Report: Augmented Lagrangian Method for Block-Structured Integer Programming

1. Overview

The paper titled "A Customized Augmented Lagrangian Method for Block-Structured Integer Programming" investigates the problem of Block-Structured Integer Programming (BSIP) and proposes a customized Augmented Lagrangian Method (ALM) to solve it. The authors highlight that BSIP problems are of significant interest due to their widespread application in areas such as train timetabling and vehicle routing. However, these problems are NP-hard because of the presence of integer variables, and there are no known polynomial-time algorithms to provide exact solutions.

The authors introduce an innovative approach by building upon the traditional ALM to address this. They propose a new decomposition technique that breaks down the minimization of the augmented Lagrangian function into several simpler subproblems. The Block Coordinate Descent (BCD) method can solve these subproblems independently. The authors also theoretically prove the strong duality between the original problem and the augmented Lagrangian dual problem, and they conduct a theoretical analysis of the convergence properties of both the BCD and ALM methods.

2. Block-Structured Integer Programming Model

We consider an integer programming problem with a block structure, which can be mathematically formulated as follows:

$$
\begin{aligned}
f^{\mathrm{IP}} := \min_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} && (1)\\
\text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b}, && (2)\\
& \mathbf{x}_j \in \mathcal{X}_j, \quad j = 1, 2, \ldots, p, && (3)
\end{aligned}
$$

Where:

- $\mathbf{x}$ is the decision variable vector, which is divided into p sub-blocks as $\mathbf{x}_j$.
- $\mathbf{c}^T\mathbf{x}$ is the linear objective function, and $\mathbf{c}$ is the coefficient vector of the objective function.
- $\mathbf{Ax} \leq \mathbf{b}$ represents the global linear inequality constraints that couple all decision variables $\mathbf{x}$ together.
- Each sub-block $\mathbf{x}_j$ needs to satisfy specific integer constraints $\mathcal{X}_j = \{\mathbf{x}_j \in \{0,1\}^{n_j} : \mathbf{B}_j\mathbf{x}_j \leq \mathbf{D}_j\}$.

3. Augmented Lagrangian Method for Solving Dual Problem Iteratively

The Augmented Lagrangian method is a strategy used to solve constrained optimization problems. It builds upon the Lagrangian function by adding a quadratic penalty term for the constraints. The following is the Augmented Lagrangian model for a block-structured integer programming problem:

$$L(\mathbf{x}, \lambda, \rho) = \mathbf{c}^T\mathbf{x} + \lambda^T(\mathbf{Ax} - \mathbf{b}) + \frac{\rho}{2}\|(\mathbf{Ax} - \mathbf{b})_+\|^2$$

Here, $\lambda \geq \mathbf{0}$ is the Lagrange multiplier vector, and $\rho > 0$ is the penalty parameter for the quadratic term. The corresponding augmented Lagrangian relaxation problem is as follows:

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

We refer to the following maximization problem as the Augmented Lagrangian Dual problem:

$$f_\rho^{\mathrm{LD}} := \max_{\lambda > \mathbf{0}} d(\lambda, \rho)$$

This bi-level dual problem seeks to find the minimum of the augmented Lagrangian function and then maximizes this minimum by adjusting the Lagrange multipliers λ.

3.1 Strong Duality

The authors prove that if the original problem is feasible and bounded, then there exists a finite penalty parameter ρ^* such that for all $\rho \geq \rho^*$, strong duality holds between the original problem f^{IP} and the augmented Lagrangian dual problem f_ρ^{LD}—meaning their optimal values are equal. Therefore, we can find the optimal solution to the primal problem by solving the dual problem.

3.2 Iteration Steps

At each iteration k, the iterative process of the Augmented Lagrangian Method typically includes the following steps:

1. Solve the subproblem to update the decision variables $\mathbf{x}^{k+1} := \arg\min_{\mathbf{x}\in\mathcal{X}} L(\mathbf{x}, \lambda^k, \rho^k)$.
2. Update the multipliers λ^{k+1}.
3. Update the penalty parameter ρ^{k+1}.

Since the Augmented Lagrangian relaxation function $d(\lambda, \rho)$ is a non-differentiable concave function, we need to use the current iterate $\mathbf{x}^k$ to construct a subgradient of this function at (λ^k, ρ^k) in order to update the multipliers λ and the penalty parameter ρ :

$$\mathbf{A}\mathbf{x}^k - \mathbf{b} \in \partial_\lambda d(\lambda^k, \rho^k), \tag{4}$$

$$\frac{1}{2}\|(\mathbf{A}\mathbf{x}^k - \mathbf{b})_+\| \in \partial_\rho d(\lambda^k, \rho^k) \tag{5}$$

Given that the multipliers λ and the penalty parameter ρ must satisfy non-negativity constraints, i.e., $\lambda \geq \mathbf{0}$ and $\rho > 0$, we can update these parameters using the Projected Subgradient Method to ensure they remain non-negative after the update:

$$\lambda^{k+1} := (\lambda^k + \alpha^k(\mathbf{A}\mathbf{x}^k - \mathbf{b}))_+, \tag{6}$$

$$\rho^{k+1} := \rho^k + \frac{\alpha^k}{2}\|(\mathbf{A}\mathbf{x}^k - \mathbf{b})_+\|. \tag{7}$$

However, in the experimental section of the paper, the method for updating the penalty parameter ρ differs. Specifically, the authors gradually increase the penalty parameter ρ

by using a scaling factor σ, thereby increasing the penalty for constraint violations. As a result, ρ is updated as follows:

$$\rho^{k+1} := \sigma \cdot \rho^k$$

3.3 Termination Condition

After each iteration, the algorithm checks whether the current solution satisfies the global constraints of the original problem:

$$\|(\mathbf{A}\mathbf{x}^k - \mathbf{b})_+\| = 0$$

When this value is 0, it indicates that the current solution is within the feasible region and satisfies all the constraints of the primal problem. At this point, the dual problem has converged to $f_\rho^{\mathrm{LD}} = f^{\mathrm{IP}}$, and thus, the algorithm can be terminated.

3.4 Selection of Initial Points

Although the paper does not provide detailed instructions, the selection of initial points in the Augmented Lagrangian Method should follow these guidelines:

- The initial decision variable $\mathbf{x}^0$ should be within the feasible region $\mathcal{X}$ to ensure that the Block Coordinate Descent (BCD) method can be executed and that the block optimal solution can be obtained after a finite number of iterations.
- The initial Lagrange multipliers λ^0 should lie within the non-negative domain $\mathbb{R}_+^m$. These multipliers are typically initialized as a zero vector, i.e., $\lambda^0 = \mathbf{0}$.
- The initial penalty parameter ρ is usually set to a small positive value and gradually increased during the iterations to ensure convergence and satisfaction of the constraints.

3.5 Step Size and Convergence

In the Augmented Lagrangian Method, the step size α^k controls the update rate of the Lagrange multipliers. By choosing an appropriate step size, the algorithm can, after a finite number of iterations, bring the value of the dual function close to the optimal value of the primal problem.

Theoretically, the step size is calculated based on the norm of the subgradient of the dual function $d(\lambda^k, \rho^k)$ and is determined by the following formula:

$$\alpha^k = \frac{\beta^k}{\|\nabla d(\lambda^k, \rho^k)\|}$$

Through theoretical analysis, the authors propose a convergence condition for β^k: $\beta^k = \sqrt{\frac{\theta}{K}}$ and provide an upper bound on the duality gap under this convergence condition:

$$\frac{\max_{\mathbf{x}\in\mathcal{X}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^4}{2} \sqrt{\frac{5\theta}{K}}$$

where K is the maximum number of iterations, and θ is the minimum distance from the initial solution (λ^0, ρ^0) to the set of optimal solutions S.

To ensure that the algorithm ultimately converges to the optimal solution of the original problem (λ^*, ρ^*), the paper further introduces two essential conditions:

- Divergence Condition: Ensures that the algorithm can iterate infinitely until convergence.

$$\sum_{k=1}^{\infty} \beta_k = +\infty$$

- Square-Sum Convergence Condition: Guarantees that the sum of the squares of the step sizes is finite, which helps ensure the stability of the algorithm.

$$\sum_{k=1}^{\infty} \beta_k^2 < +\infty$$

Although the theoretical basis for selecting the step size coefficient β^k is provided, the paper does not specify how to determine β^k in practical applications. In practice, β^k may be a positive decreasing sequence designed to ensure that the step size is neither too large, which could cause abrupt updates, nor too small, leading to slow algorithm convergence.

4. Block Coordinate Descent Method for Updating Decision Variables

During the iterative process of the Augmented Lagrangian Method, the decision variables $\mathbf{x} := \arg\min_{\mathbf{x}\in\mathcal{X}} L(\mathbf{x}, \lambda^k, \rho^k)$ do not have a closed-form solution, but they can be solved either exactly or approximately using iterative methods.

The augmented Lagrangian function achieves the decoupling of the constraints by embedding the global constraints into the objective function. However, due to the presence of the quadratic term $\frac{\rho}{2}\|(\mathbf{A}\mathbf{x}-\mathbf{b})_+\|^2$ in the function, the decision variables are coupled in the objective function. As a result, it is not possible to directly decompose the problem into p independent subproblems to solve them separately.

Based on the special structure of the original problem, the authors propose a novel Block Coordinate Descent (BCD) method. This method divides the decision variables $\mathbf{x}$ into multiple sub-blocks and minimizes the function by sequentially iterating over $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p$, thereby approximating the optimal solution to the problem:

$$\mathbf{x}_j^{t+1} := \min_{\mathbf{x}_j\in\mathcal{X}_j} L(\mathbf{x}_j|\mathbf{x}_1^{t+1}, \ldots, \mathbf{x}_{j-1}^{t+1}, \mathbf{x}_{j+1}^{t}, \ldots, \mathbf{x}_p^{t}, \lambda^k, \rho^k)$$

When optimizing sub-block j, all other blocks are kept fixed, and only the current block $\mathbf{x}_j$ is optimized.

The paper discusses two specific methods for updating block coordinates:

- Proximal Linear Update
- Classical Update

4.1 Proximal Linear Update

Given fixed parameters $\lambda \geq \mathbf{0}$ and $\rho > 0$, we can compute the gradient of $L(\mathbf{x}, \lambda^k, \rho^k)$ at $\mathbf{x}^t$ as follows:

$$g_j(\mathbf{x}^t) := \nabla_{\mathbf{x}_j} L(\mathbf{x}, \lambda, \rho) = \mathbf{c}_j + \mathbf{A}_j^T \lambda + \rho \mathbf{A}_j^T (\mathbf{A}\mathbf{x}^t - \mathbf{b})_+$$

The Proximal Linear method can utilize this first-order gradient $g_j(\mathbf{x}^t)$ to linearly approximate the augmented Lagrangian function at the point $\mathbf{x}^t$ as $g_j(\mathbf{x}^t)^T(\mathbf{x}_j - \mathbf{x}_j^t)$, while ignoring its higher-order terms. This approximation reduces the computational complexity while preserving the primary trend of the function. Then, a quadratic regularization term $\|\mathbf{x}_j - \mathbf{x}_j^t\|^2$ is added to the linearized objective function to promote solution stability and prevent too-large update steps. This results in the following expression:

$$\mathbf{x}_j^{t+1} \in \arg\min_{\mathbf{x}_j \in \mathcal{X}_j} g_j(\mathbf{x}^t)^T(\mathbf{x}_j - \mathbf{x}_j^t) + \frac{1}{\tau}\|\mathbf{x}_j - \mathbf{x}_j^t\|^2$$

where τ is the step size parameter, a positive value used to control the magnitude of the update; $\mathbf{x}_j^t$ is the current value of block j in the t-th iteration.

This problem can also be rewritten as:

$$\mathbf{x}_j^{t+1} \in \arg\min_{\mathbf{x}_j \in \mathcal{X}_j} \|\mathbf{x}_j - (\mathbf{x}_j^t - \tau g_j(\mathbf{x}^t))\|^2$$

Generally, this subproblem is not easy to solve due to the quadratic term in the objective function. However, since $\mathbf{x}$ is a binary (0-1) variable, we have:

$$\|\mathbf{x}_j\|^2 = \mathbf{1}^T \mathbf{x}_j$$

where $\mathbf{1}$ is a vector with all elements equal to 1. This allows us to linearize the quadratic part of the original problem, further simplifying it into an integer linear problem, thus facilitating the solution process:

$$\mathbf{x}_j^{t+1} \in \arg\min_{\mathbf{x}_j \in \mathcal{X}_j} \mathbf{1}^T \mathbf{x}_j - 2\mathbf{x}_j^{tT} \mathbf{x}_j + 2\tau g_j(\mathbf{x}^t)^T \mathbf{x}_j \tag{8}$$

$$= \arg\min_{\mathbf{x}_j \in \mathcal{X}_j} \left(\tau g_j(\mathbf{x}^t) + \frac{\mathbf{1}}{2} - \mathbf{x}_j^t\right)^T \mathbf{x}_j \tag{9}$$

Moreover, the choice of step size τ is crucial for the convergence and efficiency of the algorithm. If the step size is too large or too small, it can adversely affect the performance of the algorithm.

4.2 Classical Update

In the paper, to simplify the solution of subproblems, the authors propose a set of specific assumptions: the elements of the coefficient matrix $\mathbf{A}$ are either 0 or 1, and the right-hand side vector $\mathbf{b}$ is equal to $\mathbf{1}$. Although these assumptions are quite strict, they are common in many optimization problems, such as some assignment, routing, and scheduling problems.

Under these assumptions, for each block $\mathbf{x}_j$ in the Block Coordinate Descent process, the inequality $\mathbf{A}_j\mathbf{x}_j^t \leq \mathbf{1}$ always holds. With this special structure, the authors can derive a simplified form of the subproblem, linearizing the originally complex subproblem and ensuring that the algorithm produces a solution that satisfies the constraints $\mathcal{X}_j$ at each step:

$$\mathbf{x}_j^{t+1} \in \arg\min_{\mathbf{x}_j \in \mathcal{X}_j} \left(\mathbf{c}_j + \mathbf{A}_j^T \lambda + \rho \mathbf{A}_j^T \left(\sum_{l \neq j}^{p} \mathbf{A}_l \mathbf{x}_l^t - \frac{\mathbf{1}}{2} \right)_+ \right)^T \mathbf{x}_j$$

Through this simplification, the augmented Lagrangian function $\min_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}, \lambda, \rho)$ is decomposed into a series of subproblems with linear objective functions. This decomposition not only reduces the complexity of the problem but also makes each subproblem easier to solve, thereby improving the overall efficiency of the algorithm.

4.3 Step Size and Convergence

The convergence analysis provides theoretical guarantees that, under appropriate step sizes and initial conditions, the Block Coordinate Descent (BCD) method can converge to the optimal solution of the problem or to a feasible solution that meets a certain level of precision.

First, if the starting point satisfies $\mathbf{x}^0 \in \mathcal{X}$, the method is always executable and will terminate after a finite number of iterations, yielding the block optimal solution of the augmented Lagrangian relaxation problem $d(\lambda, \rho)$, i.e., the block's variables are optimally chosen when the other blocks' variables are fixed.

The paper proves that the gradient of the augmented Lagrangian function is Lipschitz continuous, which is key to ensuring global convergence. For all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, with $\kappa = \rho\|\mathbf{A}\|^2$, we have:

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

The paper further points out that in the Proximal Linear Update method, the step size τ needs to be determined according to the Lipschitz constant κ. An appropriate choice of step size ensures the non-increasing property of the augmented Lagrangian function's value after each iteration. If the step size is too large, the algorithm may fail to converge; if it is too small, the iterate $\mathbf{x}^t$ may not change after the update.

The paper proposes a theoretical bound $0 < \tau < \frac{1}{2\kappa}$ to ensure that the algorithm can converge to a τ-stationary point within a finite number of iterations and proves that, with a sufficiently large step size, the τ-stationary point is also a local minimum, thereby helping to ensure the global convergence of the BCD algorithm.

5. Finding Feasible Solutions in Block Coordinate Descent

Block Coordinate Descent (BCD) is an iterative method for solving large-scale optimization problems by gradually optimizing each sub-block. However, due to the presence of global constraints $\mathbf{Ax} \leq \mathbf{b}$, there is interdependence between the sub-blocks, which can lead to a globally infeasible solution when optimizing a single sub-block independently. Although each iteration may produce a globally infeasible solution, the algorithm can incorporate additional strategies to correct these infeasible solutions. These strategies not only help to ensure the feasibility of the solution gradually during the iterations but also provide upper bound estimates for the problem, which assists in calculating the duality gap.

Therefore, the authors also discuss two techniques for finding high-quality, globally feasible solutions in the BCD method:

- Sweeping Technique
- Packing Technique

Both techniques rely on constructing a "solution pool" $V_j^k = \{\mathbf{x}_j^1, \mathbf{x}_j^2, \ldots, \mathbf{x}_j^k\}$ for each block j. The solution pool collects all feasible solutions obtained in previous iterations, providing a rich set of candidate solutions.

5.1 Sweeping Technique

The Sweeping Technique is an intuitive and efficient method for finding feasible solutions in the Block Coordinate Descent (BCD) method. It works by sequentially checking the solutions of each block and selecting or resetting them to ensure the feasibility of the global solution.

For each block j in the optimization problem, this technique operates as follows:

1. Select a candidate solution $\mathbf{v}_j$ from the solution pool V_j^k.
2. Check whether the current candidate solution $\mathbf{v}_j$ satisfies the global constraint of the problem $\mathbf{A}_j\mathbf{v}_j + \sum_{l\neq j}^{p} \mathbf{A}_l\hat{\mathbf{x}}_l^k \leq \mathbf{b}$.
3. If the candidate solution $\mathbf{v}_j$ is feasible, it is retained as the current solution for that block $\hat{\mathbf{x}}_j^k := \mathbf{v}_j$.
4. If the candidate solution $\mathbf{v}_j$ is infeasible, set the solution for that block temporarily to zero (or another feasible initial state) and continue checking the next block in the sequence.

For large-scale problems or those with complex constraints, the Sweeping Technique may not find a good solution.

5.2 Packing Technique

The Packing Technique is used to select a set of solutions $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p\}$ from the candidate solution sets $\{V_1^k, V_2^k, \ldots, V_p^k\}$ for each block. This set of solutions must

satisfy the global constraints and collectively form the current optimal solution to the problem.

For each block j, we define $X_j^k := [\mathbf{x}_j^1; \mathbf{x}_j^2; \ldots; \mathbf{x}_j^k]$ as a matrix containing all solutions generated during the first k iterations. Then, a set of binary selection variables $\mu_j \in \{0,1\}^k$ is introduced such that $\hat{\mathbf{x}}_j = X_j^k \mu_j$. Here, the binary variables μ_j determine which solution is selected.

This approach transforms the problem into a restricted master problem, where the solutions generated during the BCD process are gradually added to the master problem using column generation techniques to find the optimal combination of solutions. The mathematical formulation of the optimization problem is as follows:

$$\min \sum_{j=1}^{p} \mathbf{c}_j^T X_j^k \mu_j \tag{10}$$

$$\text{s.t.} \quad \mu_j^T \mathbf{1} \leq 1, \quad \forall j = 1, \ldots, p \tag{11}$$

$$\sum_{j=1}^{p} \mathbf{A}_j X_j^k \mu_j \leq \mathbf{b} \tag{12}$$

Due to global constraints, there may be conflicts between candidate solutions (i.e., their combination cannot satisfy the global constraints). These conflicts prevent the solutions from being simultaneously included in the final solution. Therefore, the problem can be further transformed into a Maximal Independent Set (MIS) problem. The MIS problem aims to find the largest subset of nodes in a graph where no two nodes are adjacent, meaning any edges do not connect these nodes.

This method requires constructing a conflict graph $F = (V, E)$, where the node set V represents all candidate solutions and the edge set E represents their conflict relationships. If two solutions conflict (i.e., they cannot simultaneously satisfy certain constraints), an edge is established between them.

However, the Maximal Independent Set problem is still NP-hard. The paper mentions that the problem being solved is a relaxed version of the Maximal Independent Set problem. By relaxing the problem, the computational complexity can be reduced to some extent, and a feasible solution can be found. However, the paper does not

explicitly specify which algorithm handles this relaxed problem. One simple greedy algorithm is to choose the vertex with the fewest neighbors that have not yet been selected in the current graph and add it to the independent set until no more selections can be made.

6. Experimental Results and Conclusion

In the experimental section of the paper, the authors conducted numerical experiments to verify the effectiveness of the proposed Augmented Lagrangian Method in solving block-structured integer programming problems. The experiments involved several scenarios with practical applications, including train timetable scheduling and vehicle routing problems. These problems are large-scale and contain integer variables. The paper demonstrates the advantages compared to other existing methods in terms of solution efficiency and solution quality.

- Solution Efficiency: The algorithm exhibited fast convergence in multiple instances, achieving satisfactory solutions within fewer iterations.
- Solution Quality: The algorithm was able to find high-quality feasible solutions, often close to the global optimal solution.
- Stability: The algorithm showed good stability and robustness across different test cases, maintaining performance even with large problem sizes or data variations.

7. Code Reproduction

This repository contains the method's implementation and scripts for handling capacitated vehicle routing problem with time-windows (CVRPTW).

7.1 Dataset

In the "5.1 Capacitated Vehicle Routing Problem" section of the paper, the authors mention using the Solomon dataset to evaluate the performance of their proposed algorithm. The Solomon dataset is a standard set of test instances widely used for evaluating Vehicle Routing Problem (VRP) algorithms. These datasets include various parameters such as the number of vehicles, customers, service time windows, and

vehicle capacity limits. Here, we use the C102 instance of the Solomon dataset to reproduce the algorithm.

The C102 instance from the Solomon dataset was selected in this reproduction work. The C102 instance contains 100 customers (excluding the depot). In the paper, the authors considered different variants of the C102 instance to explore the algorithm's performance on problems of varying scales:

- C102-25: The first 25 customers were selected, and 3 vehicles were allocated.
- C102-50: The first 50 customers were selected, with 5 vehicles allocated.
- C102-100: All 100 customers were selected, with 10 vehicles allocated.

The dataset is stored in the file `c102.txt` and is processed by the `data.py` script.

The original paper does not specify the exact setting for the maximum vehicle capacity. **Based on the authors' experimental results, we temporarily set it to 200 units.**

7.2 Mathematical Model

For the Capacitated Vehicle Routing Problem (CVRP), we define the following notation:

- V: The set of nodes, where 0 represents the depot and 1 to n represent the customers.
- E: The set of edges connecting the nodes.
- $\mathbb{N}_p$: The set of vehicle indices.
- x_{st}^j: Equals 1 if vehicle j traverses the route from node s to node t, and 0 otherwise.
- w_s^j: The time when vehicle j begins servicing customer s.
- d_{st}: The distance from node s to node t.
- c_s: The demand of customer s.
- C: The maximum capacity of the vehicle.
- T_{st}: The time required to travel from node s to node t.
- a_s and b_s: The start and end times of the service time window for customer s.
- M: A sufficiently large constant used to ensure the time window constraints. **Since the paper does not specify the value of M, we set M as the maximum value of**

the service time window's end time.

The mathematical model is formulated as follows:

$$\min \sum_{j\in\mathbb{N}_p}\sum_{(s,t)\in E} d_{st}x_{st}^j \tag{13}$$

$$\text{s.t.} \quad \sum_{j\in\mathbb{N}_p}\sum_{t\in V:t\neq s} x_{st}^j = 1, \qquad \forall s \in V\setminus\{0\} \tag{14}$$

$$\sum_{t\in V\setminus\{s\}} x_{st}^j = \sum_{t\in V\setminus\{s\}} x_{ts}^j, \qquad \forall s \in V, \forall j \in \mathbb{N}_p \tag{15}$$

$$\sum_{t\in V\setminus\{0\}} x_{0t}^j = 1, \qquad \forall j \in \mathbb{N}_p \tag{16}$$

$$\sum_{s\in V}\sum_{t\in V\setminus\{s\}} c_s x_{st}^j \leq C, \qquad \forall j \in \mathbb{N}_p \tag{17}$$

$$w_s^j + T_{st} - M(1-x_{st}^j) \leq w_t^j, \qquad \forall (s,t) \in E, \forall j \in \mathbb{N}_p \tag{18}$$

$$a_s \leq w_s^j \leq b_s, \qquad \forall s \in V, \forall j \in \mathbb{N}_p \tag{19}$$

$$x_{st}^j \in \{0,1\}, \qquad \forall (s,t) \in E, \forall j \in \mathbb{N}_p, \tag{20}$$

$$w_s^j \geq 0, \qquad \forall s \in V, \forall j \in \mathbb{N}_p \tag{21}$$

Here, the objective function (11) minimizes the total distance traveled by all vehicles; constraint (12) ensures that each customer is served by exactly one vehicle; constraint (13) guarantees the flow conservation at each node, ensuring that the number of vehicles entering and leaving each node is balanced; constraint (14) ensures that each vehicle departs from and returns to the depot; constraint (15) ensures that the load on each vehicle does not exceed its capacity; constraint (16) restricts the service start time at each customer node to be within the time window; and constraint (17) requires that the arrival time of the vehicle at each customer node is within that node's time window.

Note: Although not mentioned in the original paper, constraint (16) should not apply to the depot $s = 0$, as it would otherwise render the model infeasible.

This Gurobi model is implemented in `gurobi.py`, and the solution obtained using the Gurobi solver serves as a benchmark for the algorithm's performance. The results are consistent with those reported in the paper, validating the model's accuracy.

7.3 Augmented Lagrangian Function

Here, we observe that only constraint (12) is global. Relaxing constraint (12) can decompose the problem into individual path subproblems for each vehicle. Therefore, we can implement the Augmented Lagrangian Method (ALM).

Let λ be the Lagrange multiplier associated with constraint (12), and $\rho > 0$ be the coefficient of the augmented term. The augmented Lagrangian function can then be expressed as:

$$f(\mathbf{x}) + \sum_{s \in V \setminus \{0\}} \lambda_s \left(\sum_{j \in \mathbb{N}_p} \sum_{t \in V : t \neq s} x_{st}^j - 1 \right) + \frac{\rho}{2} \sum_{s \in V \setminus \{0\}} \left(\sum_{j \in \mathbb{N}_p} \sum_{t \in V : t \neq s} x_{st}^j - 1 \right)^2$$

where $f(\mathbf{x})$ is the original objective function.

The right-hand side value of the global constraint is $\mathbf{b} = 1$. For each vehicle j, the constraint matrix block $\mathbf{A}_j$ is a $(|V| - 1) \times |E|$ matrix. Specifically, the rows of $\mathbf{A}_j$ correspond to each customer node s (excluding the depot), and the columns correspond to each possible path x_{st}^j. If the path (s, t) is included in the constraint, the corresponding matrix element is 1; otherwise, it is 0.

For instance, suppose there are three customers (nodes 1, 2, and 3) and one depot (node 0). The matrix $\mathbf{A}_j$ can be represented as:

$$\mathbf{A}_j = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

The Augmented Lagrangian Method can be found in the `alm.py` file, and the Block Coordinate Descent (BCD) method is in `bcd.py`.

7.4 Initial Values

In the paper, the authors do not provide detailed discussions regarding the initial solution $\mathbf{x}^0$, the initial values for the Lagrange multipliers λ^0, and the initial penalty parameter ρ^0 in the Augmented Lagrangian Method for the vehicle routing problem.

We used a greedy algorithm to find an appropriate initial solution $\mathbf{x}^0$ (implementation code can be found in `greedy.py`). The core idea of this algorithm is: for the current node s, while ensuring compliance with time window constraints and vehicle capacity limits, select the next node t that maximizes the value of $b_t - d_{st}$. Here, b_t represents the end time of the time window constraint for node t, and d_{st} is the distance (or time) from node s to node t. This approach allows us to construct an initial route for each vehicle. Although this algorithm does not guarantee the satisfaction of all global constraints (i.e., visiting all nodes), it does satisfy the block constraints $\mathbf{x}_j \in \mathcal{X}_j$ (flow, time window, and capacity constraints), which is consistent with the convergence requirements discussed in the paper.

The initial Lagrange multipliers λ^0 are set to the zero vector, and the initial penalty parameter ρ^0 is set to 1.

7.5 Finding Feasible Solutions

In the customized Augmented Lagrangian Method proposed in the paper, an additional heuristic algorithm is introduced to find feasible solutions during the iteration process. The implementation of this can be found in `calm.py`.

In this context, the Sweeping Technique was attempted as a heuristic method to find the optimal solution, as implemented in `feasible_heuristic.py`. However, despite using the Sweeping Technique, no feasible solutions were found during the iteration process. This could be due to issues in the code reproduction or because the constraints in the vehicle routing problem are too complex to be solved directly using such a simple heuristic method.

Given this, the Packing Technique might be a more promising solution. The Packing Technique involves constructing a conflict graph and heuristically finding the Maximal Independent Set within the graph to identify a feasible combination of solutions. This approach may be more suitable for handling optimization problems with complex constraints. However, due to time constraints, this solution has not yet been implemented.