Frequent pattern mining driven evolutionary search for cross-dock door assignment

Yongliang Lu[a], Jin-Kao Hao[b], Qinghua Wu[c], Mingjie Li[d,*]

[a]School of Economics and Management, Fuzhou University, 350108 Fuzhou, China
[b]LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France
[c]School of Management, Huazhong University of Science and Technology, 430074 Wuhan, China
[d]School of Information Engineering, Zhongnan University of Economics and Law, 430073 Wuhan, China

Computers & Operations Research 189: 107393, May 2026

Abstract

Capacitated cross-dock door assignment and uncapacitated cross-dock door assignment are two critical and challenging problems in supply chain management. This paper presents the first frequent pattern mining driven evolutionary algorithm to effectively solve these problems. The proposed approach incorporates a specialized data mining technique designed to extract significant frequent patterns from a collection of high-quality solutions, thereby guiding the search process. It also incorporates an efficient two-phase local optimization method that intensively inspects a given region to identify high-quality solutions, along with a quality-and-distance updating rule to manage the population of solutions. We evaluate the effectiveness of the proposed algorithm on popular benchmark instances of both problems. In particular, we report 26 improved best results (new upper bounds) out of 99 benchmark instances for the capacitated case and 25 improved best results out of 40 benchmark instances for the uncapacitated case. In addition, we show the importance of the two main search components, i.e., frequent pattern mining and local optimization. This research highlights the benefits of a collaboration between optimization algorithms and data mining methods. The code for our proposed algorithm will be made publicly available.

Keywords: Heuristics; Cross-dock door assignment; Frequent pattern mining; Evolutionary algorithm; Data mining driven optimization

1. Introduction

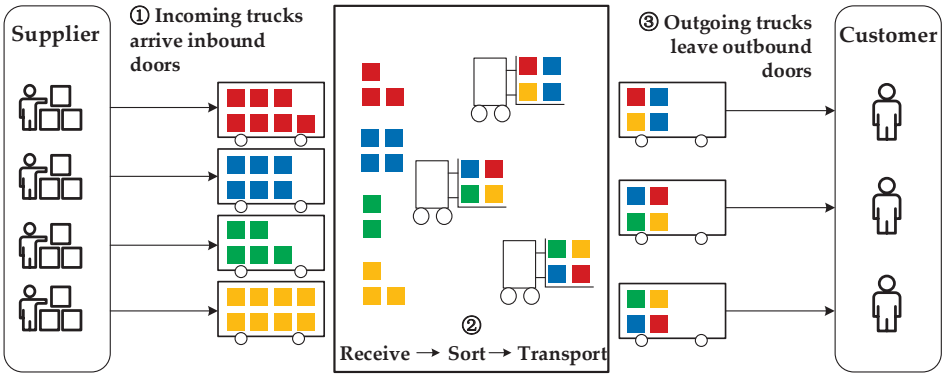
In logistics and supply chain management, efficient movement of goods through distribution centers is critical to meeting customer needs and reducing operational costs (Zhen & Li, 2022). Cross-docking (Boysen & Fliedner, 2010; Van Belle et al., 2012; Bodnar et al., 2017; Gaudioso et al., 2021), a method used to accelerate the movement of pallets of goods from inbound to outbound transportation doors without any storage (see Figure 1), has become a fundamental element in contemporary supply chain practices. Due to its advantages, cross-docking has become an effective logistics strategy, providing companies with valuable competitive benefits by accelerating material flow (Küçükoğlu & Öztürk, 2017, 2019, 2023). As

*Corresponding author
Email addresses: luyongliangly1@gmail.com (Yongliang Lu), jin-kao.hao@univ-angers.fr (Jin-Kao Hao), qinghuawu1005@gmail.com (Qinghua Wu), mingjie.li@zuel.edu.cn (Mingjie Li)

a complex system, cross-docking involves numerous optimization decisions throughout its operational cycle, including the location of cross-docks, cross-docking layout, cross-docking network design, truck-door assignment, and other related issues (Van Belle et al., 2012). Central to successful cross-docking is the cross-dock door assignment problem (CDAP) (Boysen & Fliedner, 2010), which presents a challenging optimization problem.

As shown in Figure 1, in a typical cross-docking operation, incoming trucks with pallets of commodities arrive at the cross-docking facility and unload the pallets through inbound doors. The unloaded pallets are immediately sorted and reorganized based on their destination. The reorganized pallets are then forklifted directly within the cross-dock to outbound doors where they are loaded onto outgoing trucks. The efficiency of this process heavily relies on the effective management of dock doors, which serve as the entry and exit points for incoming and outgoing trucks, respectively. The CDAP (Gelareh et al., 2020) involves the optimal assignment of incoming and outgoing trucks to the inbound and outbound dock doors in a cross-docking facility, with the objective of minimizing the total cost of transporting pallets from inbound to outbound doors while adhering to various constraints.

(a)



(b)

Figure 1: (a) Internal real scene of the cross-dock downloaded from https://smartgladiator.com/cross-docking-best-practices-with-benefits. (b) The entire process of a cross-docking system.

Over the past two decades, the CDAP has attracted considerable interest from researchers due to its critical applications in logistics and supply chain management. Various studies have examined numerous CDAP variants, taking into account a range of practical constraints. For example, different assignment

limitations have been identified, including scenarios where each door is dedicated to a single truck (Tsui & Chang, 1990, 1992; Tarhini et al., 2016), as well as situations where each door can accommodate several trucks (Zhu et al., 2009). Different capacity constraints have been investigated that address limits on outbound doors only (Tsui & Chang, 1990, 1992) or that consider both inbound and outbound doors (Gelareh et al., 2020). Additionally, various cross-docking terminal shapes, including I, L, U, T, H, and E layouts (Bartholdi & Gue, 2004), are discussed in the literature. The I-shaped layout, a rectangular design with receiving doors on one side and outbound doors on the other, is the most commonly studied (Gue, 1999; Bartholdi & Gue, 2000). This configuration allows for accurate distance simulation using rectilinear paths for forklifts along designated lanes. Moreover, various temporal constraints have been analyzed, including the placement of temporary storage buffers for goods (Yu & Egbelu, 2008), time windows for the arrival or departure of trucks (Lim et al., 2006) and the operating time assigned to each pallet (Miao et al., 2009; Gelareh et al., 2016). For an in-depth exploration of the CDAP classification, the reader is referred to (Boysen & Fliedner, 2010; Van Belle et al., 2012; Buijs et al., 2014; Gelareh et al., 2020).

This paper focuses on two well-known CDAPs: the capacitated cross-dock door assignment problem (CCDAP) and the uncapacitated cross-dock door assignment problem (UCDAP). In CCDAP, each door has specific capacity limitations. In contrast, in UCDAP, each door operates without capacity constraints and can handle, however, at most one truck at a time. To tackle the challenges associated with solving CCDAP and UCDAP, we introduce an innovative evolutionary algorithm driven by frequent pattern mining. This method aims to take advantage of combining optimization techniques with data mining within the framework of population-based evolutionary search. In particular, our proposed algorithm incorporates a frequent pattern mining procedure to extract valuable patterns that are used to guide the search. Extensive computational experiments illustrate the high performance of our algorithm compared to the best performing methods developed specifically for the CCDAP or UCDAP.

1.1. Literature review and motivations

This section provides an overview focusing on relevant studies of CCDAP and UCDAP, as well as search algorithms based on frequent pattern mining.

1.1.1. Related works on CCDAP and UCDAP

In CCDAP, capacity constraints are imposed on both inbound and outbound doors. Zhu et al. (2009) first extended the original UCDAP model of Tsui & Chang (1990) by integrating practical capacity constraints on the doors. They used the formulation of the 3-dimensional quadratic assignment problem to handle multiple-to-one assignments, treating it as a knapsack constraint affecting the capacity of the dock. They also developed a method for creating realistic test cases applicable to this problem. Two heuristics were proposed for the CCDAP by Guignard et al. (2012); one is an ad hoc approach using local search techniques, while the other is a metaheuristic termed CH, specifically designed for the 0-1 quadratic problems under linear constraints. Another mixed-integer programming formulation of CCDAP was introduced by Nassief et al. (2016), which was incorporated within the Lagrangian relaxation framework that exploits the unique structure of the problem to derive best solution bounds. To address large-scale scenarios, a primal heuristic was used in each iteration of the Lagrangian relaxation, yielding high-quality feasible solutions. Guemri et al. (2019) proposed two probabilistic tabu search algorithms tailored for the CCDAP, using an innovative large swap neighborhood structure advantageous for both CCDAP and similar problems. In their recent

research, Gelareh et al. (2020) examined eleven different mixed-integer programming models of CCDAP, establishing the equivalence among them and outlining their integrality characteristics. They also performed a comprehensive comparative analysis regarding the performance of these formulations on benchmark instances from the literature. Most recently, Li et al. (2024) investigated a flow-based mathematical model for the CCDAP that requires fewer variables and constraints than previously existing mathematical models in the literature. To address medium to large cases, they introduced an intelligent optimization algorithm. This algorithm utilized a Q-learning enhanced process to steer the search toward promising regions and employed a strategic oscillation technique to dynamically relax the capacity constraints on the doors for adaptable exploration of both feasible and infeasible solution spaces. This study provides new insights into the efficient integration of reinforcement learning with heuristics.

The UCDAP ignores door capacity constraints, where each inbound/outbound door can accommodate a maximum of one incoming/outgoing truck. Various research efforts have addressed the UCDAP challenge by introducing a range of optimization algorithms, including a genetic algorithm (Bermudez et al., 2001), a simulated annealing algorithm (Bozer & Carlo, 2008), an adaptive tabu search algorithm (Miao et al., 2014), a scatter search algorithm (Tarhini et al., 2016), a greedy algorithm (Zhang et al., 2018), and an ant colony algorithm (Zhang et al., 2018).

1.1.2. Related works on frequent pattern mining based search algorithms

Frequent patterns (Grahne & Zhu, 2003) refer to recurring combinations of items, transactions, or events that occur frequently within a dataset, offering utility for various data processing and analysis tasks. Data mining involves the identification of valuable rules and hidden patterns within datasets. Several successful examples in the literature have highlighted the usefulness of data mining in enhancing the performance of search algorithms for solving challenging optimization problems. For example, Wu et al. (2006) devised a data mining driven genetic algorithm aimed at identifying potentially high-quality genetic combinations to augment the efficiency of conventional genetic algorithms. Ribeiro et al. (2006) designed a hybrid version of the GRASP metaheuristic, incorporating a data mining process to solve the set packing problem. Kumar & Rao (2009) introduced a novel data mining based approach to ant colony optimization algorithm tailored for the flow shop scheduling problem. Their method used data mining techniques to extract insights from extensive flow shop schedules. Kshirsagar et al. (2012) developed a data mining powered genetic algorithm for intrusion detection systems. Ayadi et al. (2012) introduced a pattern-driven neighborhood search algorithm for the biclustering problem. Their approach is based on a solution representation encoded as a behavior matrix, along with a specialized neighborhood search that incorporates various pattern information. Raschip et al. (2015) developed a hybrid approach that utilizes a data mining module to guide an evolutionary algorithm in solving the constraint satisfaction problem. Tian et al. (2020) proposed an evolutionary algorithm based on pattern mining for tackling the sparse multi-objective optimization problems (SMOPs), which outperforms existing evolutionary algorithms in handling large SMOPs. Zhou et al. (2020a) presented a general frequent pattern based search method and applied the method to the classical quadratic assignment problem with great success.

In a recent study, Wu et al. (2023) introduced a frequent pattern based parallel search algorithm for scheduling agile earth observation satellites, which showed excellent performance over various problem scales. More recently, Liu et al. (2025) proposed an iterated adaptive large neighborhood search algorithm for the large-scale communication satellite range scheduling problem. Their algorithm employs a frequent

pattern mining method to extract problem-specific knowledge, which helps guide the algorithmic search process. Li et al. (2025) presented a highly effective hybrid evolutionary search approach that combines tabu search with frequent pattern mining to address the classical job shop scheduling problem. Zhang et al. (2025) proposed a frequent pattern based coevolutionary framework for solving the multi-component spectral feature selection problem.

Among these frequent pattern mining based search algorithms, frequent pattern mining based evolutionary algorithms (Wu et al., 2006; Kshirsagar et al., 2012; Tian et al., 2020; Zhou et al., 2020a; Li et al., 2025; Zhang et al., 2025) provide a robust framework for optimization problems, guiding the evolutionary search process towards solutions of superior quality. Evolutionary algorithms (Yu & Gen, 2010) serve as efficient tools for addressing optimization problems characterized by diverse properties and challenges. By embedding data mining within the framework of evolutionary algorithms, the resulting approach is expected to navigate the solution space in a targeted and informed way, effectively striking a balance between exploration and exploitation. However, there has been limited research on the application of frequent pattern mining based evolutionary algorithms to CDAPs. Therefore, in this study, we propose the frequent pattern mining driven evolutionary algorithm specifically tailored to address CDAPs. To the best of our knowledge, this is one of the first studies to apply a frequent pattern mining based evolutionary algorithm to CDAPs.

1.1.3. Motivations and contributions

The CCDAP and UCDAP addressed in this research are NP-hard problems that are computationally challenging (Guemri et al., 2019). Due to their significant practical implications and complex computational nature, a variety of methods have been proposed in the literature to tackle these problems. However, as highlighted in the review, many current approaches are designed to target either CCDAP or UCDAP individually. Even when methods are developed to solve both problems, their effectiveness is often not uniform and can show considerable variability across different benchmark cases.

Meanwhile, as shown in the existing literature (see e.g., Wu et al. (2006); Kumar & Rao (2009); Kshirsagar et al. (2012); Tian et al. (2020); Zhou et al. (2020a); Badhon et al. (2021)), pattern mining based evolutionary algorithms demonstrate superiority over other algorithms in addressing large challenging optimization problems. By exploiting the inherent structure and relationships in the data, frequent pattern mining based evolutionary algorithms provide a robust algorithmic framework for complex optimization problems, effectively guiding the exploration of the search space. To the best of our knowledge, research on the application of pattern mining based evolutionary algorithms to CDAPs is limited. This work fills that gap. Computational results show that the proposed algorithm competes favorably with the best-performing state-of-the-art CDAP methods reported in the literature.

Our key contributions can be outlined as follows:

- **Methodological contributions:** Motivated and inspired by previous studies demonstrating remarkable performance in various applications through the integration of data mining techniques with established heuristics, we propose the first frequent pattern mining driven evolutionary algorithm specifically tailored to address both CCDAP and UCDAP. By effectively integrating frequent pattern mining with powerful optimization techniques within a population-based evolutionary algorithm, our method facilitates a well-informed and focused exploration of the search space, ultimately leading

to the efficient achievement of high-quality solutions. Additionally, to enhance the search capabilities of our algorithm, an effective two-phase local exploration and focusing search, along with an advanced population management rule, are incorporated. The proposed frequent pattern mining driven algorithm represents a novel contribution to the field, offering a new perspective and potential advantages in addressing CDAPs.

- **Computational contributions:** We perform extensive experiments on two sets of 99 CCDAP benchmark instances and one set of 40 UCDAP benchmark instances, which are widely used in the literature, to evaluate the performance of our proposed algorithm. The experimental results show the competitiveness of our algorithm against the leading state-of-the-art algorithms. Specifically, for the 99 CCDAP instances, our algorithm achieves 26 new best results (new upper bounds) and matches the best-known solutions in all but one case. For the 40 UCDAP instances, our algorithm achieves 25 new best results (new upper bounds) and matches the best-known results in all but one scenario. This study provides new insights into the synergistic integration of data mining and optimization algorithms for improved efficiency and effectiveness.

Our proposed algorithm has a number of novelties compared to the existing methods. First, while most existing algorithms are designed to solve either CCDAP or UCDAP individually, our algorithm is capable of addressing both problems simultaneously. Second, our algorithm incorporates a data mining technique to extract frequent patterns from a set of solutions, which effectively guides the search process. Although the frequent pattern mining based algorithm has been applied to the generalized quadratic assignment problem (Zhou et al., 2020a), a problem similar to the one studied here, the CCDAP studied in this paper involves many-to-one assignments and includes capacity constraints, which brings significant challenges to the solution. The frequent pattern mining based algorithm proposed by Zhou et al. (2020a) for the generalized quadratic assignment problem is not applicable in this case. Moreover, the frequent pattern mining based algorithm proposed by Li et al. (2025) for the classical job shop scheduling problem is also not applicable, as the job shop scheduling problem and the CCDAP studied here are fundamentally different. Lastly, unlike existing local search-based algorithms (Guignard et al., 2012; Miao et al., 2014; Guemri et al., 2019; Li et al., 2024) for CDAPs, our approach incorporates a two-phase local search procedure that strategically explores the solution space and focuses the search to drive solution improvement.

This research presents new perspectives on the combined use of data mining and optimization techniques to achieve greater efficiency and effectiveness. Our experimental results highlight the potential of the algorithm to push the boundaries of current solutions for CDAPs addition, we will release the code of our algorithm to the public, which will support further studies on CDAPs and their practical applications.

The rest of the paper is organized as follows. In section 2, the problem is formulated along with its mathematical model. The proposed algorithm for CDAPs is detailed in section 3. Experimental results and a comparison with the latest state-of-the-art algorithms are presented in section 4. Section 5 examines the effects of the critical components of the proposed algorithm, and Section 7 concludes with final remarks.

2. Problem statement and mathematical formulation

As discussed in Section 1, various variants of the CDAP have been proposed in the literature, each incorporating different constraints (Boysen & Fliedner, 2010; Van Belle et al., 2012; Buijs et al., 2014; Gelareh

et al., 2020). This paper focuses on two widely studied versions of the CDAP: the capacitated CDAP (CCDAP) and the uncapacitated CDAP (UCDAP).

This study introduces the CCDAP and UCDAP, which address operations at an I-shaped cross-dock terminal in logistics and supply chain management. In this type of terminal, incoming trucks unload pallets at inbound doors. The pallets are then swiftly moved to outbound doors and loaded onto outgoing trucks. In CCDAP, each truck is assigned to a specific door, allowing multiple trucks to share a single door, as long as the door's capacity limits are not exceeded. In contrast, the UCDAP does not impose capacity constraints on the doors; instead, each inbound (outbound) door can accommodate a maximum of one incoming (outgoing) truck. The goal for both problems is to determine the best allocation of incoming trucks to inbound doors and outgoing trucks to outbound doors in order to minimize the overall costs associated with pallet handling within the cross-docking facility of the warehouse.

Figure 2 illustrates an example of the CCDAP considered in this study. Trucks 1, 2, and 3 are incoming trucks, each carrying three pallets, while trucks 4, 5, 6, and 7 are outgoing trucks. Each pallet is represented by a circle, and pallets of the same color are from the same incoming truck. Truck 1 needs to send $1/0/1/1$ pallet to trucks $4/5/6/7$, truck 2 needs to send $1/0/1/1$ pallet to trucks $4/5/6/7$, and truck 3 needs to send $0/2/1/0$ pallet to trucks $4/5/6/7$. Doors 1 and 2 are inbound, while doors 3 and 4 are outbound, with each door having a capacity of six pallets. The distances between each pair of inbound/outbound doors are $D_{13} = 7$, $D_{14} = 5$, $D_{23} = 8$, and $D_{24} = 10$. A solution with a total pallet-handling cost of 71 is obtained, calculated as $f = (T_{24}+T_{25})\times D_{13}+(T_{26}+T_{27})\times D_{14}+(T_{14}+T_{15}+T_{34}+T_{35})\times D_{23}+(T_{16}+T_{17}+T_{36}+T_{37})\times D_{24} = 71$ by assigning truck 2 to door 1, trucks 3 and 1 to door 2, trucks 4 and 5 to door 3, and trucks 6 and 7 to door 4.

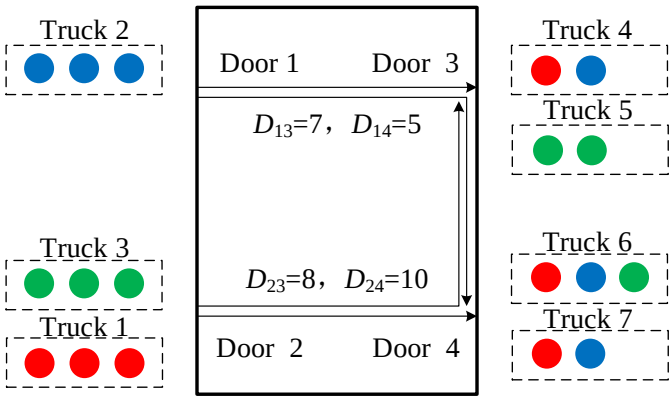


Figure 2: An example of CCDAP.

To properly define both CCDAP and UCDAP, we recall the mathematical formulation presented in Guignard et al. (2012); Zhang et al. (2018), which relies on specific notations (Table 1) and two decision variables: x_{mi}, a binary variable that takes the value 1 if the incoming truck $m \in \mathcal{M}$ is assigned to inbound door $i \in \mathcal{I}$, and equals 0 otherwise; similarly, y_{nj}, another binary variable that takes the value 1 if the outgoing truck $n \in \mathcal{N}$ is assigned to outbound door $j \in \mathcal{J}$, and equals 0 otherwise.

Using these notations and decision variables, the mathematical formulation of CCDAP (Guignard et al.,

Table 1: Notations.

Notation	Description
$\mathcal{M}$	The set of all incoming trucks
$\mathcal{N}$	The set of all outgoing trucks
$\mathcal{I}$	The set of all inbound doors
$\mathcal{J}$	The set of all outbound doors
Q_i	The capacity of the inbound door $i \in \mathcal{I}$
Q_j	The capacity of the outbound door $j \in \mathcal{J}$
D_{ij}	The distance from the inbound door $i \in \mathcal{I}$ to the outbound door $j \in \mathcal{J}$
T_{mn}	The number of pallets to be transported from incoming truck $m \in \mathcal{M}$ to outgoing truck $n \in \mathcal{N}$
P_m	The total number of pallets delivered from incoming truck $m \in \mathcal{M}$, $P_m = \sum_{n\in\mathcal{N}} T_{mn}$
L_n	The total number of pallets received by outgoing truck $n \in \mathcal{N}$, $L_n = \sum_{m\in\mathcal{M}} T_{mn}$

2012) is formally stated as:

$$\min \quad f = \sum_{m\in\mathcal{M}} \sum_{i\in\mathcal{I}} \sum_{n\in\mathcal{N}} \sum_{j\in\mathcal{J}} D_{ij} T_{mn} x_{mi} y_{nj} \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{I}} x_{mi} = 1, \ \forall m \in \mathcal{M} \tag{2}$$

$$\sum_{j\in\mathcal{J}} y_{nj} = 1, \ \forall n \in \mathcal{N} \tag{3}$$

$$\sum_{m\in\mathcal{M}} P_m x_{mi} \le Q_i, \ \forall i \in \mathcal{I} \tag{4}$$

$$\sum_{n\in\mathcal{N}} L_n y_{nj} \le Q_j, \ \forall j \in \mathcal{J} \tag{5}$$

$$x_{mi} \in \{0,1\}, \ \forall m \in \mathcal{M}, \ \forall i \in \mathcal{I} \tag{6}$$

$$y_{nj} \in \{0,1\}, \ \forall n \in \mathcal{N}, \ \forall j \in \mathcal{J} \tag{7}$$

The objective function (1) seeks to minimize the total cost of pallet transportation between inbound and outbound doors. Constraints (2) and (3) guarantee that each incoming/outgoing truck is assigned to precisely one inbound/outbound door. Constraints (4) and (5) establish the capacity limits for each inbound/outbound door. The remaining constraints are related to decision variables.

In a similar manner, the UCDAP model (Zhang et al., 2018) is elaborated with the same notation, decision variables, and objective function. The only difference between the UCDAP model and the CCDAP model is that, for the UCDAP model, there are no capacity limitations on the doors; instead, each inbound (outbound) door is allowed to accommodate at most one incoming (outgoing) truck. Therefore, to obtain the UCDAP model, we only need to replace constraints (4) and (5) in the CCDAP model by the following constraints.

$$\sum_{m\in\mathcal{M}} x_{mi} \le 1, \ \forall i \in \mathcal{I} \tag{8}$$

$$\sum_{n\in\mathcal{N}} y_{nj} \le 1, \ \forall j \in \mathcal{J} \tag{9}$$

3. Frequent pattern mining driven evolutionary algorithm for CDAPs

Since the UCDAP can be easily transformed into the CCDAP by standardizing the capacity of all doors and the number of pallets in all trucks, our work proceeds to develop a frequent pattern driven evolutionary search algorithm specifically tailored for the CCDAP.

3.1. Main scheme

Algorithm 1: Pseudocode of FEA for CDAPs

Input: Population size k, instance G with objective function f, specified number of patterns to be mined m, and running time limit $t_{\max}$
Output: The best found solution S^*

```
t_0 ← Time()
POP ← Distance based Population Initialization()                    Section 3.3
S* ← argmin{f(S_i) : i = 1, 2, ..., k}
P ← Frequent Pattern Mining(POP, m)                                 Section 3.4
while Time() − t_0 < t_max do
    W ← Pattern Selection(P)                                        Section 3.5
    S ← Frequent Pattern based Solution Construction(W)             Section 3.6
    S ← Exploring and Focusing Search(S)                            Section 3.7
    if f(S) ≤ f(S*) then
        S* ← S
    POP ← Health based Population Updating(POP, S)                  Section 3.8
    if population does not evolve anymore then
        P ← Frequent Pattern Mining(POP, m)                         Section 3.4
return S*
```

The proposed frequent pattern mining driven evolutionary algorithm (FEA) follows the general frequent pattern based search method (Zhou et al., 2020a) and adapts the method to our CCDAP problem. The core idea involves using data mining methods to extract meaningful frequent patterns from high-quality solutions. These patterns subsequently guide the optimization algorithm in exploring search areas of interest, thereby enhancing the effective investigation of the search space.

Algorithm 1 illustrates the pseudo-code of the FEA. The algorithm starts with a collection of high-quality solutions generated by a distance-based population initialization method (Section 3.3). Following this initial step, a data mining technique is used to find frequent patterns, resulting in a mined pattern set P (Section 3.4). The algorithm then enters a primary "while" loop that runs through several generations, refining the solutions within the population. In each generation, a subset $W = \{p_1, \ldots, p_n\}$ of n patterns is selected from the mined pattern set P (Section 3.5) and is used to construct a new solution (Section 3.6). This new solution undergoes further refinement through a local exploring and focusing search (Section 3.7). The refined solution is then added back into the population following the pool management rule (see Section 3.8). If the population stagnates, the data mining procedure is re-initiated to extract new frequent patterns (Section 3.4). This process continues until a stopping criterion, such as reaching a time limit, is met.

3.2. Solution representation and initial solution

In the context of a CCDAP problem, we consider a collection of incoming trucks denoted as $\mathcal{M}$, a collection of outgoing trucks represented by $\mathcal{N}$, a collection of inbound doors labeled $\mathcal{I}$, and a collection of outbound doors denoted as $\mathcal{J}$. The goal is to identify the best assignment of incoming/outgoing trucks to inbound/outbound doors that minimizes the overall pallet-handling cost. Within CCDAP, we define a solution S using a set of K truck-to-door assignment pairs, where $K = |\mathcal{M}| + |\mathcal{N}|$. Thus, a solution S can be expressed as $S = \{(t1, d_{t1}), (t2, d_{t2}) : t1 \in \mathcal{M}, t2 \in \mathcal{N}\}$, where $d_{t1} \in \mathcal{I}$ denotes the inbound door that the incoming truck $t1$ is allocated to, and $d_{t2} \in \mathcal{J}$ denotes the outbound door that the outgoing truck $t2$ is allocated to. For instance, $S = \{(1, 1), (2, 1), (3, 2), (4, 3), (5, 3), (6, 4), (7, 4)\}$ signifies that truck 1 and truck 2 are assigned to door 1, truck 3 is assigned to door 2, truck 4 and truck 5 are assigned to door 3, while truck 6 and truck 7 are assigned to door 4.

To ensure the diversity of initial solutions, we propose a hybrid method that integrates both greedy and random construction strategies to create an initial solution S consisting of K truck-to-door assignment pairs. Each truck is assigned to a door by either a greedy method or a random method. Specifically, in the random method, each unassigned incoming/outgoing truck t is allocated to an inbound/outbound door d at random. Conversely, in the greedy method, for each unassigned incoming/outgoing truck t, we prioritize its assignment to the inbound/outbound door d that results in the lowest increment of the objective function value of the partial solution S (including the partial truck-to-door assignments). In this initial solution construction, if an incoming truck t is assigned to an inbound door d, we set the decision variable $x_{td} = 1$. Similarly, if an outgoing truck t is assigned to an outbound door d, we set the decision variable $y_{td} = 1$. Thus, the objective function value of the partial truck-to-door assignments can be computed according to the objective function (1). Afterward, the initial solution S is further improved by the two-phase local exploring and focusing search procedure that is detailed in Section 3.7. It is important to highlight that this two-phase search procedure is capable of handling infeasible solutions, as the capacity constraints of the doors are ignored during both the greedy and random construction phases.

3.3. Distance based population initialization

Each solution of the initial population POP is generated by means of the greedy and random construction procedure in the above section. In order to avoid exploring similar solutions, a distance based diversification measure is used to control the insertion of a new solution into the population.

Given two solutions S_1 and S_2, the distance between them is defined as follows.

$$dist(S_1, S_2) = \frac{\sum_{t=1}^{K}(A(t))}{K}, \tag{10}$$

where $A(t) = 0$ if truck t is assigned to the same door in both S_1 and S_2 and 1 otherwise, and K is the total number of trucks.

Thus, the distance between a newly generated initial solution S_0 and the POP is defined as

$$D(S_0, POP) = \min\{dist(S_0, S_i) : S_i \in POP\} \tag{11}$$

The new initial solution S_0 is accepted into the initial population POP if $D(S_0, POP) > \tau$, where τ is a distance threshold.

We repeatedly perform the greedy and random construction procedure until k solutions are added to the initial population POP, where k denotes the population size.

3.4. Mining frequent patterns

Frequent patterns (Grahne & Zhu, 2003; Aggarwal et al., 2014) refer to recurring combinations of items, transactions, or events that occur frequently within a dataset. These patterns are crucial in revealing hidden associations, correlations, or sequential relationships among the data. In essence, frequent patterns represent the common behaviors or trends present in the dataset.

Frequent pattern mining algorithms are specialized techniques designed to efficiently discover these recurring patterns from large datasets. These mining algorithms employ various strategies and methods to identify patterns that occur above a certain threshold of frequency. By doing so, they help in extracting valuable insights and knowledge from the data, which can be utilized for decision-making, prediction, and other data-driven tasks. Popular frequent pattern mining algorithms (Aggarwal et al., 2014) include Apriori, FP-Growth, and PrefixSpan, each offering unique advantages and trade-offs in terms of scalability, efficiency, and applicability to different types of datasets and patterns.

In our study, we use a set of truck-to-door pairs to represent a solution. As a result, we represent frequent patterns as a collection of reliable truck-to-door assignments that frequently appear in high-quality solutions. To uncover consistent truck-to-door assignments that often arise in high-quality solutions, we adopt the widely recognized FPmax* algorithm (Grahne & Zhu, 2003) to create a mined pattern set P that includes m frequent patterns (truck-to-door assignments), with m indicating the number of patterns mined.

The FPmax* algorithm is one of the most popular methods for mining maximal frequent itemsets from datasets. It utilizes the FP-tree data structure, which enables efficient storage and processing of transaction data. Through recursive traversals of the FP-tree, the algorithm identifies maximal frequent itemsets. By optimizing the mining process with pruning techniques and effective recursion, FPmax* significantly enhances both the speed and scalability of the algorithm compared to traditional methods, making it particularly well-suited for large-scale data mining tasks.

3.5. Selecting mined frequent pattern

After adopting the widely recognized FPmax* algorithm to mine a frequent pattern set P that includes m frequent patterns (truck-to-door assignments), we implement the roulette-wheel selection method to choose a subset $W = \{p_1, \ldots, p_n\}$ of n patterns from the mined pattern set P, and use them to create the new solution, where n is a parameter.

Precisely, when extracting frequent patterns from a collection of high-quality solutions, we record the number of times (frequency value) that each pattern appears in these high-quality solutions. To determine the selection probabilities, the frequency values of all patterns are normalized to sum up to one. This is done by dividing each pattern's frequency value by the sum of frequency values. The resulting normalized values represent the probability distribution from which patterns will be selected. To perform selection, a random number between 0 and 1 is generated, indicating a point on the roulette wheel. The patterns are then traversed cumulatively, with their selection probabilities accumulated. The pattern whose cumulative probability range encompasses the generated random number is chosen.

3.6. Using mined patterns to build new solutions

Frequent patterns typically represent a collection of common truck-to-door assignment pairs identified in the sampled high-quality solutions, forming the basis for defining partial solutions. Our proposed solution construction procedure based on the mined patterns creates a new solution S in two steps.

Generate a partial solution S. From the mined frequent pattern set P that includes m frequent patterns (Section 3.4), we apply the pattern selection strategy (Section 3.5) to select a sample $W = \{p_1, \ldots, p_n\}$ of n patterns. These selected patterns are used to create a partial solution S, i.e., $S = \{p_1, \ldots, p_n\}$. Next, any duplicates within S, where a truck is assigned to multiple doors, are resolved by removing the extra doors and retaining only the one with the highest frequency.

Complete the partial solution S. To preserve the diversity of solution S, each unassigned incoming/outgoing truck is randomly assigned to an inbound/outbound door, regardless of the door's capacity limit.

It is important to note that the resulting solution S generated by this solution construction procedure may be infeasible due to violations of the door's capacity constraint. This infeasibility is then addressed during the following two-phase local exploring and focusing search (see Section 3.7).

3.7. Two-phase local refinement by exploring and focusing search

Algorithm 2: Pseudocode of the two-phase exploring and focusing search

Input: The starting solution S.
Output: Best solution S_{best} found so far.
1 /*__The 1st Phase: Exploring Search__*/
2 $S_{best} \leftarrow Exploring\_search(S)$
3 /*__The 2nd Phase: Focusing Search__ */
4 $S_{best} \leftarrow Focusing\_search(S_{best})$
5 **return** S_{best}

For solution improvement, we use a two-phase local search process, i.e., an infeasible local exploration search phase (identical to the one used by Li et al. (2024)), followed by an intensification focusing search phase. In the first phase, the search process investigates the search space by considering infeasible solutions, which involves relaxing the constraints related to the capacity of doors. In the subsequent phase, the emphasis shifts to intensifying the local search based on the frequent patterns that have been identified, with the goal of obtaining more refined solutions.

The proposed two-phase local refinement procedure is summarized in Algorithm 2. Beginning with an input solution, the algorithm initiates the infeasible local exploring search phase, followed by a local focusing search on the best solution, denoted as S_{best}, identified in the first phase.

The two-phase local refinement procedure combines the exploration capability of the infeasible local exploration with the exploitation capabilities of the feasible local focusing search to effectively navigate the search space. As shown in the computational experiments outlined in Section 5.2, the later local focusing search phase significantly improves the quality of solutions generated during the first local exploration phase.

3.7.1. Move operators

The two-phase local exploring and focusing search is based on a joint use of the following four distinct move operators.

- **Shift (N_1):** Alter the assignment of a truck from its present door to an alternate door.
- **Swap (N_2):** Swap two trucks situated at two different doors.
- **Multi-shift (N_3):** Change the allocation of all trucks in door i to another door j.
- **Multi-swap (N_4):** Interchange all trucks positioned between two distinct doors i and j.

N_1, N_2, N_3, and N_4 denote four neighborhoods induced by the above four move operators.

3.7.2. The exploring search phase

The local exploring search phase utilizes an adaptive tabu search that examines both feasible and infeasible solutions to help the search for better solutions. Local search encompasses exploration across both feasible and infeasible solution spaces, offering greater search flexibility compared to approaches limited to feasible spaces alone. Numerous studies (Liu et al., 2014; Qin et al., 2016; Li et al., 2022) have demonstrated that temporarily considering infeasible solutions within the solution space can significantly enhance the efficacy of local search algorithms. This ability facilitates smoother transitions between structurally distinct feasible solutions, thereby boosting overall performance.

In this search phase, an extended penalty-based evaluation function F is employed to assess the quality of both feasible and infeasible solutions. For a given solution S, let $g(S)$ represent the penalty function related to the degree of infeasibility of S. This is calculated based on the total excess degree of all doors against their capacity limits, expressed as $g(S) = \sum_{k \in \mathcal{I} \cup \mathcal{J}} (\max\{W_k - Q_k, 0\})$, where W_k ($k \in \mathcal{I} \cup \mathcal{J}$) denotes the total number of pallets assigned to door k. Therefore, the extended penalty-based evaluation function $F(S)$ is formulated as follows.

$$F(S) = f(S) + \varphi \times g(S), \tag{12}$$

where φ is a self-adjusting penalty coefficient.

Algorithm 3 describes the local exploration search procedure. Once the variables are initialized, the procedure undergoes multiple iterations. In each iteration, it jointly explores four neighborhoods N_1, N_2, N_3, and N_4 of the current solution S, generated by the four move operators (Shift, Swap, Multi-shift, and Multi-shift) as described in Section 3.7.1. The procedure then selects the best admissible neighboring solution (i.e., one that is not prohibited by the tabu strategy and has the minimum extended evaluation function value $F(S)$) to replace the current solution S. During the search, the parameter φ within the evaluation function F is consistently updated. Specifically, if u_c consecutively accessed solutions are all infeasible, φ is doubled. On the other hand, if u_c consecutively accessed solutions are all feasible, φ is halved. The search terminates and returns the best found feasible solution S_b if there is no improvement after ω consecutive iterations, where ω denotes the search depth.

The local exploration search procedure employs a mixed tabu strategy introduced in Li et al. (2024). This strategy combines the attribute-based (Glover, 1997) and the solution-based (Woodruff & Zemel, 1993; Wang et al., 2017; Wu et al., 2020) tabu strategies to avoid long-term or short-term search cycles, while

Algorithm 3: Pseudocode of the exploring search

Input: The starting solution S.
Output: The best found feasible solution S_b.

$\varphi \leftarrow 1$
$counter \leftarrow 0$
$O_b \leftarrow +\infty$
while $counter < \omega$ **do**
  $S' \leftarrow$ The best admissible neighboring solution of S
  if S' is a feasible solution and $f(S') < O_b$ **then**
    $O_b \leftarrow f(S'), S_b \leftarrow S', S \leftarrow S', counter \leftarrow 0$
  else if S' is not feasible and $F(S') < F(S)$ **then**
    $S \leftarrow S', counter \leftarrow 0$
  else
    $S \leftarrow S', counter \leftarrow counter + 1$
  if u_c solutions generated by u_c consecutive iterations are all feasible solutions **then**
    $\varphi \leftarrow \varphi/2$
  else if u_c solutions generated by u_c consecutive iterations are all infeasible **then**
    $\varphi \leftarrow \varphi \times 2$
return S_b

maintaining efficient search capabilities. However, outright prohibition of all previously visited solutions may prevent the algorithm from exploring potentially fruitful search paths. To address this, the exploration search procedure utilizes hashing techniques (Woodruff & Zemel, 1993) to keep track of how often each solution is accessed during the search period, limiting each solution to a maximum of β visits during the exploration search phase. Furthermore, in the subsequent tt iterations, reversing the current move operation is prohibited to prevent the search from quickly returning to the recently accessed solution. For the detailed implementation of this mixed tabu strategy, please see Li et al. (2024).

3.7.3. The focusing search phase

In the local focusing search phase (see Algorithm 4), it first applies the pattern selection strategy (see Section 3.5) to select a sample $W = \{p_1, \ldots, p_n\}$ of n patterns from the mined pattern set P, and then intensively explores the four neighborhoods $N_1 - N_4$ in a random order to find a high-quality local optimal solution. Specifically, for each given neighborhood N_i (where $i = 1, 2, 3, 4$), the procedure evaluates all candidate moves within the neighborhood of the current solution S in random order, and accepts the first improving feasible move that does not disrupt any pattern in W. Since frequent patterns usually represent common truck-to-door assignment pairs found in the sampled high-quality solutions, when conducting a neighborhood search on the current solution S, we retain these common truck-to-door assignment pairs in W and do not break down them. During the focusing search phase, for any move operator, if it disrupts any pattern in W, such a move is disregarded. This mechanism significantly speeds up the search process.

3.8. Population updating rule

To improve diversity within the population, we implement a population updating strategy that emphasizes both solution quality and distances between the solutions in the population. The population updating procedure is outlined in Algorithm 5. Given an improved solution S' and the population POP, S' is first

Algorithm 4: Pseudocode of the focusing search

```
Input: The current solution S.
Output: updated S.
W ← PatternSelection(P)                    /*the frequent pattern sample W, see Section 3.5*/
NL ← {N1, N2, N3, N4}
flag ← true
while flag = true do
    flag ← false
    Randomly shuffle all the neighboring operators in NL
    for each neighborhood N ∈ NL do
        Randomly shuffle all the possible moves in neighborhood N(S)
        for each move mv ∈ N(S) do
            if mv does not disrupt any pattern in W then
                S' ← S ⊕ mv
                if S' is a feasible solution and f(S') < f(S) then
                    S ← S'
                    flag ← true
                    break;
return S
```

tentatively added to POP resulting in $POP = POP \cup \{S'\}$. Then, the quality-and-distance score is calculated for each solution in the population POP using Equation (13). Finally, the solution S_w with the lowest score is determined and then eliminated from the population.

Algorithm 5: Population updating

```
Input: Population POP, solution to be inserted S'.
Output: Updated population POP.
Add S' to POP: POP = POP ∪ {S'}
for each solution S ∈ POP do
    Calculate the quality-and-distance score Score(S) of S according to Equation (13)
Identify the solution Sw that possesses the smallest quality-and-distance score in POP'
Delete Sw from POP: POP = POP\{Sw}
return POP
```

The quality-and-distance score for a solution $S \in POP$ is calculated by

$$Score(S) = \lambda \times \frac{f(S) - f_{min}}{f_{max} - f_{min}} + (1 - \lambda) \times \frac{D(S, POP) - D_{min}}{D_{max} - D_{min}}, \tag{13}$$

where λ is a parameter set to be 0.5, $f(S)$ represents the objective value of solution S, f_{max} is the maximum objective value found among the solutions in the population POP, f_{min} is the lowest objective value within the same population POP, D_{max} is the maximum distance measured between a solution $S \in POP$ and the population POP, and D_{min} is the minimum distance between a solution $S \in POP$ and the population POP. The distance from a solution $S \in POP$ to the population POP is defined as $D(S, POP) = \min\{dist(S, S_i) : S_i \in POP \backslash \{S\}\}$.

4. Computational experiments

This section presents a comprehensive computational evaluation and analysis of the proposed FEA by testing it on CCDAP and UCDAP benchmark instances in the literature.

4.1. Benchmark instances

For CCDAP, we use the following two sets of 99 popular benchmark instances proposed by Guignard et al. (2012).

- **Set A.** This set comprises 50 test instances. They are generated as follows: the quantity of incoming trucks matches that of outgoing trucks, drawn from $\{8, 9, 10, 11, 12, 15, 20\}$; the number of inbound doors equals the number of outbound doors, drawn from 4, 5, 6, 7, 10. The distance between pairs of inbound/outbound doors is determined by their positions in the warehouse. The distance between two directly facing doors is set to 8, while for non-facing doors, it is set to 8 plus I, where I is the sequential number of the door starting from the facing door. The number of pallets shifted from incoming to outgoing trucks is specified by a freight matrix. This matrix is created by randomly assigning 25% of its elements with integers ranging from 10 to 50, with the remaining elements being zero. Each door's capacity is calculated uniformly by dividing the overall pallet total from all incoming trucks by the number of doors, then incorporating a capacity slack that may be 5%, 10%, 15%, 20%, or 30%.

- **Set B.** This set includes 49 large instances, created in a manner similar to Set A. The number of incoming/outgoing trucks is selected from $\{25, 50, 75, 100\}$, while the quantity of inbound/outbound doors is taken from $\{10, 20, 30, 40\}$. Each test instance is labeled using the format "xx×yySzz", where "xx" indicates the number of incoming/outgoing trucks, "yy" represents the number of inbound/outbound doors, and "zz" refers to the capacity slack.

For UCDAP, we use the same 40 benchmark instances proposed by Zhang et al. (2018). Zhang et al. (2018) utilized data from Yellow Transportation company to generate these 40 instances, labeled FC and SC. In Table 2, the detailed parameters for both FC and SC cases are provided. Each category (FC and SC) includes five varying proportions of test cases, with four distinct traffic patterns for each case.

Table 2: UCDAP Benchmark instances.

Instance	Number of inbound doors	Number of outbound doors	Pallet range	Distance range
FC12	5	7	[10, 500]	[5, 15]
FC24	10	14	[10, 400]	[5, 24]
FC48	20	28	[10, 300]	[5, 36]
FC96	45	51	[10, 200]	[5, 60]
FC192	90	102	[10, 100]	[5, 110]
SC48	24	24	[10, 300]	[8, 31]
SC64	32	32	[10, 250]	[8, 39]
SC96	48	48	[10, 200]	[8, 55]
SC192	96	96	[10, 150]	[8, 103]
SC300	150	150	[10, 100]	[8, 157]

4.2. Experimental settings

The proposed FEA[1] was implemented in C++ and compiled with the GNU g++ compiler using the -O3 optimization flag. Experiments were conducted on a 2.6 GHz Intel E5-2670 system equipped with 2 GB of RAM and running Linux.

4.3. Parameter tuning

The proposed FEA has nine parameters (see Table 3). These parameters were tuned using Irace (López-Ibáñez et al., 2016), a racing algorithm designed for offline parameter configuration of parameterized algorithms. Irace begins by generating initial candidate configurations from a predefined parameter space. In each round, candidates are tested on real tasks; poor performers are eliminated, retaining winners. Then, new candidates are generated by mutating or crossing these winners. This process repeats until convergence, outputting the optimal configuration. The predefined spaces, listed in Table 3, were determined based on preliminary experiments. In this tuning experiment, Irace was executed on 30 randomly chosen instances. The tuning process had a budget of 1000 FEA executions and a time limit of 400 seconds per execution. Table 3 presents the parameter values suggested by Irace, which are considered as the algorithm's default settings and were applied across all experiments reported in the paper.

Table 3: Settings of the parameters.

Parameter	Section	Description	Value range	Final value
k	3.3	Population size	{10, 20, 30, 40, 50}	20
τ	3.3	Distance threshold	{0.3, 0.5, 0.6, 0.7, 0.8}	0.5
m	3.4	Number of mined patterns	{10, 15, 20, 25, 30}	20
n	3.5	Number of patterns selected	{3, 5, 10, 15, 20}	10
ω	3.7.2	Search depth of the exploring search	{100, 200, 300, 400, 500}	200
φ	3.7.2	Self-adjusting penalty coefficient	{5, 10, 15, 20, 25}	10
u_c	3.7.2	frequency of updating φ	{1, 3, 5, 10, 15}	5
tt	3.7.2	Tabu tenure	{5, 10, 15, 20, 25}	10
β	3.7.2	Maximum number of times allowed to be visited for each solution	{2, 5, 10, 15, 20}	5

4.4. Computational results and comparison on CCDAP instances

To evaluate the effectiveness of the proposed FEA, we compare its results with the best known solutions (BKS) documented in the existing literature and evaluate its performance along with six top performing state-of-the-art heuristic algorithms specifically designed for the CCDAP, including two local search algorithms (LS1 and LS2) (Guignard et al., 2012), a convex hull relaxation algorithm (CHR) (Guignard et al., 2012), two probabilistic tabu search algorithms (PTS1 and PTS2) (Guemri et al., 2019), and a reinforcement learning based strategic oscillation algorithm (RL-SO) (Li et al., 2024). Following Guemri et al. (2019), the FEA was run 10 times independently on each instance and each run is limited to $3|\mathcal{M}|$ seconds, where $|\mathcal{M}|$ denotes the number of incoming trucks in the corresponding instance. Since the source code of the six reference algorithms is not available, their computational results have been taken directly from their respective publications. Performing a completely fair comparison between the proposed FEA and these reference algorithms is challenging due to differences in computing platforms, programming languages,

[1]The code of the proposed algorithm will be made available with the publication of the paper.

Table 4: Comparative results of the proposed FEA on 50 "SetA" CCDAP instances with LS1 (Guignard et al., 2012), LS2 (Guignard et al., 2012), CHR (Guignard et al., 2012), PTS1 (Guemri et al., 2019), PTS2 (Guemri et al., 2019), and RL-SO (Li et al., 2024).

		LS1		LS2		CHR		PTS1			PTS2			RL-SO			FEA		
Instance	BKS	OF	T(s)	OF	T(s)	OF	T(s)	OF	AVG	T(s)	OF	AVG	T(s)	OF	AVG	T(s)	OF	AVG	T(s)
8X4S5	**5174***	**5174**	1	**5174**	2	**5174**	13	**5174**	5174.0	3.9	**5174**	5174.0	1.7	**5174**	5174.0	0.1	**5174**	5174.0	0.1
8X4S10	**5169***	**5169**	2	**5169**	3	**5169**	14	**5169**	5169.0	3.7	**5169**	5169.0	1.1	**5169**	5169.0	0.1	**5169**	5169.0	0.1
8X4S15	**5112***	**5112**	2	**5112**	2	**5112**	9	**5112**	5112.0	3.5	**5112**	5112.0	1.0	**5112**	5112.0	0.1	**5112**	5112.0	0.2
8X4S20	**5086***	**5086**	2	**5086**	3	**5086**	17	**5086**	5086.0	3.3	**5086**	5086.0	0.9	**5086**	5086.0	0.1	**5086**	5086.0	0.1
8X4S30	**5063***	**5063**	2	**5063**	2	**5063**	17	**5063**	5063.0	3.2	**5063**	5063.0	0.8	**5063**	5063.0	0.1	**5063**	5063.0	0.2
9X4S5	**6047***	**6047**	2	**6047**	3	**6047**	15	**6047**	6047.0	4.5	**6047**	6047.0	1.3	**6047**	6047.0	0.1	**6047**	6047.0	0.1
9X4S10	**6027***	**6027**	2	**6027**	3	**6027**	17	**6027**	6027.0	4.1	**6027**	6027.0	1.1	**6027**	6027.0	0.1	**6027**	6027.0	0.1
9X4S15	**5976***	**5976**	3	**5976**	3	**5976**	21	**5976**	5976.0	3.9	**5976**	5976.0	0.9	**5976**	5976.0	0.1	**5976**	5976.0	0.2
9X4S20	**5937***	**5937**	2	**5937**	2	**5937**	19	**5937**	5937.0	3.9	**5937**	5937.0	1.0	**5937**	5937.0	0.1	**5937**	5937.0	0.2
9X4S30	**5904***	**5904**	2	**5904**	2	**5904**	14	**5904**	5904.0	3.7	**5904**	5904.0	0.9	**5904**	5904.0	0.1	**5904**	5904.0	0.5
10X4S5	**6518***	**6518**	5	**6518**	6	**6518**	16	**6518**	6518.0	5.1	**6518**	6518.0	1.4	**6518**	6518.0	0.1	**6518**	6518.0	0.1
10X4S10	**6325***	**6325**	3	**6325**	3	**6325**	30	**6325**	6325.0	4.9	**6325**	6325.0	1.1	**6325**	6325.0	0.1	**6325**	6325.0	0.1
10X4S15	**6296***	**6296**	3	**6296**	3	**6296**	28	**6296**	6296.0	4.6	**6296**	6296.0	1.0	**6296**	6296.0	0.1	**6296**	6296.0	0.1
10X4S20	**6267***	**6267**	3	**6267**	4	**6267**	26	**6267**	6267.0	4.4	**6267**	6267.0	1.0	**6267**	6267.0	0.1	**6267**	6267.0	0.2
10X4S30	**6193***	**6193**	3	**6193**	3	**6193**	19	**6193**	6193.0	4.1	**6193**	6193.0	1.1	**6193**	6193.0	0.1	**6193**	6193.0	0.1
10X5S5	**6616***	**6616**	6	**6616**	6	**6616**	13	**6616**	6616.0	4.5	**6616**	6616.0	1.8	**6616**	6616.0	0.1	**6616**	6616.0	0.1
10X5S10	**6476***	**6476**	4	**6476**	5	**6476**	23	**6476**	6476.0	4.5	**6476**	6476.0	1.5	**6476**	6476.0	0.1	**6476**	6476.0	0.2
10X5S15	**6397***	**6397**	3	**6397**	4	**6397**	16	**6397**	6397.0	4.1	**6397**	6397.0	1.4	**6397**	6397.0	0.1	**6397**	6397.0	0.1
10X5S20	**6342***	6374	3	6363	4	6354	18	**6342**	6342.0	4.0	**6342**	6342.0	1.3	**6342**	6342.0	0.1	**6342**	6342.0	0.1
10X5S30	**6308***	6324	3	6333	4	6333	18	**6308**	6308.0	3.9	**6308**	6308.0	1.3	**6308**	6308.0	0.1	**6308**	6308.0	0.2
11X5S5	**7812***	**7812**	12	**7812**	12	**7812**	16	**7812**	7812.0	5.2	**7812**	7812.0	2.1	**7812**	7812.0	0.1	**7812**	7812.0	0.1
11X5S10	**7572***	**7572**	5	**7572**	6	**7572**	18	**7572**	7572.0	5.0	**7572**	7572.0	1.6	**7572**	7572.0	0.2	**7572**	7572.0	0.2
11X5S15	**7535***	7542	5	7542	5	7542	20	**7535**	7535.0	4.8	**7535**	7535.0	1.4	**7535**	7535.0	0.3	**7535**	7535.0	0.3
11X5S20	**7439***	7465	5	7465	5	7478	17	**7439**	7439.0	4.5	**7439**	7439.0	1.3	**7439**	7439.0	0.4	**7439**	7439.0	0.6
11X5S30	**7420***	7428	4	7424	5	**7420**	18	**7420**	7420.0	4.2	**7420**	7421.6	1.4	**7420**	7420.0	0.1	**7420**	7420.0	0.1
12X5S5	**8072***	**8072**	5	**8072**	7	**8072**	13	**8072**	8072.0	5.8	**8072**	8072.0	1.8	**8072**	8072.0	0.9	**8072**	8072.0	0.5
12X5S10	**7978***	**7978**	4	7988	5	7991	15	**7978**	7978.0	5.5	**7978**	7978.0	1.6	**7978**	7978.0	0.4	**7978**	7978.0	0.3
12X5S15	**7939***	**7939**	4	**7939**	6	**7939**	18	**7939**	7939.0	5.4	**7939**	7939.0	1.5	**7939**	7939.0	0.5	**7939**	7939.0	0.3
12X5S20	**7939***	**7939**	5	**7939**	6	**7939**	18	**7939**	7939.0	5.1	**7939**	7939.0	1.4	**7939**	7939.0	0.6	**7939**	7939.0	0.2
12X5S30	**7923***	7942	6	7925	6	7925	21	**7923**	7923.0	4.7	**7923**	7923.0	1.4	**7923**	7923.0	0.7	**7923**	7923.0	0.9
12X6S5	**10891***	**10891**	10	10894	14	10894	14	**10891**	10891.0	5.4	**10891**	10891.0	2.7	**10891**	10891.0	0.3	**10891**	10891.0	0.7
12X6S10	**10456***	10475	6	10496	7	10480	14	**10456**	10456.0	5.1	**10456**	10456.0	2.0	**10456**	10456.0	1.0	**10456**	10456.0	1.1
12X6S15	**10362***	10395	5	10420	6	10374	17	**10362**	10362.0	4.9	**10362**	10378.5	1.8	**10362**	10362.0	0.6	**10362**	10362.0	0.9
12X6S20	**10312***	10331	7	10339	6	10323	16	**10312**	10312.0	4.6	**10312**	10312.0	1.9	**10312**	10312.0	0.8	**10312**	10312.0	1.0
12X6S30	**10228***	**10228**	6	**10228**	7	10277	20	**10228**	10228.0	4.2	**10228**	10228.0	1.8	**10228**	10228.0	0.6	**10228**	10228.0	0.5
15X6S5	**13927***	13960	13	**13927**	16	13983	18	**13927**	13927.0	7.3	**13927**	13927.0	2.5	**13927**	13927.0	0.7	**13927**	13927.0	0.6
15X6S10	**13803***	13856	11	13852	11	13872	15	**13803**	13803.0	7.0	**13803**	13810.7	2.2	**13803**	13803.0	0.9	**13803**	13803.0	0.5
15X6S15	**13765***	13769	9	13799	12	**13765**	18	**13765**	13765.0	6.5	**13765**	13792.3	2.2	**13765**	13765.0	1.0	**13765**	13765.0	2.2
15X6S20	**13720***	13769	10	13754	11	**13720**	17	**13720**	13720.0	6.2	**13720**	13750.0	2.3	**13720**	13720.0	1.0	**13720**	13720.0	1.3
15X6S30	**13567***	**13567**	10	**13567**	10	13626	16	**13567**	13567.0	5.7	**13567**	13585.2	2.2	**13567**	13567.0	0.8	**13567**	13567.0	1.0
15X7S5	**15054***	**15054**	25	**15054**	25	15096	26	**15054**	15054.0	6.9	**15054**	15063.1	3.4	**15054**	15054.0	1.0	**15054**	15054.0	1.1
15X7S10	**14810***	14841	15	14818	17	**14810**	22	**14810**	14810.0	6.5	**14810**	14843.1	2.7	**14810**	14810.0	1.0	**14810**	14810.0	1.9
15X7S15	**14657***	14678	13	14680	14	14678	22	**14657**	14657.2	6.2	**14657**	14658.2	2.7	**14657**	14657.0	1.4	**14657**	14657.0	1.8
15X7S20	**14514***	14596	14	14533	16	14533	19	**14514**	14514.0	5.8	**14514**	14537.6	2.7	**14514**	14514.0	1.1	**14514**	14514.0	1.7
15X7S30	**14409***	14415	13	14423	13	14414	19	**14409**	14409.0	5.3	**14409**	14413.2	2.6	**14409**	14409.0	0.9	**14409**	14409.0	0.7
20X10S5	**29907**	29945	254	30033	45	29933	87	**29907**	29909.6	9.2	**29907**	30004.4	6.4	**29907**	29907.0	0.7	**29907**	29907.0	1.5
20X10S10	**29236**	29286	36	29286	36	29498	52	**29236**	29253.3	8.5	**29236**	29567.3	5.0	**29236**	29236.0	1.1	**29236**	29236.0	2.0
20X10S15	**29134**	29278	35	29184	37	29529	65	29135	29135.5	7.8	29135	29345.7	4.8	29135	29135.0	1.6	29135	29135.0	2.2
20X10S20	**28945**	29130	38	28992	40	29089	63	**28945**	28951.2	7.2	**28945**	29051.5	4.8	**28945**	28945.0	1.0	**28945**	28945.0	0.9
20X10S30	**28533**	28800	39	28541	41	28571	62	**28533**	28539.6	6.9	**28533**	28741.3	4.8	**28533**	28533.0	1.2	**28533**	28533.0	1.1
AVG	**10741.8**	10764.7	13.6	10755.5	10.3	10768.5	22.8	10741.9	10742.5	5.2	10741.9	10764.4	2.0	10741.9	10741.9	0.5	10741.9	10741.9	0.6

BKS is marked with an asterisk* when it is proved to be the optimal value.

compilation settings, and termination criteria. Consequently, the comparative study focused on the quality of the solutions based on objective values while the timing information was provided for indicative purposes only.

Tables 4 and 5 show the comprehensive computational results of the FEA along with six comparative algorithms on Set A and Set B, respectively. In these tables, the first column identifies the names of each instance. The column labeled "BKS" refers to the best-known solutions documented in existing literature. The "OF" and "AVG" columns show the best and average objective value achieved by each algorithm, respectively. The "T(s)" column presents the average computational time (in seconds) required by each

Table 5: Comparative results of the proposed FEA on 49 "SetB" CCDAP instances with LS1 (Guignard et al., 2012), LS2 (Guignard et al., 2012), CHR (Guignard et al., 2012), PTS1 (Guemri et al., 2019), PTS2 (Guemri et al., 2019), and RL-SO (Li et al., 2024).

		LS1		LS2		CHR		PTS1			PTS2			RL-SO			FEA		
Instance	BKS	OF	T(s)	OF	T(s)	OF	T(s)	OF	AVG	T(s)	OF	AVG	T(s)	OF	AVG	T(s)	OF	AVG	T(s)
25X10S5	**49013**	49144	194	49335	198	49904	105.5	**49013**	49014.8	26.1	**49013**	49013.0	13.8	**49013**	49013.0	0.1	**49013**	49013.0	0.1
25X10S10	**48672**	48949	116	48941	129	49332	138.9	**48672**	48699.3	23.5	48740	48869.5	12.5	**48672**	48672.0	0.2	**48672**	48672.0	0.2
25X10S15	**48407**	48556	122	48504	131	48770	132.5	**48407**	48415.6	20.1	**48407**	48477.0	12.1	**48407**	48407.0	1.5	**48407**	48407.0	1.2
25X10S20	**47926**	48215	85	48235	90	48338	120.0	47934	47949.3	19.1	**47926**	47977.6	12.2	**47926**	47926.0	0.3	**47926**	47926.0	0.2
25X10S30	**47314**	47480	77	47426	81	47652	113.5	**47314**	47356.5	19.6	**47314**	47368.1	12.3	**47314**	47314.0	0.4	**47314**	47314.0	0.1
25X20S30	**51523**	51921	181	51741	200	52447	112.0	51533	51618.4	19.8	51562	52334.2	28.5	**51523**	51523.0	13.0	**51523**	51523.0	11.3
50X10S5	**191040**	191773	4039	191788	4496	192114	178.0	191160	191241.3	78.3	191186	192350.7	26.6	**191040**	191062.0	56.9	**191040**	191048.5	78.3
50X10S10	**189134**	189409	5345	189833	5313	190508	155.9	189166	189478.5	66.2	189573	190316.3	26.2	**189134**	189190.0	77.0	**189134**	189147.3	68.8
50X10S15	**187256**	188006	3363	188264	3120	187753	164.0	187315	187417.9	69.2	187377	188834.9	26.5	**187256**	187257.2	60.1	**187256**	187256.0	59.6
50X10S20	**185975**	186800	3854	186578	3729	187901	181.7	186085	186246.2	72.3	**185975**	186788.2	26.2	**185975**	185996.9	95.9	**185975**	185975.9	91.8
50X10S30	**183145**	183961	1652	184013	1567	184532	195.8	183249	183373.7	79.0	183234	183943.9	25.9	**183145**	183178.3	94.3	**183145**	183155.5	69.4
50X20S5	237396	238048	7074	239673	7308	240288	504.0	237656	238244.5	59.3	239835	241379.3	50.7	237396	237490.2	69.8	**237352**	237369.4	96.5
50X20S10	**233319**	235178	7877	234807	7741	236124	626.8	233341	234045.4	44.6	235326	236932.9	51.6	**233319**	233843.4	74.5	**233319**	233592.5	67.2
50X20S15	229342	230758	7815	230666	7490	233324	604.8	229638	230429.2	45.6	230375	233110.0	48.0	229342	229756.7	84.3	**229269**	229328.4	68.0
50X20S20	**226051**	227698	7161	227883	7328	228918	472.0	226448	227134.2	45.7	228332	230201.4	50.9	**226051**	226512.0	74.0	**226051**	226219.0	77.7
50X20S30	**220291**	221892	7138	222060	6951	223687	357.0	220748	221313.8	48.5	221322	223423.5	45.9	**220291**	220659.3	80.2	**220291**	220294.1	83.1
50X30S15	272504	275973	5975	275121	6950	276237	262.7	273166	274455.9	50.5	276938	278798.1	91.2	272504	273507.1	105.6	**272450**	273140.1	95.3
50X30S20	**260923**	264199	2467	263790	3056	264971	389.9	261725	263099.9	43.6	264729	267802.5	86.3	**260923**	262330.6	95.7	**260923**	261690.1	83.8
50X30S30	**251956**	254056	1879	254795	1962	257529	260.9	252639	253680.2	45.2	255151	258277.8	77.0	**251956**	252318.5	75.4	**251956**	252875.6	95.1
50X43S30	329309	332318	6977	330661	7381	332947	323.0	330285	332378.3	47.5	335036	341233.4	121.1	329309	330412.3	93.4	**329201**	330249.1	83.6
75X10S5	**438575**	440248	9114	440420	9446	442749	168.7	439055	439478.3	158.1	440181	441088.8	44.8	**438575**	438712.0	108.3	**438575**	438661.0	127.1
75X10S10	433876	435985	8842	435970	9178	437642	219.8	434275	435134.7	157.9	435595	437051.9	44.9	433876	434143.1	126.3	**433842**	433923.5	134.3
75X10S15	**429758**	431405	8429	431686	8521	432086	209.8	430005	430781.3	171.2	430663	432183.0	45.3	**429758**	430152.5	115.3	**429758**	429865.4	111.9
75X10S20	426385	427468	8772	427522	8127	429513	78.2	426385	427176.8	179.4	427168	428920.9	44.7	426431	426680.4	138.3	**426364**	426465.3	136.2
75X10S30	419526	420851	6754	420660	6643	421983	215.0	419651	420123.3	198.3	420619	421566.1	44.5	419526	419565.3	110.6	**419515**	419529.4	81.1
75X20S5	528769	531762	8174	532873	8746	533701	789.0	529131	529964.2	86.1	532968	535724.6	72.9	528769	529105.9	131.8	**528634**	528822.5	103.0
75X20S10	**519927**	523447	9269	521970	9665	525098	803.0	520127	521708.1	86.1	524001	526829.3	72.9	**519927**	520569.6	99.7	**519927**	520156.4	115.0
75X20S15	512114	514760	9778	514981	9614	513166	602.0	512170	512788.7	87.1	515098	519040.7	73.9	512114	512366.2	121.1	**511968**	512107.8	116.2
75X20S20	**504434**	506346	8617	506114	9793	508862	627.8	504502	505141.8	90.3	506313	511407.4	72.3	**504434**	504823.7	120.7	**504434**	504530.3	121.9
75X20S30	491774	493417	9549	493977	9855	494897	412.9	491796	492429.4	97.8	493403	498179.1	73.1	491774	492037.6	123.4	**491488**	491784.1	109.1
75X30S10	628585	636697	6898	634304	6986	636740	2087.7	630259	631982.2	78.2	637957	644569.3	113.9	628585	629374.6	152.9	**628433**	628605.6	153.6
75X30S15	611439	620356	7481	618688	7491	617984	1454.7	613001	614840.5	81.8	621149	626365.0	112.7	611439	612324.0	147.0	**611420**	611910.5	141.6
75X30S20	597785	600422	8697	601199	8389	604486	1773.0	599329	601011.5	84.9	605055	610181.4	114.4	597785	598392.3	140.4	**597505**	597861.4	131.7
75X30S30	576926	580490	9236	582766	9159	582388	1203.0	577843	579640.3	88.7	583551	586215.3	104.0	576926	578071.2	174.8	**576192**	576443.0	101.3
100X10S5	**771074**	773971	5571	773498	5339	777008	229.0	771172	771976.4	319.1	771368	774128.9	68.9	**771074**	771644.8	141.2	**771074**	771220.1	218.9
100X10S10	761741	764866	5024	763908	5399	766119	241.0	762282	763320.3	311.7	763469	765036.1	69.6	761741	762470.3	151.0	**761621**	762014.0	109.1
100X10S15	754485	757159	5832	757046	5333	757289	296.6	755040	755955.7	348.7	756236	758154.4	68.0	754485	754810.0	178.2	**754463**	754673.5	172.5
100X10S20	748605	750658	5150	750394	5160	751662	274.0	748611	749327.1	367.7	750047	751747.3	68.5	748605	749108.6	220.4	**748377**	748649.0	141.7
100X10S30	735783	738033	4375	737694	4886	739024	295.5	736248	737063.8	416.0	737505	739087.7	68.7	735783	736235.5	149.0	**735635**	735927.6	153.2
100X20S5	961620	966474	4920	970189	4500	970464	465.0	961900	964422.8	142.3	968861	973939.2	104.3	961620	962419.6	181.4	**960502**	961582.9	120.6
100X20S10	945453	951882	4474	949715	4940	958608	499.0	945835	948003.6	145.1	952317	958369.8	102.8	945453	946951.9	159.5	**944985**	945902.4	148.4
100X20S15	930568	935443	5047	936227	5007	936565	499.7	931525	933518.5	151.6	935246	940053.2	102.7	930568	931289.5	197.6	**929775**	930272.4	169.0
100X20S20	**916096**	921746	4938	922768	5203	925019	360.0	916505	918597.7	157.2	920851	923581.8	102.5	**916096**	917149.3	180.5	**916096**	917182.6	165.9
100X20S30	892547	894685	5555	896656	5332	903289	360.0	892755	894825.9	172.0	896202	899657.8	104.2	892547	893693.6	176.1	**892129**	893422.0	228.6
100X30S5	**1150550**	1170457	2818	1167044	2729	1173214	1814.0	1154077	1159790.8	121.8	1164617	1174570.3	141.4	**1150550**	1152976.9	248.1	**1150550**	1152804.3	163.9
100X30S10	1125044	1145700	4153	1142881	4307	1144807	1042.7	1127161	1130487.3	126.7	1140965	1149830.1	145.9	1125044	1127892.4	199.4	**1124567**	1126545.1	157.3
100X30S15	1101454	1113552	4482	1119040	4616	1121865	1116.0	1103176	1105138.4	130.4	1116441	1123978.2	146.0	1101454	1104339.4	196.4	**1101300**	1101709.9	199.4
100X30S20	1080706	1093126	4928	1096146	4806	1097480	943.0	1081933	1086086.7	133.1	1093174	1100436.0	143.8	1080706	1082370.1	222.4	**1080360**	1081154.6	182.6
100X30S30	1043537	1052682	5038	1057544	4958	1057666	900.0	1044499	1046736.6	140.5	1055745	1061462.5	138.2	1043537	1045586.4	200.5	**1043154**	1043900.3	144.4
AVG	**500604.7**	504253.5	5414.4	504448.9	5496.9	506013.3	517.9	501137.6	502307.1	117.4	504369.7	507363.0	70.5	500605.7	501257.7	117.7	500466.5	500853.5	108.0

algorithm to reach its final results.

As shown in Table 4, the FEA proposed in this study successfully attains optimal solutions (or the best-known solutions) for 49 out of the 50 instances in Set A, with the exception of the single case 20X10S15. Considering the relatively small scale of the instances in Set A, the performance of the FEA exhibits minimal differences when compared to the PTS1, PTS2, and RL-SO algorithms regarding these specific problem instances. In Set B, which contains 49 instances, Table 5 indicates that the FEA surpasses the best-known solution (BKS) reported in the literature in 26[2] of the 49 cases. Furthermore, it matches the best-known results for the remaining 23 instances. These results highlight the ability of the proposed FEA to provide high-quality solutions to the CCDAP problem.

To conduct a more thorough evaluation of the performance of the proposed FEA, Table 6 presents the

[2]The improved best-known solutions found by FEA are publicly available at: https://github.com/lyldft/CDAP.

number of instances in which the FEA outperformed, matched, or fell short compared to the best-known solution, referred to as "BKS" as documented in the existing literature, along with six leading algorithms for CCDAP, across the 99 benchmark test cases. This evaluation considers both the best and average results. To assess whether there are statistically significant differences among these comparative results, we employed the Wilcoxon signed-rank test, with the resulting p-values outlined in Table 6.

Table 6: Summarized comparisons of FEA with the best-known solutions and six reference algorithms on 99 benchmark CCDAP instances, the symbol '-' representing results that have not been reported.

	Best results				Average results			
Algorithm pair	#Win	#Tie	#Lose	p-value	#Win	#Tie	#Lose	p-value
FEA vs. BKS	**26**	72	1	6.28E-06	-	-	-	-
FEA vs. LS1	**71**	28	0	2.43E-13	-	-	-	-
FEA vs. LS2	**72**	27	0	1.66E-13	-	-	-	-
FEA vs. CHR	**72**	27	0	1.66E-13	-	-	-	-
FEA vs. PTS1	**45**	54	0	5.18E-09	**55**	44	0	1.11E-10
FEA vs. PTS2	**44**	55	0	7.62E-09	**64**	35	0	3.53E-12
FEA vs. RL-SO	**26**	73	0	8.30E-06	**41**	56	2	1.50E-07

According to Table 6, the proposed FEA shows excellent competitive performance when compared with the best-known solution and the six leading algorithms for CCDAP. The Wilcoxon signed-rank test provides a p-value significantly lower than 0.05, highlighting the statistical significance of the performance differences between the FEA and its competitors.

4.5. Computational results and comparison on UCDAP instances

To assess the performance of the FEA for solving the UCDAP, we compare it with five top performing algorithms specifically designed for the problem in the literature, including a greedy algorithm (GD) (Zhang et al., 2018), a scatter search algorithm (SS) (Tarhini et al., 2016), a genetic algorithm (GA) (Bermudez et al., 2001), an ant colony system algorithm (ACS) (Zhang et al., 2018), and a reinforcement learning based strategic oscillation algorithm (RL-SO) (Li et al., 2024). We use the same time limit stopping criterion from Zhang et al. (2018) for FEA as shown in Table 7.

Table 7: Time limits for the UCDAP instances.

Instance	Time limit(s)	Instance	Time limit(s)
FC12	0.2	SC48	3
FC24	1	SC64	5
FC48	3	SC96	15
FC96	15	SC192	100
FC192	100	SC300	200

Table 8 details the computational results obtained by the FEA and the five reference algorithms for the 40 UCDAP benchmark instances. To assess the performance of the proposed FEA more comprehensively, Table 9 presents a comparison of the FEA with the best-known solutions found in the literature, along with the five reference algorithms evaluated across the 40 UCDAP instances.

Tables 8 and 9 show that the proposed FEA has achieved 25 new record-breaking results and matched the best-known results for all but one instance. When compared to GD, SS, GA, ACS, and RL-SO, the FEA exhibits exceptional competitiveness in both the best and average solutions. Notably, FEA consistently

Table 8: Comparative results of the proposed FEA on 40 UCDAP instances with GD (Zhang et al., 2018), SS (Tarhini et al., 2016), GA (Bermudez et al., 2001), ACS (Zhang et al., 2018), and RL-SO (Li et al., 2024).

		GD	SS		GA		ACS		RL-SO			FEA		
Instance	BKS	OF	OF	AVG	OF	AVG	OF	AVG	OF	AVG	T(s)	OF	AVG	T(s)
FC12a	**19151.06**	19790.84	**19151.06**	19164.22	**19151.06**	19361.71	**19151.06**	19190.55	**19151.06**	19151.06	0.0	**19151.06**	19151.06	0.0
FC12b	**36643.11**	39000.21	**36643.11**	36661.25	**36643.11**	36882.78	**36643.11**	36643.11	**36643.11**	36643.11	0.0	**36643.11**	36643.11	0.0
FC12c	**56695.52**	60453.89	**56695.52**	56697.11	**56695.52**	56777.77	**56695.52**	56695.52	**56695.52**	56695.52	0.0	**56695.52**	56695.52	0.0
FC12d	**87732.20**	90425.56	**87732.20**	88312.19	**87732.20**	88120.47	**87732.20**	87732.20	**87732.20**	87732.20	0.0	**87732.20**	87732.20	0.0
FC24a	**56295.83**	73829.01	56660.19	61509.70	58960.66	61227.21	56349.74	56764.96	**56295.83**	56295.83	0.1	**56295.83**	56295.83	0.0
FC24b	**138190.85**	166894.92	138696.30	146341.73	140748.27	146973.05	**138190.85**	141568.18	**138190.85**	138190.85	0.0	**138190.85**	138190.85	0.0
FC24c	**227278.56**	271550.22	230661.22	234741.43	231110.10	234914.21	**227278.56**	230489.16	**227278.56**	227278.56	0.1	**227278.56**	227278.56	0.0
FC24d	**340710.01**	372523.80	342875.16	346313.55	344238.56	347422.15	**340710.01**	342590.87	**340710.01**	340710.01	0.1	**340710.01**	340710.01	0.1
FC48a	**208094.23**	291638.75	228944.22	240301.34	233844.41	249233.61	214678.32	222881.03	**208094.23**	209159.74	1.7	**208094.23**	208386.37	1.8
FC48b	549417.03	700302.36	572269.02	594699.07	598401.58	619560.75	563028.09	575359.25	549417.03	553747.66	1.4	**546790.80**	547770.91	1.3
FC48c	1039297.50	1216516.44	1055544.92	1086056.44	1086430.53	1104812.88	1051025.67	1063927.48	1039297.50	1042290.10	2.3	**1032633.79**	1033457.54	2.4
FC48d	1481308.79	1643617.72	1492960.38	1519199.33	1521410.64	1538809.15	1491426.12	1502554.95	1481308.79	1482501.89	1.6	**1474467.52**	1476770.95	2.3
FC96a	1115370.06	1611540.72	1176080.77	1237016.32	1315069.72	1366079.32	1146614.90	1192038.88	1115370.06	1120371.91	8.2	**1102861.31**	1108065.20	7.0
FC96b	2866490.01	3612130.64	2964379.62	3027248.75	3106296.61	3212751.50	2919418.33	2971180.02	2866490.01	2878524.85	6.4	**2854888.75**	2860433.52	9.0
FC96c	4692190.09	5509100.39	4789727.59	4856046.84	4963458.85	5061254.58	4782081.58	4812146.75	4692190.09	4711156.22	7.3	**4684513.27**	4690618.14	7.0
FC96d	7188103.89	7832802.77	7265340.38	7312004.01	7393625.95	7459814.73	7236850.63	7271295.54	7188103.89	7192722.60	7.9	**7163385.83**	7175802.27	8.5
FC192a	4891863.53	6798878.87	5209296.17	5270447.67	5796335.14	5946671.38	5178822.30	5243792.56	4891863.53	4925118.50	59.3	**4889079.59**	4901182.49	56.0
FC192b	11540994.27	13964856.36	11870904.81	11951968.07	12568533.12	12768173.58	11855968.63	11918365.21	11540994.27	11558457.94	63.5	**11481983.23**	11500920.07	57.6
FC192c	18330991.61	20722213.60	18648235.98	18764655.03	19354990.05	19613982.17	18609180.31	18713729.83	18330991.61	18348774.78	55.7	**18306455.68**	18313804.29	48.5
FC192d	26903925.51	28921027.07	27155541.75	27243191.59	27730048.36	27923306.76	27149822.94	27202833.20	26903925.51	26919385.47	35.7	**26846924.72**	26877057.71	38.2
SC48a	**242960.63**	287139.83	243659.41	247053.88	244863.69	251402.99	243029.61	245955.32	**242960.63**	242960.63	0.9	**242960.63**	242960.63	0.4
SC48b	**558890.55**	620569.20	560020.56	567203.62	562387.72	568986.22	559026.09	562369.29	**558890.55**	558890.55	0.3	**558890.55**	558890.55	0.1
SC48c	**949636.78**	1041376.04	951372.20	956636.16	955850.07	960743.09	**949636.78**	953221.73	**949636.78**	949636.78	0.3	**949636.78**	949636.78	0.2
SC48d	**1334062.19**	1410908.81	1335715.58	1339204.16	1336933.98	1343062.22	**1334062.19**	1338859.53	**1334062.19**	1334062.19	0.4	**1334062.19**	1334062.19	0.2
SC64a	445411.12	548131.48	446653.50	455557.26	455187.34	464936.15	447630.84	452989.42	445411.12	445935.82	2.3	**445154.09**	445284.28	2.7
SC64b	1024634.17	1191997.35	1028973.38	1041385.88	1037583.36	1050611.90	1025689.93	1033733.93	1024634.17	1024676.14	2.4	**1024606.60**	1024606.60	2.9
SC64c	**1629865.46**	1804211.30	1633641.39	1643659.05	1642391.08	1649550.25	1632907.57	1637768.39	**1629865.46**	1630201.05	2.4	**1629865.46**	1630740.20	4.4
SC64d	2256563.85	2374841.05	2257335.06	2267789.10	2266725.65	2274744.09	2257359.48	2264420.02	2256563.85	2256629.96	3.1	**2256481.56**	2256485.09	3.5
SC96a	1087201.96	1294226.62	1093760.68	1112837.80	1120573.50	1138542.86	1097244.29	1114206.19	1087201.96	1087394.26	6.3	**1086188.07**	1086509.05	6.8
SC96b	2499351.20	2849290.12	2506841.11	2533649.08	2537052.25	2560055.06	2501125.05	2522159.93	2499351.20	2500021.61	10.0	**2498831.11**	2499111.77	8.9
SC96c	3852744.90	4164282.98	3867741.11	3889926.09	3892131.30	3916201.93	3854540.75	3880443.27	3852744.90	3853090.52	8.7	**3852057.12**	3852231.65	6.2
SC96d	5380130.63	5711507.32	5382754.44	5407996.25	5411594.57	5439763.81	5380388.39	5410317.62	5380130.63	5380361.62	8.4	**5380015.46**	5380096.86	9.4
SC192a	5726836.88	7118906.79	5800374.05	5877762.37	5970586.38	6046223.64	5777475.55	5852008.89	5726836.88	5728903.87	58.5	**5721167.46**	5722516.78	62.4
SC192b	12852429.05	14642126.67	12928441.98	13021704.61	13103982.22	13237988.15	12921073.98	13002310.47	12852429.05	12854418.54	49.8	**12849422.63**	12852240.02	56.5
SC192c	19653535.27	21414021.44	19736893.76	19835367.87	19915108.44	20045848.79	19738043.91	19825934.94	19653535.27	19656832.40	58.8	**19651746.51**	19653375.29	55.9
SC192d	27860099.45	29452942.33	27935918.04	28026272.43	28111883.59	28173283.24	27910134.02	28006066.81	27860099.45	27861921.84	54.4	**27858676.80**	27859529.55	50.5
SC300a	**4320268.66**	17032541.30	14431764.18	14561572.14	14864949.90	15054465.63	**4320268.66**	14489309.81	14276938.29	14285147.95	143.6	14271977.53	14273340.39	101.0
SC300b	31816411.01	35669638.69	31923025.74	32155457.57	32571902.72	32741426.25	31913740.31	32093159.21	31816411.01	31842164.15	126.2	**31772374.74**	31809386.68	85.1
SC300c	49872756.00	53981808.28	50057517.64	50212334.74	50534100.07	50737469.08	50021193.15	50162149.78	49872756.00	49949773.12	156.7	**49855010.36**	49883596.53	139.5
SC300d	68228377.88	71478678.01	68376779.86	68469274.90	68714408.02	68877012.86	68299555.48	68411154.51	68228377.88	68237238.63	116.7	**68143632.66**	68146215.32	143.8
AVG	8084072.78	9200205.99	8397438.10	8442880.52	8547348.01	8609711.20	8133644.87	8423007.96	8332989.52	8339629.26	26.53	8323438.35	8327944.57	24.5

Table 9: Summarized comparisons of FEA with the best-known solutions and five reference algorithms on 40 benchmark UCDAP instances, the symbol '-' representing results that have not been reported.

	Best results				Average results			
Algorithm pair	#Win	#Tie	#Lose	p-value	#Win	#Tie	#Lose	p-value
FEA vs. BKS	**25**	14	1	1.46E-04	-	-	-	-
FEA vs. GD	**40**	0	0	3.57E-08	-	-	-	-
FEA vs. SS	**36**	4	0	1.68E-07	**40**	0	0	3.57E-08
FEA vs. GA	**36**	4	0	1.68E-07	**40**	0	0	3.57E-08
FEA vs. ACS	**30**	9	1	2.11E-05	**37**	3	0	1.14E-07
FEA vs. RL-SO	**26**	14	0	8.30E-06	**27**	12	1	5.86E-06

produces results at least as good as those from any of the reference algorithms and significantly outperforms these algorithms in numerous instances. The results of the Wilcoxon signed-rank test substantiate these observations, indicating p-values below 0.05, which confirm the statistical relevance of the performance differences between the FEA and the five reference algorithms. Collectively, these comparative outcomes clearly indicate the FEA's superiority over the five leading UCDAP algorithms in the literature.

In summary, the proposed frequent pattern mining driven algorithm is versatile and has the potential to be applied to a range of CDAPs.

5. Analyses and discussions

This section shows an analysis of two key components of FEA: the frequent pattern mining technique and the two-phase local search method.

5.1. The impact of the frequent pattern mining

Frequent pattern mining plays a pivotal role in the proposed FEA approach. As outlined in Section 3.1, the mined frequent patterns guide the FEA in exploring relevant areas of the search space, thereby enhancing the efficiency of the search process. To assess the benefits of frequent pattern mining, we developed a variant of the FEA, named FEA1, which excludes the pattern mining. In this variant, a basic backbone crossover operator, as described in Zhou et al. (2020b), is used to generate new solutions, replacing the use of mined patterns. This backbone crossover operator preserves and transfers common components from the parent solutions to the offspring. In this experiment, the backbone crossover operator generates a partial solution by preserving the common truck-to-door assignments from the two selected solutions in the population, and then completes the partial solution in a random manner.

Table 10 presents the summarized comparative results of FEA with and without pattern mining across all instances. These results include the number of times FEA outperforms, matches, or falls short of FEA1, in terms of both the best and average results, across all benchmark instances. To evaluate whether there are statistically significant differences in these results, we applied the Wilcoxon signed-rank test, with the corresponding p-values provided in Table 10. The findings clearly indicate that FEA, which integrates mined patterns to generate new solutions, consistently outperforms its counterpart without pattern mining in both the best and average objective values, highlighting the effectiveness of frequent pattern mining.

Table 10: Summarized comparisons of FEA with the FEA1 across all the benchmark instances.

		Best results				Average results			
Algorithm pair	Instance	#Win	#Tie	#Lose	p-value	#Win	#Tie	#Lose	p-value
FEA vs. FEA1	SetA (CCDAP)	0	50	0	-	0	50	0	-
FEA vs. FEA1	SetB (CCDAP)	42	7	0	1.65E-08	43	6	0	8.26E-08
FEA vs. FEA1	FC (UCDAP)	12	8	0	3.35E-03	12	8	0	4.74E-03
FEA vs. FEA1	SC (UCDAP)	15	5	0	6.55E-04	15	5	0	8.20E-05

5.2. Advantage of two-phase local search

As detailed in Section 3.7, the local refinement approach employs a two-phase local exploring and focusing search procedure. The first phase, known as the infeasible local exploration search phase, focuses on discovering diverse solutions, while the second phase, an intensification-driven local focusing search phase, aims to refine the solutions found. To evaluate the benefits of this combined two-phase local search approach, we compare the performance of the FEA with two of its variants. The first variant FEA2 omits the infeasible local exploration phase, and the second variant FEA3 excludes the intensification-driven local focusing phase. This experiment was conducted on 30 most challenging CCDAP benchmark instances from the literature.

The best and average performance results for the three FEA are presented in Figure 3, where the y-axis indicates the percentage gap between the latest best-known solution and the results obtained by each

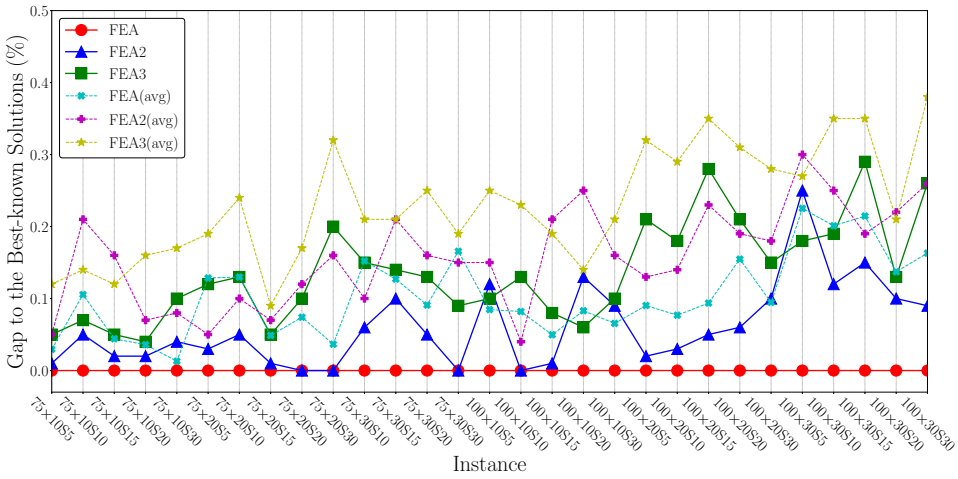


Figure 3: Comparison between FEA and two FEA variants.

algorithm. The comparison, as illustrated in Figure 3, clearly shows that FEA surpasses its two variations (FEA2 and FEA3), thereby confirming the efficacy of its two-phase local search method.

6. Real-world example

To demonstrate the practical applicability of our proposed algorithm, we apply it to solve a real-world cross-dock door assignment problem from Hubei Meritar Logistics Co., Ltd. in China, an innovative company that offers smart logistics services to manufacturing industry clients. In this scenario, the cross-dock network is in a situation where the number of available doors is significantly smaller than the number of trucks, and the cross-dock capacity is also limited.

Figure 4 presents an example illustrating the real-world cross-docking door assignment problem. In this scenario, Trucks 1 to 7 each carry three pallets from different suppliers, and these pallets need to be assigned to cross-docking doors or handled through outdoor operations. The arrival and departure time for each truck t are denoted as A_t and E_t, respectively. Circles are used to represent these pallets, where pallets of the same color indicate they come from the same truck. As shown in Figure 4, the cargo of Trucks 1 to 4 is processed through indoor operations, while the cargo of Trucks 5 to 7 is handled outdoors. The shipping list on the left details the transportation information of all relevant trucks.

In this real-world case, three types of constraints must be satisfied:

1. **Time window constraints**: Trucks with overlapping time windows cannot be assigned to the same warehouse door.
2. **Operation time constraints**: when truck m needs to transport goods to truck n, the arrival time of goods from m to n must not be later than n's departure time. For example, $A_3 + D_{47} \leq E_4$.
3. **Internal capacity constraints**: the cross-dock's capacity must not be exceeded at any event time, such as truck arrival or departure time. For example, during the period from time A_1 to E_3, the transportation volume T_{13} continuously occupies the internal capacity.

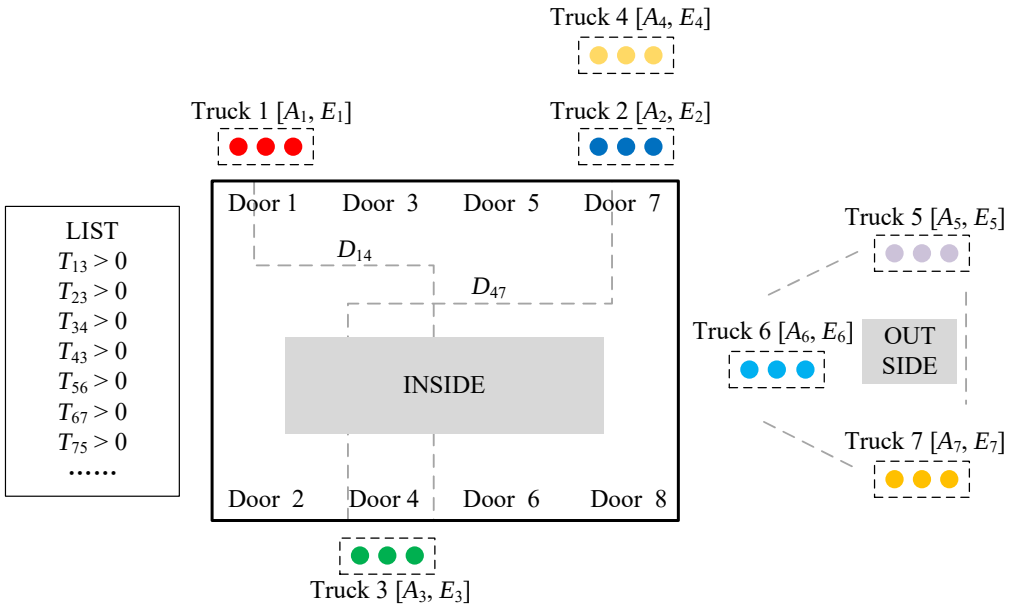


Figure 4: An example of real-world application.

Outdoor operations require additional manual intervention, resulting in a penalty on the corresponding quantity of cargo. Let T_{mn} represent the number of pallets transported from truck m to truck n, D_{ij} denote the transportation time for cargo from warehouse door i to warehouse door j, and P_{mn} represent the penalty per unit of cargo transported from truck m to truck n through outdoor operations. Specifically, the total cost of this cross-docking pallet handling plan can be expressed as: $f = T_{13} \times D_{14} + (T_{23} + T_{34} + T_{43}) \times D_{47} + T_{56} \times P_{56} + T_{67} \times P_{67} + T_{75} \times P_{75}$.

To evaluate the performance of our proposed algorithm, we collected five real-world instances from Hubei Meritar Logistics Co., Ltd., and adapted our algorithm to solve these real-world cases. We compare our proposed algorithm with the greedy heuristic (GH) currently used by Hubei Meritar Logistics Co., Ltd. Table 11 presents the solutions obtained by both FEA and GH, along with the percentage gap between the results.

Table 11: Computational results of FEA and GH.

Instance	GH	FEA	$Gap(\%)$
1	4293	4271	0.51
2	12723	11611	8.74
3	6531	6385	2.24
4	8394	8170	2.67
5	11254	10360	7.94
Avg.			4.42

As shown in Table 11, FEA significantly outperforms the greedy heuristic (GH), achieving an average improvement of 4.42%. This real-world application further demonstrates the practical significance of the proposed algorithm.

7. Conclusion

Cross-dock door assignment problems are useful models in logistics and supply chain management. They pose significant challenges due to their direct influence on the efficiency, cost, and service quality of cross-dock operations. In this study, we introduce the first evolutionary algorithm driven by frequent pattern mining to address two extensively studied CDAPs, with and without capacity limits for each door (i.e., CCDAP and UCDAP).

Our proposed algorithm introduces several innovative components. First, it employs a specialized data mining technique aimed at extracting significant frequent patterns, which are used to generate new promising solutions. By effectively combining frequent pattern mining with robust optimization techniques within a population-based evolutionary algorithm, our approach ensures a focused and informed exploration of the search space, ultimately leading to the efficient identification of high-quality solutions. Second, the algorithm integrates an efficient two-phase local search approach, consisting of an infeasible local exploration search phase followed by an intensification-driven local focusing search phase. This combination allows for a comprehensive exploration of specific search regions, facilitating the discovery of high-quality solutions.

To assess our algorithm's effectiveness, we conducted comprehensive tests on two sets of 99 CCDAP benchmark instances and a set of 40 UCDAP benchmark instances taken from existing literature. The computational results indicate that our algorithm surpasses other prominent algorithms. Specifically, for the CCDAP instances, our algorithm establishes 26 new best upper bounds and matches 72 remaining best-known solutions. In the case of UCDAP, it discovers 25 new best upper bounds and matches the best-known results for all but one instance. In addition, we performed additional experiments to assess the impacts of crucial elements, such as the frequent pattern mining techniques and the two-phase local search method.

Our proposed algorithm demonstrates strong effectiveness and robustness, delivering high-quality solutions in a reasonable timeframe. Its underlying strategies could be modified to create effective algorithms for other CDAPs that involve additional constraints such as time windows, vehicle categories, and constraints related to loading and unloading equipments. It would be an interesting to explore how the algorithm adapts and performs in other complex CDAP cases. Meanwhile, in several benchmark scenarios, the algorithm's ability to find high-quality solutions is limited, suggesting that improvements could be made. One way to overcome these limitations would be to develop other neighborhoods and faster neighborhood evaluation techniques to further enhance the algorithm's performance. Finally, since the proposed algorithm is a heuristic method, it does not guarantee optimal solutions. Further research is necessary to develop efficient exact methods.

Acknowledgment

We are grateful to the editors and reviewers for their comments, which helped us to significantly improve this paper. We also thank Hubei Meritar Logistics Co., Ltd. for generously sharing their real-world cross-dock door assignment application and data. This work was partially supported by the National Natural Science Foundation of China (Grant No. 72371076, 72471100, 72122006, 72501305), the China Postdoctoral Science Foundation - Hubei Joint Support Program (Grant No. 2025T099HB), the Postdoctoral Fellowship Program of China Postdoctoral Science Foundation (Grant No. GZC20242029), and the Hubei Provincial Natural Science Foundation of China (Grand No. 2025AFC038).

References

Aggarwal, C. C., Bhuiyan, M. A., & Hasan, M. A. (2014). *Frequent pattern mining algorithms: A survey*. Springer.

Ayadi, W., Elloumi, M., & Hao, J.-K. (2012). Pattern-driven neighborhood search for biclustering of microarray data. *BMC bioinformatics*, *13*, S11.

Badhon, B., Kabir, M. M. J., Xu, S., & Kabir, M. (2021). A survey on association rule mining based on evolutionary algorithms. *International Journal of Computers and Applications*, *43*, 775–785.

Bartholdi, J. J., & Gue, K. R. (2000). Reducing labor costs in an ltl crossdocking terminal. *Operations Research*, *48*, 823–832.

Bartholdi, J. J., & Gue, K. R. (2004). The best shape for a crossdock. *Transportation Science*, *38*, 235–244.

Bermudez, R., Cole, M. H., & Center, M.-B. T. (2001). *Genetic Algorithm Approach to Door Assignments in Breakbulk Terminals*. Technical Report.

Bodnar, P., de Koster, R., & Azadeh, K. (2017). Scheduling trucks in a cross-dock with mixed service mode dock doors. *Transportation Science*, *51*, 112–131.

Boysen, N., & Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega*, *38*, 413–422.

Bozer, Y. A., & Carlo, H. J. (2008). Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions*, *40*, 1007–1018.

Buijs, P., Vis, I. F., & Carlo, H. J. (2014). Synchronization in cross-docking networks: A research classification and framework. *European Journal of Operational Research*, *239*, 593–608.

Gaudioso, M., Monaco, M. F., & Sammarra, M. (2021). A lagrangian heuristics for the truck scheduling problem in multi-door, multi-product cross-docking with constant processing time. *Omega*, *101*, 102255.

Gelareh, S., Glover, F., Guemri, O., Hanafi, S., Nduwayo, P., & Todosijević, R. (2020). A comparative study of formulations for a cross-dock door assignment problem. *Omega*, *91*, 102015.

Gelareh, S., Monemi, R. N., Semet, F., & Goncalves, G. (2016). A branch-and-cut algorithm for the truck dock assignment problem with operational time constraints. *European Journal of Operational Research*, *249*, 1144–1152.

Glover, F. (1997). Tabu search and adaptive memory programming—advances, applications and challenges. In *Interfaces in Computer Science and Operations Research* (pp. 1–75). Springer.

Grahne, G., & Zhu, J. (2003). Efficiently using prefix-trees in mining frequent itemsets. In *FIMI* (p. 65). volume 90.

Gue, K. R. (1999). The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*, *33*, 419–428.

Guemri, O., Nduwayo, P., Todosijević, R., Hanafi, S., & Glover, F. (2019). Probabilistic tabu search for the cross-docking assignment problem. *European Journal of Operational Research*, *277*, 875–885.

Guignard, M., Hahn, P. M., Pessoa, A. A., & da Silva, D. C. (2012). Algorithms for the cross-dock door assignment problem. In *Proceedings of the Fourth International Workshop on Model-based Metaheuristics* (pp. 145–162).

Kshirsagar, V. K., Tidke, S. M., & Vishnu, S. (2012). Intrusion detection system using genetic algorithm and data mining: An overview. *International Journal of Computer Science and Informatics ISSN (PRINT)*, *2231*.

Küçükoğlu, İ., & Öztürk, N. (2017). Two-stage optimisation method for material flow and allocation management in cross-docking networks. *International Journal of Production Research*, *55*, 410–429.

Küçükoğlu, İ., & Öztürk, N. (2019). A hybrid meta-heuristic algorithm for vehicle routing and packing problem with cross-docking. *Journal of Intelligent Manufacturing*, *30*, 2927–2943.

Küçükoğlu, İ., & Öztürk, N. (2023). A new hybrid genetic algorithm to optimize distribution and operational plans for cross-docking satellites. *Soft Computing*, *27*, 18723–18738.

Kumar, S., & Rao, C. (2009). Application of ant colony, genetic algorithm and data mining-based techniques for scheduling. *Robotics and Computer-Integrated Manufacturing*, *25*, 901–908.

Li, M., Hao, J.-K., & Wu, Q. (2022). Learning-driven feasible and infeasible tabu search for airport gate assignment. *European Journal of Operational Research*, *302*, 172–186.

Li, M., Hao, J.-K., & Wu, Q. (2024). A flow based formulation and a reinforcement learning based strategic oscillation for cross-dock door assignment. *European Journal of Operational Research*, *312*, 473–492.

Li, M., Hao, J.-K., & Wu, Q. (2025). Combining hash-based tabu search and frequent pattern mining for job-shop scheduling. *IISE Transactions*, (pp. 1–24).

Lim, A., Ma, H., & Miao, Z. (2006). Truck dock assignment problem with time windows and capacity constraint in transshipment network through crossdocks. In *Computational Science and Its Applications-ICCSA 2006: International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part III 6* (pp. 688–697). Springer.

Liu, R., Xie, X., & Garaix, T. (2014). Hybridization of tabu search with feasible and infeasible local searches for periodic home health care logistics. *Omega*, *47*, 17–32.

Liu, Z., Liu, J., Liu, X., Yan, J., Cheng, Y., & Chen, Y. (2025). An iterated adaptive large neighborhood search algorithm for the large-scale communication satellite range scheduling problem. *Expert Systems with Applications*, *278*, 127377.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58.

Miao, Z., Cai, S., & Xu, D. (2014). Applying an adaptive tabu search algorithm to optimize truck-dock assignment in the crossdock management system. *Expert Systems with Applications*, *41*, 16–22.

Miao, Z., Lim, A., & Ma, H. (2009). Truck dock assignment problem with operational time constraint within crossdocks. *European journal of operational research*, *192*, 105–115.

Nassief, W., Contreras, I., & As' Ad, R. (2016). A mixed-integer programming formulation and lagrangean relaxation for the cross-dock door assignment problem. *International Journal of Production Research*, *54*, 494–508.

Qin, J., Xu, X., Wu, Q., & Cheng, T. (2016). Hybridization of tabu search with feasible and infeasible local searches for the quadratic multiple knapsack problem. *Computers & Operations Research*, *66*, 199–214.

Raschip, M., Croitoru, C., & Stoffel, K. (2015). Using association rules to guide evolutionary search in solving constraint satisfaction. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 744–750). IEEE.

Ribeiro, M. H., Plastino, A., & Martins, S. L. (2006). Hybridization of grasp metaheuristic with data mining techniques. *Journal of Mathematical Modelling and Algorithms*, *5*, 23–41.

Tarhini, A. A., Yunis, M. M., & Chamseddine, M. (2016). Natural optimization algorithms for the cross-dock door assignment problem. *IEEE Transactions on Intelligent Transportation Systems*, *17*, 2324–2333.

Tian, Y., Lu, C., Zhang, X., Cheng, F., & Jin, Y. (2020). A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems. *IEEE transactions on cybernetics*, *52*, 6784–6797.

Tsui, L. Y., & Chang, C.-H. (1990). A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering*, *19*, 309–312.

Tsui, L. Y., & Chang, C.-H. (1992). An optimal solution to a dock door assignment problem. *Computers & Industrial Engineering*, *23*, 283–286.

Van Belle, J., Valckenaers, P., & Cattrysse, D. (2012). Cross-docking: State of the art. *Omega*, *40*, 827–846.

Wang, Y., Wu, Q., & Glover, F. (2017). Effective metaheuristic algorithms for the minimum differential dispersion problem. *European Journal of Operational Research*, *258*, 829–843.

Woodruff, D. L., & Zemel, E. (1993). Hashing vectors for tabu search. *Annals of Operations Research*, *41*, 123–137.

Wu, J., Yao, F., Song, Y., He, L., Lu, F., Du, Y., Yan, J., Chen, Y., Xing, L., & Ou, J. (2023). Frequent pattern-based parallel search approach for time-dependent agile earth observation satellite scheduling. *Information Sciences*, *636*, 118924.

Wu, Q., Wang, Y., & Glover, F. (2020). Advanced tabu search algorithms for bipartite boolean quadratic programs guided by strategic oscillation and path relinking. *INFORMS Journal on Computing*, *32*, 74–89.

Wu, Y.-T., An, Y. J., Geller, J., & Wu, Y.-T. (2006). A data mining based genetic algorithm. In *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)* (pp. 6–pp). IEEE.

Yu, W., & Egbelu, P. J. (2008). Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research*, *184*, 377–396.

Yu, X., & Gen, M. (2010). *Introduction to evolutionary algorithms*. Springer Science & Business Media.

Zhang, P., Rong, J., Tian, Y., Zhang, Y., Yang, S., & Zhang, X. (2025). A frequent pattern-based coevolutionary framework for multi-component spectral feature selection. *Swarm and Evolutionary Computation*, *98*, 102077.

Zhang, Y.-H., Gong, Y.-J., Chen, W.-N., Gu, T.-L., Yuan, H.-Q., & Zhang, J. (2018). A dual-colony ant algorithm for the receiving and shipping door assignments in cross-docks. *IEEE Transactions on Intelligent Transportation Systems*, *20*, 2523–2539.

Zhen, L., & Li, H. (2022). A literature review of smart warehouse operations management. *Frontiers of Engineering Management*, *9*, 31–55.

Zhou, Y., Hao, J.-K., & Duval, B. (2020a). Frequent pattern-based search: A case study on the quadratic assignment problem. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*, 1503–1515.

Zhou, Y., Hao, J.-K., Fu, Z.-H., Wang, Z., & Lai, X. (2020b). Variable population memetic search: A case study on the critical node problem. *IEEE Transactions on evolutionary computation*, *25*, 187–200.

Zhu, Y.-R., Hahn, P. M., Liu, Y., & Guignard, M. (2009). New approach for the cross-dock door assignment problem. In *Proceedings of the XLI Brazilian Symposium on Operations Research* (pp. 1226–1236).