The freight allocation problem with all-units quantity-based discount: A heuristic algorithm

Hu Qin\textsuperscript{a,b}, Meifeng Luo\textsuperscript{c,*}, Xiang Gao\textsuperscript{b}, Andrew Lim\textsuperscript{b}

\textsuperscript{a} School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China
\textsuperscript{b} Department of Management Sciences, City University of Hong Kong, Tat Chee Ave, Kowloon Tong, Hong Kong
\textsuperscript{c} Department of Logistics and Maritime Studies, Faculty of Business, Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

\begin{abstract}
This paper studies a problem encountered by a buying office for one of the largest retail distributors in the world. An important task for the buying office is to plan the distribution of goods from Asia to various destinations across Europe. The goods are transported along shipping lanes by shipping companies, which offer different discount rates depending on the freight quantity. To increase the reliability of transportation, the shipper imposes a quantity limit on each shipping company on each shipping lane. To guarantee a minimum business volume, each shipping company requests a minimum total freight quantity over all lanes if it is contracted. The task involves allocating projected demand of each shipping lane to shipping companies subject to the above conditions such that the total cost is minimized.

Existing work on this and related problems employs commercial linear programming software to solve their models. However, since the problem is \textit{NP}-hard in the strong sense, it is unlikely to be solvable optimally in reasonable time for large cases. Hence, we propose the first heuristic-based algorithm for the problem, which combines a filter-and-fan search scheme with a tabu search mechanism. Experiments on randomly generated test instances show that as the size of the problem increases, our algorithm produces superior solutions in less time compared to a leading mixed-integer programming solver.
\end{abstract}

1. Introduction

This study is motivated by a project awarded by a Hong Kong-based buying office (henceforth referred to as the \textit{shipper}) for one of the largest international retail corporations in the world with over 2000 outlets in Europe, Africa and Asia. The shipper annually procures diverse products, from textiles and food stuffs to major electrical appliances, from over one thousand suppliers across Asia to satisfy the demands of parent company's sales divisions that are distributed across Europe. Long-distance ocean shipping is the main transportation mode for the shipper for the delivery of the procured products, accounting for around 95% of its total annual turnover on average. Hence, the shipper maintains close relationships with most of the leading international shipping companies, which offer different discount rates depending on the freight quantity. To increase the reliability of transportation, the shipper imposes a quantity limit on each shipping company on each shipping lane. To guarantee a minimum business volume, each shipping company requests a minimum total freight quantity over all lanes if it is contracted.

This paper examines the problem of allocating the freight quantity for all lanes to the carriers such that the total transportation cost is minimized.

This paper studies a problem encountered by a buying office for one of the largest retail distributors in the world. An important task for the buying office is to plan the distribution of goods from Asia to various destinations across Europe. The goods are transported along shipping lanes by shipping companies, which offer different discount rates depending on the freight quantity. To increase the reliability of transportation, the shipper imposes a quantity limit on each shipping company on each shipping lane. To guarantee a minimum business volume, each shipping company requests a minimum total freight quantity over all lanes if it is contracted. The task involves allocating projected demand of each shipping lane to shipping companies subject to the above conditions such that the total cost is minimized.

The carrier offers discount rates to the shipper according to the total freight quantity it obtains across all lanes, and if it is contracted this total freight quantity must exceed its requested MQC. As a safeguard against the inability of a carrier to fulfill its contractual obligations due to unforeseen circumstances, the shipper also specifies a minimum number of carriers for each lane.

This paper examines the problem of allocating the freight quantity for all lanes to the carriers such that the total transportation cost is minimized.

The shipper performs this freight allocation at the strategic level. At the beginning of every fiscal year, the shipper forecasts the total quantity of freight (demand) for the coming year on each lane, taking into account possible market fluctuations. Price quotes are collected from each carrier along with its discount information and minimum quantity commitment (MQC) [1]. The carrier offers discount rates to the shipper according to the total freight quantity it obtains across all lanes, and if it is contracted this total freight quantity must exceed its requested MQC. As a safeguard against the inability of a carrier to fulfill its contractual obligations due to unforeseen circumstances, the shipper also specifies a minimum number of carriers for each lane.

We call this problem the freight allocation problem with all-units quantity-based discount (FAPAQD). While some research has been done on problems of this nature, in general they employ exact linear and integer programming solvers to produce optimal solutions on small instances. However, the problem is \textit{NP}-hard in the strong sense, and therefore such approaches are unlikely to be successful for the large and practical scenarios faced by the shipper. Consequently, we developed the first tailored heuristic for the FAPAQD, which makes use of the polynomial-time...
solvable min-cost network flow problem to generate solutions, and then uses a filter-and-fan search heuristic with tabu search to locate good solutions.

In our problem, the discount rates offered by each carrier are expressed as discount intervals. If the discount interval adopted for each carrier is determined, the resultant model can be solved in polynomial-time. Thus, identifying the best discount interval to select for each carrier would allow us to find the optimal solution to this problem. Our approach is based on searching the space of combinations of discount intervals for the carriers. In this paper, we propose a filter-and-fan technique with tabu search (F&F) for this purpose; computational experiments on randomly generated instances of practical size show that our approach outperforms CPLEX 11.0 in terms of both solution quality and computation time. Note that other than the F&F, we have also investigated using a standard simulated annealing algorithm or tabu search algorithm with similar neighborhood structures and a variety of parameter settings, but preliminary experiments indicate that our F&F outperforms both these approaches.

The rest of the paper is organized as follows. In Section 2, we provide an overview of existing research that considers the MQC constraint or discounts. We then describe the FAPAQD in detail in Section 3, where we explain the notations used in the remainder of the paper and formulate the problem as a mixed-integer programming (MIP) model. This is followed in Section 4 with a description of the F&F for this problem. Our experimental results are given in Section 5, and we conclude our paper in Section 6 with some closing remarks.

2. Literature review

The MQC constraint has been previously studied in the analysis of supply contracts [2–4]. In commitment-purchase contracts, buyers commit in advance to purchasing a minimum quantity of products from the supplier, and the unit price of the product is based on the total quantity or total dollar amount purchased. In recent years, the transportation service procurement problem with the MQC constraint has also received attention. Lim et al. [1] incorporated the MQC constraint into the traditional transportation problem, which makes the problem intractable. The authors proposed an MIP model defined by a number of strong facets and applied a branch-and-cut scheme, a linear programming rounding heuristic, and a greedy approximation method. This work was extended in Lim et al. [5] by considering a fixed selection cost associated with each carrier; for this extended problem it was shown that not only is finding an optimum solution NP-hard in the strong sense, but finding a feasible solution is also N’P-hard. The problem studied in our paper can be regarded as another extension of Lim et al. [1].

There is a substantial amount of literature related to procurement problems involving discounts. Two commonly used discount policies are the all-units discount and the incremental discount. Under the all-units discount policy, the discounted price applies to all units purchased, while the discounted price applies only for quantities within the associated discount level for the incremental discount policy. The discount can be based on total business volume, which is the total dollar amount of business across all products purchased from the suppliers [6]; or based on total quantity, in which the discount is given according to the total number of units of all products purchased from the suppliers [7]. If these problems are distinguished by the number of suppliers and products, then procurement problems with discounts can be classified into the following four categories: (1) single supplier and single product [8–11]; (2) single supplier and multiple products [12–17]; (3) multiple suppliers and single product [18–21]; and (4) multiple suppliers and multiple products [6,7,22–26]. The FAPAQD can be viewed as a multiple-supplier, multiple product procurement problem; the rest of this section provides an overview of existing work of this type.

Katz et al. [22] and Sadrian and Yoon [6] introduced a Procurement Decision Support System (PDSS) that improved the cost-effectiveness of purchasing activities of regional Bell telephone companies. The PDSS uses the optimization software LINGO, but the detailed numerical results were not revealed by the authors. The model in Crama et al. [23] considers a company that manufactures a set of products, each of which can be obtained by blending a set of ingredients according to certain recipes; ingredients are purchased from a number of suppliers who offer discount schedules. This model is more complex than the one examined in our study because it requires the concurrent determination of the recipe used for each product along with the quantity of each ingredient purchased from each supplier. Small test instances for this problem were solved by a branch-and-bound algorithm embedded in the XA solver. Xia and Wu [24] formulated the procurement problem as a multi-objective MIP model and utilized the optimization toolbox in MATLAB to solve the problem. Stadlter [25] presented a general model that is applicable to both all-units and incremental discount policies, and solved the model using the standard MIP solver Xpress-MP optimizer. Goossens et al. [7] proposed a min-cost network flow based branch-and-bound algorithm that uses the commercial MIP solver CPLEX 8.1 to solve the procurement problem under a total quantity discount structure optimally. However, the experimental results show that this algorithm is only applicable to small instances as its performance is worse than CPLEX 8.1 when solving medium to large instances. Sawik [26] studied procurement models that simultaneously consider discount and some other influence factors, such as price, quality of purchased parts and reliability of on time delivery. The author conducted the experiments using the AMPL programming language and the CPLEX 11.0 solver with the default settings.

Importantly, the test data employed for the evaluation of the above approaches are all much smaller than the hundreds of lanes and dozens of carriers that our problem must consider; for example, the largest instances considered by Goossens et al. [7] consist of only 50 suppliers and 100 products. While problem instances of this scale are appropriate for the procurement problems examined in these publications, they are insufficient for our purposes.

3. Problem formulation

We modeled the problem faced by the shipper in the following manner. There is a set of candidate carriers \(I = \{1, 2, \ldots, n\}\) and a set of lanes \(J = \{1, 2, \ldots, m\}\). Not all carriers can operate on all lanes; the set \(N\) contains \((i, j)\) pairs, \(i \in I, j \in J\), indicating that carrier \(i\) operates on lane \(j\). The projected demand for lane \(j\) in the upcoming fiscal year is given by \(d_j\).

Each carrier \(i\) has an MQC, denoted by \(b_i\), which defines the minimum quantity that must be assigned to that carrier if it is selected. The regular price quoted by carrier \(i\) to transport one unit of product on lane \(j\) is denoted by \(p_{ij}\). Each carrier \(i\) also defines a set of discount intervals \(K_i = \{1, 2, \ldots, k_i\}\) that describe the percentage discount on all units assigned to that carrier for each quantity range. Table 1 shows examples of discount intervals for two carriers along with their corresponding MQCs, where both carriers have defined 4 discount intervals. We denote the discount lower bound of the \(k\)th interval for carrier \(i\) by \(p_{ik}\) and the discount value of the \(k\)th interval for carrier \(i\) by \(a_{ik}\). In the example, the values for Carrier 1 are \(p_{11} = 2400\), \(p_{12} = 3000\), \(a_{11} = 15\), \(a_{12} = 20\), \(a_{13} = 25\), \(a_{14} = 30\), and for Carrier 2, \(p_{21} = 2800\), \(p_{22} = 3500\), \(a_{21} = 20\), \(a_{22} = 25\), \(a_{23} = 30\), \(a_{24} = 35\).
The objective function (1) minimizes the total procurement cost, comprising the total freight space cost and the total penalty cost across all lanes. Constraints (2) state that on each lane, the sum of the freight capacity purchased from all carriers and the unfulfilled quantity must satisfy demand. Constraints (3) guarantee that if carrier $i$ offers discount interval $k$ to the shipper, then the total freight capacity purchased from carrier $i$ across all lanes must be greater than or equal to the discount lower bound $b_k$. Conversely, if carrier $i$ does not offer discount interval $k$ to the shipper, then all corresponding decision variables $x_{ijk}$ must equal zero, as required by constraints (4). Constraints (5) ensure that at most one discount interval per carrier can be selected. When a carrier $i$ cannot provide service on lane $j$ (i.e., $(i, j) \notin N$), then the corresponding decision variables $x_{ijk} = 0$ for all $k$, as given by constraints (6). Finally, the quantity limit for each carrier on each lane is controlled by constraints (7). In our model, the variables $x_{ijk}$ and $u_j$ are non-negative real numbers.

The MIP model implicitly includes the MQC constraints. Given a feasible solution $(x, u, z)$, if $\sum_{k \in K} z_{ik} = 1$, then carrier $i$ must be selected and there must exist exactly one $z_{ik} = 1, 1 \leq k \leq k_i$. According to constraints (3), we have $\sum_{j \in N} x_{ijk} \geq b_k \geq b_i$, which guarantees that the total quantity purchased from carrier $i$ satisfies its MQC. Otherwise, if $\sum_{k \in K} z_{ik} = 0$, then all the decision variables $x_{ijk}$ relating to carrier $i$ must be zero, which implies that carrier $i$ is not selected on all lanes.

**Theorem 1.** The FAPAQD is \(\mathcal{NP} - \text{hard in the strong sense.}\)

**Proof.** We prove this by showing that the Transportation Problem with MQC, which is \(\mathcal{NP} - \text{hard in the strong sense,}\) is a special case of the FAPAQD. The details can be found in Appendix A. □

### 4. Solution procedure

In this section, we present an algorithm combining a filter-and-fan search scheme with a tabu search mechanism (F&F) to solve the FAPAQD. It comprises two main components, namely, a local search to identify a locally optimal solution and an F&F tree search to explore larger neighborhoods. Our approach relies on a transformation of the FAPAQD into the min-cost network flow problem in order to produce optimal solutions for a given set of selected discount intervals.

We first describe the problem transformation. Next, we explain how the local search is incorporated into an F&F tree search. We then give the details of how the mechanism of tabu search is used to help our algorithm escape from local optima. This is followed by a description of two measures that we used to speed-up our algorithm: a hashtable to avoid the unnecessary re-evaluation of previously evaluated solutions; and a rule that rapidly identifies unpromising solutions. Finally, we explain how these components are incorporated into the main procedure of our algorithm.

### Table 1  Example of discount intervals.

<table>
<thead>
<tr>
<th>Carrier 1</th>
<th>Carrier 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interval</td>
<td>Discount</td>
</tr>
<tr>
<td>$\geq 0$</td>
<td>$&lt; 2400$</td>
</tr>
<tr>
<td>$\geq 2400$</td>
<td>$&lt; 3000$</td>
</tr>
<tr>
<td>$\geq 3000$</td>
<td>$&lt; 3800$</td>
</tr>
<tr>
<td>$\geq 3800$</td>
<td>$&lt; 5000$</td>
</tr>
<tr>
<td>$\geq 5000$</td>
<td>7%</td>
</tr>
</tbody>
</table>

$\beta_{13} = 3800$, $\beta_{14} = 5000$, $\alpha_{11} = 0.0$, $\alpha_{12} = 0.025$, $\alpha_{13} = 0.05$, and $\alpha_{14} = 0.07$. It is assumed that $b_i = \beta_{11} < \beta_{12} < \ldots < \beta_{1k}$ for all $i$, and $z_{ik}$ is a non-decreasing function of $\beta_{1k}$.

In order to increase the reliability of service, the shipper requires that each lane must be assigned to a minimum number of carriers based on the characteristics of the lane such as traveling distance, annual quantity of freight and safety level. By imposing a minimum number of carriers on each lane, the shipper protects against the unavailability of carriers due to insufficient cargo hold space on the day or other unforeseen circumstances. However, assigning a minimum number of carriers for each lane is insufficient. For example, if we impose that the number of carriers for a certain lane must be at least 4, then the resulting allocation pattern may be 97%, 1%, 1%, 1%, which is insufficient cargo hold space on the day or other unforeseen circumstances.

After conferring with the shipper, we decided to model this requirement as a maximum percentage allocation $q_i$ of the freight quantity for each carrier on lane $j$, i.e., the amount of freight quantity assigned to each carrier servicing lane $j$ can be at most $q_i d_j$. Similar applications of such quantity limit measures can be found in Sadrian and Yoon [6] and Goossens et al. [7].

For a lane $j$, the number of selected carriers that can operate on it can be expressed as $c_j = (i, j) \in N$ and carrier $i$ is selected]. For a given set of selected carriers, the number of carriers that can operate on a lane may be insufficient to cater for the demand, i.e., it is possible that there exists some lane $j$ such that $q_i d_j < 1$. When this occurs in practice, the shipper would likely have to purchase freight space from carriers in the open or spot market, who usually charge a much higher rate than regular carriers under long-term contract (e.g., double the price or more). We modeled this aspect by the value $r_j$, which is the unit penalty cost for unfulfilled demand for lane $j$. It can be used to reflect the added cost of purchasing freight space from the spot market.

The objective of the FAPAQD is to assign a freight quantity to each carrier-lane pair (that is in $N$) or spot market such that the demand for each lane is fulfilled at minimum cost, subject to various constraints; our MIP formulation for the problem is as follows. The decision variable $x_{ijk}$ is the amount of freight space purchased from carrier $i$ on lane $j$ subject to discount interval $k$. The value of $x_{ijk}$ controls two other values: the amount of unfulfilled demand $u_j$ on lane $j$; and the binary variable $z_{ik}$ that equals 1 if discount interval $k$ for carrier $i$ is activated and equals 0 otherwise. The value $M$ is a sufficiently large positive number.

**MIP:** Minimize $\omega \sum_{i \in I} \sum_{j \in J} \sum_{k \in K} (1-x_{ijk}) p_i x_{ijk} + \gamma \sum_{j \in J} r_j u_j$  
subject to $\sum_{i \in I} \sum_{j \in J} x_{ijk} + u_j \geq d_j v_j \forall j \in J$  
$\sum_{j \in J} x_{ijk} + (1-x_{ijk}) M \geq \beta_k \forall i \in I, k \in K_i$  
$\sum_{j \in J} x_{ijk} \leq M z_{ik} \forall i \in I, k \in K_i$
4.1. Problem transformation

If all binary variables $z_{ik}$ are determined in the MIP model, then the resultant model becomes a linear programming problem with only continuous variables. We define a vector set $Z$ as

$$Z = \{ (z_{11}, \ldots, z_{1k}, \ldots, z_{n1}, \ldots, z_{nk}) : \sum_{k \in K_i} z_{ik} \leq 1 \text{ and } z_{ik} \in (0,1), \forall i \in I \}$$

Hence, the set $Z$ contains all vectors that describe a possible selection of discount intervals from the $n$ carriers. For a particular $z \in Z$, let $I_1$ be the set of selected carriers where $\sum_{k \in K_i} z_{ik} = 1$, and let $k(i)$ be the discount interval offered by carrier $i$, where $k(i) = k$ if $z_{ik} = 1$ and $k(i) = 0$ if $i \notin I_1$. Given an element $z \in Z$, we can derive a model as follows:

$$\text{MIP}(z) : \text{Minimize } \omega(z) = \sum_{i \in I_1} \sum_{j \in J} \left( 1 - z_{ik(i)} \right) \beta_{ik(i)} X_{ijk(i)} + \sum_{j \in J} r_j u_j$$

s.t. $\sum_{i \in I_1} x_{ijk(i)} + u_j \geq d_j, \forall j \in J$ (8)

$$\sum_{j \in J} x_{ijk(i)} \geq \beta_{ik(i)}, \forall i \in I_1$$ (9)

$$x_{ijk(i)} = 0, \forall (i,j) \notin N, i \in I_1, j \in J$$ (10)

$$x_{ijk(i)} = q_d, \forall j, i \in I_1$$ (11)

$$u_j = 0, \forall j \in J$$ (12)

$$x_{ijk(i)} \geq 0, \forall i \in I_1, j \in J$$

MIP($z$) is a min-cost network flow problem, which can be solved in polynomial-time using the network simplex algorithm; we use CPLEX 11.0 for this purpose. For a given $z \in Z$, the solution of MIP($z$) represents an optimal allocation of freight capacity to the discount intervals corresponding to $z$, which is a solution to the original FAPAQD. However, note that for the given $z$, it is possible that $\sum_{j \in J} q_j d_j < \beta_{ik(i)}$ for some $i \in I_1$, which implies that the discount interval combination $z$ leads to an infeasible solution of the original FAPAQD. This possibility is explicitly handled in our final algorithm.

Given the current solution $z \in Z$, we define the neighborhood $N(z)$ of $z$ as: $N(z) = \{ z' : z' \in Z \text{ and there exists exactly one carrier } i \text{ such that } k(i) \neq k'(i) \text{ where } k(i) \text{ and } k'(i) \text{ are the discount intervals selected for carrier } i \text{ in solution } z \text{ and } z' \text{, respectively} \}$. Therefore, the neighborhood of $z$ consists of solutions where a different discount interval is selected for a single carrier (including interval 0, which denotes that the carrier is not selected). Changing from a current solution $z$ to any member of $N(z)$ is defined as a move. If $z$ leads to an infeasible FAPAQD solution, we set $\omega(z) = +\infty$. For any two vectors $z_1, z_2 \in Z$, if $\omega(z_1) \leq \omega(z_2)$ and $z_1$ is feasible, then $z_1$ is considered better than $z_2$.

4.2. Filter-and-fan procedure

The F&F procedure is a meta-heuristic that alternates between a local search to achieve a locally optimal solution and an F&F search to escape from the local optimum so as to explore other neighborhoods. This search scheme has been successfully implemented for the traveling salesman problem [27], the uncapacitated facility location problem [28], and the job shop scheduling problem [29], etc.

Our F&F procedure begins with an initial feasible solution $z'$ to the original FAPAQD. The simplest such solution can be obtained by setting all $z_{ik}$ to 0, which represents the case where the shipper uses only spot market carriers. We then use a standard 1-step look ahead greedy algorithm (i.e., choose the best neighbor from

![Diagram](attachment:image.png)

---

Fig. 1. Example of filter-and-fan procedure ($\eta_1 = 3, \eta_2 = 2, L = 4$).
Here is a direct transcription of the text:

Once $z^*$ is identified, the F&F procedure generates a series of neighborhood trees where each branch represents a move and each node corresponds to a solution in $Z$. There are three user-defined parameters for this procedure: $L$, $\eta_1$, and $\eta_2$. The maximum number of levels in each neighborhood tree is $L$, and at each level there are at most $\eta_1$ selected nodes. In each neighborhood tree, the node corresponding to the locally optimal solution is regarded as level 0, and its $\eta_1$ best feasible neighbors comprise level 1. For each node in level $l \geq 1$, the best $\eta_2$ feasible neighbors are found, thereby generating $\eta_1 \cdot \eta_2$ candidate nodes for level $l+1$. Then, the best $\eta_1$ nodes are selected from the $\eta_1 \cdot \eta_2$ candidates and constitute level $l+1$. The generation of $\eta_1 \cdot \eta_2$ candidate nodes is called the fan candidate list strategy, and the selection of the best $\eta_1$ nodes is called the filter candidate list strategy [28]. This level propagation procedure is repeated until either an improved solution $z'$ that is better than the previously best-known solution is found, or the depth of the neighborhood tree reaches $L$. In the first situation, the greedy algorithm is again triggered to seek another local optimum based on $z'$, and the tree generation process repeats. In the second situation, the procedure terminates. Fig. 1 below gives an example of the F&F procedure.

The F&F procedure is as follows:

**step 0.** Initialize $\eta_1 \cdot \eta_2 \cdot L$ and generate an initial feasible solution $z$;

**step 1.** Perform a 1-step look ahead greedy algorithm on $z$ to find a locally optimal solution $z^*$;

**step 2.** Identify the best $\eta_1$ feasible neighbors of $z^*$ to construct level 1 of the F&F tree, and set $l=1$;

**step 3.** Identify the best $\eta_2$ feasible neighbors for each node in level $l$ and select the best $\eta_1$ moves from the $\eta_1 \cdot \eta_2$ candidates to construct level $l+1$;

**step 4.** If an improved solution $z'$ is found, set $z'=z'$ and go to step 1; otherwise, continue to step 5.

**step 5.** If $l+1=L$, terminate; otherwise, set $l=l+1$ and go to step 3.

### 4.3. Tabu conditions

To escape from local optima during the F&F search procedure, we incorporate tabu search mechanism [30] on the selection of the $\eta_2$ neighbors associated with each node. Recall that a move changes the discount interval of exactly one carrier. We therefore define the tabu condition as: for the current move, if the discount interval of carrier $i$ is changed from $k \in K \cup \{0\}$ to any other value $k' \in K \cup \{0\}$, then the discount interval $k$ cannot be selected for carrier $i$ for the next $\xi$ moves, where $\xi$ is the tabu tenure. Each node has several tabu conditions inherited from its parent node and one tabu condition upon creation. We also use an aspiration criterion, i.e., the tabu conditions are overridden if the move yields an improved solution; this type of aspiration criterion is commonly used in previous tabu search implementations.

Fig. 2 provides an example of the propagation of tabu conditions, where the node chain corresponds to a path in Fig. 1. For instance, node 4 inherits two tabu conditions “cannot select discount interval 0 of carrier 2 for the next 1 move” and “cannot select discount interval 3 of carrier 5 for the next 2 moves” from node 3. When node 4 is created, a new tabu condition “cannot select discount interval 3 of carrier 0 for the next 3 moves” is generated. Each node can lead to neighbors that either satisfy all tabu conditions or that correspond to improved solutions (or both). Note that each path in Fig. 1 can be viewed as a separate tabu search path; hence, the F&F tree search can be

---

**Fig. 2. Tabu conditions on one tree path.**
viewed as a modified tabu search algorithm with multiple search paths.

4.4. Algorithm speed-up

The computational effort required to identify the best \( \eta_2 \) feasible neighbors is a critical factor that significantly affect the running time of the algorithm, since it requires the solving of the MIP(\( z' \)) model for each discount interval for each carrier. In this subsection, we describe two measures that we used to speed-up the running time of this procedure: a memorization technique using a hashtable, and a pruning rule to avoid evaluating solutions that are likely to be poor.

4.4.1. Function memorization

As shown in Fig. 2, the value of \( z \in Z \) can be coded as a numeric string. For example, the string 320013 encodes the solution where \( k(0)=3, k(1)=2, k(2)=0, k(3)=0, k(4)=1 \) and \( k(5)=3 \). We memorize our function for evaluating the neighboring solutions of \( z \) into a hashtable with the numeric string encoding of \( z \) as the key; if the solution is infeasible, then \( +\infty \) is recorded. Prior to solving an MIP(\( z' \)) model, the algorithm first checks if \( z \) exists in the hashtable; if so, the value of \( \alpha(z) \) is directly retrieved from the hashtable.

We implemented our hashtable using the C++ STL hash map template class, which allows the insertion and retrieval of the \( \alpha(z) \) values in amortized constant time.

4.4.2. Pruning rule

Observe that constraints (2) allow the shipper to purchase more freight capacity than is required. There are some cases where purchasing more than the required amount may result in a lower overall cost since the carriers offer a higher per-unit discount when a larger amount of freight capacity is purchased \([6,23]\). However, if the amount of capacity exceeds the demand by a large amount, then the cost of unused capacity will most likely more than offset any savings achieved by the higher discount. Therefore, if a particular solution \( z \) requires that the shipper purchases too much excess capacity, then it is likely to be an inferior solution.

Notice that for a given \( z \in Z \), in order to make MIP(\( z \)) feasible, constraints (10) must be satisfied and accordingly \( \sum_{i \in I} \sum_{j \in A} \beta_{ijk} \leq \sum_{i \in I} \delta_{ij} d_j \). Assuming the set of selected carriers is able to cover the demand (i.e., \( \sum_{i \in I} \beta_{ik0} \geq \sum_{j \in J} d_j \)), then the amount of excess freight capacity bought from carriers is at least as large as \( \sum_{i \in I} \sum_{j \in A} \beta_{ijk} - \sum_{j \in J} d_j \). Our pruning rule can be stated as follows: for a move resulting in solution \( z \), if \( z \) satisfies \( \sum_{i \in I} \sum_{j \in A} \beta_{ijk} \geq (1+\delta) \sum_{j \in J} d_j \), where \( \delta \) is a controlling parameter, then \( z \) is considered unpromising, and we do not solve MIP(\( z' \)).

4.5. Candidate generation procedure

Our final procedure that identifies the best \( \eta_2 \) feasible neighbors for a solution \( z \) is given in Algorithm 1. The moves associated with \( z \) can be divided into three categories: (1) the set of promising moves where their associated instances of MIP(\( z' \)) are solved and have feasible solutions; (2) the set of moves that were considered unpromising by the pruning rule; and (3) the set of moves where their MIP(\( z' \)) is infeasible. In our implementation, we maintain the moves in categories (1), (2) and (3) in lists \( L_1 \), \( L_2 \) and \( L_3 \), respectively.

Algorithm 1 Procedure for identifying the best \( \eta_2 \) feasible neighbours

1: Inputs. Solution \( z \), hashtable \( H \) and value of the current best solution \( \omega^* \).
2: Set of selected moves \( M_z \leftarrow \emptyset \).
3: Lists \( L_1, L_2, L_3, L_{move} \leftarrow \emptyset \).
4: for each element \( z' \in N(z) \) do
5:   if \( z' \in H \) then
6:     Retrieve \( \omega(z') \) from \( H \).
7:     if \( \omega(z') \neq +\infty \) then
8:       Put \( z' \) into \( L_1 \).
9:     else
10:       Put \( z' \) into \( L_2 \).
11:   end if
12:   else
13:     if \( z' \) is unpromising then
14:       Put \( z' \) into \( L_2 \).
15:     else
16:       Solve MIP(\( z' \)) and put \((z', \omega(z'))\) into \( H \).
17:     if \( \omega(z') \neq +\infty \) then
18:       Put \( z' \) into \( L_1 \).
19:     else
20:       Put \( z' \) into \( L_3 \).
21:     end if
22:   end if
23: end if
24: end for
25: Sort elements in \( L_1 \) and \( L_2 \).
26: Concatenate \( L_{move} = L_1 + L_2 + L_3 \).
27: for each element \( z' \in L_{move} \) from 1 to \(|L_{move}|\) do
28:   if \( z' \) is not tabu then
29:     Put \( z' \) into \( M_z \).
30:   else if \( z' \in L_2 \) and \( \omega(z') < \omega^* \) then
31:     Put \( z' \) into \( M_z \).
32:   end if
33: end if
34: if \(|M_z| = \eta_2 \) then
35:   return \( M_z \).
36: end if
37: end for
38: return \( M_z \).

Before solving the MIP(\( z' \)) model, we first check if its value has already been computed and stored in the hashtable (line 5). If so, and the stored value does not indicate that it is infeasible, then we put \( z' \) in \( L_1 \). If not, then we use our pruning rule to check if \( z' \) is unpromising (line 13) and place it in \( L_2 \) if this is the case.

Otherwise, \( z' \) is a promising new solution and MIP(\( z' \)) is solved, which allows us to either place \( z' \) into \( L_1 \) if it is feasible, or into \( L_2 \) if it is infeasible.

We sort the elements of \( L_1 \) in ascending order of their \( \omega(z') \) values, and sort the elements in \( L_2 \) in ascending order according to their values of \( \sum_{i \in I} \beta_{ik0} - \sum_{j \in J} d_j \); the list \( L_3 \) remains unsorted (line 25). The three lists \( L_1, L_2 \) and \( L_3 \) are then concatenated together in that order to form \( L_{move} \) from which the best \( \eta_2 \) candidate moves are selected. We consider each move in \( L_{move} \) in turn, which we retain if it is not tabu or if it improves on the current best solution (aspiration). This continues until we have found \( \eta_2 \) candidate solutions. Note that it is possible that all \( \eta_2 \) candidate solutions for a current level \( l \) are infeasible; we allow the algorithm to continue despite this happenstance as long as \( l < L \) because a superior feasible solution might exist in the neighborhood of these infeasible solutions and be discovered in later levels.

5. Computational experiments

We tested our algorithm on a large number of instances that were randomly generated based on typical values supplied by the
shipper. As a comparison, we also applied the branch & cut search scheme provided by CPLEX 11.0 to solve the MIP model. However, it was necessary to set a time limit for each run of CPLEX due to practical considerations; the time limit varied according to the size of the test instance. Furthermore, preliminary experiments showed that CPLEX requires a large amount of memory to store the search tree information when solving these instances, often resulting in premature termination due to “out-of-memory” exceptions. Therefore, we stored the branch & cut tree generated by CPLEX in the hard disk when performing our experiments. We implemented both algorithms in C++ and ran all experiments on an Intel Xeon(R) 2.66 GHz server with 3 GB RAM. Computational times reported here are in CPU seconds on this server.

5.1. Test instance generation

Our test instances were generated according to the following scheme.

**Instance size** \((n, m)\): we divide our instances into 3 groups based on the number of carriers \(n\) and the number of lanes \(m\). Small instances have values (20, 50), (20, 100) and (40, 50). Medium instances have values (20, 300), (20, 500), (20, 1000), (40, 300) and (40, 500). Large instances have values (40, 1000), (80, 300), (80, 500) and (80, 1000).

**No service**: the percentage of lanes to which carrier \(i\) does not provide service is chosen from \([0.2, 0.5]\), where \(U[a, b]\) denotes the uniform distribution in the interval \([a, b]\). Given this percentage value, the lanes to which the carrier does not provide service are uniformly randomly selected.

**Standard price**: for each lane \(j\), we first randomly choose a mean rate \(p_{ij}\) from \([20, 100]\), and a deviation factor \(s\) from \([0.05, 0.30]\). Then, the standard price \(p_i\) quoted by carrier \(i\) for lane \(j\) is selected from \(U[p_{ij}(1-t), p_{ij}(1+t)]\).

**Forecasted demand**: the demand of each lane is selected from \([10, 100]\).

**Quantity limit**: the value of \(q_i\) is randomly selected from the set \([0.3, 0.4, 0.6]\) with equal probabilities. These approximately correspond to having a minimum of 4, 3 and 2 carriers for that lane, respectively.

**Discount intervals**: the number of discount intervals is fixed at 5 for all carriers. Given the total demand \(D = \sum_{j=1}^{m} d_j\), the discount interval lower bounds \(b_1, ..., b_5\) were generated from \([D/13.5, D/12.5], [D/11.5, D/10.5], [D/9.5, D/8.5], [D/7.5, D/6.5] and [D/5.5, D/4.5]\), respectively.

**Discount values**: the discount increment for each interval \(k \geq 1\) has a 40%, 40% and 20% probability to be 1.5%, 2% and 2.5%, respectively. For example, if the discount increments for carrier \(i\) were selected to be 1.5%, 1.5%, 2% and 2.5%, then the discount coefficients associated with the five intervals would be 1.5%, 3.0%, 5.0%, 7.0% and 8.5%.

**Penalty for unfulfilled demand**: the unit cost for unfulfilled demand \(f = 2 max_{x \in S} p_x\), which is twice the quoted standard price for the most expensive carrier on that lane.

Unfortunately, due to privacy issues, we were unable to acquire actual data for testing purposes. The various parameters in this scheme were chosen after discussion with the shipper, and reflect typical values encountered in actual data.

For each instance size, we generated 20 random instances, for a total of 240 instances (60 small, 100 medium and 80 large). All instances and experimental results can be found in the online repository at [http://www.tigerqin.com/fapaqd](http://www.tigerqin.com/fapaqd).

5.2. Parameter tuning

Our algorithm includes a number of controlling parameters. We used the same values of \(\eta_1 = 2\eta_2\) as previous work on the filter-and-fan algorithm [28,29]. A full \(2^4\) experimental design process was conducted for the values \(\eta_1, L\) and \(\zeta\). After some preliminary experiments, the following low and high levels were selected for each parameter: 40 and 60 for \(\eta_1\), 50 and 100 for \(L\), 25 and 50 for \(\zeta\), and 1.5% and 2.5% for \(\delta\). Each of the 16 parameter combinations was tested on a set of 60 instances, which includes 5 instances for each of the 12 instance sizes. For each instance, we divided each of the 16 objective values by the smallest value to generate a ratio. For each parameter setting, the average ratio of all 60 instances was used to measure its impact. Finally, we chose the parameter setting with the smallest average ratio, and if there exists multiple parameter settings with the same average ratio, we chose the one resulting in less computation time. After this tuning process, the values of the parameters were set as \(\eta_1 = 40\), \(\eta_2 = 20\), \(L = 100\), \(\zeta = 25\) and \(\delta = 1.5\%\).

5.3. Results and analysis

We compared the performance of our F&F with CPLEX in terms of the gap between the total costs of their best solutions

$$\text{Gap} = \frac{\text{CPLEX} - \text{F&F}}{\text{CPLEX}} \times 100\%$$

The computational results for the small instances are shown in Table 2. When solving these instances, we imposed a time limit of 1 h on CPLEX; for all instances, CPLEX was unable to identify verifiably optimal solutions within this time limit. For each instance size, the column “Average F&F Time” gives the average amount of time used by our F&F algorithm and the column “Average Gap (%)” presents the average gap. The average gap values for all entries in this table are negative, which indicates that CPLEX obtained higher quality solutions on average for the small instances. The column “Number of Negative Gaps” gives the number of instances (out of 20) where CPLEX achieved better solutions than F&F.

Although CPLEX was able to produce superior solutions, F&F consumed significantly less computation time than the 1 h used by CPLEX. Moreover, the gap between the solutions found by F&F and CPLEX is not less than −0.49% and the overall average gap is only −0.07%. Consequently, for applications where a quick, high quality solution is required, our F&F approach may be preferred even for small instances.

The results for the medium and large instances are summarized in Table 3. We first ran F&F on these instances, and then set the time limit for CPLEX to be up to 10 times the running time of F&F for each instance. The columns Gap\((x = \{1, 2, 4, 6, 8, 10\})\) give the average gap between the best solutions obtained by CPLEX at \(x\) times the running time of F&F and the solutions obtained by F&F. From this table, we find that on average F&F achieved higher quality solutions than CPLEX even when CPLEX is given 10 times the running time of F&F. CPLEX achieved better solutions within 10 times the running time of F&F for only 10 out of 180 instances, most of which are of medium size. Note that the amount of improvement in the solutions generated by CPLEX increases very slowly as computation time increases. This suggests that there are severe diminishing returns in terms of computation time to using

**Table 2**

<table>
<thead>
<tr>
<th>Size</th>
<th>Average F&amp;F time</th>
<th>Average gap (%)</th>
<th>Number of negative gaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20, 50)</td>
<td>82.0</td>
<td>−0.06</td>
<td>9</td>
</tr>
<tr>
<td>(20, 100)</td>
<td>128.4</td>
<td>−0.08</td>
<td>13</td>
</tr>
<tr>
<td>(40, 50)</td>
<td>130.7</td>
<td>−0.07</td>
<td>12</td>
</tr>
</tbody>
</table>
The heuristic approach was necessary because the running time of F&F for CPLEX to find solutions comparable to the solutions found by F&F. These results suggest that it would take much more than 10 times the running time of F&F for CPLEX to solve this problem; in particular, extrapolating the computational results of the medium and large instances.

### Table 3

<table>
<thead>
<tr>
<th>Size</th>
<th>Average F&amp;F time</th>
<th>Average Gap1 (%)</th>
<th>Gap2 (%)</th>
<th>Gap4 (%)</th>
<th>Gap6 (%)</th>
<th>Gap8 (%)</th>
<th>Gap10 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(20, 500)</td>
<td>293.9</td>
<td>0.51</td>
<td>0.47</td>
<td>0.42</td>
<td>0.40</td>
<td>0.36</td>
<td>0.22</td>
</tr>
<tr>
<td>(20, 500)</td>
<td>516.8</td>
<td>0.48</td>
<td>0.40</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.23</td>
</tr>
<tr>
<td>(20, 1000)</td>
<td>1229.4</td>
<td>0.69</td>
<td>0.49</td>
<td>0.45</td>
<td>0.34</td>
<td>0.31</td>
<td>0.21</td>
</tr>
<tr>
<td>(40, 300)</td>
<td>577.0</td>
<td>1.00</td>
<td>0.66</td>
<td>0.44</td>
<td>0.32</td>
<td>0.27</td>
<td>0.16</td>
</tr>
<tr>
<td>(40, 500)</td>
<td>889.3</td>
<td>0.85</td>
<td>0.70</td>
<td>0.43</td>
<td>0.31</td>
<td>0.28</td>
<td>0.18</td>
</tr>
<tr>
<td>Average</td>
<td>–</td>
<td>0.71</td>
<td>0.55</td>
<td>0.42</td>
<td>0.35</td>
<td>0.31</td>
<td>0.20</td>
</tr>
<tr>
<td>(40, 1000)</td>
<td>2228.9</td>
<td>1.28</td>
<td>0.90</td>
<td>0.71</td>
<td>0.62</td>
<td>0.55</td>
<td>0.44</td>
</tr>
<tr>
<td>(80, 500)</td>
<td>1272.6</td>
<td>1.37</td>
<td>1.34</td>
<td>1.04</td>
<td>0.92</td>
<td>0.77</td>
<td>0.48</td>
</tr>
<tr>
<td>(80, 500)</td>
<td>2587.8</td>
<td>1.86</td>
<td>1.82</td>
<td>1.62</td>
<td>1.38</td>
<td>1.11</td>
<td>0.71</td>
</tr>
<tr>
<td>(80, 1000)</td>
<td>5636.3</td>
<td>1.81</td>
<td>1.78</td>
<td>1.74</td>
<td>1.73</td>
<td>1.68</td>
<td>1.17</td>
</tr>
<tr>
<td>Average</td>
<td>–</td>
<td>1.58</td>
<td>1.46</td>
<td>1.27</td>
<td>1.16</td>
<td>1.03</td>
<td>0.70</td>
</tr>
</tbody>
</table>

...and all lanes are sufficiently large. We can derive a reduced problem RP1 that is a special case of the FAPAQD as follows:

\[
\sum_{i,j} x_{ij} + (1-z_{i1})M \geq b_i, \forall i \in I
\]

(A.3)

Since no upper bound is imposed on variables \( x_{ijk} \), as long as there exists a carrier \( i \) that operates on lane \( j \), then we can always fulfill the demand \( d_j \) by increasing the value of \( x_{ijk} \) as necessary. So \( u_j \) must be equal to zero in optimal solutions of RP1, and it can be removed without affecting the optimal solutions. Replacing \( x_{ijk}, z_i, b_i, \) and \( b_i \) with \( x_{ij}, y_i, f_i, \) and \( b_i \), respectively, RP1 can be transformed into the following equivalent model:

\[
\text{RP2: Minimize } \sum_{j \in J} \sum_{i \in I} p_{ij} x_{ij} + \sum_{i \in I} M y_i
\]

(A.5)

\[
s.t. \sum_{i \in I} x_{ij} \geq d_j, \forall j \in J
\]

(A.6)

\[
by_j \sum_{i \in I} x_{ij} \leq M y_i, \forall i \in I
\]

(A.7)

where variable \( x_{ij} \) represents the quantity purchased from carrier \( i \) on lane \( j \); and binary variable \( y_i = 1 \) if carrier \( i \) is used and \( y_i = 0 \) otherwise. The Transportation Problem with MQC was shown to be \( \mathcal{NP} \)-hard in the strong sense in Appendix A of Lim et al. [1] by reduction from the Cover by 3-Sets (X3C) problem; the Transportation Problem with MQC is identical to RP2 except that constraints (A.6) are strictly equal (rather than greater than or equal). However, this does not affect the proof, and therefore RP2 can be shown to be \( \mathcal{NP} \)-hard in the strong sense in the same manner. □

### References


