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:

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

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.

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

- *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.

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 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- x 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 semide nite 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 de ne 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 nding the global optimal solution for practical problems such as train timetabling, we introduce re nement 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 xed 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 satis ed. 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 signi cantly 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 speci c 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 signi cantly in that we handle inequality constraints directly, without introducing slack variables. This approach has two key bene ts: 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 $N_m := \{1, 2, \ldots, m\}$, $N_p := \{1, 2, \ldots, p\}$ and $R^m_+ := \{x \in R^m : x_i \geq 0 \text{ for all } i\}$. The superscript "$>$" means

"transpose". Denote a_i by the i-th row of the matrix A and b_i by the i-th element of the vector b. For convenience, we let $A_{I,j}$ denote the submatrix consisting of columns of A_j indexed by I and b_I denote the subvector consisting of entries of $b \in R^m$ indexed by I. A neighborhood of a point x is a set $N(x;1)$ consisting of all points x such that $\|x - x\|^2 \leq 1$. Let 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:

$$\min_{x\in X} L(x;\lambda) := \sum_{j=1}^{p} c_j^\top x_j + \lambda^\top \Big(\sum_{j=1}^{p} A_j x_j - b\Big); \quad (2)$$

where $\lambda \in R_+^m$ 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

$$L(x;\lambda;\rho) = \sum_{j=1}^{p} c_j^\top x_j + \lambda^\top \Big(\sum_{j=1}^{p} A_j x_j - b\Big) + \frac{\rho}{2}\Big\|\sum_{j=1}^{p} A_j x_j - b\Big\|_+^2; \quad (3)$$

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

$$d(\lambda;\rho) := \min_{x\in X} L(x;\lambda;\rho). \quad (4)$$

We call the following maximization problem the AL dual problem:

$$f^{LD} := \max_{\lambda \in R_+^m} d(\lambda;\rho). \quad (5)$$

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

$$\hat{L}(x;\lambda;\rho) = c^\top x + \frac{\rho}{2}\Big\|Ax - b + \frac{\lambda}{\rho}\Big\|_+^2 - \frac{\|\lambda\|^2}{2\rho}. \quad (6)$$

We prefer using the form of the AL function (3) rather than the classical version (6) due to the absence of 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 ε such that for any $i \in N_m$,

$$0 < \varepsilon < \min_{x\in X}\{(a_i x - b_i)^2 : a_i x > b_i\}; \quad (7)$$

then there exists a finite $\bar{\rho} \in (0; +\infty)$ such that

$$f^{LD} = \min_{Ax-b;x\in X} c^\top x.$$

Proof For any $\lambda \in R_+^m$ and $\rho > 0$, since $\{x \in X : Ax - b \leq 0\} \subseteq X$, we have

$$d(\lambda;\rho) \min_{\substack{x\in X\\ Ax-b\leq 0}} L(x;\lambda;\rho) \min_{\substack{x\in X\\ Ax-b\leq 0}} c^\top x = f^{IP}. \quad (8)$$

Then $f^{LD} \leq f^{IP}$.

Now it suffices to find a finite $\bar{\rho}$ such that $f^{LD} \geq f^{IP}$. We first let x^0 be any arbitrary feasible solution of (1), that is, $x^0 \in X$ and $Ax^0 \leq b$. 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., $-\infty < f^{LP} \leq c^\top x^0 < +\infty$, we set $\bar{\rho} = 2(c^\top x^0 - f^{LP})/\varepsilon$, then $0 < \varepsilon < \bar{\rho} < +\infty$. Moreover,

$$f^{LD} \max_{\lambda \in R_+^m} d(\lambda;\bar{\rho}) \geq d(\lambda;\bar{\rho}) = \min_{x\in X} L(x;\lambda;\bar{\rho}); \quad (9)$$

where $\lambda \in R_+^m$ is a given parameter. Let $I := \{i \in N_m : a_i x - b_i > 0; x \in X\}$, we consider following two cases:

Case 1: $I = \emptyset$. In this case we have $Ax \leq b$ for all $x \in X$. By letting $\lambda = 0$, we can obtain that

$$\begin{aligned} L(x;\lambda;\bar{\rho}) &= c^\top x + \lambda^\top (Ax - b) + \frac{\bar{\rho}}{2}\|(Ax - b)_+\|^2 \\ &= c^\top x \geq f^{IP}. \end{aligned} \quad (10)$$

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

$$\begin{aligned} L(x;\lambda^{LP};\bar{\rho}) &= c^\top x + (\lambda^{LP})^\top (Ax - b) + \frac{\bar{\rho}}{2}\sum_{i\in I}(a_i x - b_i)^2 \\ &\quad + \frac{\bar{\rho}}{2}\sum_{i\notin I}((a_i x - b_i)_+)^2 \\ &= c^\top x + (\lambda^{LP})^\top (Ax - b) + \frac{\bar{\rho}}{2}\sum_{i\in I}(a_i x - b_i)^2. \end{aligned}$$

Since

$$\frac{\bar{\rho}}{2}\sum_{i\in I}(a_i x - b_i)^2 \geq \frac{\bar{\rho}}{2}\min_{i\in I}(a_i x - b_i)^2 \overset{(7)}{>} \frac{\bar{\rho}}{2}\varepsilon = c^\top x^0 - f^{LP};$$

it yields

$$\begin{aligned} L(x;\lambda^{LP};\bar{\rho}) &> c^\top x + (\lambda^{LP})^\top (Ax - b) + (c^\top x^0 - f^{LP}) \\ &\geq f^{LP} + (c^\top x^0 - f^{LP}) = c^\top x^0 \geq f^{IP}; \end{aligned} \quad (11)$$

where the second inequality holds due to the de nition of ^{LP}. Thus the inequalities (10) and (11) by letting be ^{LR} and imply that

$$d(\ ; \ ) = \min_{x2X} \ L(x; \ ; \ ) \quad f^{IP}:$$

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

$$f^{LD} = d(\ ; \ ) = f^{IP}:$$

Hence we complete the proof.

One can observe from the proof that there exists a nite value such that for all , the strong duality still holds. Therefore, given a suf ciently large penalty parameter , we can achieve a satisfactory feasibility of (1). Specifically, as increases beyond , 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 $d(\ ; \ )$, we apply the projected subgradient method for solving (5) since the dual function is not differentiable. We rst give the de nition of subgradient and subdifferential.

De nition 2.2. Let $h : R^m \ ! \ R$ be a convex function. The vector $s \ 2 \ R^m$ is called a subgradient of h at $x \ 2 \ R^m$ if

$$h(x) \quad h(x) \quad s^> (x \quad x); \ 8x \ 2 \ R^m :$$

The subdifferential of h at x is the set of all subgradients of h at x which is given by

$$@h(x) = f s \ 2 \ R^m : h(x) \quad h(x) \quad s^> (x \quad x); \ 8x \ 2 \ R^m g:$$

Since $d(\ ; \ )$ is concave, we adjust De nition 2.2 to correspond to the set $@(\ d(\ ; \ ))$, allowing us to apply properties of the subdifferential of a convex function to a concave function.

Proposition 2.1. Consider the dual function $d(\ ; \ ) : R^{m+1} \ ! \ R$. Then the subdifferentials of d at and satisfy

$$A x \quad b \ 2 \ @d(\ ; \ ); \quad \frac{1}{2} k(A x \quad b)_+ k^2 \ 2 \ @d(\ ; \ ); \quad (12)$$

where x is a solution of (4) with input $(\ ; \ )$.

Proof Let x be a solution of (4) with input $(\ ; \ )$. Then for any pair $(\hat{}; \ \hat{}) \ 2 \ R^m_+ \quad R_+$, it holds that

$$\begin{aligned} d(\hat{}; \ \hat{}) \quad & c^> x + \hat{}^> (A x \quad b) + \frac{\hat{}}{2} k(A x \quad b)_+ k^2 \\ = \ & c^> x + \ ^> (A x \quad b) + \frac{\hat{}}{2} k(A x \quad b)_+ k^2 \\ & +(\hat{} \quad)^> (A x \quad b) + \frac{\hat{}}{2} k(A x \quad b)_+ k^2 \\ = \ & d(\ ; \ ) + (\hat{} \quad)^> (A x \quad b) + \frac{\hat{}}{2} k(A x \quad b)_+ k^2; \end{aligned}$$

where the last equality holds due to the optimality of x. Then by the de nition 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 rst use the primal information x to construct the subgradient of d at $(\ ; \ )$. Since and satisfy 0 and $> \ 0$, then we apply the projected subgradient method to update parameters and . Starting from $^0 \ 2 \ R^m_+$ and $^0 > \ 0$, we can update and at $(k+1)$-th iteration by

$$^{k+1} = \ ^k + \ ^k (A x^{k+1} \quad b)_+ ; \quad (13a)$$

$$\begin{aligned} ^{k+1} = \ & ^k + \frac{k}{2} (A x^{k+1} \quad b)_+ \ ^2 \ ! \ + \\ = \ & ^k + \frac{k}{2} (A x^{k+1} \quad b)_+ \ ^2 ; \end{aligned} \quad (13b)$$

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

$$x^{k+1} \ 2 \ \arg\min_{x2X} \ L(x; \ ^k; \ ^k): \quad (14)$$

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

Solving the 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 x-subproblem (14) in the ALM framework. The BCD method minimizes the function L by iterating cyclically in order $x_1; :::; x_p$, xing the previous iteration during each iteration. Denote $x^t = x^t_1; x^t_2; :::; x^t_p$ where x^t_j is the value of x_j at its t-th update. Let

$$L^t_j (x_j ; \ ; \ ) = L(x^{t+1}_1 ; :::; x^{t+1}_{j-1} ; x_j ; x^t_{j+1} ; :::; x^t_p ; \ ; \ );$$

and

$$x^t(j) = x^{t+1}_1 ; x^{t+1}_2 ; :::; x^{t+1}_{j-1} ; x^t_j ; x^t_{j+1} ; :::; x^t_p :$$

Therefore $x^t(1) = x^t_1; x^t_2; :::; x^t_p = x^t$ and $x^t(p+1) = x^{t+1}_1 ; x^{t+1}_2 ; :::; x^{t+1}_p = x^{t+1}$: Given xed parameters 0 and $> \ 0$, we can also calculate the gradient of $L(x^t(j); \ ; \ )$ at x_j by

$$\begin{aligned} g_j(x^t) \quad := \quad & r_{x_j} L(x^t(j); \ ; \ ) \\ = \quad & c_j + A^>_j \quad + A^>_j \quad A x^t(j) \quad b \ _+ : \end{aligned}$$

At each step, we consider two types of updates for every $x_j \ 2 \ X_j$:

$$\text{Classical: } x^{t+1}_j \ 2 \ \arg\min_{x_j \ 2X_j} \ L^t_j (x_j ; \ ; \ ); \quad (15a)$$

Proximal linear:

$$x^{t+1}_j \ 2 \ \arg\min_{x_j \ 2X_j} \ hx_j \quad x^t_j ; g_j(x^t)i + \frac{1}{2} kx_j \quad x^t_j k^2 \ ; \quad (15b)$$

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

$$P_X(v) \in \arg\min\{\|u - v\| : u \in X\};$$

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

$$x_j^{t+1} = P_{X_j}\left[x_j^t - g_j(x^t)\right]:$$

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 x_j. Now we summarize the ALM for solving (1) in Algorithm 1, which allows each 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 x^0, ^{0}; ^{0}.
Output: A feasible solution x^{k+1}.

```
for k = 0; 1; :::; k_max do
    for t = 0; 1; :::; t_max do
        for j = 1; 2; :::; p do
            Compute x_j^(t+1) by (15a) or (15b);
        if x^(t+1) = x^(t), then let x^{k+1} = x^(t+1) and break;
    if k(A x^{k+1} - b)_+ k_2 = 0, then Teminate;
    Update the Lagrangian multipliers ^{k+1} by (13a)
      and the penalty coefficient ^{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 $v \in R^n$ is defined by

$$T(v) = \arg\min_{u \in \Omega} v^\top u;$$

where Ω is a nonempty and closed set.

Benefiting from the good characteristics of x being a binary variable, we have

$$\|x_j\|^2 = 1^\top x_j; \quad \text{for } \forall j \in N_p; \tag{16}$$

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

$$\begin{aligned} x_j^{t+1} &\in \arg\min_{x_j \in X_j} \; x_j^\top g_j(x^t) + \frac{1}{2}1^\top x_j - x_j^\top x_j^t \\ &= T_{X_j}\left[g_j(x^t) + \frac{1}{2} - x_j^t\right]: \end{aligned} \tag{17}$$

Due to the discrete property of the feasible set X_j, it is crucial to choose the step size β appropriately. If β is too large, the BCD method will not converge. If β 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 A are either 1 or 0. The vector b equals 1.

Under this assumption, we consider the subproblem (15a). For each $j \in N_p$, if the condition $A_j x_j^{t+1} \geq 1$ always holds for each iteration of the BCD method, then

$$\begin{aligned} x_j^{t+1} &\in \arg\min_{x_j \in X_j} L_j^t(x_j; \cdot; \cdot) \\ &= T_{X_j}\left[c_j + A_j^\top \cdot 0 + A_j^\top \cdot \beta\Big(\sum_{l \neq j} A_l x_l^t(j) - \frac{1}{2}A \cdot 1\Big)_+\right]: \end{aligned} \tag{18}$$

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

$$\begin{aligned} I &:= \{i \in N_m : \textstyle\sum_{l \neq j} A_{i,l} x_l^t(j) = 0\}; \\ \bar{I} &:= \{i \in N_m : \textstyle\sum_{l \neq j} A_{i,l} x_l^t(j) \geq 1\}; \end{aligned} \tag{19}$$

where $x_l^t(j) = \begin{cases} x_l^{t+1}; & \text{if } l < j; \\ x_l^t; & \text{if } l > j: \end{cases}$ Obviously, $I \cap \bar{I} = \emptyset$ and $I \cup \bar{I} = N_m$. Therefore, we have

$$\sum_{l \neq j} A_{I,l} x_l^t(j) - \frac{1_I}{2} = \sum_{l \neq j} A_{\bar{I},l} x_l^t(j) - \frac{1_{\bar{I}}}{2}; \tag{20}$$

$$\sum_{l \neq j} A_{I,l} x_l^t(j) - \frac{1_I}{2} = 0; \tag{21}$$

and

$$\begin{aligned} A_{I,j} x_j + \sum_{l \neq j} A_{I,l} x_l^t(j) - 1_I &= (A_{I,j} x_j - 1_I)_+ = 0; \\ A_{\bar{I},j} x_j + \sum_{l \neq j} A_{\bar{I},l} x_l^t(j) - 1_{\bar{I}} &\geq 0: \end{aligned} \tag{22}$$

Since the element in $A_j x_j$ is either 1 or 0, we have

$$\|A_j x_j\|^2 = 1^\top (A_j x_j); \quad \forall j \in N_p: \tag{23}$$

Denote $C = \left\| \sum_{l \neq j} A_{I,l} x_l^t(j) - 1_I \right\|^2$. Then

$$\begin{aligned} &\left\| A_j x_j + \sum_{l \neq j} A_l x_l^t(j) - 1 \right\|_+^2 \\ \overset{(22)}{=}\; &\|A_{\bar{I},j} x_j\|^2 + 2(A_{\bar{I},j} x_j)^\top \sum_{l \neq j} A_{\bar{I},l} x_l^t(j) - 1_{\bar{I}} + C \\ \overset{(23)}{=}\; &2(A_{\bar{I},j} x_j)^\top \sum_{l \neq j} A_{\bar{I},l} x_l^t(j) - \frac{1_{\bar{I}}}{2} + C \\ \overset{(20)}{=}\; &2(A_{\bar{I},j} x_j)^\top \sum_{l \neq j} A_{\bar{I},l} x_l^t(j) - \frac{1_{\bar{I}}}{2}\Big\|_+ + C \\ \overset{(21)}{=}\; &2(A_{\bar{I},j} x_j)^\top \sum_{l \neq j} A_{\bar{I},l} x_l^t(j) - \frac{1_{\bar{I}}}{2}\Big\|_+ \\ &+ 2(A_{I,j} x_j)^\top \sum_{l \neq j} A_{I,l} x_l^t(j) - \frac{1_I}{2}\Big\|_+ + C \\ =\; &2(A_j x_j)^\top \sum_{l \neq j} A_l x_l^t(j) - \frac{1}{2}\Big\|_+ + C: \end{aligned}$$

Then the iterative scheme for x-update is given by

$$\begin{aligned} x_j^{t+1} &= \arg\min_{x_j \in X_j} L_j^t(x_j; \cdot; \cdot) \\ &= \arg\min_{x_j \in X_j} c_j^\top x_j + \cdot^\top (A_j x_j - 1) \\ &\quad + \tfrac{\cdot}{2} \| A_j x_j + \sum_{l \neq j} A_l x_l^t(j) - 1 \|_2^2 \\ &\overset{(24)}{=} \arg\min_{x_j \in X_j} c_j^\top x_j + \cdot^\top (A_j x_j - 1) \\ &\quad + (A_j x_j)^\top \sum_{l \neq j} A_l x_l^t(j) + \tfrac{1}{2} \| \cdot \|_2^2 \\ &= T_{X_j}\left(-c_j + A_j^\top \cdot - A_j^\top \sum_{l \neq j} A_l x_l^t(j) - \tfrac{1}{2} \cdot \right) : \end{aligned}$$

We can observe that the condition $A x - 1$ 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 $A_j x_j$ 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_{A_j} the indices of columns of the matrix A_j containing only zeros and c_{j_s} the s-th element of c_j. At least one of the following holds.

(i) For all $j \in N_p$, there exists $x_j \in X_j$ such that $A_j x_j - 1$ and $\{ s \in N_{n_j} : c_{j_s} \neq 0 \} \subseteq T_{A_j}$.

(ii) For all $j \in N_p$, there is at most one nonzero element in the vector c_j and $\{ x_j \in R^{n_j} : A_j x_j - 1 \} \subseteq \{ x_j \in R^{n_j} : x_j \in X_j \}$.

Remark: (a) Assumption 3.2 (i) requires that if an element x_{j_s} satisfies $x_{j_s} = 1$, then at least one of c_{j_s} and the s-th column vector of A_j is zero. Assumption 3.2 (ii) requires that if the block constraint set X_j is large enough such that $A_j x_j - 1$ holds, then the weight c_j corresponding to each block variable has only one nonzero element. (b) Assumption 3.2 usually holds when A and 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 x^0, the solution generated by the BCD method with the classical update at each iteration satisfies $A_j x_j^{t+1} - 1$ for all $j \in N_p$ and $t = 0, 1, 2, \ldots$.

Proof Let 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 $i \in N_m$ such that $A_{i,j} x_j^{t+1} > 1$. Then for any $x_j \in X_j$ satisfying $A_j x_j - 1$, we have

$$L_j^t(x_j^{t+1}; \cdot; \cdot) < L_j^t(x_j; \cdot; \cdot), \quad \forall j \in N_p,$$

which implies that for any $j \in N_p$,

$$\left(c_j + A_j^\top \cdot + A_j^\top \left(\sum_{l \neq j} A_l x_l^t(j) - \frac{1}{2} A \cdot \right)_+ \right)^\top (x_j^{t+1} - x_j) < 0,$$

then

$$\begin{aligned} &c_j^\top (x_j^{t+1} - x_j) + \cdot^\top (A_j x_j^{t+1} - A_j x_j) + \\ &\left(\sum_{l \neq j} A_l x_l^t(j) - \tfrac{1}{2} \cdot \right)_+^\top (A_j x_j^{t+1} - A_j x_j) < 0. \end{aligned} \tag{24}$$

We consider following two cases corresponding to Assumption 3.2:

Case 1: Assumption 3.2 (i) implies $c_{j_s} = 0$ for any $s \in S_j := \{ s \in N_{n_j} : x_{j_s}^{t+1} = 1, A_{i,j_s} = 1 \}$. Taking $x_j = (x_{j_1}^{t+1}, \ldots, x_{j_s}^{t+1} - 1, \ldots, x_{j_{n_j}}^{t+1})$ with $s \in S_j$ such that $A_j x_j - 1$ gives us

$$c_j^\top (x_j^{t+1} - x_j) = \sum_s c_{j_s} = 0 \text{ and } A_{i,j} x_j^{t+1} - A_{i,j} x_j - 1. \tag{25}$$

This contradicts (24) since $\cdot > 0$.

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

$$x_j = (x_{j_1}^{t+1}, \ldots, x_{j_{s^0}}, \ldots, x_{j_s}^{t+1} - 1, \ldots, x_{j_{n_j}}^{t+1}) \text{ with } x_{j_{s^0}} = 1 \tag{26}$$

for any $s \in S_j$ and $s^0 \neq s$ such that $A_j x_j - 1$. If $c_{j_{s^0}} = 0$, we can apply the same procedure in Case 1 and arrive at the contradiction. If $c_{j_{s^0}} < 0$, we take x_j in (26), then $x_{j_{s^0}}^{t+1} - x_{j_{s^0}} \geq 0$. Hence,

$$c_j^\top (x_j^{t+1} - x_j) = c_{j_{s^0}} (x_{j_{s^0}}^{t+1} - x_{j_{s^0}}) < 0, \; A_{i,j} x_j^{t+1} - A_{i,j} x_j - 1,$$

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 $V_j^k = \{ x_j^1, x_j^2, \ldots, x_j^k \}, j = 1, \ldots, p$. 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 $\hat{x}^k$

```
while the termination is not satisfied do
    for j = 1; 2; :::; p do
        Select v_j ∈ V_j^k;
        if A_j v_j + Σ_{l=1}^{j-1} A_l x̂_l^k ≤ b then
            let x̂_j^k = v_j,
        else
            let x̂_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 $j \in \mathcal{N}_p$, we introduce a set of binary variables λ_j that defines the current selection:

$$\hat{x}_j^k = X_j^k \lambda_j; \quad \lambda_j \in \{0,1\}^{g^k};$$

where $X_j^k = [x_j^1; x_j^2; :::; x_j^{g^k}]$. We consider the following problem:

$$\min_{\lambda_j \in \{0,1\}^{g^k}} \sum_{j=1}^{p} c_j^\top X_j^k \lambda_j \tag{27a}$$

$$\text{s.t.} \quad \sum_j \lambda_j \geq 1; \ \forall j = 1; :::; p; \tag{27b}$$

$$\sum_{j=1}^{p} A_j X_j^k \lambda_j \leq b: \tag{27c}$$

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 x which satisfies $Ax \leq b$ 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 K. Therefore, the output feasible point x_j corresponds to $v_j \in K$ for each $j \in \mathcal{N}_p$. If $v_j \notin K$, then $x_j = 0$. 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 x^k, last conflict graph $F^{k-1} = (V^{k-1}; E^{k-1})$
Output: A feasible solution x^k

```
Step 1: Conflict graph update
for j = 1; 2; :::; p do
    Collect new candidate paths V̄_j^k for block j;
    (1.1 node-update) V^k ← V^{k-1} + V̄_j^k;
    (1.2 self-check) E^k ← E^{k-1} + {(p; p') | ∀p ≠ p', p, p' ∈ V̄_j^k; p' ∈ V_j^k};
    (1.3 edge-completion) E^k ← E^k + {(p; p') |
        if p; p' are compatible for ∀p ∈ V̄_j^k; p' ∈ V^k ∩ V_j^k},
        here `compatible' means that these two nodes
        (block variables) satisfy the binding constraints;
Step 2: Maximal independent set (MIS) for a
    feasible solution
Select a candidate solution set K ← V^k;
for v ∈ K do
    Compute a maximal independent set K(v) with
        respect to v in O(|E^k|) iterations;
Compute x = arg max_v K(v).
```

Algorithm 4: A customized ALM

Input: $x^0; \lambda^0; \mu^0 > 0$ and the best objective function value $f^* = +\infty$. Set $k = 0$.
Output: A local (global) optimal solution x^*.

```
while the termination is not satisfied do
    Step 2: Construct solution using Alg. 1
    Update the BCD solution x^{k+1} by the procedure
        in lines 2-5 of Alg. 1;
    Step 3: Generate a feasible solution
    if the BCD solution is not feasible then
        transform the BCD solution to a feasible
            solution x^{k+1} by calling a refinement
            method in Alg. 2 or 3;
    else
        let x^{k+1} = x^{k+1};
    Step 4: Update the best solution
    if c^⊤ x^{k+1} < f* then
        Set f* = c^⊤ x^{k+1} and x* = x^{k+1};
    Update the Lagrangian multipliers λ^{k+1} by (13a)
        and the penalty coefficient μ^{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 $L(x;\ ;\ )$ at x is Lipschitz continuous with constant $\ $ on X, namely,

$$\|\nabla_x L(x;\ ;\ ) - \nabla_x L(x;\ ;\ )\| \quad \|x \quad x\|$$

for all $x;x \ 2\ X$, where $\ = \ \|A\|_2^2$. Furthermore,

$$L(x;\ ;\ ) \quad L(x;\ ;\ ) + \langle x \quad x;\nabla_x L(x;\ ;\ )\rangle + \frac{}{2}\|x \quad x\|^2 \tag{28}$$

for all $x;x \ 2\ X$.

Proof For any $x;x \ 2\ X$,

$$\begin{aligned}
&\|\nabla_x L(x;\ ;\ ) - \nabla_x L(x;\ ;\ )\| \\
= \ & A^{>}(Ax \quad b)_+ \quad A^{>}(Ax \quad b)_+ \\
& \|A\|_2\|(Ax \quad b)_+ \quad (Ax \quad b)_+\| \\
& \|A\|_2\|Ax \quad Ax\| \quad \|A\|_2^2\|x \quad x\|.
\end{aligned}$$

Then we have

$$\begin{aligned}
&\langle \nabla_x L(x;\ ;\ ) - \nabla_x L(x;\ ;\ );x \quad x\rangle \\
&\|\nabla_x L(x;\ ;\ ) - \nabla_x L(x;\ ;\ )\|\|x \quad x\| \quad \|x \quad x\|^2.
\end{aligned}$$

It follows from the convexity of the function $f(x) = \frac{}{2}\|x\|^2$ $L(x;\ ;\ )$ that

$$f(x) \quad f(x) + \langle \nabla f(x)^{>}(x \quad x);$$

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 $\ $ and $\ $, a feasible solution x is called a blockwise optimal solution of the problem (4) if for each $j \ 2\ N_p$, we have for all $x = (x_1;\dots;x_{j\ 1};x_j;x_{j+1};\dots;x_p) \ 2\ X$, $L(x\ ;\ ;\ ) \quad L(x;\ ;\ )$:

Lemma 4.1. Suppose Assumptions 3.1 and 3.2 hold. If the starting point satisfies $x^0 \ 2\ X$, 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 $\{x^t\}$ generated by the BCD method exists. Thus the sequence of objective function values $\{L(x^t;\ ;\ )\}$ can only take finitely many different values. This together with the monotonically decreasing property of the function $L(x;\ ;\ )$ yields that $L(x^t;\ ;\ )$ 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 $x^{t+1} \ \not\in\ x^t$ and the step size is chosen properly.

Lemma 4.2. Let $\{x^t\}_{t\ 2\ N}$ be a sequence generated by (15b), we obtain

$$(\frac{1}{2} \quad \frac{}{2})\|x^{t+1} \quad x^t\|^2 \quad L(x^t;\ ;\ ) \quad L(x^{t+1};\ ;\ ): \tag{29}$$

Proof Since

$$x_j^{t+1} \ = \ T_{X_j} \quad g_j(x^t)^{>}x_j + \frac{1}{2} \quad x_j^t$$

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

$$g_j(x^t) + \frac{1}{2}\langle x_j^t - x_j^{t+1}, x_j^t - x_j^{t+1}\rangle \le g_j(x^t) + \frac{1}{2}\langle x_j^t - x_j^t, x_j^t - x_j^t\rangle,$$

which implies that

$$\langle x_j^{t+1} - x_j^t, g_j(x^t)\rangle + \frac{\rho}{2}\|x_j^{t+1} - x_j^t\|^2 \le 0. \quad (30)$$

Proposition 4.1 tells us that

$$\mathcal{L}_\rho(x^{t+1}, \lambda^t, \beta^t) - \mathcal{L}_\rho(x^t, \lambda^t, \beta^t) \le \sum_{j=1}^p \langle x_j^{t+1} - x_j^t, g_j(x^t)\rangle + \frac{\rho}{2}\sum_{j=1}^p \|x_j^{t+1} - x_j^t\|^2. \quad (31)$$

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

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

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

$$x^t = T_{\mathcal{X}}\left(x^t - \frac{1}{2}\gamma x^t\right).$$

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

$$2\gamma\langle g(x^t), (x^t - x)\rangle \le 2\gamma\|g(x^t)\|k x^t - xk\|x^t - xk < \|x^t - xk^2,$$

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

$$\langle g(x^t) + \frac{1}{2}\gamma(x^t - x^t), x^t - x\rangle < 0.$$

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

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

Definition 4.1. For the AL relaxation problem (4), if a point x satisfies

$$x = T_{\mathcal{X}}\left(x - \frac{1}{2}\gamma g(x)\right)$$

with $\gamma > 0$, then it is called a γ-stationary point.

For the augmented Lagrangian relaxation problem (4), given parameters λ and β, we say that x is a δ-local minimizer if there is an integer $\eta > 0$ such that

$$\mathcal{L}_\rho(x; \lambda, \beta) \le \mathcal{L}_\rho(x', \lambda, \beta), \text{ for all } x' \in \mathcal{N}_\eta(x) \cap \mathcal{X}.$$

Note that a η-local minimizer is a global minimizer due to the fact $\|x - x'\| \le \eta$ for all $x, x' \in \mathcal{X}$. The following important result reveals a relationship between a γ-stationary point and a global minimizer of the problem (4).

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

(i) If x is a local minimizer, then x is a γ-stationary point for any step size $0 < \gamma < 1/\rho$.

(ii) If x is a γ-stationary point with $\gamma > \rho/2$, then x is a δ-local minimizer of the problem (4) under the assumption that the entries in A, b, c, λ and β are integral.

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

$$\mathcal{L}_\rho(x', \lambda, \beta) \ge \mathcal{L}_\rho(x, \lambda, \beta) \ge \mathcal{L}_\rho(x', \lambda, \beta) + \langle \nabla\mathcal{L}_\rho(x', \lambda, \beta), x - x'\rangle + \frac{\rho}{2}\|x - x'\|^2.$$

Therefore,

$$\langle \nabla\mathcal{L}_\rho(x', \lambda, \beta), x - x'\rangle \ge -\frac{\rho}{2}\|x - x'\|^2 \ge -\frac{\gamma}{2}\|x - x'\|^2,$$

which implies that x is a γ-stationary point.

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

$$\langle g(x) + \frac{1}{2}\gamma(x - x), x - x\rangle \ge 0. \quad (32)$$

For any $x \in \mathcal{X} \setminus \mathcal{N}_\eta(x)$, we define the index sets $J := \{l \in \mathcal{N}_n : x_l = 0, x_l = 1\}$ and $J := \{l \in \mathcal{N}_n : x_l = 1, x_l = 0\}$. Then

$$\begin{aligned}
&\langle g(x) + \frac{1}{2}\gamma(x - x), (x - x)\rangle \\
&= \sum_{l\in J} g_l(x) + \frac{1}{2}\gamma\sum_{l\in J} g_l(x) - \frac{1}{2}\gamma\sum_{l\in J} g_l(x) + \frac{1}{2}\gamma\sum_{l\in J} g_l(x) + \frac{\gamma}{2},
\end{aligned}$$

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

$$\sum_{l\in J} g_l(x) - \sum_{l\in J} g_l(x) - \frac{\rho}{2} > -1.$$

Since the entries in A, b, c, λ and β are integral, then $g_l(x)$ is an integer for every $l \in \mathcal{N}_n$. This implies that

$$\sum_{l\in J} g_l(x) - \sum_{l\in J} g_l(x) \ge 0.$$

Then for any $x \in \mathcal{X} \setminus \mathcal{N}_\eta(x)$, it follows from the convexity of $\mathcal{L}_\rho$ that

$$\begin{aligned}
&\mathcal{L}_\rho(x', \lambda, \beta) - \mathcal{L}_\rho(x, \lambda, \beta) \\
&\ge \langle \nabla\mathcal{L}_\rho(x, \lambda, \beta), x' - x\rangle \\
&= \sum_{l\in J} g_l(x)(x_l' - x_l) + \sum_{l\in J} g_l(x)(x_l - x_l') \\
&= \sum_{l\in J} g_l(x) - \sum_{l\in J} g_l(x) \ge 0.
\end{aligned}$$

Therefore, 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 γ-stationary point. Conversely, if the entries in A, b, c, λ and β are integral and $\gamma > \frac{1}{2}\max_j\{n_j\}$, then every γ-stationary point of (4) is blockwise optimal. These results on relationships between γ-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 $\{x^t\}_{t\in\mathcal{N}}$ be a sequence generated by (15b). If the step size satisfies $0 < \gamma < \frac{1}{\rho}$, then we have

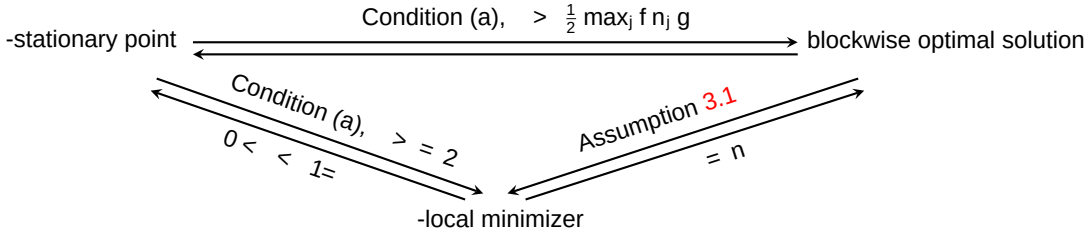

Fig. 1: The relationships among -stationary point, blockwise optimal solution and local minimizer. Condition(a): the entries in A ; b; c; and are integral.

(i) The sequence$\{L(x^t; ;)\}_{t\in N}$ is nonincreasing and

$$\frac{}{2}\|x^{t+1} \quad x^t\|^2 \quad L(x^t; ;) \quad L(x^{t+1}; ;): \quad (33)$$

(ii) The sequence$\{x^t\}$ converges to a -stationary point after at most $\lceil \frac{2C\sqrt{\bar{n}+\ n}}{} \rceil$ iterations, where $C = \max_{x\in X} \|r L(x; ;)\|$.

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

(ii) Given the parameters $2 R^m_+$ and > 0, we can observe that the model (4) can be equivalently written as

$$\min F(x) := L(x; ;) + \ _X(x);$$

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

Firstly, since the set $X = \{x \in \{0;1\}^n : Bx \quad d\}$ is semi-algebraic, then $_X(x)$ is a Kurdyka- ojasiewicz (KL) function [38], [39]. It is obvious to see that $L(x; ;)$ is also a KL function, and hence the function $F(x)$ satis es the KL property, which is a crucial condition ensuring convergence.

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

Thirdly, for any $x; x \in X$ with $x = (x_1; :::; x_j; :::; x_p)$,

$$\begin{aligned} &\|r_{x_j} L(x; ;) \quad r_{x_j} L(x; ;)\| \\ = \quad &\|A_j^{>}(Ax \quad b)_+ \quad A_j^{>}(Ax \quad b)_+\| \\ &\|A_j\|_2 \|(Ax \quad b)_+ \quad (Ax \quad b)_+\| \\ &\|A_j\|_2 \|Ax \quad Ax\| \quad \|A_j\|_2^2 \|x_j \quad x_j\|: \end{aligned}$$

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

Suppose$x$ is a -stationary point, then for every $x \in X$, we have

$$\begin{aligned} &L(x; ;) \quad L(x ; ;) \\ &hr \quad L(x ; ;); x \quad x i + \frac{}{2}\|x \quad x\|^2 \\ &C\|x \quad x\| + \frac{}{2}\|x \quad x\|^2 \quad C\sqrt{\bar{n}} + \frac{n}{2}: \end{aligned} \quad (34)$$

If each iteration always nds a new point until arriving at x , we get

$$\begin{aligned} &\frac{T}{2} \quad \sum_{t=0}^{T} L(x^t; ;) \quad L(x^{t+1}; ;) \\ &L(x^0; ;) \quad L(x ; ;); \end{aligned} \quad (35)$$

where the rst inequality holds due to the conclusion in (i) and the fact $\|x^{t+1} \quad x^t\| \quad 1$. Thus combining (34) with (35) yields that $T \quad d\frac{2C\sqrt{\bar{n}+\ n}}{}e$. It implies that we only need at most $d\frac{2C\sqrt{\bar{n}+\ n}}{}e$ steps to arrive at a -stationary point.

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

4.2 Convergence of ALM

Assume the BCD method returns a global minimizer 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:

$$\begin{aligned} S \quad &:= \quad \arg\max_{2R^m_+; > 0} d(;); \\ &:= \quad \min_{(;)2S} \| \quad ^0 \quad \|^2 + (\quad ^0 \quad)^2: \end{aligned}$$

Let d_g^k denote the subgradient of $d(\ ^k; \ ^k)$. 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 $_k = \frac{k}{\|d_g^k\|}$ with $_k = \sqrt{\frac{}{K}}$, then,

$$f^{IP} \quad \max_{k\in\{1;2;:::;K\}} g d(\ ^k; \ ^k) \quad \frac{}{2}\sqrt{\frac{5}{K}};$$

where $= \max_{x\in X} \|Ax \quad b\|^4$. Furthermore, if the positive sequence$\{ _k\}_{k\in N}$ is bounded and $\sum_{k\in N} \ ^2_k < +1$: Then $(\ ^k; \ ^k)$ converges to some$(\ ; \ ) \in S$.

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

$$f^{IP} \quad \max_{k\in\{1;2;:::;K\}} g d(\ ^k; \ ^k) \quad \frac{+ \sum_{k=1}^{K} (\ ^k\|d_g^k\|)^2}{2\sum_{k=1}^{K} \ ^k}: \quad (36)$$

For all $k \in N$, we have

$$\|d_g^k\|^2 = \|Ax^k \quad b\|^2 + \frac{1}{4}\|(Ax^k \quad b)_+\|^4 \quad \frac{5}{4}\|Ax^k \quad b\|^4 \quad \frac{5}{4}:$$

Then the right-hand side of (36) satis es

$$\frac{+ \sum_{k=1}^{K} (\ ^k\|d_g^k\|)^2}{2\sum_{k=1}^{K} \ ^k} = \frac{\sqrt{5}(+ \sum_{k=1}^{K} \ ^2_k)}{4\sum_{k=1}^{K} \ ^k} = \frac{}{2}\sqrt{\frac{5}{K}}:$$

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 $V = \{0; 1; \ldots; n\}$ 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^j_{st}: binary variable equal to 1 if edge $(s; t)$ is used by vehicle j, 0 otherwise. (ii) w^j_s: continuous variable indicating the start of service time at customer s by vehicle j. Then the block structured integer linear programming formulation is:

$$\min \sum_{j \in N_p} \sum_{(s;t) \in E} d_{st} x^j_{st} \tag{37a}$$

$$\text{s.t.} \sum_{j \in N_p} \sum_{t \in V : te s} x^j_{st} = 1; \qquad s \in V \setminus 0 \tag{37b}$$

$$\sum_{t \in V \setminus s} x^j_{st} = \sum_{t \in V \setminus s} x^j_{ts}; \qquad i \in V; j \in N_p \tag{37c}$$

$$\sum_{t \in V \setminus 0} x^j_{0t} = 1; \qquad j \in N_p \tag{37d}$$

$$\sum_{s \in V} \sum_{t \in V \setminus s} c_s x^j_{st} \le C; \qquad j \in N_p \tag{37e}$$

$$w^j_s + T_{st} \le M(1 - x^j_{st}) \le w^j_t; (s; t) \in E; j \in N_p \tag{37f}$$

$$a_s \le w^j_s \le b_s; \qquad s \in V; j \in N_p \tag{37g}$$

$$x^j_{st} \in \{0; 1\}; \qquad (s; t) \in E; j \in N_p \tag{37h}$$

The block structure lies in the routing variables x^j_{st} 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 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 $j \in N_p$, 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

$$\max \sum_{j \in N_p} \sum_{e \in E^j} p_e x_e \tag{38a}$$

$$\text{s.t.} \sum_{e \in \delta_j^+(o')} x_e \le 1, \quad j \in N_p \tag{38b}$$

$$\sum_{e \in \delta_j^-(v)} x_e = \sum_{e \in \delta_j^+(v)} x_e, \quad j \in N_p, \ v \in V \setminus \{o', d'\} \tag{38c}$$

$$\sum_{e \in \delta_j^-(d')} x_e \le 1, \quad j \in N_p \tag{38d}$$

$$\sum_{v' \in N(v)} \sum_{j \in T(v')} \sum_{e \in \delta_j^-(v')} x_e \le 1, \quad v \in V \tag{38e}$$

$$\sum_{e \in C} x_e \le 1, \quad C \in \mathcal{C} \tag{38f}$$

$$x_e \in \{0, 1\}, \quad e \in E, \tag{38g}$$

where p_e is the "profit" of using a certain arc e. o' and d' denote artificial origin and destination nodes, respectively. $T(v)$ and $N(v)$ 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 $x_j = \{x_e \mid e \in E_j\}$. 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 timetabling 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 $^0 = 20$ and $= 1.2$ 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 $^0 = 10^3$ and $= 2$ for the instances No. 1-2, $^0 = 10^2$ and $= 1.1$ for the instances No. 3 and No. 5, $^0 = 10^3$ and $= 1.1$

*. 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.	jVj	jFj	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 x^0 and ^{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 gap ^{2} := (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. Bene ting 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.	jJj	f	OR-Tools		Gurobi			ALM-P			ALM-C		
			f	"	f	"	t	f	"	t	f	"	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 re nement 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 nding 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 semide nite 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. Koutecký, A. Levin, and S. Onn, "An algorithmic theory of integer programming," *arXiv preprint arXiv:1904.01361*, 2019.

[6] J. Cslovjecsek, M. Koutecký, 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, "`$_p$-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