Journal of Industrial and Management Optimization

https://www.aimspress.com/journal/jimo

JIMO, 22(7): 3284–3309.
DOI:10.3934/jimo.2026120
Received: 19 March 2026
Revised: 20 May 2026
Accepted: 28 May 2026
Published: 22 June 2026

Research article

An augmented Lagrangian decomposition method for the single-source capacitated facility location problem

Deshan Hong and Rui Wang*

School of Mathematics, Southwestern University of Finance and Economics, Chengdu 611130, China

* **Correspondence:** Email: ruiwang@bicmr.pku.edu.cn.

Abstract: We studied the single-source capacitated facility location problem (SSCFLP), which can also be viewed as a representative structured binary optimization model with assignment, activation, and capacity constraints. The main difficulty comes from the discrete structure of the model and the coupling constraints, which make large-scale instances hard to solve. To exploit this structure, we introduced a split formulation that separates the original variables into two blocks connected by an equality constraint. For the reformulated model, we proposed an augmented Lagrangian decomposition method with tractable primal subproblems. We further showed that the augmented Lagrangian dual has zero duality gap with the original binary problem when the penalty parameter is sufficiently large. This result provides a theoretical basis for applying the augmented Lagrangian method to this discrete optimization problem. In addition, we used a primal recovery step and a final refinement step to improve feasible solutions. Numerical results demonstrated that the proposed method can produce high-quality feasible solutions on both synthetic and benchmark instances, especially for large instances in time-limited settings.

Keywords: 0–1 integer programming; augmented Lagrangian decomposition; single-source capacitated facility location problem; split formulation; exact augmented Lagrangian duality
Mathematics Subject Classification: 90C10, 90C26, 90C30

1. Introduction

Large-scale 0–1 integer programming (IP) models arise in many decision problems, including resource allocation, logistics, network design, scheduling, and production planning. They are challenging to solve due to the combinatorial nature of binary decisions and the strong coupling among variables induced by multiple constraints.

In this paper, we study the single-source capacitated facility location problem (SSCFLP), in which each customer must be assigned to exactly one facility, each facility has a limited capacity, and a fixed

opening cost is incurred whenever a facility is open. The SSCFLP is a classical binary optimization problem with exact-assignment constraints and capacity-opening coupling. It appears in capacitated assignment, facility planning, service system design, and related location-allocation settings [1–4]. The main difficulty of the SSCFLP lies in the interaction between two different types of constraints. The assignment constraints require each customer to choose exactly one facility, while the capacity constraints link the total assigned demand to the facility-opening decisions. Each part is relatively simple when considered alone, but their combination creates a hard combinatorial structure. As a result, directly solving the original IP model may not fully exploit the internal structure of the problem, especially on large-scale instances.

Substantial literature has studied exact and heuristic methods for the SSCFLP and related capacitated location models. Among exact methods, Holmberg et al. [5] studied an exact algorithm for the SSCFLP. Yang et al. [6] developed a cut-and-solve framework, and Gadegaard et al. [7] later improved this framework. More recently, Weninger and Wolsey [8] studied Benders-type branch-and-cut algorithms for capacitated facility location with single sourcing. Heuristic methods have also been widely studied. Multi-exchange local search, scatter search, and tabu-search-based procedures were shown to work well on large instances [9–11]. Lagrangian heuristics and relaxation-based bounds were well-established for location and allocation models [12]. These studies provide useful algorithmic references for the present problem. However, most existing methods for the SSCFLP are designed either as exact global-search procedures or as problem-specific heuristics. This suggests the need for a method that uses the internal structure of the model more directly, especially the different roles of the assignment decisions and the facility-opening decisions.

Reformulation and decomposition are standard ways to exploit structure in difficult binary optimization models. Classical approaches include Lagrangian relaxation [13], Dantzig–Wolfe decomposition [14], Benders decomposition [15], and later developments in reformulation and decomposition for integer programming [16–19]. For the SSCFLP, these ideas suggest that the assignment structure and the facility-opening structure should be separated whenever possible. After such a reformulation, the main remaining issue is how to enforce the linking constraints between the resulting blocks in an effective manner.

The augmented Lagrangian (AL)-based methods provide a natural framework for handling the linking equation while keeping the block subproblems simple. Recent work has shown that augmented Lagrangian schemes can support both decomposition and exactness in discrete optimization. Wang et al. [20] developed an augmented Lagrangian method for block-structured integer programming. Takapoui et al. [21] proposed a simple and effective heuristic for embedded mixed-integer quadratic programming. Boland and Eberhard [22] showed that the augmented Lagrangian dual achieves strong duality under a sufficiently large fixed penalty for pure integer programming. Feizollahi et al. [23] established exact augmented Lagrangian duality for mixed-integer linear programming and gave a primal interpretation of the resulting bound. More recently, Sun et al. [24] studied decomposition methods for globally solving two-block mixed-integer linear programs by combining splitting ideas with augmented Lagrangian-based approaches. These results indicate that augmented Lagrangian schemes can be useful for structured discrete models. However, it is still unclear how to design a split augmented Lagrangian framework that keeps the block subproblems simple, supports practical primal recovery, and admits structural properties tied to the discrete consensus relation.

Motivated by these observations, we reformulate the SSCFLP by introducing two copies of the assignment variable and linking them through a consensus equation. This split formulation separates the original model into two blocks: an assignment block and a capacity-opening block. After the split, the first block admits simple customer-wise updates, while the second block reduces to independent fixed-charge binary knapsack-type subproblems for each facility, which can be solved in parallel. Based on this reformulation, we develop an augmented Lagrangian decomposition framework in which the primal variables are updated by alternating block minimization and the linking equation is handled by multiplier updates. To improve practical performance, we also incorporate a primal recovery step and a local improvement procedure. Theoretical guarantees for basic decomposition schemes in discrete optimization are usually different from those for classical continuous augmented Lagrangian methods. Therefore, we focus on the main structural issues, namely, the exactness of the split augmented Lagrangian model and several basic properties of the resulting iteration, such as residual control and the role of the multiplier sequence. These properties are analyzed in the binary split formulation and serve as the theoretical foundation of the proposed method. Based on the above discussion, Table 1 presents a brief comparison between related approaches and the method developed in this paper. The main contributions of this paper are summarized as follows.

(i) We propose a split formulation for the SSCFLP by separating the assignment decisions from the facility-opening decisions and linking them through an equality constraint. The reformulated model preserves the original feasible set and objective value, while exposing a structure that is easier to decompose.
(ii) We develop an augmented Lagrangian decomposition method for the reformulated model. The assignment block admits simple closed-form updates, and the capacity-opening block reduces to independent fixed-charge binary knapsack-type subproblems for each facility. We further combine the basic decomposition iteration with primal recovery and local improvement to obtain feasible solutions of good quality in practice.
(iii) We show that the augmented Lagrangian dual achieves zero duality gap with the original binary problem when the penalty parameter is sufficiently large, providing a theoretical justification for the proposed framework in the discrete setting.

The rest of the paper is organized as follows. In Section 2, we present the SSCFLP and its split reformulation, and establish the zero duality gap result for the augmented Lagrangian dual problem. Section 3 develops the augmented Lagrangian decomposition method together with the primal recovery and refinement procedures. Section 4 studies several structural properties of the proposed iteration. Section 5 reports numerical results on synthetic and benchmark instances. Section 6 concludes the paper.

2. Problem description and formulation

In this section, we first present the original binary optimization model and its compact form. We then introduce a consensus reformulation that separates the assignment and capacity blocks, and establish an exactness property for the associated augmented Lagrangian penalization. These results provide the structural foundation for the decomposition algorithm developed in the next section.

Table 1. Related approaches for the SSCFLP and structured integer programming.

Approach	Key technique	Constraint handling	Solution strategy
Exact SSCFLP methods [5–8]	Branch-and-cut, cut-and-solve, Benders enumeration	Direct modeling, no block separation	Search incumbent
Heuristic SSCFLP methods [10, 25, 26]	Local search, scatter search, tabu search	No explicit block separation	Direct search over feasible set
Lagrangian relaxation [27]	Relax coupling constraints, update multipliers	Dualization of one constraint group	Recovery from dual solution
Classical decomposition [14, 15]	Master–subproblem with cuts or columns	Cuts/columns coordinate subproblems	Search incumbent
AL-based methods for IP [22, 23, 28, 29]	Quadratic penalty and multiplier on linking constraints	Consensus or linking between blocks	Algorithm-specific recovery
This paper	AL decomposition with recovery and refinement	Assignment and capacity blocks linked by $\boldsymbol{p} = \boldsymbol{q}$	Recovery and local improvement

2.1. Problem formulation

We consider the single-source capacitated facility location problem (SSCFLP) with a set $\mathcal{O} := \{1, 2, \ldots, n\}$ of customers and a set $\mathcal{B} := \{1, 2, \ldots, m\}$ of candidate facilities. Each customer must be assigned to exactly one facility, while each facility has a limited capacity and incurs a fixed opening cost when used. This model arises in several settings, including supply chain design, service deployment, and capacitated resource allocation.

For each $i \in \mathcal{O}$ and $b \in \mathcal{B}$, let $c_{ib} \geq 0$ be the assignment cost of assigning customer i to facility b. Let $w_i > 0$ be the demand of customer i, let $C_b > 0$ be the capacity of facility b, and let $\alpha_b \geq 0$ be the fixed opening cost of facility b. We introduce the binary variables

$$x_{ib} = \begin{cases} 1, & \text{if customer } i \text{ is assigned to facility } b, \\ 0, & \text{otherwise,} \end{cases} \qquad z_b = \begin{cases} 1, & \text{if facility } b \text{ is open,} \\ 0, & \text{otherwise.} \end{cases}$$

The problem is then written as

$$\min_{x,z} \quad \sum_{i\in\mathcal{O}} \sum_{b\in\mathcal{B}} c_{ib} x_{ib} + \sum_{b\in\mathcal{B}} \alpha_b z_b \tag{2.1a}$$

$$\text{s.t.} \quad \sum_{b\in\mathcal{B}} x_{ib} = 1, \qquad \forall i \in \mathcal{O}, \tag{2.1b}$$

$$\sum_{i\in\mathcal{O}} w_i x_{ib} \leq C_b z_b, \qquad \forall b \in \mathcal{B}, \tag{2.1c}$$

$$x_{ib} \in \{0,1\}, \qquad \forall i \in O,\ \forall b \in \mathcal{B}, \tag{2.1d}$$

$$z_b \in \{0,1\}, \qquad \forall b \in \mathcal{B}. \tag{2.1e}$$

In (2.1), the objective consists of the total assignment cost and the total facility opening cost. Constraint (2.1b) enforces exact assignment. Constraint (2.1c) is a capacity-linking constraint: if $z_b = 0$, then no customer can be assigned to b; if $z_b = 1$, then the total assigned demand cannot exceed C_b.

To expose the structural decomposition of the model, it is useful to rewrite (2.1) in a compact form. Let $\boldsymbol{x} \in \{0,1\}^{nm}$ collect all variables x_{ib}, and let $\boldsymbol{z} \in \{0,1\}^m$ collect all variables z_b. Let $\mathbf{c} \in \mathbb{R}^{nm}$ collect the coefficients c_{ib}, and let $\boldsymbol{\alpha} \in \mathbb{R}^m$ collect the coefficients α_b. Then (2.1) can be written as

$$\min_{\boldsymbol{x},\boldsymbol{z}} \quad \mathbf{c}^\top \boldsymbol{x} + \boldsymbol{\alpha}^\top \boldsymbol{z} \tag{2.2a}$$

$$\text{s.t.} \quad A\boldsymbol{x} = e, \tag{2.2b}$$

$$W\boldsymbol{x} - C\boldsymbol{z} \le 0, \tag{2.2c}$$

$$\boldsymbol{x} \in \{0,1\}^{nm}, \qquad \boldsymbol{z} \in \{0,1\}^m, \tag{2.2d}$$

where A is the assignment matrix associated with (2.1b), W is the demand aggregation matrix, C is the diagonal matrix of capacities, and e is the all-one vector of suitable dimension.

The compact formulation (2.2) highlights two distinct structural components of the model: the exact-assignment equations (2.2b) and the capacity-opening constraints (2.2c). Each block is relatively simple when considered alone. The main difficulty comes from the combinatorial coupling in the feasible set. This structural observation motivates a variable-splitting reformulation, in which separate copies of the assignment variable are introduced for the two constraint blocks and linked through a consensus equation $\boldsymbol{p} = \boldsymbol{q}$.

2.2. Reformulation and finite exactness

Variable splitting and Lagrangian decomposition are classical techniques to separate coupled parts of an optimization model [30, 31]. Related ideas have also been used in capacitated location models, where the problem can be decomposed into transportation and knapsack-type subproblems [32], and in later decomposition or alternating direction method of multipliers (ADMM)-based methods for integer programming [33, 34]. In this paper, we use this idea for the SSCFLP by assigning one copy to the exact-assignment constraints and the other copy to the capacity-opening constraints. We now formalize the split reformulation. Variable $\boldsymbol{p}$ is associated with the exact-assignment block, and variable $\boldsymbol{q}$ is associated with the capacity block. The two copies are linked by a consensus equation, which leads to the following equivalent reformulation.

$$\min_{\boldsymbol{p},\boldsymbol{q},\boldsymbol{z}} \quad \sum_{i\in O}\sum_{b\in\mathcal{B}} c_{ib}q_{ib} + \sum_{b\in\mathcal{B}} \alpha_b z_b \tag{2.3a}$$

$$\text{s.t.} \quad \sum_{b\in\mathcal{B}} p_{ib} = 1, \qquad \forall i \in O, \tag{2.3b}$$

$$\sum_{i\in O} w_i q_{ib} \le C_b z_b, \qquad \forall b \in \mathcal{B}, \tag{2.3c}$$

$$p_{ib} = q_{ib}, \qquad \forall i \in O,\ \forall b \in \mathcal{B}, \tag{2.3d}$$

$$p_{ib}, q_{ib}, z_b \in \{0,1\}, \qquad \forall i \in O,\ \forall b \in \mathcal{B}. \tag{2.3e}$$

The reformulated problem (2.3) is equivalent to the original model (2.1). Indeed, if $(\boldsymbol{x}, \boldsymbol{z})$ is feasible for (2.1), then $(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{z}) = (\boldsymbol{x}, \boldsymbol{x}, \boldsymbol{z})$ is feasible for (2.3). Conversely, if $(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{z})$ is feasible for (2.3), then (2.3d) gives $\boldsymbol{p} = \boldsymbol{q}$, and hence $(\boldsymbol{q}, \boldsymbol{z})$ is feasible for (2.1). Therefore, the two formulations have the same optimal value.

For convenience, we define

$$\mathcal{P} := \left\{ \boldsymbol{p} \in \{0,1\}^{nm} : \ \sum_{b \in \mathcal{B}} p_{ib} = 1, \ \forall i \in \mathcal{O} \right\}$$

and

$$\mathcal{Q} := \left\{ (\boldsymbol{q}, \boldsymbol{z}) \in \{0,1\}^{nm} \times \{0,1\}^{m} : \ \sum_{i \in \mathcal{O}} w_i q_{ib} \le C_b z_b, \ \forall b \in \mathcal{B} \right\}.$$

Then (2.3) can be written as

$$\min_{\boldsymbol{p} \in \mathcal{P}, \ (\boldsymbol{q}, \boldsymbol{z}) \in \mathcal{Q}} \{ f(\boldsymbol{q}, \boldsymbol{z}) : \ \boldsymbol{p} - \boldsymbol{q} = 0 \},$$

where

$$f(\boldsymbol{q}, \boldsymbol{z}) := \sum_{i \in \mathcal{O}} \sum_{b \in \mathcal{B}} c_{ib} q_{ib} + \sum_{b \in \mathcal{B}} \alpha_b z_b$$

is the objective function and depends only on $(\boldsymbol{q}, \boldsymbol{z})$, since $\boldsymbol{p}$ is introduced solely to isolate the assignment constraints.

We introduce an augmented Lagrangian (AL) function associated with the consensus equation. Let $\boldsymbol{\lambda} \in \mathbb{R}^{nm}$ be the multiplier vector and let $\rho > 0$ be the penalty parameter. Then the AL function is defined by

$$\mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{z}; \boldsymbol{\lambda}) = f(\boldsymbol{q}, \boldsymbol{z}) + \boldsymbol{\lambda}^\top (\boldsymbol{p} - \boldsymbol{q}) + \frac{\rho}{2} \|\boldsymbol{p} - \boldsymbol{q}\|^2. \tag{2.4}$$

Based on (2.4), we define the augmented Lagrangian dual function

$$g_\rho(\boldsymbol{\lambda}) := \min_{\boldsymbol{p} \in \mathcal{P}, \ (\boldsymbol{q}, \boldsymbol{z}) \in \mathcal{Q}} \mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{z}; \boldsymbol{\lambda}),$$

and the corresponding augmented Lagrangian dual problem

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^{nm}} \ g_\rho(\boldsymbol{\lambda}). \tag{2.5}$$

We next present a finite exactness property of the augmented Lagrangian dual problem, which shows that the AL dual value coincides with the optimal value of the original binary program (2.1) if the penalty parameter is sufficiently large.

Theorem 2.1. *Assume that* (2.1) *is feasible, and let* f^* *denote the optimal value of* (2.1)*. Then there exists a finite constant* $\bar{\rho} > 0$ *such that, for every* $\rho > \bar{\rho}$*, the augmented Lagrangian dual problem* (2.5) *satisfies*

$$\max_{\boldsymbol{\lambda} \in \mathbb{R}^{nm}} \ g_\rho(\boldsymbol{\lambda}) = f^*.$$

Proof. Define

$$\mathcal{X} := \{ (\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{z}) : \ \boldsymbol{p} \in \mathcal{P}, \ (\boldsymbol{q}, \boldsymbol{z}) \in \mathcal{Q} \}.$$

Since $\mathcal{P}$ and $\mathcal{Q}$ are finite, the set $\mathcal{X}$ is finite. Let

$$\mathcal{X}_{\text{fea}} := \{(\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X} : \ \boldsymbol{p} = \boldsymbol{q}\}, \qquad \mathcal{X}_{\text{inf}} := \{(\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X} : \ \boldsymbol{p} \neq \boldsymbol{q}\}.$$

Because (2.1) is feasible and (2.3) is equivalent to (2.1), the set $\mathcal{X}_{\text{fea}}$ is nonempty, and

$$f^* = \min \ \{f(\boldsymbol{q}, z) : \ (\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X}_{\text{fea}}\}.$$

We consider the following two cases.

Case (i): If $\mathcal{X}_{\text{inf}} = \emptyset$, then every point in $\mathcal{X}$ already satisfies $\boldsymbol{p} = \boldsymbol{q}$. Hence, for every $\rho > 0$ and every $\boldsymbol{\lambda} \in \mathbb{R}^{nm}$,

$$g_\rho(\boldsymbol{\lambda}) = \min_{(\boldsymbol{p},\boldsymbol{q},z)\in\mathcal{X}_{\text{fea}}} f(\boldsymbol{q}, z) = f^*,$$

and the conclusion follows immediately.

Case (ii): If $\mathcal{X}_{\text{inf}} \neq \emptyset$, since $\mathcal{X}_{\text{inf}}$ is finite and $\boldsymbol{p}, \boldsymbol{q}$ are binary, we have

$$\delta := \min \{\|\boldsymbol{p} - \boldsymbol{q}\|^2 : \ (\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X}_{\text{inf}}\} > 0.$$

We define

$$M := \max \{f^* - f(\boldsymbol{q}, z) : \ (\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X}_{\text{inf}}\},$$

which is finite because $\mathcal{X}_{\text{inf}}$ is finite. Let

$$\bar{\rho} := \max \left\{0, \frac{2M}{\delta}\right\}.$$

Fix any $\rho > \bar{\rho}$. For every $(\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X}_{\text{inf}}$, we have

$$\mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, z; \boldsymbol{0}) = f(\boldsymbol{q}, z) + \frac{\rho}{2}\|\boldsymbol{p} - \boldsymbol{q}\|^2 \geq f(\boldsymbol{q}, z) + \frac{\rho}{2}\delta > f(\boldsymbol{q}, z) + M \geq f^*.$$

On the other hand, for every $(\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X}_{\text{fea}}$, we have $\boldsymbol{p} = \boldsymbol{q}$, and therefore

$$\mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, z; \boldsymbol{0}) = f(\boldsymbol{q}, z).$$

It follows that

$$g_\rho(\boldsymbol{0}) = \min_{(\boldsymbol{p},\boldsymbol{q},z)\in\mathcal{X}} \mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, z; \boldsymbol{0}) = \min_{(\boldsymbol{p},\boldsymbol{q},z)\in\mathcal{X}_{\text{fea}}} f(\boldsymbol{q}, z) = f^*.$$

Next, let $\boldsymbol{\lambda} \in \mathbb{R}^{nm}$ be arbitrary. For every $(\boldsymbol{p}, \boldsymbol{q}, z) \in \mathcal{X}_{\text{fea}}$, since $\boldsymbol{p} = \boldsymbol{q}$, we have

$$\mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, z; \boldsymbol{\lambda}) = f(\boldsymbol{q}, z).$$

Hence

$$g_\rho(\boldsymbol{\lambda}) \leq \min_{(\boldsymbol{p},\boldsymbol{q},z)\in\mathcal{X}_{\text{fea}}} \mathcal{L}_\rho(\boldsymbol{p}, \boldsymbol{q}, z; \boldsymbol{\lambda}) = \min_{(\boldsymbol{p},\boldsymbol{q},z)\in\mathcal{X}_{\text{fea}}} f(\boldsymbol{q}, z) = f^*.$$

Therefore,

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^{nm}} \ g_\rho(\boldsymbol{\lambda}) \leq f^*.$$

Since $g_\rho(\boldsymbol{0}) = f^*$, we also have

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^{nm}} \ g_\rho(\boldsymbol{\lambda}) \geq g_\rho(\boldsymbol{0}) = f^*.$$

Combining the two inequalities yields

$$\max_{\boldsymbol{\lambda}\in\mathbb{R}^{nm}} \ g_\rho(\boldsymbol{\lambda}) = f^*,$$

which completes the proof. □

Theorem 2.1 is in the same spirit as exact augmented Lagrangian duality results for integer optimization [22, 23]. In the present split formulation, the constant $\bar{\rho}$ can be viewed as an exactness threshold for the penalty parameter. Once ρ is sufficiently large, the augmented Lagrangian dual value coincides with the optimal value of the original binary problem. The exact value of $\bar{\rho}$ is not known in advance, since it depends on the unknown optimal value f^* and on the infeasible split points used in the proof. This motivates the adaptive penalty strategy used in the algorithm developed in the next section. We next present this decomposition algorithm.

3. Augmented Lagrangian decomposition method

Based on the reformulation in Section 2, we now develop an augmented Lagrangian decomposition (ALD) method for (2.3), which alternates between two primal block updates and one multiplier update. The first primal block corresponds to the exact-assignment constraints, while the second corresponds to the capacity-opening constraints. To enhance practical solution quality, we further incorporate primal recovery, local improvement, and an optional final refinement stage.

3.1. The ALD iteration and block updates

At iteration k, given the current iterate $(\boldsymbol{p}^k, \boldsymbol{q}^k, z^k)$, the multiplier vector $\boldsymbol{\lambda}^k$, and the penalty parameter $\rho_k > 0$, we perform the following updates:

$$\begin{cases} \boldsymbol{p}^{k+1} & \in \arg\min_{\boldsymbol{p}\in\mathcal{P}} \ \mathcal{L}_{\rho_k}(\boldsymbol{p}, \boldsymbol{q}^k, z^k; \boldsymbol{\lambda}^k), & (3.1) \\ (\boldsymbol{q}^{k+1}, z^{k+1}) & \in \arg\min_{(\boldsymbol{q},z)\in\mathcal{Q}} \ \mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}, z; \boldsymbol{\lambda}^k), & (3.2) \\ \boldsymbol{\lambda}^{k+1} & = \boldsymbol{\lambda}^k + \rho_k(\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}). \end{cases}$$

We next discuss how the two primal subproblems can be solved.

The $\boldsymbol{p}$ update. Given $(\boldsymbol{q}^k, z^k, \boldsymbol{\lambda}^k, \rho_k)$, the $\boldsymbol{p}$ subproblem (3.1) is

$$\boldsymbol{p}^{k+1} \in \arg\min_{\boldsymbol{p}\in\mathcal{P}} \ \mathcal{L}_{\rho_k}(\boldsymbol{p}, \boldsymbol{q}^k, z^k; \boldsymbol{\lambda}^k) = \arg\min_{\boldsymbol{p}\in\mathcal{P}} \left\{ {\boldsymbol{\lambda}^k}^\top \boldsymbol{p} + \frac{\rho_k}{2}\|\boldsymbol{p} - \boldsymbol{q}^k\|^2 \right\}, \tag{3.3}$$

where all terms involving $(\boldsymbol{q}^k, z^k)$ only are constant with respect to $\boldsymbol{p}$ and can therefore be dropped. Since the structure of $\mathcal{P}$ implies that the above problem is separable over the customers $i \in O$, (3.3) can be decomposed into n independent subproblems:

$$\min \left\{ \sum_{b\in\mathcal{B}} \lambda_{ib}^k p_{ib} + \frac{\rho_k}{2} \sum_{b\in\mathcal{B}} (p_{ib} - q_{ib}^k)^2 : \ \sum_{b\in\mathcal{B}} p_{ib} = 1, \ p_{ib} \in \{0,1\}, \ \forall b \in \mathcal{B} \right\}. \tag{3.4}$$

We now fix $i \in O$ and analyze problem (3.4). Due to the constraint $\sum_{b\in\mathcal{B}} p_{ib} = 1$ and the binary restriction $p_{ib} \in \{0,1\}$, exactly one component of $(p_{ib})_{b\in\mathcal{B}}$ must be equal to one, and all others must be zero. Suppose that customer i is assigned to facility b. Then $p_{ib} = 1, p_{ib'} = 0, \forall b' \in \mathcal{B}\backslash\{b\}$. Substituting this into the objective function in (3.4), we obtain

$$\sum_{b'\in\mathcal{B}} \lambda_{ib'}^k p_{ib'} + \frac{\rho_k}{2} \sum_{b'\in\mathcal{B}} (p_{ib'} - q_{ib'}^k)^2 = \lambda_{ib}^k + \frac{\rho_k}{2}(1 - q_{ib}^k)^2 + \frac{\rho_k}{2} \sum_{b'\neq b} (q_{ib'}^k)^2$$

$$= \lambda_{ib}^k + \frac{\rho_k}{2}(1 - q_{ib}^k) + \frac{\rho_k}{2}\sum_{b' \neq b} q_{ib'}^k$$
$$= \lambda_{ib}^k - \rho_k q_{ib}^k + \frac{\rho_k}{2}\sum_{b' \in \mathcal{B}} q_{ib'}^k + \frac{\rho_k}{2}, \tag{3.5}$$

where the second equation is due to $q_{ib}^k \in \{0, 1\}$. For a fixed i, choosing the best b amounts to minimizing the above expression over $b \in \mathcal{B}$. Since the last two terms in (3.5) are independent of the choice of b, an exact solution of the $\boldsymbol{p}$ subproblem is obtained by selecting

$$b_i^{k+1} \in \arg\min_{b \in \mathcal{B}} \lambda_{ib}^k - \rho_k q_{ib}^k, \tag{3.6}$$

and then setting

$$p_{ib_i^{k+1}}^{k+1} = 1, \qquad p_{ib}^{k+1} = 0, \quad \forall b \neq b_i^{k+1}. \tag{3.7}$$

In summary, the $\boldsymbol{p}$ subproblem admits an exact closed-form solution. For each $i \in O$, the update only requires a linear scan over all $b \in \mathcal{B}$. Hence, the total computational cost of the $\boldsymbol{p}$ update is $O(nm)$.

The $(\boldsymbol{q}, z)$ update. With $\boldsymbol{p}^{k+1}$ fixed, the $(\boldsymbol{q}, z)$ subproblem (3.2) is

$$(\boldsymbol{q}^{k+1}, z^{k+1}) \in \arg\min_{(\boldsymbol{q},z)\in Q} \mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}, z; \boldsymbol{\lambda}^k)$$
$$= \arg\min_{(\boldsymbol{q},z)\in Q} \left\{ f(\boldsymbol{q}, z) - \boldsymbol{\lambda}^{k\top}\boldsymbol{q} + \frac{\rho_k}{2}\|\boldsymbol{p}^{k+1} - \boldsymbol{q}\|^2 \right\}. \tag{3.8}$$

We next expand the quadratic term in (3.8):

$$\|\boldsymbol{p}^{k+1} - \boldsymbol{q}\|^2 = \sum_{i\in O}\sum_{b\in\mathcal{B}} \left((p_{ib}^{k+1})^2 - 2p_{ib}^{k+1} q_{ib} + q_{ib}^2 \right).$$

Since both $\boldsymbol{p}$ and $\boldsymbol{q}$ are binary, we have $(p_{ib}^{k+1})^2 = p_{ib}^{k+1},\ q_{ib}^2 = q_{ib}, \forall i \in O,\ \forall b \in \mathcal{B}$. Substituting these identities into (3.8), and removing all terms independent of $(\boldsymbol{q}, z)$, we obtain the equivalent problem

$$(\boldsymbol{q}^{k+1}, \boldsymbol{z}^{k+1}) \in \arg\min_{(\boldsymbol{q},z)\in Q} \sum_{i\in O}\sum_{b\in\mathcal{B}} \left(\left(c_{ib} - \lambda_{ib}^k \right) + \frac{\rho_k}{2}\left(1 - 2p_{ib}^{k+1} \right) \right) q_{ib} + \sum_{b\in\mathcal{B}} \alpha_b z_b.$$

Define

$$\psi_{ib}^k := \left(c_{ib} - \lambda_{ib}^k \right) + \frac{\rho_k}{2}\left(1 - 2p_{ib}^{k+1} \right), \quad \forall i \in O,\ \forall b \in \mathcal{B}. \tag{3.9}$$

Then the subproblem becomes

$$(\boldsymbol{q}^{k+1}, z^{k+1}) \in \arg\min_{(\boldsymbol{q},z)\in Q} \sum_{b\in\mathcal{B}} \left(\sum_{i\in O} \psi_{ib}^k q_{ib} + \alpha_b z_b \right). \tag{3.10}$$

The set Q is separable over the facilities $b \in \mathcal{B}$ since the capacity constraints are imposed independently for each facility and there is no coupling between different facilities. Therefore, (3.10) decomposes into m independent subproblems. For each fixed $b \in \mathcal{B}$, we solve

$$\begin{aligned} \min_{q_{\cdot b}, z_b} \quad & \sum_{i\in O} \psi_{ib}^k q_{ib} + \alpha_b z_b \\ \text{s.t.} \quad & \sum_{i\in O} w_i q_{ib} \le C_b z_b, \\ & z_b \in \{0, 1\},\ q_{ib} \in \{0, 1\}, \quad \forall i \in O, \end{aligned} \tag{3.11}$$

which is a fixed-charge binary knapsack subproblem. Its structure can be described directly from the value of z_b. We consider the following two cases:

(i) If $z_b = 0$, then the capacity constraint implies $\sum_{i\in O} w_i q_{ib} \le 0$. Since $w_i > 0$ and $q_{ib} \in \{0, 1\}$, this forces

$$q_{ib} = 0, \quad \forall i \in O.$$

In this case, the objective value is 0.

(ii) If $z_b = 1$, then the subproblem reduces to

$$\min \left\{ \alpha_b + \sum_{i\in O} \psi_{ib}^k q_{ib} : \sum_{i\in O} w_i q_{ib} \le C_b,\ q_{ib} \in \{0, 1\},\ \forall i \in O \right\}.$$

Equivalently, we may choose a subset $S \subseteq O$ such that $\sum_{i\in S} w_i \le C_b$, and the corresponding objective value is $\alpha_b + \sum_{i\in S} \psi_{ib}^k$.

Therefore, the optimal value of (3.11) can be written as

$$\min \left\{ 0,\ \alpha_b + \min \left\{ \sum_{i\in S} \psi_{ib}^k : S \subseteq O,\ \sum_{i\in S} w_i \le C_b \right\} \right\}. \tag{3.12}$$

In summary, the variable split transforms the coupled assignment-capacity structure into a collection of independent facility-wise subproblems. As a result, the $(\boldsymbol{q}, z)$ update decomposes into m independent subproblems, one for each $b \in \mathcal{B}$, and these subproblems can be solved in parallel.

Multiplier and penalty parameter update. After the two primal updates, the multiplier is updated by

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \rho_k(\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}). \tag{3.13}$$

We measure the violation of the linking equation by

$$r^{k+1} := \|\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}\|_1. \tag{3.14}$$

Since $\boldsymbol{p}^{k+1}$ and $\boldsymbol{q}^{k+1}$ are binary, the residual r^{k+1} is integer-valued. In particular, choosing $\varepsilon < 1$ enforces exact consensus, i.e., $r^{k+1} = 0$.

Following the discussion after Theorem 2.1, the penalty parameter is updated according to the residual sequence. The purpose is to increase the penalty when the violation of the linking equation does not decrease sufficiently. Let $\bar{K}$ be a positive integer and let $\theta \in (0, 1)$ be a fixed threshold. At iteration $k + 1$, we define

$$\mathcal{I}_{k+1} = \{\max\{1, k - \bar{K} + 2\}, \ldots, k + 1\}$$

and

$$R_{k+1}^{\min} = \min_{t\in\mathcal{I}_{k+1}} r^t, \qquad R_{k+1}^{\max} = \max_{t\in\mathcal{I}_{k+1}} r^t.$$

If $R_{k+1}^{\min} \ge \theta R_{k+1}^{\max}$, then the residual reduction over the recent iterations is regarded as insufficient, and we set

$$\rho_{k+1} = \min\{\gamma\rho_k, \rho_{\max}\}, \tag{3.15}$$

where $\gamma > 1$ is a fixed growth factor and $\rho_{\max} > 0$ is a prescribed upper bound. Otherwise, we set $\rho_{k+1} = \rho_k$.

The algorithm is terminated when the residual becomes sufficiently small, namely,

$$r^{k+1} \le \varepsilon,$$

where $\varepsilon > 0$ is a given tolerance.

3.2. Primal recovery and refinement

In addition to the main ALD iteration, we use the information contained in the current split variables to generate and improve feasible solutions for the original model. At iteration k, the block $\boldsymbol{p}^k$ reflects the current assignment pattern of customers to facilities, while the block $(\boldsymbol{q}^k, \boldsymbol{z}^k)$ provides complementary information on facility opening and capacity usage. Based on these two sources of information, we apply a primal recovery procedure during the ALD process and a final refinement procedure after the main loop. These steps are used to strengthen the quality of the incumbent solution while keeping the main ALD iteration unchanged.

Primal recovery. During the ALD iteration, the current split variables are used to construct feasible solutions of the original SSCFLP. At iteration k, the block p_k gives an assignment pattern satisfying the exact-assignment constraints, while (q_k, z_k) gives information on facility opening and capacity usage. We use these two sources to build a candidate facility set for each customer.

For each customer i, we construct a candidate facility set

$$C_i^k = \{b : \boldsymbol{p}_{ib}^k = 1\} \cup \{b : \boldsymbol{q}_{ib}^k = 1\} \cup \{b : \boldsymbol{z}_b^k = 1\} \cup \mathcal{N}_i,$$

where $\mathcal{N}_i$ contains a fixed number of facilities with the smallest assignment costs for customer i. Customers are processed in non-increasing order of demand. When customer i is considered, we first search the facilities in C_i^k that have enough residual capacity. Among these facilities, we assign i to the one with the smallest increase in the original objective value. If this facility is not yet open, it is opened and its fixed opening cost is included in the objective increase.

If no facility in C_i^k can accommodate customer i, we apply a fallback step and search all facilities with enough residual capacity. Customer i is then assigned to the feasible facility with the smallest objective increase. If no feasible facility exists, the recovery attempt is discarded and the incumbent is not updated at this iteration. After all customers have been assigned, empty facilities are closed and the objective value is recomputed. The recovered solution is then passed to the local improvement step.

Local improvement. After a feasible solution is recovered, we further improve it by simple feasibility-preserving neighborhood moves, including single-customer reassignment and facility-closing moves. These moves are applied to the incumbent and are accepted only when the objective value strictly decreases. A basic move is to reassign one customer from its current facility to another facility with sufficient residual capacity. Suppose that customer i is moved from facility b_0 to facility b. Let $\delta_b^+ \in \{0, 1\}$ indicate whether facility b is newly opened by this move, and let $\delta_{b_0}^- \in \{0, 1\}$ indicate whether facility b_0 becomes empty after this move. Then the corresponding objective change is

$$\Delta_{i:b_0 \to b} = (c_{ib} - c_{ib_0}) + \alpha_b \delta_b^+ - \alpha_{b_0} \delta_{b_0}^-.$$

The move is accepted only when the resulting solution remains feasible and $\Delta_{i:b_0 \to b} < 0$.

We also consider facility closing moves. In this case, one currently active facility is selected, and all customers assigned to it are reassigned to other facilities in a feasible way. The move is accepted only when feasibility is preserved and the total objective value decreases. These local improvement steps help remove redundant facility openings and improve the assignment pattern obtained from the recovery step.

Final refinement. After the main ALD loop terminates, we further apply a final refinement procedure to the best incumbent found during the run. This stage uses the same simple move types as the in-loop local improvement step, but with additional passes over the incumbent solution. As before, every candidate move must preserve feasibility and is accepted only when it improves the incumbent value. Therefore, the final refinement stage can only improve the best feasible solution already obtained during the ALD process.

Overall, the recovery and refinement stages provide a simple way to extract feasible solutions from the split iterates and improve the incumbent during the ALD process. They help maintain a high-quality feasible incumbent throughout the run and provide an additional polishing step after termination.

3.3. Overall ALD framework

We now summarize the overall ALD framework in Algorithm 1. We use the recovery and local improvement steps in Section 3.2 to maintain and improve a feasible incumbent solution of the original model. After the main loop terminates, we may further polish the best incumbent by a final refinement stage. We also show the decomposition and solving process in Figure 1.

Algorithm 1 ALD with primal recovery and final refinement

Initialize $(\boldsymbol{p}^0, \boldsymbol{q}^0, z^0)$, $\boldsymbol{\lambda}^0$, ρ_0, $k_{\max}$, and tolerance ε.
Initialize the best incumbent feasible solution $(\hat{\boldsymbol{x}}, \hat{z})$ as empty.
for $k = 0, 1, \ldots, k_{\max}$ **do**
  Update $\boldsymbol{p}^{k+1}$ by (3.6)–(3.7).
  Update $(\boldsymbol{q}^{k+1}, z^{k+1})$ by solving the independent subproblems (3.11).
  Update the multiplier $\boldsymbol{\lambda}^{k+1}$ by (3.13).
  Compute the residual r^{k+1} by (3.14).
  Perform primal recovery and local improvement to obtain a feasible solution $(\bar{\boldsymbol{x}}^{k+1}, \bar{z}^{k+1})$ from $(\boldsymbol{p}^{k+1}, \boldsymbol{q}^{k+1}, z^{k+1}, \boldsymbol{\lambda}^{k+1})$.
  If $(\bar{\boldsymbol{x}}^{k+1}, \bar{z}^{k+1})$ improves the incumbent value, update $(\hat{\boldsymbol{x}}, \hat{z}) \leftarrow (\bar{\boldsymbol{x}}^{k+1}, \bar{z}^{k+1})$.
  if $r^{k+1} \le \varepsilon$ **then**
    break
  end if
  Update the penalty parameter ρ_{k+1} according to (3.15).
end for
Apply the final refinement stage to the incumbent $(\hat{\boldsymbol{x}}, \hat{z})$.
Return the final split iterate $(\boldsymbol{p}^{k+1}, \boldsymbol{q}^{k+1}, z^{k+1})$ and the best incumbent solution $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$.

Algorithm 1 presents the complete ALD framework. Starting from an initial state $(\boldsymbol{p}^0, \boldsymbol{q}^0, z^0, \boldsymbol{\lambda}^0)$, the method alternates between the $\boldsymbol{p}$ update, the $(\boldsymbol{q}, z)$ update, and the multiplier update, with the penalty parameter adjusted according to the residual progress. At each iteration, the current split variables are used to recover a feasible solution, which is then improved by lightweight local moves and compared with the current incumbent. After termination, a final refinement step is applied to the best incumbent found during the run. The main computational advantage comes from the two primal block updates: the $\boldsymbol{p}$-block admits a simple customer-wise exact update, while the $(\boldsymbol{q}, z)$-block decomposes into independent fixed-charge binary knapsack subproblems that can be solved in parallel.

The recovery and refinement stages are built upon the ALD iterates and used to further improve the incumbent solution.

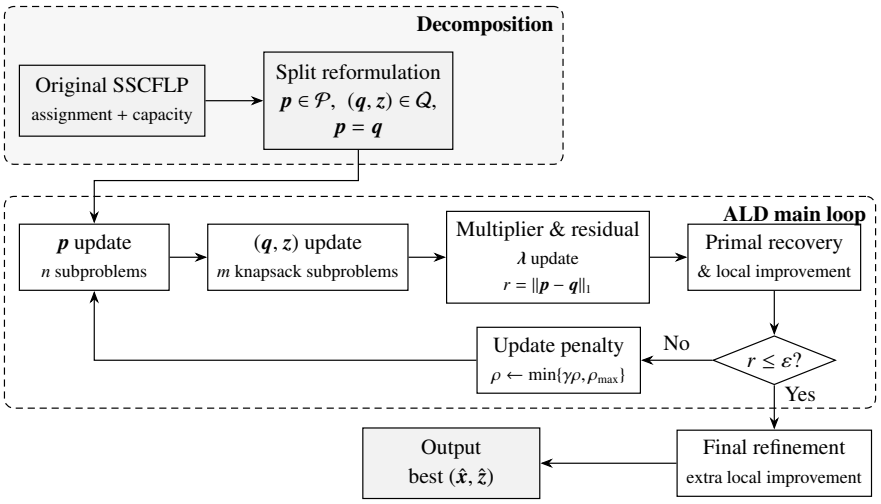


Figure 1. Decomposition and solving process of the proposed method.

4. Structural properties of the ALD iteration

In this section, we study several structural properties of the proposed ALD iteration. Since the SSCFLP is a binary optimization problem, the analysis is different from that of classical continuous augmented Lagrangian methods. We focus on properties that follow directly from the split formulation and the algorithmic updates. These include residual-based feasibility control, multiplier identities, and the descent property of one primal sweep with a fixed multiplier and penalty parameter.

Recall that the residual at iteration k is defined by

$$r^k := \|\boldsymbol{p}^k - \boldsymbol{q}^k\|_1,$$

which measures the disagreement between the two copies of the assignment variable. Since $\boldsymbol{p}^k$ and $\boldsymbol{q}^k$ are binary vectors, r^k is a nonnegative integer. Thus, if the stopping tolerance satisfies $\varepsilon < 1$, the condition $r^k \le \varepsilon$ is equivalent to $r^k = 0$, and hence to exact consensus $\boldsymbol{p}^k = \boldsymbol{q}^k$. In this case, $\boldsymbol{p}^k \in \mathcal{P}$ and $(\boldsymbol{q}^k, z^k) \in \mathcal{Q}$ imply that $(\boldsymbol{q}^k, z^k)$ is feasible for the original SSCFLP. For any scalar $t \in \mathbb{R}$, $(t)_+ := \max\{t, 0\}$ denotes its positive part. The next lemma shows that this residual also controls the violation of the original assignment and capacity constraints.

Lemma 4.1. *For any iterate $(\boldsymbol{p}^k, \boldsymbol{q}^k, z^k)$ generated by Algorithm 1, the following statements hold:*

$$\sum_{i \in O} \left| \sum_{b \in \mathcal{B}} q_{ib}^k - 1 \right| \le r^k, \tag{4.1}$$

$$\sum_{b \in \mathcal{B}} \left(\sum_{i \in O} w_i p_{ib}^k - C_b z_b^k \right)_+ \le w_{\max} r^k, \tag{4.2}$$

where $w_{\max} := \max_{i \in O} w_i$.

Proof. Since $\boldsymbol{p}^k \in \mathcal{P}$, we have

$$\sum_{b\in\mathcal{B}} p_{ib}^k = 1, \qquad \forall i \in \mathcal{O}.$$

Hence, for each $i \in \mathcal{O}$,

$$\left|\sum_{b\in\mathcal{B}} q_{ib}^k - 1\right| = \left|\sum_{b\in\mathcal{B}} \left(q_{ib}^k - p_{ib}^k\right)\right| \le \sum_{b\in\mathcal{B}} \left|q_{ib}^k - p_{ib}^k\right|.$$

Summing over $i \in \mathcal{O}$ gives (4.1).

Next, since $(\boldsymbol{q}^k, z^k) \in \mathcal{Q}$, we have

$$\sum_{i\in\mathcal{O}} w_i q_{ib}^k \le C_b z_b^k, \qquad \forall b \in \mathcal{B}.$$

Therefore,

$$\sum_{i\in\mathcal{O}} w_i p_{ib}^k - C_b z_b^k \le \sum_{i\in\mathcal{O}} w_i (p_{ib}^k - q_{ib}^k), \qquad \forall b \in \mathcal{B}.$$

Taking positive parts yields

$$\left(\sum_{i\in\mathcal{O}} w_i p_{ib}^k - C_b z_b^k\right)_+ \le \left(\sum_{i\in\mathcal{O}} w_i \left(p_{ib}^k - q_{ib}^k\right)\right)_+ \le \left|\sum_{i\in\mathcal{O}} w_i \left(p_{ib}^k - q_{ib}^k\right)\right| \le \sum_{i\in\mathcal{O}} w_i \left|p_{ib}^k - q_{ib}^k\right|.$$

Summing over $b \in \mathcal{B}$ and using $w_i \le w_{\max}$ gives

$$\sum_{b\in\mathcal{B}} \left(\sum_{i\in\mathcal{O}} w_i p_{ib}^k - C_b z_b^k\right)_+ \le \sum_{i\in\mathcal{O}} \sum_{b\in\mathcal{B}} w_i \left|p_{ib}^k - q_{ib}^k\right| \le w_{\max} \sum_{i\in\mathcal{O}} \sum_{b\in\mathcal{B}} \left|p_{ib}^k - q_{ib}^k\right| = w_{\max} r^k,$$

which proves (4.2). □

Lemma 4.1 shows that when r^k is small, $\boldsymbol{q}^k$ is close to satisfying the exact-assignment equations, while $(\boldsymbol{p}^k, z^k)$ is close to satisfying the capacity constraints of the original model.

Lemma 4.2. *For every $K \ge 1$, the multiplier sequence generated by Algorithm 1 satisfies*

$$\sum_{j=0}^{K-1} \rho_j (\boldsymbol{p}^{j+1} - \boldsymbol{q}^{j+1}) = \boldsymbol{\lambda}^K - \boldsymbol{\lambda}^0. \tag{4.3}$$

Moreover, for every $k \ge 0$,

$$\boldsymbol{\lambda}^{k\top} (\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}) + \frac{\rho_k}{2} \|\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}\|^2 = \frac{1}{2\rho_k} \left(\|\boldsymbol{\lambda}^{k+1}\|^2 - \|\boldsymbol{\lambda}^k\|^2\right). \tag{4.4}$$

Consequently,

$$\mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}^{k+1}, z^{k+1}; \boldsymbol{\lambda}^k) = f(\boldsymbol{q}^{k+1}, z^{k+1}) + \frac{1}{2\rho_k} \left(\|\boldsymbol{\lambda}^{k+1}\|^2 - \|\boldsymbol{\lambda}^k\|^2\right). \tag{4.5}$$

Proof. The multiplier update (3.13) gives

$$\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k = \rho_k(\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}).$$

Summing this identity from $k = 0$ to $K - 1$ yields (4.3).

Next,

$$\|\boldsymbol{\lambda}^{k+1}\|^2 = \|\boldsymbol{\lambda}^k\|^2 + 2\rho_k\boldsymbol{\lambda}^{k\top}(\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}) + \rho_k^2\|\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}\|^2.$$

Rearranging terms gives (4.4). Substituting (4.4) into the definition of the augmented Lagrangian yields (4.5). □

Lemma 4.2 follows directly from the multiplier update. It links the weighted violation of the consensus equation to the change in the multiplier and rewrites the coupling terms in the augmented Lagrangian through the multiplier variation. The multiplier update also gives a direct interpretation of the multiplier variation. From

$$\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k = \rho_k(\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}),$$

we have

$$\|\boldsymbol{\lambda}^{k+1} - \boldsymbol{\lambda}^k\|_\infty = \rho_k\|\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}\|_\infty.$$

Since $\boldsymbol{p}^{k+1}$ and $\boldsymbol{q}^{k+1}$ are binary vectors,

$$\|\boldsymbol{p}^{k+1} - \boldsymbol{q}^{k+1}\|_\infty \in \{0, 1\}.$$

Thus, the multiplier does not change when the two assignment copies agree. If they differ in at least one component, then the infinity norm of the multiplier change is exactly ρ_k.

Proposition 4.1. *For every iteration k, the updates of* $\boldsymbol{p}$ *and* $(\boldsymbol{q}, z)$ *with fixed multiplier* $\boldsymbol{\lambda}^k$ *and penalty parameter* ρ_k *satisfy*

$$\mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}^{k+1}, z^{k+1}; \boldsymbol{\lambda}^k) \le \mathcal{L}_{\rho_k}(\boldsymbol{p}^k, \boldsymbol{q}^k, z^k; \boldsymbol{\lambda}^k). \tag{4.6}$$

Proof. Since $\boldsymbol{p}^{k+1}$ is an exact minimizer of the p subproblem (3.1), we have

$$\mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}^k, z^k; \boldsymbol{\lambda}^k) \le \mathcal{L}_{\rho_k}(\boldsymbol{p}^k, \boldsymbol{q}^k, z^k; \boldsymbol{\lambda}^k).$$

Similarly, since $(\boldsymbol{q}^{k+1}, z^{k+1})$ is an exact minimizer of the (q, z) subproblem (3.2), we have

$$\mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}^{k+1}, z^{k+1}; \boldsymbol{\lambda}^k) \le \mathcal{L}_{\rho_k}(\boldsymbol{p}^{k+1}, \boldsymbol{q}^k, z^k; \boldsymbol{\lambda}^k).$$

Combining the two inequalities yields (4.6). □

Proposition 4.1 records the decrease produced by one primal sweep when λ^k and ρ_k are held fixed. It shows that the two exact primal block updates decrease the current augmented Lagrangian value. Together with the residual and multiplier relations above, this property describes how the main ALD iteration proceeds.

5. Numerical experiments

In this section, we test the practical performance of the proposed method on both random and benchmark SSCFLP instances. The experiments assess the quality of the feasible solutions produced by the method, the reliability of the proposed framework on standard benchmark instances, and its performance on large instances.

5.1. Experimental setup

We consider two versions of the proposed method. ALDR combines the ALD iteration, primal recovery, and local improvement. ALDRR is the full version, which further applies the final refinement step to the best feasible solution produced by ALDR. In the implementation, each facility-wise subproblem in the (q, z) update is solved exactly by a dynamic programming procedure for the corresponding fixed-charge knapsack problem. Gurobi is used as a benchmark solver in the numerical comparison. All experiments are carried out on a computer with an Intel Core i5-11400H @ 2.70GHz CPU and 16 GB RAM. The proposed method is implemented in MATLAB. All runs use one thread.

ALDR and ALDRR are terminated when the stopping criterion of the algorithm is satisfied, subject to the prescribed iteration limit. By contrast, Gurobi is run with a fixed time limit. For ALDR and ALDRR, we report the best feasible value found and the actual running time at termination. For Gurobi, we report the best feasible value found within the given time limit. The main parameters of the proposed method are the maximum number of outer iterations $k_{\max}$, the stopping tolerance ε, the initial penalty parameter ρ_0, the penalty update factor γ, and the time budget T_{ref} used in the final refinement step. In all experiments, we set $k_{\max} = 1000, \varepsilon = 10^{-6}$. The initial penalty parameter ρ_0 is chosen according to the problem scale. Specifically, we set $\rho_0 = 0.65$ for the random instances, $\rho_0 = 0.5$ for the medium-size benchmark instances, and $\rho_0 = 0.45$ for the large-size benchmark instances. The penalty update uses the residual-based rule described in Section 3.1 with growth factor $\gamma = 1.25$. The same update rule is used for all instances, and no instance-specific tuning is applied. The time budget used in the final refinement step is set to $T_{\mathrm{ref}} = 200$ seconds.

5.2. Results on random instances

We first test the method on randomly generated feasible SSCFLP instances of increasing scale. The purpose of this test is to examine how the proposed framework behaves as the problem scale grows, to evaluate the contribution of the final refinement step, and to compare the best feasible values obtained by the two methods.

We consider three size groups, namely, $(n, m) = (500, 100)$, $(1000, 200)$, and $(2000, 400)$, and generate four independent instances for each group. The random instances are generated to preserve the main structural features of the SSCFLP. Customer demands are drawn from several distributions with different ranges and dispersions. Facility capacities are then generated from the total demand with a prescribed capacity ratio, which guarantees the feasibility of every instance. The assignment cost matrix combines a random background component with a small set of preferred low-cost facilities for each customer. As a result, the generated instances retain a clear location-allocation structure rather than becoming completely unstructured random cost matrices. In this test, the fixed opening costs are set to a common value, so that the comparison focuses on the interaction among assignment, opening, and capacity decisions.

For this test set, we are mainly interested in whether the final refinement step improves the feasible solution returned by ALDR, and how the solution quality of ALDRR compares with that of Gurobi under the same total time budget. To measure these two aspects, we use the following two quantities:

$$\mathrm{Gain}_{\mathrm{RR}}(\%) = 100 \times \frac{f_{\mathrm{ALDR}} - f_{\mathrm{ALDRR}}}{f_{\mathrm{ALDR}}}, \qquad \mathrm{Gap}_{\mathrm{G}}(\%) = 100 \times \frac{f_{\mathrm{ALDRR}} - f_{\mathrm{Gurobi}}}{f_{\mathrm{Gurobi}}}, \tag{5.1}$$

where f_{ALDR}, f_{ALDRR}, and f_{Gurobi} denote the best feasible objective values returned by ALDR, ALDRR,

and Gurobi, respectively. A larger value of Gain_{RR} means that the final refinement step gives a larger improvement over ALDR, while a negative value of Gap_{G} means that ALDRR gives a better feasible objective value than the best feasible value found by Gurobi within the given time limit.

Table 2. Performance comparison of ALDRR, ALDR, and Gurobi on large-scale random instances.

Scale	Inst.	ALDRR		ALDR		Gurobi	Gap_{G}	Gain_{RR}
		Obj	Time	Obj	Time	Obj		
$n = 500$ $m = 100$	I1	3514.0	13.3	4456.0	8.0	3623.0	-3.01	21.14
	I2	3590.0	18.6	5064.0	8.8	3708.0	-3.18	29.11
	I3	3324.0	19.4	4454.0	9.0	3508.0	-5.25	25.37
	I4	4266.0	29.3	5933.0	10.0	4367.0	-2.31	28.10
	Avg.	3673.5	20.2	4976.8	8.9	3801.5	-3.44	25.93
$n = 1000$ $m = 200$	I1	11162.0	78.0	17434.0	51.7	12117.0	-7.88	35.98
	I2	12317.0	79.9	15357.0	64.2	12460.0	-1.15	19.80
	I3	13053.0	155.4	18018.0	63.5	13645.0	-4.34	27.56
	I4	14331.0	55.2	18049.0	24.6	14371.0	-0.22	20.62
	Avg.	12715.6	92.1	17214.5	51.0	13148.3	-3.39	25.99
$n = 2000$ $m = 400$	I1	43763.0	492.0	47640.0	404.8	45581.0	-3.99	8.14
	I2	40198.0	341.6	43951.0	314.3	48071.0	-16.38	8.54
	I3	41985.0	453.1	47068.0	348.9	41736.0	0.60	10.80
	I4	42386.0	396.1	46172.0	366.5	47761.0	-11.25	8.20
	Avg.	42083.0	420.7	46207.8	358.6	45172.0	-7.76	8.92

*Note: Obj = best feasible objective value. For Gurobi, Obj denotes the best feasible value obtained within the 1000-second time limit. Negative values of Gap_{G} mean that ALDRR gives a better feasible value than Gurobi.

The results are summarized in Table 2. It is worth noting that the Time column for ALDR and ALDRR records the time at which the reported best value is first reached, while for Gurobi, it reports the total time limit of 1000 seconds. We first compare ALDRR with ALDR. On all tested instances, ALDRR improves upon ALDR. The average improvement measured by Gain_{RR} is 25.93%, 25.99%, and 8.92% for $(n, m) = (500, 100)$, $(1000, 200)$, and $(2000, 400)$, respectively. This shows that the final refinement step makes a clear contribution beyond the solutions already obtained by ALDR. We next compare ALDRR with the best feasible values found by Gurobi within the 1000-second time limit. On all three instance groups, ALDRR gives a lower average objective value than Gurobi. The average gaps are -3.44%, -3.39%, and -7.76%, respectively. ALDRR outperforms Gurobi on most individual instances, and the advantage becomes more pronounced as the problem scale increases.

Overall, the random-instance results show that the final refinement step makes a clear contribution and that the full method can produce high-quality feasible solutions efficiently.

5.3. Impact of capacity tightness

To examine how capacity constraint tightness affects the convergence behavior and solution quality of the proposed method, we conduct an experiment at the largest random-instance scale $(n, m) = (2000, 400)$. The tightness ratio is defined as $\tau = \sum_{b\in\mathcal{B}} C_b / \sum_{i\in O} w_i$, where smaller τ corresponds to tighter capacity. We generate eight instances with the same customer set and assignment costs but with different total capacity levels, giving $\tau = 1.5, 2.5, \ldots, 8.5$. Apart from capacity scaling, the generation procedure follows that described in Section 5.2, which guarantees the feasibility of every instance. Gurobi is run with a 1000-second time limit on each instance.

We use $\mathrm{Gap}_{\mathrm{G}}$ defined in (5.1) to measure solution quality relative to Gurobi, and report the total running time T_{total} of ALDRR. The results are summarized in Table 3, which shows that ALDRR remains effective under different capacity tightness levels. For all tested values of τ, ALDRR obtains a lower objective value than Gurobi within less than 70 seconds, while Gurobi is given a 1000-second time limit. The running time increases when the capacity becomes tighter, but it remains moderate for all cases. Moreover, all runs terminate with $r^k = 0$, which means that exact consensus $\boldsymbol{p}^k = \boldsymbol{q}^k$ is reached.

Table 3. Effect of the capacity tightness ratio τ on solution quality and running time for instances with $(n, m) = (2000, 400)$.

τ	f_{ALDR}	f_{ALDRR}	f_{Gurobi}	Gap_G (%)	T_{total} (s)
1.5	110774	110442	110790	-0.31	67.35
2.5	70388	67642	68574	-1.36	47.67
3.5	50289	48558	50882	-4.57	26.11
4.5	42885	41417	43722	-5.27	22.73
5.5	39271	37826	45275	-16.45	21.16
6.5	34770	34033	42265	-19.48	18.93
7.5	30019	27320	31316	-12.76	19.04
8.5	30080	29321	39473	-25.72	26.17

*Note: $\mathrm{Gap}_G = 100\,(f_{\mathrm{ALDRR}} - f_{\mathrm{Gurobi}})/f_{\mathrm{Gurobi}}$. T_{total} is the total running time of ALDRR. All runs reached $r^k = 0$ (exact consensus $p^k = q^k$).

5.4. Results on benchmark instances

We next test the method on benchmark instances from the Beasley data set [35]. These instances examine the practical behavior of the proposed method on standard SSCFLP test sets. We distinguish between medium-sized and large-sized benchmark instances because the computational behavior is clearly different on these two scales.

Medium-sized instances We first consider the medium-sized benchmark instances `cap61`–`cap134`. This set is organized into six groups, namely ,`cap6`, `cap7`, `cap9`, `cap10`, `cap12`, and `cap13`, and each group contains four instances. Hence, the medium benchmark set contains 24 instances in total.

Accordingly, Table 4 reports summary results by group, where #Inst denotes the number of instances in each group.

Since these instances are relatively small, Gurobi can solve them very quickly and certify optimality. Therefore, the main question on this set is not whether ALDRR is faster than Gurobi, but whether the proposed framework remains reliable on standard benchmark data. For this reason, we use the match rate (the percentage of instances where ALDRR achieves the optimal solution) as the main summary measure.

Table 4. Performance comparison of ALDRR, ALDR, and Gurobi on medium benchmark instances.

Scale	Group	#Inst	Match rate (%)	ALDRR Avg. time	Gurobi Avg. time	ALDRR Max. time
$(n, m) =$ $(50, 16)$	cap6	4	100	0.37	0.10	0.42
	cap7	4	100	0.26	0.09	0.40
$(n, m) =$ $(50, 25)$	cap9	4	100	0.18	0.15	0.24
	cap10	4	100	0.25	0.13	0.36
$(n, m) =$ $(50, 50)$	cap12	4	100	0.38	0.28	0.45
	cap13	4	100	0.30	0.13	0.53
	All	24	100	0.29	0.15	0.53

The results in Table 4 show that ALDRR matches the Gurobi optimum on all 24 tested instances, giving a match rate of 100%. At the same time, ALDRR is computationally efficient on this set. Its average running time over all 24 instances is 0.29 second, and the maximum running time is 0.53 second. These results show that the proposed framework is reliable on medium-sized standard benchmark instances.

Large-scale instances We next consider the large benchmark instances `capa1`–`capc4`. All these instances have scale $(n, m) = (1000, 100)$. For this test set, the main question is the quality of the feasible incumbent under different time budgets. Accordingly, Table 5 gives the best feasible values obtained within 10, 30, 100, and 200 seconds, together with the corresponding Best_Time, that is, the first time at which the reported best feasible value is attained. We also report

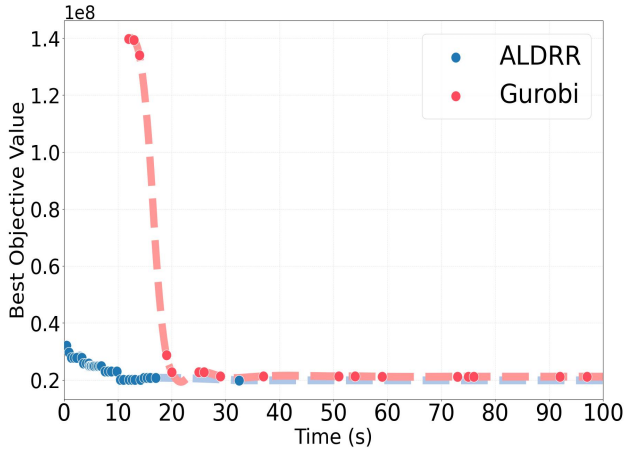
$$\text{Gap@30s}(\%) = 100 \times \frac{f^{30}_{\text{ALDRR}} - f^{30}_{\text{Gurobi}}}{f^{30}_{\text{Gurobi}}},$$

where f^{30}_{ALDRR} and f^{30}_{Gurobi} denote the best feasible objective values obtained by ALDRR and Gurobi within 30 seconds, respectively. A negative value of Gap@30s means that ALDRR gives a better incumbent at 30 seconds.

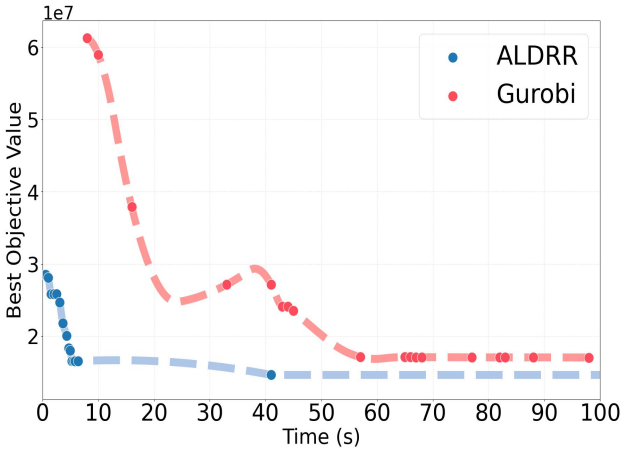
Table 5. Time-limited comparison on large benchmark instances. Objective values are reported in units of 10^6.

Inst.	Method	10s	30s	100s	200s	Best_Time	Gap@30s
capa1	ALDRR	20.63	19.85	19.85	19.85	32.1	-6.69
	Gurobi	–	21.27	21.14	19.96	199.0	
capa2	ALDRR	21.74	19.86	18.63	18.63	63.4	0.63
	Gurobi	33.69	19.73	19.60	18.83	195.1	
capa3	ALDRR	21.92	18.57	18.57	18.57	30.6	-6.78
	Gurobi	28.33	19.92	19.82	18.70	199.0	
capa4	ALDRR	20.67	17.55	17.55	17.55	30.1	-3.44
	Gurobi	26.89	18.17	18.17	17.75	198.0	
capb1	ALDRR	15.65	14.63	14.63	14.63	41.2	-60.14
	Gurobi	61.30	36.71	16.62	13.66	199.0	
capb2	ALDRR	15.22	14.37	14.07	14.07	60.4	-8.42
	Gurobi	16.72	15.69	14.09	13.72	198.0	
capb3	ALDRR	15.44	14.03	13.85	13.85	81.3	-10.36
	Gurobi	15.65	15.65	13.94	13.86	200.0	
capb4	ALDRR	15.37	13.86	13.10	13.10	46.3	-9.73
	Gurobi	16.96	15.36	14.16	13.72	198.0	
capc1	ALDRR	12.89	12.61	11.65	11.65	33.3	-11.20
	Gurobi	18.51	14.20	12.50	11.65	199.0	
capc2	ALDRR	12.25	11.86	11.86	11.86	31.3	-16.77
	Gurobi	14.75	14.25	12.92	12.25	199.0	
capc3	ALDRR	11.97	11.57	11.57	11.57	15.6	-9.56
	Gurobi	14.38	12.79	11.77	11.56	196.0	
capc4	ALDRR	11.79	11.64	11.64	11.64	61.2	-5.33
	Gurobi	54.22	12.51	11.86	11.72	196.0	

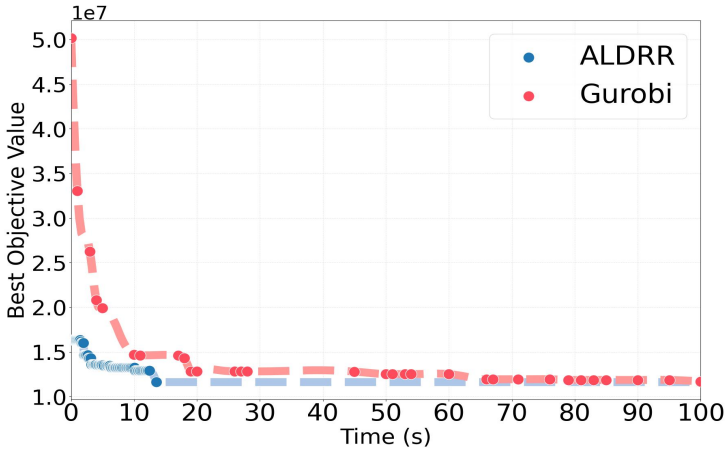
*Note: Entries at 10s, 30s, 100s, and 200s are the best feasible objective values obtained within the corresponding time budget. Best_Time denotes the first time at which the reported best feasible value is attained. Gap@30s is computed as $(f^{30}_{\text{ALDRR}} - f^{30}_{\text{Gurobi}})/f^{30}_{\text{Gurobi}} \times 100\%$. A negative Gap@30s value means that ALDRR achieves a lower objective value at 30 seconds. All objective values are scaled by 10^6. "–" means that no feasible value was available within the corresponding time budget.



(a) Instance `capa1`



(b) Instance `capb1`



(c) Instance `capc1`

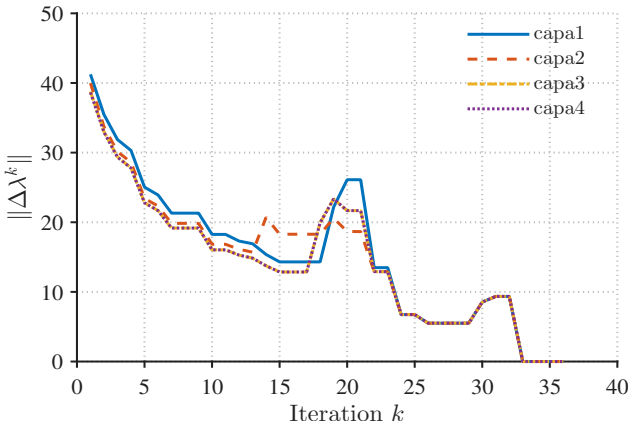
Figure 2. Time–objective curves on three representative large benchmark instances.

Several observations can be made from Table 5. First, ALDRR is usually more competitive in the early stage. At 10 seconds, it gives a much better incumbent than Gurobi on most tested instances, and this advantage remains visible at 30 seconds in most cases. In particular, `Gap@30s` is negative on 11 of the 12 instances, which shows that ALDRR usually provides a better feasible solution within this time budget. The advantage is especially clear on the `capb` group. For example, on instance `capb1`, the 30-second incumbent of ALDRR is 14.63, while the corresponding Gurobi value is 36.71, giving `Gap@30s` $= -60.14\%$. Second, ALDRR often reaches its final reported value much earlier than Gurobi. For all 12 instances, the best value of ALDRR is first reached within about 15.6 to 81.3 seconds. By contrast, the best values reported for Gurobi are typically obtained close to the 200-second limit shown in the table. This indicates that ALDRR tends to identify a strong incumbent early, while Gurobi often needs a longer search process before approaching the same quality level. Third, the advantage of ALDRR mainly lies in early-stage solution quality, rather than uniform dominance across all entries. For example, on `capa2`, the 30-second value of Gurobi is slightly better, and on `capc1`, the two methods reach the same best value in the end. Even in such cases, however, ALDRR reaches a strong incumbent earlier and remains competitive throughout the run. Taken together, these results show that the proposed

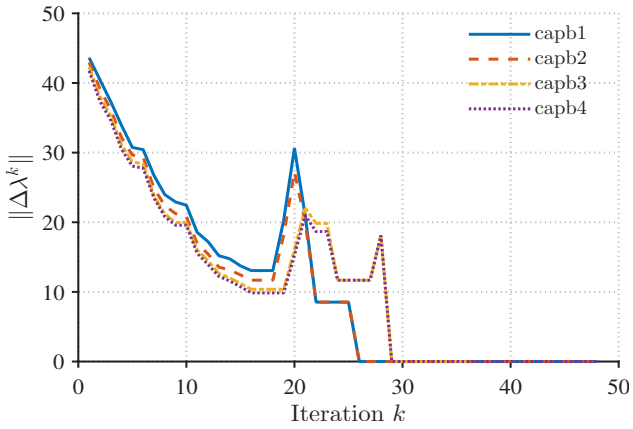
framework is particularly effective when the goal is to obtain a high-quality feasible solution within a limited time budget.

The time–objective curves in Figure 2 further support the same conclusion on three representative instances, namely, capa1, capb1, and capc1. Since the main advantage of ALDRR lies in the early stage of the run, this figure focuses on the first 100 seconds. On capa1, ALDRR decreases rapidly to its final incumbent value, while Gurobi improves more gradually and only approaches a comparable value much later. On capb1, the difference is even more pronounced: ALDRR reaches a strong incumbent very early, whereas Gurobi starts from a much weaker incumbent and needs substantially more time to close the gap. On capc1, both methods eventually reach the same best value, but ALDRR again gets there substantially earlier. Overall, ALDRR produces high-quality feasible solutions quickly, and this early incumbent advantage is the main strength of the proposed method on large benchmark instances.

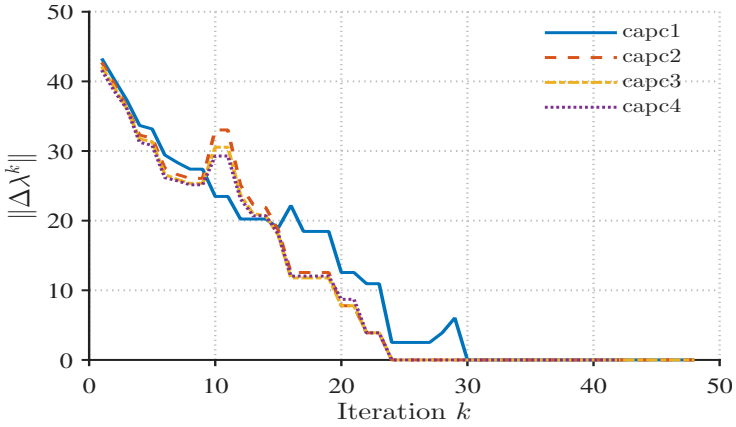
To further illustrate the behavior of the multiplier update, Figure 3 displays the multiplier change $\|\lambda^{k+1} - \lambda^k\|$ during the ALD iteration on all twelve large benchmark instances, grouped by instance family (capa, capb, capc). Across all instances, the multiplier change decreases to zero. This behavior is consistent with the multiplier identity in Section 4, where the multiplier variation directly reflects whether consensus holds at each iteration.



(a) Instances capa1–capa4



(b) Instances capb1–capb4



(c) Instances capc1–capc4

Figure 3. Multiplier change $\|\lambda^{k+1} - \lambda^k\|$ on large benchmark instances.

6. Conclusions

In this paper, we study a structured binary optimization model arising from the single-source capacitated facility location problem. To exploit its structure, we introduce a split reformulation that separates the assignment variables from the capacity-opening variables and links them through consensus equations. Based on this reformulation, we propose an augmented Lagrangian decomposition framework with tractable primal subproblems. The ALDRR method integrates the basic ALD iteration with primal recovery, local improvement, and a final refinement step to improve the incumbent solution during the run.

We prove that the augmented Lagrangian dual problem achieves zero duality gap for the original binary problem when the penalty parameter is sufficiently large. We also establish several basic properties of the proposed iteration, including the relation between the consensus residual and primal feasibility, multiplier identities, the interpretation of multiplier variation, and the descent property of one primal sweep.

The numerical results show that the proposed framework is effective on both random and benchmark SSCFLP instances. On medium-sized benchmark instances, ALDRR matched the Gurobi optimum on all tested cases. On large benchmark instances, the main advantage of the proposed method is that it produces strong feasible solutions within limited time budgets. The results also show that the final refinement step makes a clear contribution to the overall performance. Although this paper focuses on the SSCFLP, the same splitting idea may also be useful for other structured binary programs with coupled constraint blocks. Examples include the generalized assignment problem [36], capacitated p-median problems [37], and two-stage stochastic binary programs with capacity linking [38]. For these problems, the main idea is still to separate the constraint blocks and handle the linking relation by an augmented Lagrangian term. The subproblem solvers and recovery rules should be adapted to the specific model. A detailed study of these extensions is left for future work. Overall, the proposed framework provides an effective way to combine augmented Lagrangian decomposition with problem-specific incumbent improvement for large-scale structured binary optimization.

Author contributions

Rui Wang: Writing–original draft, methodology, conceptualization, writing–review and editing, supervision, resources; Deshan Hong: Writing–review and editing, methodology, data curation, validation, visualization, conceptualization.

Use of Generative-AI tools declaration

During the preparation of this manuscript, the authors utilized Claude Code and ChatGPT for assistance with proofreading and language polishing. The authors thoroughly reviewed and revised all AI-assisted content and are solely responsible for the final version.

Acknowledgments

This research was supported by the Sichuan Science and Technology Program under Grant 2026NSFSC0789 and the National Natural Science Foundation of China under Grant 12401414.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. A. Klose, A. Drexl, Facility location models for distribution system design, *European J. Oper. Res.*, **162** (2005), 4–29. https://doi.org/10.1016/j.ejor.2003.10.031
2. Z.-J. M. Shen, Integrated supply chain design models: A survey and future research directions, *J. Ind. Manag. Optim.*, **3** (2007), 1–27. https://doi.org/10.3934/jimo.2007.3.1
3. M. T. Melo, S. Nickel, F. Saldanha-da-Gama, Facility location and supply chain management — A review, *European J. Oper. Res.*, **196** (2009), 401–412. https://doi.org/10.1016/j.ejor.2008.05.007
4. Y. Costa, T. Melo, Facility location modeling in supply chain network design: Current state and emerging trends, In: *The Palgrave Handbook of Supply Chain Management*, Palgrave Macmillan, Cham, 2022, 1–36. https://doi.org/10.1007/978-3-030-89822-9_101-1
5. K. Holmberg, M. Rönnqvist, D. Yuan, An exact algorithm for the capacitated facility location problems with single sourcing, *European J. Oper. Res.*, **113** (1999), 544–559. https://doi.org/10.1016/S0377-2217(98)00008-3
6. Z. Yang, F. Chu, H. Chen, A cut-and-solve based algorithm for the single-source capacitated facility location problem, *European J. Oper. Res.*, **221** (2012), 521–532. https://doi.org/10.1016/j.ejor.2012.03.047
7. S. L. Gadegaard, A. Klose, L. R. Nielsen, An improved cut-and-solve algorithm for the single-source capacitated facility location problem, *EURO J. Comput. Optim.*, **6** (2018), 1–27. https://doi.org/10.1007/s13675-017-0084-4
8. D. Weninger, L. A. Wolsey, Benders-type branch-and-cut algorithms for capacitated facility location with single-sourcing, *European J. Oper. Res.*, **310** (2023), 84–99. https://doi.org/10.1016/j.ejor.2023.02.042
9. R. K. Ahuja, J. B. Orlin, S. Pallottino, M. P. Scaparra, M. G. Scutellà, A multi-exchange heuristic for the single-source capacitated facility location problem, *Manage. Sci.*, **50** (2004), 749–760. https://doi.org/10.1287/mnsc.1030.0193
10. I. A. Contreras, J. A. Díaz, Scatter search for the single source capacitated facility location problem, *Ann. Oper. Res.*, **157** (2008), 73–89. https://doi.org/10.1007/s10479-007-0193-1
11. S. C. Ho, An iterated tabu search heuristic for the single source capacitated facility location problem, *Appl. Soft Comput.*, **27** (2015), 169–178. https://doi.org/10.1016/j.asoc.2014.11.004
12. R. D. Galvão, V. Marianov, Lagrangean relaxation-based techniques for solving facility location problems, In: *Foundations of Location Analysis*, Springer, 2011, 391–420. https://doi.org/10.1007/978-1-4419-7572-0_17
13. M. L. Fisher, The Lagrangian relaxation method for solving integer programming problems, *Manage. Sci.*, **27** (1981), 1–18. https://doi.org/10.1287/mnsc.27.1.1

14. G. B. Dantzig, P. Wolfe, Decomposition principle for linear programs, *Oper. Res.*, **8** (1960), 101–111. https://doi.org/10.1287/opre.8.1.101

15. J. F. Benders, Partitioning procedures for solving mixed-variables programming problems, *Numer. Math.*, **4** (1962), 238–252. https://doi.org/10.1007/BF01386316

16. F. Vanderbeck, L. A. Wolsey, Reformulation and decomposition of integer programs, In: *50 Years of Integer Programming 1958–2008*, Springer, Berlin, Heidelberg, 2010, 431–502. https://doi.org/10.1007/978-3-540-68279-0_13

17. F. Vanderbeck, M. W. P. Savelsbergh, A generic view of Dantzig–Wolfe decomposition in mixed integer programming, *Oper. Res. Lett.*, **34** (2006), 296–306. https://doi.org/10.1016/j.orl.2005.05.009

18. R. Rahmaniani, T. G. Crainic, M. Gendreau, W. Rei, The Benders decomposition algorithm: A literature review, *European J. Oper. Res.*, **259** (2017), 801–817. https://doi.org/10.1016/j.ejor.2016.12.005

19. J. P. Vielma, Mixed integer linear programming formulation techniques, *SIAM Rev.*, **57** (2015), 3–57. https://doi.org/10.1137/130915303

20. R. Wang, C. Zhang, S. Pu, J. Gao, Z. Wen, A customized augmented Lagrangian method for block-structured integer programming, *IEEE Trans. Pattern Anal. Mach. Intell.*, **46** (2024), 9439–9455. https://doi.org/10.1109/TPAMI.2024.3416514

21. R. Takapoui, N. Moehle, S. Boyd, A. Bemporad, A simple effective heuristic for embedded mixed-integer quadratic programming, *Internat. J. Control*, **93** (2020), 2–12. https://doi.org/10.1080/00207179.2017.1316016

22. N. L. Boland, A. C. Eberhard, On the augmented Lagrangian dual for integer programming, *Math. Program.*, **150** (2015), 491–509. https://doi.org/10.1007/s10107-014-0763-3

23. M. J. Feizollahi, S. Ahmed, A. Sun, Exact augmented Lagrangian duality for mixed integer linear programming, *Math. Program.*, **161** (2017), 365–387. https://doi.org/10.1007/s10107-016-1012-8

24. K. Sun, M. Sun, W. Yin, Decomposition methods for global solution of mixed-integer linear programs, *SIAM J. Optim.*, **34** (2024), 1206–1235. https://doi.org/10.1137/22M1487321

25. C.-H. Chen, C.-J. Ting, Applying multiple ant colony system to solve single source capacitated facility location problem, In: *Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2006)*, Lecture Notes in Computer Science, Springer, 2006, 508–509. https://doi.org/10.1007/11839088_55

26. E. S. Correa, M. T. A. Steiner, A. A. Freitas, C. Carnieri, A genetic algorithm for solving a capacitated p-median problem, *Numer. Algorithms*, **35** (2004), 373–388. https://doi.org/10.1023/B:NUMA.0000021767.42899.31

27. J. E. Beasley, Lagrangean heuristics for location problems, *European J. Oper. Res.*, **65** (1993), 383–399. https://doi.org/10.1016/0377-2217(93)90118-7

28. X. Gu, S. Ahmed, S. S. Dey, Exact augmented Lagrangian duality for mixed integer quadratic programming, *SIAM J. Optim.*, **30** (2020), 781–797. https://doi.org/10.1137/19M1271695

29. X. Bai, J. Sun, X. Zheng, An augmented Lagrangian decomposition method for chance-constrained optimization problems, *INFORMS J. Comput.*, **33** (2021), 1056–1069. https://doi.org/10.1287/ijoc.2020.1001

30. K. O. Jörnsten, M. Näsberg, P. A. Smeds, *Variable splitting: A new Lagrangean relaxation approach to some mathematical programming models,* Technical Report MAT-R-85-04, Department of Mathematics, Linköping Institute of Technology, Linköping, Sweden, 1985.

31. M. Guignard, S. Kim, Lagrangean decomposition: A model yielding stronger Lagrangean bounds, *Math. Program.*, **39** (1987), 215–228. https://doi.org/10.1007/BF02592954

32. J. Barcelo, E. Fernandez, K. O. Jörnsten, Computational results from a new Lagrangean relaxation algorithm for the capacitated plant location problem, *European J. Oper. Res.*, **53** (1991), 38–45. https://doi.org/10.1016/0377-2217(91)90091-9

33. E. J. Alenezy, Solving capacitated facility location problem using Lagrangian decomposition and volume algorithm, *Adv. Oper. Res.*, **2020** (2020), 1–7. https://doi.org/10.1155/2020/5239176

34. B. Wu, B. Ghanem, Lp-Box ADMM: A versatile framework for integer programming, *IEEE Trans. Pattern Anal. Mach. Intell.*, **41** (2019), 1695–1708. https://doi.org/10.1109/TPAMI.2018.2845842

35. J. E. Beasley, OR-Library: Distributing test problems by electronic mail, *J. Oper. Res. Soc.*, **41** (1990), 1069–1072. https://doi.org/10.1057/jors.1990.166

36. D. G. Cattrysse, L. N. Van Wassenhove, A survey of algorithms for the generalized assignment problem, *European J. Oper. Res.*, **60** (1992), 260–272. https://doi.org/10.1016/0377-2217(92)90077-M

37. A. Ceselli, G. Righini, A branch-and-price algorithm for the capacitated p-median problem, *Networks*, **45** (2005), 125–142. https://doi.org/10.1002/net.20059

38. L. Lozano, J. C. Smith, A binary decision diagram based algorithm for solving a class of binary two-stage stochastic programs, *Math. Program.*, **191** (2022), 381–404. https://doi.org/10.1007/s10107-018-1315-z

AIMS Press