dissertation, Georgia Institute of Technology, 2015.

[19] R. Takapoui, "The alternating direction method of multipliers for mixed-integer optimization applications," Ph.D. dissertation, Stanford University, 2017.

[20] Y. Yao, X. Zhu, H. Dong, S. Wu, H. Wu, L. C. Tong, and X. Zhou, "ADMM-based problem decomposition scheme for vehicle routing problem with time windows," *Transportation Research Part B: Methodological*, vol. 129, pp. 156–174, 2019.

[21] Q. Zhang, R. M. Lusby, P. Shang, and X. Zhu, "Simultaneously re-optimizing timetables and platform schedules under planned track maintenance for a high-speed railway network," *Transportation Research Part C: Emerging Technologies*, vol. 121, p. 102823, 2020.

[22] Y. Kanno and S. Kitayama, "Alternating direction method of multipliers as a simple effective heuristic for mixed-integer nonlinear optimization," *Structural and Multidisciplinary Optimization*, vol. 58, no. 3, pp. 1291–1295, 2018.

[23] E. L. Johnson, G. L. Nemhauser, and M. W. Savelsbergh, "Progress in linear programming-based algorithms for integer programming: An exposition," *Informs Journal on Computing*, vol. 12, no. 1, pp. 2–23, 2000.

[24] J. Feldman, M. J. Wainwright, and D. R. Karger, "Using linear programming to decode binary linear codes," *IEEE Transactions on Information Theory*, vol. 51, no. 3, pp. 954–972, 2005.

[25] M. Laurent and F. Rendl, "Semidefinite programming and integer programming," *Handbooks in Operations Research and Management Science*, vol. 12, pp. 393–514, 2005.

[26] P. Wang, C. Shen, and A. Van Den Hengel, "A fast semidefinite approach to solving binary quadratic problems," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1312–1319.

[27] A. Camisa, I. Notarnicola, and G. Notarstefano, "A primal decomposition method with suboptimality bounds for distributed mixed-integer linear programming," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3391–3396.

[28] R. Vujanic, P. M. Esfahani, P. J. Goulart, S. Mariéthoz, and M. Morari, "A decomposition method for large scale MILPs, with performance guarantees and a power system application," *Automatica*, vol. 67, pp. 144–156, 2016.

[29] P. Muts, I. Nowak, and E. M. Hendrix, "The decomposition-based outer approximation algorithm for convex mixed-integer nonlinear programming," *Journal of Global Optimization*, vol. 77, no. 1, pp. 75–96, 2020.

[30] W. Murray and K.-M. Ng, "An algorithm for nonlinear optimization problems with binary variables," *Computational Optimization and Applications*, vol. 47, no. 2, pp. 257–288, 2010.

[31] S. Lucidi and F. Rinaldi, "Exact penalty functions for nonlinear integer programming problems," *Journal of Optimization Theory and Applications*, vol. 145, no. 3, pp. 479–488, 2010.

[32] C. Yu, K. L. Teo, and Y. Bai, "An exact penalty function method for nonlinear mixed discrete programming problems," *Optimization Letters*, vol. 7, no. 1, pp. 23–38, 2013.

[33] M. J. Feizollahi, S. Ahmed, and A. Sun, "Exact augmented Lagrangian duality for mixed integer linear programming," *Mathematical Programming*, vol. 161, no. 1, pp. 365–387, 2017.

[34] C. E. Blair and R. G. Jeroslow, "The value function of a mixed integer program: I," *Discrete Mathematics*, vol. 19, no. 2, pp. 121–138, 1977.

[35] J. A. González and J. Castro, "A heuristic block coordinate descent approach for controlled tabular adjustment," *Computers & Operations Research*, vol. 38, no. 12, pp. 1826–1835, 2011.

[36] S. Jäger and A. Schöbel, "The blockwise coordinate descent method for integer programs," *Mathematical Methods of Operations Research*, vol. 91, no. 2, pp. 357–381, 2020.

[37] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *Journal of Optimization Theory and Applications*, vol. 109, no. 3, pp. 475–494, 2001.

[38] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1, pp. 459–494, 2014.

[39] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel

TABLE 6: Performance comparison between the Gurobi solver, ADMM, ALM-P and ALM-C on the large networks for maximizing total revenue in Table 5.

No.	Gurobi			ADMM			ALM-P			ALM-C		
	f	gap^2	Time	f	gap^2	Time	f	gap^2	Time	f	gap^2	Time
1	6.18e+04	13.6%	7200.0	6.94e+04	3.0%	**54.1**	6.93e+04	3.1%	84.3	**6.98e+04**	**2.5%**	72.2
2	7.61e+04	45.3%	7200.0	1.34e+05	3.5%	**96.2**	1.34e+05	3.4%	144.4	**1.35e+05**	**3.0%**	126.1
3	9.69e+04	61.5%	7200.0	2.33e+05	7.3%	558.3	2.36e+05	6.3%	414.9	**2.36e+05**	**6.3%**	**396.6**
4	1.23e+05	74.5%	7200.0	4.38e+05	9.6%	1422.7	4.42e+05	8.7%	924.3	**4.45e+05**	**8.1%**	**714.9**
5	2.95e+05	74.0%	7200.0	9.15e+05	19.5%	3600.0	9.60e+05	15.6%	2094.2	**9.63e+05**	**15.3%**	**1722.7**

TABLE 7: Performance comparison between the Gurobi solver, ADMM, ALM-P and ALM-C on the large networks for maximizing the number of trains in the timetable in Table 5

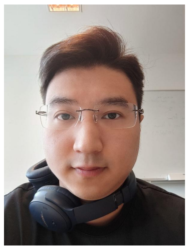
No.	f^*	Gurobi			ADMM			ALM-P			ALM-C		
		f	gap^1	Time	f	gap^1	Time	f	gap^1	Time	f	gap^1	Time
1	–	**27**	–	3836.0	**27**	–	3.4	**27**	–	**3.4**	**27**	–	3.5
2	50	**50**	**0**	2965.8	**50**	**0**	8.2	**50**	**0**	**8.1**	**50**	**0**	8.2
3	100	11	89.0%	7200.0	89	11.0%	3001.4	**100**	**0**	127.4	**100**	**0**	**96.4**
4	150	14	90.7%	7200.0	148	1.3%	3005.3	**150**	**0**	99.5	**150**	**0**	**96.4**
5	292	30	89.7%	7200.0	286	2.1%	3003.8	**292**	**0**	203.1	**292**	**0**	**172.1**

methods," *Mathematical Programming*, vol. 137, no. 1-2, pp. 91–129, 2013.

[40] A. Ruszczynski, *Nonlinear Optimization*. Princeton University Press, 2011.

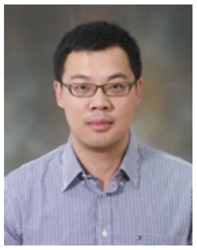
[41] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[42] A. Caprara, M. Fischetti, and P. Toth, "Modeling and solving the train timetabling problem," *Operations Research*, vol. 50, no. 5, pp. 851–861, 2002.



Shanwen Pu received his B.Sc. degree from Nanjing University, Nanjing, China, in 2019. He is currently pursuing his Ph.D. at the Shanghai University of Finance and Economics, Shanghai. His research is focused on large-scale and distributed mixed integer programming, as well as the integration of artificial intelligence and optimization.

Rui Wang received the BSc degree and the PhD degree in operational research from Beijing Jiaotong University, Beijing, China, in 2015 and 2021. She held a postdoctoral position at Peking University, Beijing, China from 2021 to 2023. She currently serves as a lecturer at South Western University of Finance and Economics, Chengdu, China. Her current research interests include large-scale classification optimization problems, sparse optimization, and numerical computing.



Jianjun Gao received his B.E. degree from the University of Science and Technology of China, Hefei, China, in 2003, and his M. Phil and Ph.D degrees in Systems Engineering and Engineering Management from the Chinese University of Hong Kong, Hong Kong, in 2005 and 2009, respectively. He joined the Shanghai Jiao Tong University in China in 2012 the research professor. Starting from 2016, he is the associate professor in Shanghai University of Finance and Economics. His research interests include optimization theory, stochastic optimal control with applications in finance and management science.

Chu wen Zhang received the B.E. degree in Industrial Engineering from Shanghai Jiao Tong University, Shanghai, China, in 2015. He receiced a M.Sc. degree in Operations Research from the University of Texas at Austin, in 2017. He is currently a Ph.D. student with Shanghai University of Finance and Economics, Shanghai. His research interests include large-scale optimization and public sector operations research. Before that, he was an algorithm expert at Cardinal Operations Co., Ltd.

Zaiwen Wen is a Professor at Peking University. He holds a Ph.D in Operations Research from Columbia University (2009). His research interests include optimization algorithms and theory and their applications in machine learning. He was awarded the China Youth Science and Technology Award in 2016 and Beijing Outstanding Youth Zhongguancun Award in 2020. He was funded by the National Ten Thousand Talents Program for Science and He is an associate editor of "Journal of Scientific Computing", "Communications in Mathematics and Statistics", "Journal of the Operations Research Society of China", "Journal of Computational Mathematics" and a technical editor of "Mathematical Programming Computation".