A hybrid algorithm for large-scale service network design considering a heterogeneous fleet

Abstract

Service network design addresses decisions related to transportation services selection and origin-to-destination commodity flow distribution. In this paper, we consider the usage of a heterogeneous fleet to provide services for huge transportation network. We propose both arc-based and cycle-path models to formulate the problem. A hybrid algorithm is presented to solve large-scale instances. The method includes pricing and cutting techniques to achieve stronger lower bounds, as well as a local search algorithm to obtain high-quality solutions. Computational study indicates the effectiveness and efficiency of the proposed algorithm when compared with the state-of-the-art solver CPLEX.

Keywords: service network design; heterogeneous fleet; column generation; cutting plane; local search

1 Introduction

Freight transportation plays an increasingly important role because of rapid development of e-commerce. Carriers are facing tremendous pressure from growing competition and start to pay more attention to optimize their transportation network. Service network design (SND) formulations address issues related to the tactical planning for consolidation-based freight transportation systems. Specifically, such formulations make decisions on selecting transportation services and distributing origin-destination (O-D) commodity flow. Network links represent services with operational assets (e.g. vehicles, crews and power units) in SND formulations. Asset management concentrates on the operation and schedule of assets which require balance at each terminal in order to be available for services.

Most researchers assume that transportation system is serviced by homogenous assets which have the same property. Nevertheless, it is a common practice to employ a heterogeneous fleet in transportation and logistics management for carriers. In the literature, only a small amount of efforts are devoted to considering a heterogeneous fleet, e.g. Kim et al. (1999) and Li et al. (2017). Moveover, with the improvement of logistics infrastructure, carriers are confronted with increasingly huge transportation network. Since service network design problem is NP-hard, it is difficult to find a feasible solution for large-scale instances. Therefore, there is an urgent need for efficient algorithms to reduce costs for carriers in real-life application.

Our goal of this paper is to address this issue related to heterogeneous assets and take vehicles as an example to formulate service network design problem. Furthermore, efficient algorithms are introduced to help carriers reduce logistics-related costs when dealing with huge transportation network. Specifically, this paper makes the following contributions to the literature: First, we present both arc-based and cycle-path formulations for service network design considering a heterogeneous fleet. Second, a hybrid algorithm including column generation, cutting plane and local search methods is introduced to solve large-scale instances. Moreover, the proposed algorithm provides high-quality both lower bounds and feasible solutions efficiently, and outperforms CPLEX running 10 hours of CPU time.

The outline of this paper is as follows. In Section 2, a brief literature review is elaborated. We describe the problem and propose both arc-based and cycle-path formulations in Section 3. Section 4 is devoted to the proposed hybrid algorithm, whereas experimental results is presented in Section 5. Concluding remarks is given in Section 6.

2 Literature review

Service network design contains the tactical planning decisions on selecting transportation services to satisfy O-D demands which is generally formulated as fixed-cost capacitated multicommodity network design problem (CMND), see Magnanti and Wong (1984) and Minoux (1989). In addition to scheduling services and distributing flow, asset management is marginally considered in Crainic (2000), Smilowitz et al. (2003) and Crainic (2003). Since carriers attach increasing importance to utilize assets economically and efficiently, SND with asset management attracts increasing attention in the literature (Andersen et al. 2009b, Teypaz et al. 2010). Afterwards design-balanced constraints are proposed by Pedersen et al. (2009), which state that the number of assets entering each terminal must equal the number of leaving. The design-balanced constraints induce a cyclic structure for the assets movements to complete service operations and flow distribution.

With respect to the way of formulating SND, it is general to propose arc-based formulations in the literature. While Andersen et al. (2009b) present four alternative formulations and analyze strengths and weaknesses of each formulation. The experimental results show that formulations based on cycle variables outperform other formulations in both solution quality and time. However, the number of generated cycles grows exponentially and requires more efficient enumeration algorithm. Consequently, Andersen et al. (2011) propose an efficient branch-and-price scheme for the cycle-based formulation to avoid the enumeration of cycles. Their algorithm contains two subproblems for the dynamical construction of paths for commodities and cycles for vehicles.

Since SND problem is NP-hard, heuristic methods are preferred choices. Pedersen et al. (2009)

present a two-phase tabu search meta-heuristic framework for the arc-based formulation. The proposed algorithm includes exploration phase to build neighborhoods and feasibility phase to implement an infeasibility-monitoring procedure for reaching good feasible solutions. Teypaz et al. (2010) present a three-step decomposition heuristic to solve large-scale freight transportation formulation and build comprehensive solutions for real-life size instances. Vu et al. (2013) propose a three-phase meta-heuristic method combining neighbourhood-based and exact methods. Additionally, computational results on large-scale benchmark instances indicate the efficiency of the algorithm. Another meta-heuristic combining a cutting-plane scheme to obtain lower bound and a variable fixing procedure to cut down the dimension of the problem is introduced in Chouman and Crainic (2014). Crainic et al. (2016) propose a new service network design model with resource constraints and design a solution approach combining column generation and slope scaling methods to find high-quality solutions.

Only a single asset type is considered in the aforementioned literature. As for heterogeneous assets, especially multiple types of vehicles, Kim et al. (1999) introduce three different but equivalent formulations and their potentially significant difference in computational time. They also present two sets of valid inequalities to strengthen lower bounds. However, no efficient algorithm is mentioned to solve those formulations in that paper. Li et al. (2017) propose a tabu search based meta-heuristic to solve arc-based formulation for design-balanced capacitated multicommodity network design with heterogeneous assets. However, the numbers of each fleet type employed by each service are not decided explicitly in their formulation. Moreover the scale of those experimental instances solved by the proposed algorithm cannot satisfy the real-life demand. The biggest instance in that paper only considers 25 nodes, 50 commodities and 215 arcs.

3 Problem statement and mathematical formulation

In this section, we describe the problem and propose two equivalent formulations based on arc and cycle-path respectively.

3.1 Problem description

Since most literature assume homogeneous fleet is employed which may not accord with the common practices, we take heterogeneous fleet issue into consideration in this paper. The example network with a heterogeneous fleet is shown in Fig. 1. Under this assumption, the number of different type of vehicles should be decided on each arc. In Fig. 1, there are two types of fleet providing services and arcs in solid line are labeled by the number of specific type of vehicles. At each node the numbers of same type of vehicles entering and leaving equals. A cycle consists of a sequence of arcs satisfying design-balance requirements of the same type of vehicles, e.g., $\{(A, D), (D, C), (C, B), (B, A)\}$. A path starts at the origin of a commodity and ends at its destination. Suppose there is a commodity with origin A and destination G, thus $\{(A, D), (D, G)\}$ and $\{(A, D), (D, F), (F, G)\}$ are two paths. Note that arc (A, D) employs both two types of vehicles to provide transportation services. Dashed-line arcs like (E, A), (C, F) and (G, E) mean they are not selected as a service to open.

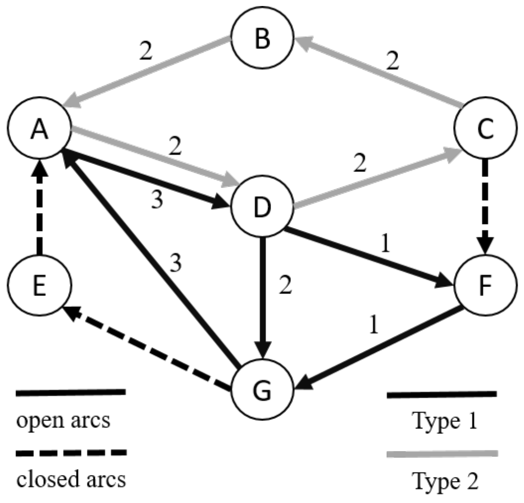

Figure 1: Example of heterogeneous fleet

Given a directed graph $G = (N, A)$ with node set N and arc set A. Each commodity k in commodity set K has a known demand d^k to be transited from its origin node $o(k)$ to destination node $d(k)$. Let F denote the set of fleet types and each type $f \in F$ has a certain capacity u^f. There is no limit to the number of each types of vehicles.

3.2 Arc-based formulation

Let h_{ij}^f denote the fixed cost of utilizing fleet type $f \in F$ with respect to arc (i, j). The unit flow cost for transporting commodity $k \in K$ on arc (i, j) is denoted c_{ij}^k. We define $N_i^+ = \{j \in N | (i, j) \in A\}, N_i^- = \{j \in N | (j, i) \in A\}$. For each commodity k and node i, define

$$\omega_i^k = \begin{cases} d^k & if\ i = o(k) \\ -d^k & if\ i = d(k) \\ 0 & otherwise \end{cases}$$

Two sets of decision variables are defined as follows.

- Continuous variable: x_{ij}^k represents the amount of flow with respect to commodity $k \in K$ on arc $(i,j) \in A$;
- Integer variable: y_{ij}^f denotes the number of fleet type $f \in F$ providing transportation services on arc $(i,j) \in A$.

The model is formulated as follows:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k + \sum_{f \in F} \sum_{(i,j) \in A} h_{ij}^f y_{ij}^f \tag{1}$$

s.t.

$$\sum_{j \in N_i^+} x_{ij}^k - \sum_{j \in N_i^-} x_{ji}^k = w_i^k \quad \forall i \in N, \forall k \in K, \tag{2}$$

$$\sum_{j \in N_i^+} y_{ij}^f - \sum_{j \in N_i^-} y_{ji}^f = 0 \quad \forall i \in N, \forall f \in F, \tag{3}$$

$$\sum_{k \in K} x_{ij}^k \leq \sum_{f \in F} u^f y_{ij}^f \quad \forall (i,j) \in A, \tag{4}$$

$$x_{ij}^k \geq 0 \quad \forall (i,j) \in A, \forall k \in K, \tag{5}$$

$$y_{ij}^f \in Z_0^+ \quad \forall (i,j) \in A, \forall f \in F, \tag{6}$$

The objective function (1) minimizes the sum of the total commodity flow costs and the fixed costs for selecting and operating transportation service with respect to a heterogeneous fleet. Constraints (2) represent the flow conversation relations for each node and each commodity. Equations (3) are the design-balance constraints, which ensure the same number of each type of vehicles entering and leaving a node. Through constraints (4), we enforce the total flow cannot exceed the total capacity provided by all vehicles on each arc. Finally, variable-type restrictions appear in constraints (5) and (6).

3.3 Cycle-path formulation

Let Q^f represent the set of all design cycle with respect to fleet type $f \in F$. And we define the set of all paths with respect to commodity $k \in K$ as P^k. Two sets of decision variables are given as follows.

- Cycle integer variables: y_q^f denotes the number of vehicle type f used by cycle $q \in Q^f$;
- Path continuous variables: x_p^k denotes the volume of commodity k on path $p \in P^k$.

We define two sets of indicator variables:

- α_{ij}^q: equals 1 if arc (i,j) is included in cycle q, and 0 otherwise;

- δ_{ij}^p: equals 1 if arc (i,j) is included in path p, and 0 otherwise.

The fixed cost of cycle q with respect to fleet type f is denoted h_q^f, which equals $\sum_{(i,j)\in A} h_{ij}^f \alpha_{ij}^q$. Similarly, the flow cost of path p with respect to commodity k is denoted c_p^k, which equals $\sum_{(i,j)\in A} c_{ij}^k \delta_{ij}^p$.

$$\min \sum_{k\in K}\sum_{p\in P^k} c_p^k x_p^k + \sum_{f\in F}\sum_{q\in Q^f} h_q^f y_q^f \tag{7}$$

s.t.

$$\sum_{k\in K}\sum_{p\in P^k} \delta_{ij}^p x_p^k \le \sum_{f\in F}\sum_{q\in Q^f} u^f \alpha_{ij}^q y_q^f \quad \forall (i,j)\in A \tag{8}$$

$$\sum_{p\in P^k} x_p^k = d^k \quad \forall k\in K \tag{9}$$

$$x_p^k \ge 0 \quad \forall p\in P^k, \forall k\in K \tag{10}$$

$$y_q^f \in Z_0^+ \quad \forall q\in Q^f, \forall f\in F \tag{11}$$

The objective function (7) minimizes the total cost calculated as the sum of the flow costs on paths plus the fixed costs for selected cycles with respect to specific type of vehicle.Through constraints (8), we enforce service capacity restrictions on each arc. Constraints (9) ensure demand satisfaction corresponding to the flow conservation equations for each commodity. Finally, constraints (10) and (11) give variable-type restrictions.

3.4 Summary of two formulations

As for comparisons of the two ways to formulate service network design problem, Andersen et al. (2009b) have performed computational study to indicate their strengths and weaknesses. The formulation based on arc is intuitively related to the description of the problem and easier to understand. Since paths and cycles are composed of arcs, the arc-based formulation is the basis of cycle-path formulation. Furthermore, the structure of cycle-path formulation is suitable for column generation approach because each path and cycle may be added to the model as a column. In this paper, local search approach which will be introduced in Section 4 is based on the arc-based formulation.

4 Solution method

Since service network design problem is NP-hard, it is difficult for exact algorithms to solve the large-scale instances optimally. Heuristics and metaheuristics are the preferred choices in real-life application for a minimization problem. But the drawback of heuristics algorithm is that the solutions cannot be evaluated by lower bounds. Therefore we propose a hybrid algorithm combining exact and heuristic techniques to solve the problem. The exact part mainly contains column generation and cutting plane method to obtain tight lower bounds. Moreover, we employ the linear solution provided by column-and-cut generation approach as a initial neighborhood and use local search approach to find high-quality feasible solutions. The sketch of the proposed hybrid algorithm is shown in Algorithm 1.

Algorithm 1: Hybrid solution method

```
1  while find promising columns or cuts do
2      column generation;
3      cutting plane;
4  end
5  obtain lower bound;
6  use the linear solution as a initial neighborhood;
7  local search approach;
8  return lower bound and upper bound;
```

4.1 Column-and-cut generation (C&CG)

Column generation approach has been proved to be efficient to solve cycle-path formulation by Andersen et al. (2011). With respect to cutting-plane procedure, it is also proved to be effective in calculating tight lower bounds for CMND-related problems, see Chouman et al. (2009, 2016), Chouman and Crainic (2014). We combine the two methods to strengthen lower bounds for the proposed formulations.

4.1.1 Column generation (CG)

The linear relaxation of cycle-path formulation constitutes the master problem(MP) which is solved by CG approach. Since the number of variables in the MP is exponentially increasing, it is necessary to concentrate on a restricted master problem(RMP) which contains only a subset of variables(columns) in the MP. The CG approach starts with some necessary cycles and paths to prevent the constraints violation. We generate one path for each commodity to satisfy its O-D demand. A reversed path from the destination of each commodity to its origin is generated to constitute a cycle with the former path. Therefore, we add a path and a cycle for each commodity as the initial columns to ensure the RMP feasible. The method adds new paths and cycles dynamically to the RMP until no additional column with negative reduced cost can be found.

The dual variable values in the solution of the RMP are passed to the subproblem for generating new columns. Dual variables η_{ij} and σ_k are associated with constraints (8) and (9) respectively. The reduced costs associated with cycle-related variable y_q^f and path-related variable x_p^k are given by:

$$RC_{fq} = h_q^f + \sum_{(i,j)\in A} u^f \alpha_{ij}^q \eta_{ij} = \sum_{(i,j)\in q} (h_{ij}^f + u^f \eta_{ij}) \tag{12}$$

$$RP_{kp} = c_p^k - \sum_{(i,j)\in A} \delta_{ij}^p \eta_{ij} - \sigma_k = \sum_{(i,j)\in p} (c_{ij}^k - \eta_{ij}) - \sigma_k \tag{13}$$

Algorithm 2: Column generation procedure

```
add initial sets of cycles and paths to the RMP;
while find columns with negative reduced cost do
    solve the RMP;
    determine dual variables $\eta_{ij}$ and $\sigma_k$;
    foreach node do
        find the shortest cycle with respect to arc costs $h_{ij}^f + u^f \eta_{ij}$;
        let $\theta_q$ denote the cost of the shortest cycle;
        if $\theta_q < 0$ then
            add the corresponding cycle to the RMP;
        end
    end
    foreach commodity do
        find the shortest path with respect to arc costs $c_{ij}^k - \eta_{ij}$;
        let $\theta_p$ denote the cost of the shortest path;
        if $\theta_p - \sigma_k < 0$ then
            add the corresponding path to the RMP;
        end
    end
end
```

We search for cycles and paths with negative reduced cost and add them to the RMP. The column generation procedure is presented in Algorithm 2. When searching for the shortest cycle, we add a virtual node which is as same as the starting node as the destination node. Such that we can use the shortest path algorithm to find the shortest cycle. Note that dual variable $\eta_{ij} \leq 0$, then the arc costs may be negative when searching for the shortest cycle. Therefore, we employ the label-correcting algorithm implemented by Ahuja et al. (1993) to identify the shortest path.

4.1.2 Cutting plane

The cutting plane procedure employs the generation of valid inequalities to strengthen lower bounds in relatively short time comparing with state-of-the-art software. We introduce two sets of valid inequalities including strong inequalities and cutset inequalities. Strong inequalities(SI) have been widely used to improve the quality of lower bound, see Gendron and Crainic (1994), Gendron et al. (1999) and Chouman and Crainic (2014). We extend SI to be suitable for the proposed formulations in our paper which are defined in inequalities (14). The main effects of SI is to restrict the flow of every commodity on an arc with no fleet to be 0.

$$x_{ij}^k \leq d^k \sum_{f\in F} y_{ij}^f \quad \forall (i,j) \in A, \forall k \in K. \tag{14}$$

Cutset inequalities(CI) have been used by Chouman and Crainic (2014) to strengthen the lower bound for CMND with design-balanced constraints. As for the circumstance of considering a heterogeneous fleet, CI have been presented by Kim et al. (1999) which are defined as

$$\sum_{f\in F} u^f Y_{S,\bar{S}}^f \geq D_{S,\bar{S}} \tag{15}$$

where we define cutset $(S,\bar{S})$ by partitioning the node set N into any nonempty subset S and its complement $\bar{S} = N\backslash S$. An arc (i,j) that connects node i in S to node j in $\bar{S}$ belongs to the cutset $(S,\bar{S})$. Let $Y_{S,\bar{S}}^f$ denote the total number of type f vehicles used on the cutset $(S,\bar{S})$ arcs, i.e., $\sum_{(i,j)\in(S,\bar{S})} y_{ij}^f$ for the arc-based formulation and $\sum_{q\in Q^f}\sum_{(i,j)\in(S,\bar{S})} \alpha_{ij}^q y_q^f$ for the cycle-path formulation. Let $D_{S,\bar{S}}$ denote the aggregate demand of all commodities with their origin in S and destination in $\bar{S}$. CI ensure the total demand that must flow from S to $\bar{S}$ will be satisfied by enough capacity on the arcs of $(S,\bar{S})$. In general, the linear solution will not violate CI. Therefore, we lift CI by using a integer rounding procedure to produce *Chvátal-Gomory* (C-G) cuts which are proved to be valid by Kim et al. (1999).

$$\sum_{f\in F}\left(\left\lceil \frac{u_f}{u_l}\right\rceil Y_{S,\bar{S}}^f\right) \geq \left\lceil \frac{D_{S,\bar{S}}}{u_l}\right\rceil \quad \forall l \in F, \text{for all cutsets } \{\mathrm{S},\bar{\mathrm{S}}\} \tag{16}$$

The challenge in using cutset inequalities is how to deal with large amounts of potential cutsets, it is impractical to enumerate all the associated inequalities. Moveover, it becomes more computationally expensive to lift the inequalities as the scale of the problem increases. Therefore, we only generate single-node cutset inequalities, see Chouman et al. (2016) for the details.

The dynamic generation of valid inequalities is embedded in the column generation approach. Once there exist violated cuts, column generation procedure is executed again. The price-and-cut loop stops only if neither negative reduced cost columns nor violated cuts are found.

4.2 Local search

Since the application of SND employing a heterogeneous fleet by logistics enterprises will deal with large-scale network, it is necessary to present effective algorithms to find high-quality solutions efficiently. Therefore, we propose a two-stage local search algorithm to find feasible solutions. Based on the linear solution obtained from C&CG approach, we consider it as a initial neighborhood to search for better solutions.

After we obtain the lower bound, C&CG approach will provide the values of $\bar{x}_{ij}^k$ and $\bar{y}_{ij}^f$ which are defined in arc-based formulation. Then define the arc subset including arcs with nonzero flows as A_1, i.e., $A_1 = \{(i,j) \in A : \sum_{k\in K} \bar{x}_{ij}^k > 0\}$, and another arc subset including arcs with nonzero vehicles as A_2, i.e., $A_2 = \{(i,j) \in A : \sum_{f\in F} \bar{y}_{ij}^f > 0\}$. It is easy to find that $A_1 \subseteq A_2 \subseteq A$ because of the capacity constraints. The first stage satisfies O-D demands and determines flows on each arc $(i,j) \in A_1$. The second stage ensures the balance of vehicles and provides enough capacity for flows on each arc $(i,j) \in A_2$. The corresponding formulations in two stages are defined as follows.

4.2.1 Stage 1: flow distribution problem

Without taking service selection and fleet assignment into account, the formulation of flow distribution problem is actually a capacitated multi-commodity minimum cost flow problem (CMCF). The values of $\sum_{(i,j)\in A_1} u^f \bar{y}_{ij}^f$ decides the capacity u_{ij} of arc (i,j) in A_1.

$$\min \sum_{(i,j)\in A_1} \sum_{k\in K} c_{ij}^k x_{ij}^k \tag{17}$$

s.t.

$$\sum_{j\in N_i^+} x_{ij}^k - \sum_{j\in N_i^-} x_{ji}^k = w_i^k \quad \forall i \in N, \forall k \in K, \tag{18}$$

$$\sum_{k\in K} x_{ij}^k \leq u_{ij} \quad \forall (i,j) \in A_1, \tag{19}$$

$$x_{ij}^k \geq 0 \quad \forall (i,j) \in A_1, \forall k \in K. \tag{20}$$

After solving the CMCF and redistributing the flow on each arc, the arcs without flow are removed from A_1, i.e., $A_1 = A_1 \backslash \{(i,j) \in A_1 : \sum_{k\in K} x_{ij}^k = 0\}$.

4.2.2 Stage 2: fleet assignment problem

Once completing the flow distribution and satisfying the demands of all O-D commodities, we need to assign a heterogeneous fleet to provide transportation services. Let Ω_{ij} be the total flows of all commodities distributed on arc $(i,j) \in A_1$, which equals $\sum_{k \in K} x_{ij}^k$ provided by solving CMCF in the first stage. The mathematical formulation of Stage 2 defined on arc set A_2 is as follow:

$$\min \sum_{f \in F} \sum_{(i,j) \in A_2} h_{ij}^f y_{ij}^f \tag{21}$$

s.t.

$$\sum_{j \in N_i^+} y_{ij}^f - \sum_{j \in N_i^-} y_{ji}^f = 0 \quad \forall i \in N, \forall f \in F \tag{22}$$

$$\sum_{f \in F} u^f y_{ij}^f \geq \Omega_{ij} \quad \forall (i,j) \in A_1 \tag{23}$$

$$y_{ij}^f \in Z_0^+ \quad \forall (i,j) \in A_2, \forall f \in F \tag{24}$$

Constraints (23) ensure each service arc will provide sufficient vehicle capacity for the total flow.

4.2.3 Cycle-based neighborhoods

After the two formulations are solved, we obtain a feasible solution of the original problem. The quality of the feasible solution depends on the two arc subsets A_1 and A_2. To find promising A_1 and A_2, we introduce cycle-based neighborhoods to search for better feasible solutions. The cycle-based neighborhoods have been presented and proved to be efficient when solving CMND-related problems, see Ghamlouche et al. (2003, 2004), Li et al. (2017).

The cycle-based neighborhoods aim at redirecting flow from one path to another, and exploring the space of arc subsets A_1 and A_2 through opening or closing some arcs. To illustrate, considering the partial network in Fig. 2, there is a cycle formed by paths $\{(A,B),(B,C),(C,F)\}$ and $\{(A,E),(E,F)\}$. If we redirect the flow from the former to the latter, a new neighborhood will be generated by opening arcs $(A,E),(E,F)$ and closing empty arcs $(A,B),(B,C),(C,F)$. The procedure of finding a cycle-based neighborhood can be summarized as follows.

- Identify a cycle containing two paths connecting two points.
- Redirect the flow from one path to another and ensure that at least one open arc becomes empty.
- Open all formerly closed arcs in the cycle and close all formerly open arcs if they are empty after the flow redirection.

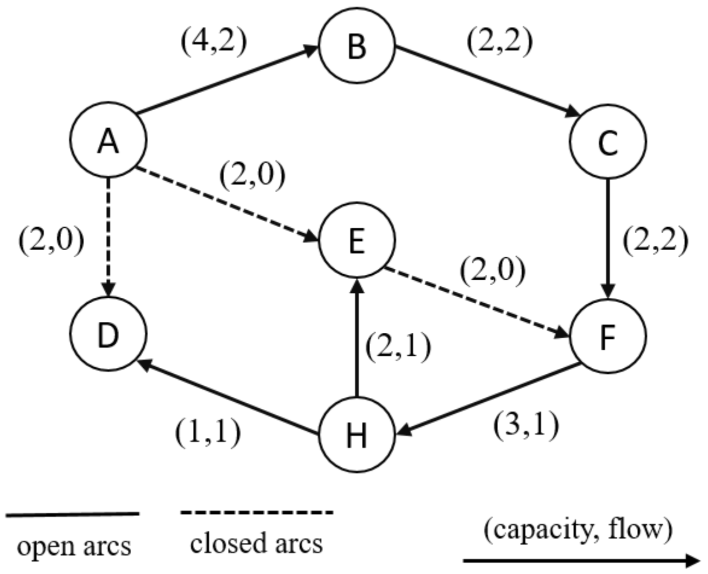

Figure 2: Example of cycle in partial network

It is very important to implement an efficient procedure to find a promising neighborhood since complete evaluation of all moves is impractical, especially facing huge networks. Note that since we have to close at least one previously open arc, it seems suitable to start from a nonempty open arc (i^*, j^*) with $\gamma = \sum_{k\in K} x^k_{i^*j^*}$ units of flow. Since the arc (i^*, j^*) will be closed after the flow redirection, the residual capacity of any cycle associated with (i^*, j^*) must be at least γ. A residual network must be built to efficiently search for low-cost cycles corresponding to promising moves without exhaustive exploration and exact evaluation.

To build such a residual network associated to (i^*, j^*) and γ units of flow redirection, we include two types of arc sets in the new network, one with positive arc cost for additional flows and the other with negative arc cost for reduced flows. Each arc in the original network will be replaced by at most two arcs $(i, j)^+$ and $(j, i)^-$ from the two arc sets respectively. Define $\bar{c}_{ij}$ as the average unit flow cost on arc (i, j), i.e. $\bar{c}_{ij} = \sum_{k\in K} c^k_{ij}/|K|$. Let $\bar{F}_{ij}$ denote the average fixed cost of arc (i, j) to evaluate the impact of fleet assignment because of opening or closing arcs, which equals $\sum_{f\in F} h^f_{ij}/|F|$.

Arc $(i, j)^+$ will be included in the residual network if the residual capacity of arc (i, j) is no less than γ, i.e. $u_{ij} - \sum_{k\in K} x^k_{ij} \geq \gamma$. The cost c^+_{ij} corresponding to $(i, j)^+$ approximates the additional cost of distribute γ units of flow on arc (i, j). It is calculated as the additional flow cost, plus the average fixed cost if arc (i, j) is closed at present, i.e. $(i, j) \notin A_2$.

$$c^+_{ij} = \begin{cases} \bar{c}_{ij}\gamma + \bar{F}_{ij} & if\ (i,j) \notin A_2, \\ \bar{c}_{ij}\gamma & otherwise. \end{cases}$$

Since arc $(j,i)^-$ is employed to represent flow reduction on arc (i,j), it will be included in the residual network only if the current flow on (i,j) is no less than γ. Symmetrically, the cost c_{ji}^- associated to $(j,i)^-$ approximates the reduced cost of decreasing γ units of flow on arc (i,j). It is calculated as the reduced flow cost, minus the average fixed cost if arc (i,j) will be empty after reduction of γ units of flow.

$$c_{ji}^- = \begin{cases} -\bar{c}_{ij}\gamma - \bar{F}_{ij} & if \sum\limits_{k\in K} x_{ij}^k = \gamma, \\ -\bar{c}_{ij}\gamma & if \sum\limits_{k\in K} x_{ij}^k > \gamma. \end{cases}$$

Such a residual network aims at identifying the lowest-cost cycle containing arc (i^*, j^*) to redirect γ units of flow. Moreover, it takes into account the influence of opening or closure of arcs to search for better feasible solutions. To illustrate how to find the lowest-cost cycle in the residual network, consider the network example of Fig. 3. Suppose arc (A,B) is the candidate arc to be closed, the residual network associated with (A,B) is shown in Fig. 3. The average unit flow costs and fixed costs of all arcs in the example are assumed to be 2 and 3 respectively. Arcs in the residual network are labeled by their costs. Note that there are two cycles associated with arc (A,B): $(A,E),(E,F),(F,C),(C,B),(B,A)$ and $(A,E),(E,B),(B,A)$. The two cycles have costs of -7 and 4, respectively. Since the former cycle has the lowest cost, 2 units of flow are moved from path $\{(A,B),(B,C),(C,F)\}$ to path $\{(A,E),(E,F)\}$. Therefore, a new neighborhood is obtained by opening arcs $(A,E),(E,F)$ and closing empty arcs $(A,B),(B,C),(C,F)$.

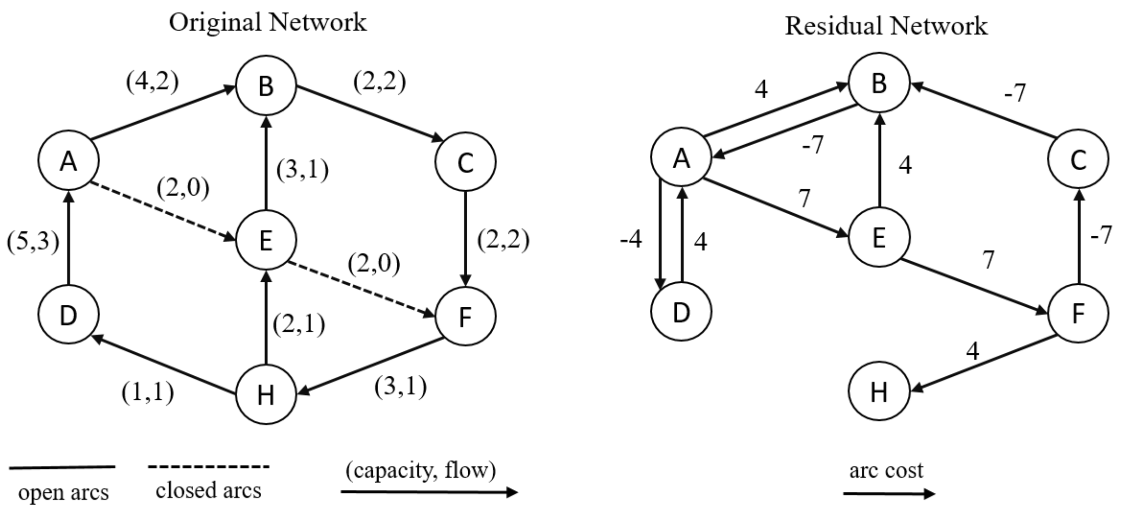

Figure 3: Network example associated with (A,B)

4.2.4 Main local search loop

Since each new neighborhood is associated with a candidate arc, let A^+ denote the candidate arcs, which equals $\{(i,j) \in A : \sum_{k \in K} x_{ij}^k > 0\}$. With respect to each arc (i^*, j^*) in A^+, we build a residual network except arc (i^*, j^*) and find the shortest path from i^* to j^* to complete the cycle. We still employ the label-correcting algorithm implemented by Ahuja et al. (1993) to identify the shortest path.

Note that the closure of some arcs may lead to infeasible neighboring solutions. Therefore it is necessary to check whether at least one path links the origin and destination of each commodity before comparing the cycle costs. Although it can not ensure the feasibility of the first stage when solving the flow distribution problem, it still decreases the appearance of infeasible local search moves.

Once the lowest cost cycle is found, A_1 is updated by the move including opening and closure of some arcs. Then we solve the flow distribution problem and update A_1 again by closing empty arcs. Before solving the fleet assignment problem, A_2 is updated by $A_1 \cup A_2$. If the two stages are both feasible, we obtain a feasible solution of the original problem. Furthermore, A_1 and A_2 in the next iteration are determined by this feasible solution. To explore more extensive search space, we assign a tabu status to the arc associated with the feasible move. The local search approach terminates if either of the two stages is infeasible or the maximum iteration is reached. The sketch of the local search algorithm is shown in Algorithm 3.

Algorithm 3: Local search algorithm

```
Input: linear solution obtained by C&CG
1  while termination criterion not met do
2      determine $A_1, A_2$ and $A^+$;
3      foreach $(i^*, j^*) \in A^+$ do
4          create residual network;
5          determine the lowest-cost cycle associated with $(i^*, j^*)$;
6      end
7      find the cycle with minimum cost;
8      implement the move;
9      update $A_1$ then solve flow distribution problem, stop if infeasible;
10     update $A_1$ and $A_2$;
11     solve fleet assignment problem, stop if infeasible;
12     update the current solution;
13     assign a tabu status to $(i^*, j^*)$;
14     if find better solution then
15         update best feasible solution;
16     end
17 end
Output: best feasible solution
```

5 Numerical experiments and analyses

The computation study is implemented in C++, using CPLEX 12.63 as the linear programming solver. All experiments are performed using a computer with four 64-bit 2.4 GHz Intel Core processors and 4 GB of RAM, operating under Windows 10. Computing time is reported in seconds.

5.1 Data instances

Since there are no suitable instances available in the literature, we randomly generate 30 large-scale instances based on the classic network instance sets described in Crainic et al. (2001) and used in several papers (Ghamlouche et al. (2003), Pedersen et al. (2009) and Chouman and Crainic (2014)). The arc flow cost is an integer randomly generated in the interval [2,7]. The demand of each commodity is generated over [2,15]. The capacity and fixed cost of each type of fleet is a random number in the interval [2,15] and [100,300] respectively. Each instance is characterized by its numbers of fleet types, nodes, commodities and arcs, noted $|F| \in \{5, 10\}$, $|N| \in \{30, 35, 40\}$, $|K| \in \{200, 300, 400\}$ and $|A| \in \{600, 800\}$, respectively.

5.2 Parameter setting

During the process of computing lower bound, to avoid instability, we use the CPLEX recommended value, a tolerance of 10^{-6} for relative change in objective function value in column generation approach. That is, columns will be added only if their reduced costs is less than -10^{-6}.

As for the maximum iteration of local search algorithm, we choose two instances randomly from the generated 30 instances, i.e., instance 15 and instance 23 whose properties are listed in Table 1 in detail. We record the change of total cost for the two instances in each local search iteration to illustrate the evolution of solution values. As Fig. 4 shows, the solution values are improved very significantly in previous several iterations. But the improvement tendency of the solution value becomes weak near 50 iterations. Therefore, we set the maximum iteration number as 50.

Furthermore, we find it easy to obtain a good solution in a short time when solving the second stage in local search procedure. There is no need to solving the second stage optimally. Therefore the fleet assignment problem is solved by CPLEX with a time limit of 30 seconds. Each instance is only solved once by the proposed hybrid algorithm due to the stability of results.

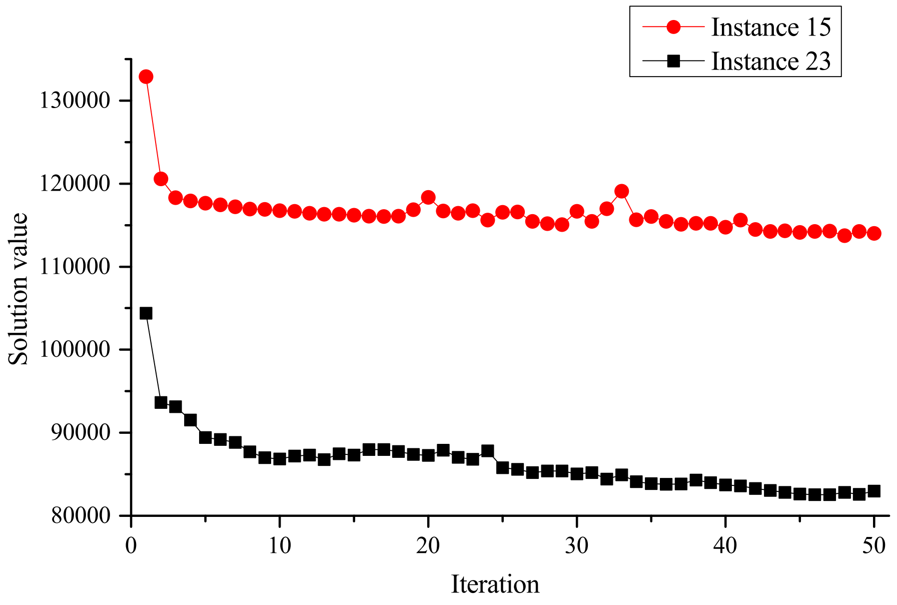

Figure 4: Evolution of solution values obtained by local search approach

5.3 Computational results

The scope of the section is to evaluate the performance of the proposed solution method. In particular, We compare the results with CPLEX running 10 hours of CPU time to show clearly its effectiveness and efficiency in strengthening lower bounds and finding high-quality feasible solutions. The comparison results with respect to 30 large-scale instances are shown in Table 1. Without loss of generality, the values of all lower bound, upper bound and computing time are round to integers to show in Table 1.

The lower bounds and upper bounds obtained by CPLEX running 10 hours of CPU time are shown in Column LB_1 and UB_1. Let GAP_1 denote the percentage gap between LB_1 and UB_1, which is calculated as $\frac{\mathrm{UB}_1-\mathrm{LB}_1}{\mathrm{UB}_1}\times 100$. The lower bounds, total number of cuts and total computing time obtained by column-and-cut generation approach are displayed in column LB_2, Cuts and T_1, respectively. The percentage gap between LB_1 and LB_2 is shown in column GAP_2, which is calculated as $\frac{\mathrm{LB}_2-\mathrm{LB}_1}{\mathrm{LB}_2}\times 100$. Column UB_2 and T_2 display the upper bounds and computing time of local search approach. Column GAP_3 denotes the percentage gap between UB_1 and UB_2, i.e., $\frac{\mathrm{UB}_2-\mathrm{UB}_1}{\mathrm{UB}_2}\times 100$. The last column GAP_4 shows the percentage gap between UB_2 and LB_2, which is calculated as $\frac{\mathrm{UB}_2-\mathrm{LB}_2}{\mathrm{UB}_2}\times 100$.

We calculate the distribution of four relative percentage gaps listed in Table 1 and show the results in Table 2. GAP_1 and GAP_4 denote the relative gaps between lower bounds and upper bounds obtained by CPLEX and the proposed hybrid algorithm with respect to 30 instances. The gaps of most instances are more than 50% when solved by CPLEX, by contrast, the gaps of most instances are between 10% and 20% when solved by the proposed hybrid algorithm. In general, CPLEX achieves relative gap by 50.90% on average after 10 hours of CPU time. Nevertheless, the proposed hybrid algorithm achieves relative gap by 18.14% and only takes 2124 seconds averagely.

Table 1: Performance comparisons with CPLEX

No.	$\lvert F\rvert$	$\lvert N\rvert$	$\lvert K\rvert$	$\lvert A\rvert$	CPLEX(10h)			Column & cut generation				Local search			GAP_4
					UB_1	LB_1	GAP_1 (%)	LB_2	GAP_2 (%)	Cuts	T_1	UB_2	GAP_3 (%)	T_2	(%)
1	5	30	200	600	56924	49789	12.53	**50703**	1.80	243	167	60976	6.65	473	16.85
2	5	30	200	800	72651	43794	39.72	**45007**	2.70	265	476	**55676**	-30.49	534	**19.16**
3	5	30	300	600	128337	67196	47.64	**68497**	1.90	284	266	**79869**	-60.68	1275	**14.24**
4	5	30	300	800	111534	61256	45.08	**63313**	3.25	299	583	**75897**	-46.95	926	**16.58**
5	5	30	400	800	145367	80274	44.78	**83075**	3.37	403	537	**96035**	-51.37	2043	**13.50**
6	5	35	200	600	60604	52138	13.97	**52579**	0.84	195	340	63426	4.45	918	17.10
7	5	35	200	800	103486	49710	51.96	**50371**	1.31	228	621	**63999**	-61.70	579	**21.29**
8	5	35	300	600	166677	74780	55.13	**75134**	0.47	238	317	**85966**	-93.89	1705	**12.60**
9	5	35	300	800	150886	73057	51.58	**74124**	1.44	283	708	**88060**	-71.34	1268	**15.82**
10	5	35	400	800	169514	91011	46.31	**91923**	0.99	330	628	**106591**	-59.03	3180	**13.76**
11	5	40	200	600	136708	59145	56.74	**59162**	0.03	225	378	**69868**	-95.67	1608	**15.32**
12	5	40	200	800	140979	50949	63.86	**51558**	1.18	246	786	**64225**	-119.51	1166	**19.72**
13	5	40	300	600	205291	81987	60.06	**82239**	0.31	251	348	**94092**	-118.18	2207	**12.60**
14	5	40	300	800	152466	78037	48.82	**78353**	0.40	279	868	**92660**	-64.54	1968	**15.44**
15	5	40	400	800	210357	97754	53.53	**98586**	0.84	331	847	**113725**	-84.97	3360	**13.31**
16	10	30	200	600	54034	41679	22.86	**43106**	3.31	270	326	**53750**	-0.53	547	**19.80**
17	10	30	200	800	85816	39018	54.53	**41318**	5.57	277	361	**56274**	-52.50	222	**26.58**
18	10	30	300	600	159767	60743	61.98	**63196**	3.88	368	236	**75971**	-110.30	1378	**16.82**
19	10	30	300	800	112950	57601	49.00	**61501**	6.34	342	835	**77457**	-45.82	1181	**20.60**
20	10	30	400	800	144109	75363	47.70	**80385**	6.25	447	696	**92690**	-55.47	2555	**13.28**
21	10	35	200	600	95446	45739	52.08	**46752**	2.17	338	279	**59730**	-59.80	817	**21.73**
22	10	35	200	800	148665	44270	70.22	**46056**	3.88	272	1062	**63544**	-133.95	761	**27.52**
23	10	35	300	600	142254	68884	51.58	**69931**	1.50	370	452	**82531**	-72.36	1904	**15.27**
24	10	35	300	800	136667	59582	56.40	**62267**	4.31	411	574	**76888**	-77.75	1947	**19.02**
25	10	35	400	800	203376	78123	61.59	**81104**	3.68	505	622	**98203**	-107.10	2982	**17.41**
26	10	40	200	600	157310	50049	68.18	**50306**	0.51	299	569	**66488**	-136.60	1581	**24.34**
27	10	40	200	800	119555	46457	61.14	**47406**	2.00	306	1048	**63623**	-87.91	876	**25.49**
28	10	40	300	600	180064	73285	59.30	**74031**	1.01	416	409	**90454**	-99.07	2053	**18.16**
29	10	40	300	800	167173	65169	61.02	**66717**	2.32	450	842	**85347**	-95.87	2229	**21.83**
30	10	40	400	800	197232	83307	57.76	**85543**	2.61	507	702	**105818**	-86.39	2589	**19.16**
Average					-	-	50.90	-	**2.34**	323	**563**	-	**-72.29**	**1561**	**18.14**

Table 2: Distribution of relative percentage gaps

	[10,20)	[20,30)	[30,40)	[40,50)	[50,∞)
GAP_1	2	1	1	7	19
GAP_4	22	8	0	0	0
GAP_2	[0,1.5)	[1.5,3)	[3,4.5)	[4.5,6)	[6,∞)
	13	7	7	1	2
GAP_3	[$-\infty$,-120)	[-120,-80)	[-80,-40)	[-40,0)	[0,∞)
	2	11	13	2	2

5.3.1 Evaluation of column-and-cut generation approach

Table 1 displays the performance of column-and-cut generation approach with respect to lower bound. Comparing with CPLEX running 10 hours, C&CG approach finds better lower bounds in very short time. The values of GAP_2 being positive means C&CG approach can find tighter lower bounds. In general, C&CG approach achieves stronger lower bounds by 2.34% on average comparing with CPLEX, and only takes 563 seconds averagely.

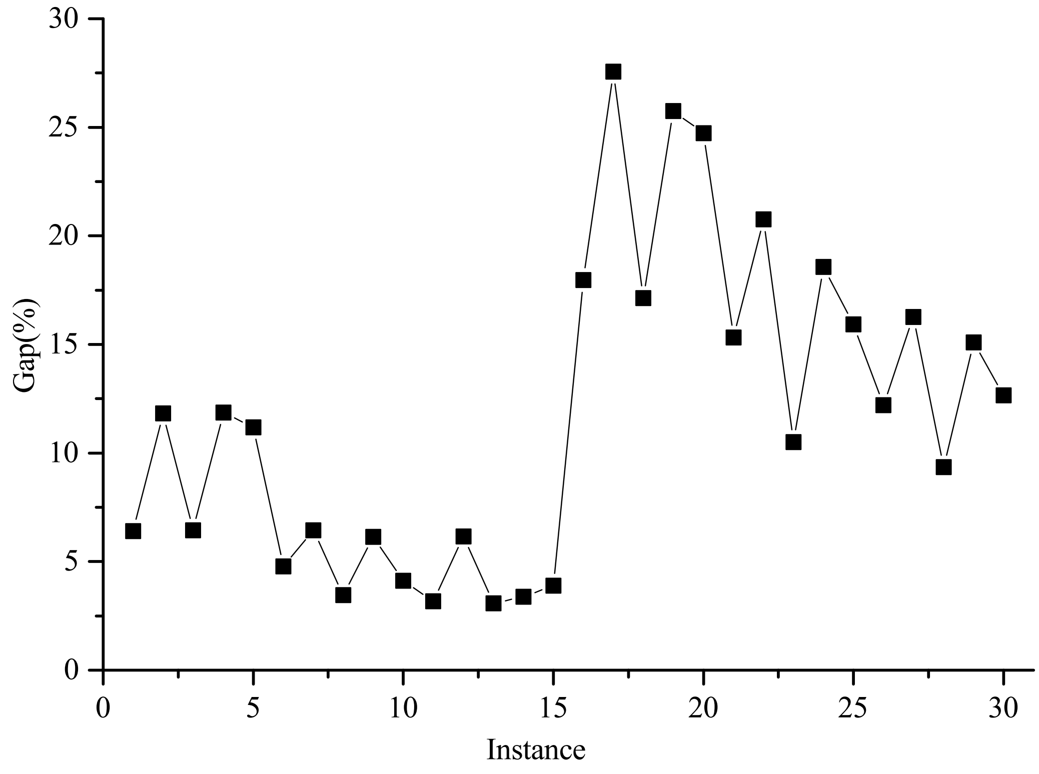

Figure 5: The gaps between LB_2 and the initial linear relaxation bounds

The results show clearly the effectiveness of C&CG approach in improving the quality of the lower bounds. Although the improvements are not significantly, it is still impressive considering the computing time. It only takes hundreds of seconds to outperform CPLEX running 10 hours. This could signify the efficiency of C&CG method in searching for promising columns to reduce the problem size. Moreover, we also calculate the improvements comparing with the first linear relaxation (LP) bounds, which is denoted $\frac{\text{LB}_2 - \text{LP}}{\text{LB}_2} \times 100$. The results of 30 instances are shown in Fig. 5. It demonstrates the effectiveness of these inequalities whose number is 323 on average for each instance in strengthening the lower bounds. As we can see from Fig. 5, the effects of adding cuts vary with each instance.

5.3.2 Evaluation of local search approach

The results with respect to local search approach demonstrate its efficiency in searching for high-quality solutions for large-scale instances, which are illustrated in Table 1. The improvements calculated through GAP_3 of several instances may exceed -100%. In general, local search approach improves the upper bounds by -72.29% on average comparing with CPLEX, and only takes 1561 seconds averagely. Although local search approach fails to find better solutions than CPLEX for two instances, the distribution of GAP_3 in Table 2 still displays the significant improvement of upper bounds.

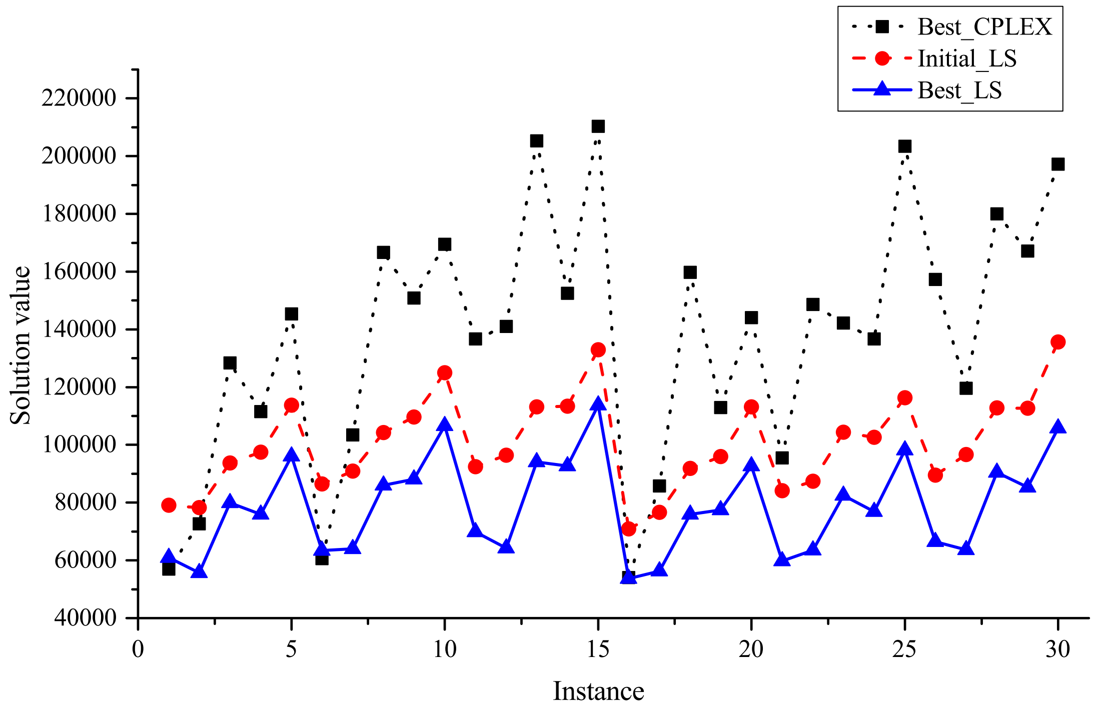

Figure 6: Comparative upper bounds of CPLEX and local search (LS) approach

Fig. 6 shows the comparative upper bounds of CPLEX and local search approach. We display both the initial solution values at the first iteration and the best solution values obtained by local search approach. A majority of the initial solution values line is under the line of the best solution value by CPLEX. Therefore, it confirms that the linear solutions obtained by C&CG approach provide good neighborhood for local search approach to find high-quality initial solutions.

5.3.3 Impact of heterogeneous fleet issue

This section aims to examine the impact of heterogeneous fleet issue and demonstrate its necessity. In this respect, two small-scale test instances which can be solved optimally by CPLEX are randomly generated. Different numbers of fleet type provide transportation services over the two networks. We generate multiple combinations for each certain number of fleet type and take an average as the total cost. Main parameters of these networks and average total costs are shown in Table 3.

Table 3: Parameters and total costs of two networks

Network	$\|N\|$	$\|K\|$	$\|A\|$	$\|F\|$				
				1	3	5	7	10
1	8	15	50	4606.00	4074.13	3815.17	3745.50	3726.00
2	10	20	90	5569.20	4898.63	4625.83	4540.00	4527.00

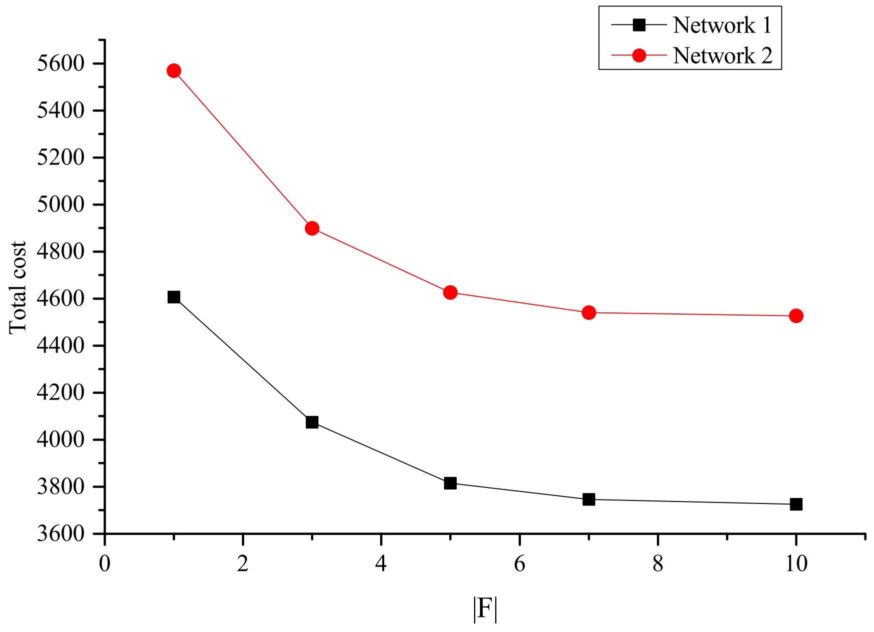

Figure 7: Change of total costs as $|F|$ increases

We employ Fig. 7 to intuitively show the change of total costs as the number of fleet type increases. As Fig. 7 illustrates, the diversification of fleet types reduces total cost over the same network. It is a reasonable result which accords with practical experience.

6 Conclusions

We address heterogeneous fleet issue for service network design problem to accord with the real-life demand of transportation carriers and present two equivalent formulations based on arc and cycle-path respectively. Faced with increasingly huge logistics network, carriers are in urgent need of effective and efficient algorithms to help reduce total cost. Therefore, a hybrid algorithm combing exact and heuristic techniques is introduced to solve the proposed formulations. The method includes a column-and-cut generation procedure to efficiently achieve tight lower bounds and a local search algorithm to find high-quality solutions.

The computational study indicates the merit of the proposed column-and-cut generation procedure in strengthening lower bounds and providing a good neighborhood for the local search approach to identify high-quality initial feasible solutions. The results also show the effectiveness of cycle-based neighborhood in searching for promising solutions. Both exact and heuristic techniques are proved to outperform the state-of-the-art MIP solver CPLEX running 10 hours of CPU time. This performance appears more remarkable when comparing the computational efforts required by identifying lower bounds and upper bounds for the large-scale instances. We also perform computational experiments to illustrate the impact of heterogeneous fleet issue. The results show that the diversification of fleet types helps reduce total cost for carriers.

References

Ahuja, Ravindra K, Thomas L Magnanti, James B Orlin. 1993. *Network flows: theory, algorithms, and applications*. Prentice hall.

Andersen, Jardar, Marielle Christiansen, Teodor Gabriel Crainic, Roar Grønhaug. 2011. Branch and price for service network design with asset management constraints. *Transportation Science* **45**(1) 33–49.

Andersen, Jardar, Teodor Gabriel Crainic, Marielle Christiansen. 2009b. Service network design with asset management: Formulations and comparative analyses. *Transportation Research Part C: Emerging Technologies* **17**(2) 197–207.

Chouman, Mervat, Teodor Gabriel Crainic. 2014. Cutting-plane matheuristic for service network design with design-balanced requirements. *Transportation Science* **49**(1) 99–113.

Chouman, Mervat, Teodor Gabriel Crainic, Bernard Gendron. 2016. Commodity representations and cut-set-based inequalities for multicommodity capacitated fixed-charge network design. *Transportation Science* **51**(2) 650–667.

Chouman, Mervat, Teodor Gabriel, Crainic Bernard Gendron, Teodor Gabriel Crainic, Bernard Gendron. 2009. A cutting-plane algorithm for multicommodity capacitated fixed-charge network design. *Discrete Applied Mathematics* **112**(1) 73–99.

Crainic, Teodor Gabriel. 2000. Service network design in freight transportation. *European Journal of Operational Research* **122**(2) 272–288.

Crainic, Teodor Gabriel. 2003. Long-haul freight transportation. *Handbook of transportation science*. Springer, 451–516.

Crainic, Teodor Gabriel, Antonio Frangioni, Bernard Gendron. 2001. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* **112**(1) 73–99.

Crainic, Teodor Gabriel, Mike Hewitt, Michel Toulouse, Duc Minh Vu. 2016. Service network design with resource constraints. *Transportation Science* **50**(4).

Gendron, Bernard, Teodor Gabriel Crainic. 1994. Relaxations for multicommodity capacitated network design problems. Tech. rep., Univ., CRT.

Gendron, Bernard, Teodor Gabriel Crainic, Antonio Frangioni. 1999. *Multicommodity capacitated network design*. Springer US.

Ghamlouche, Ilfat, Teodor Gabriel Crainic, Michel Gendreau. 2003. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations research* **51**(4) 655–667.

Ghamlouche, Ilfat, Teodor Gabriel Crainic, Michel Gendreau. 2004. Path relinking, cycle-based neighbourhoods and capacitated multicommodity network design. *Annals of Operations Research* **131**(1) 109–133.

Kim, Daeki, Cynthia Barnhart, Keith Ware, Gregory Reinhardt. 1999. Multimodal express package delivery: A service network design application. *Transportation Science* **33**(4) 391–407.

Li, Xiangyong, Kai Wei, YP Aneja, Peng Tian. 2017. Design-balanced capacitated multicommodity network design with heterogeneous assets. *Omega* **67** 145–159.

Magnanti, Thomas L, Richard T Wong. 1984. Network design and transportation planning: Models and algorithms. *Transportation science* **18**(1) 1–55.

Minoux, Michel. 1989. Networks synthesis and optimum network design problems: Models, solution methods and applications. *Networks* **19**(3) 313–360.

Pedersen, Michael Berliner, Teodor Gabriel Crainic, Oli BG Madsen. 2009. Models and tabu search metaheuristics for service network design with asset-balance requirements. *Transportation Science* **43**(2) 158–177.

Smilowitz, Karen R, Alper Atamtürk, Carlos F Daganzo. 2003. Deferred item and vehicle routing within integrated networks. *Transportation Research Part E: Logistics and Transportation Review* **39**(4) 305–323.

Teypaz, Nicolas, Susann Schrenk, Van-Dat Cung. 2010. A decomposition scheme for large-scale service network design with asset management. *Transportation Research Part E: Logistics and Transportation Review* **46**(1) 156–170.

Vu, Duc Minh, Teodor Gabriel Crainic, Michel Toulouse. 2013. A three-phase matheuristic for capacitated multi-commodity fixed-cost network design with design-balance constraints. *Journal of Heuristics* **19**(5) 757–795.