Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics & Operations Research

Solving the competitive hub location problem

Florine van Meer
Student number: 435101
Supervisor: Nemanja Milovanović MSc
Co-reader: Dr. Twan Dollevoet
July 8, 2018

Abstract

A mixed integer programming model and three heuristics are proposed to maximize market shares of a smaller liner shipping provider. The aim of this liner shipping provider is to compete with a bigger established liner shipping provider. The problem is essential to the smaller liner shipping provider, since otherwise the dominant larger alliances might force it to go out of business. Because solving the exact model turns out to be hard in practice, this thesis implements a Lagrangian heuristic, tabu search and a genetic algorithm. All heuristics are considerably faster than the exact model and for small instances, good quality solutions are found. Both the tabu search and genetic algorithm are significantly faster than the Lagrangian heuristic. The tabu search provides better objective values for a larger sized instance as compared to the Lagrangian heuristic, whereas the genetic algorithm outperformed both the Lagrangian relaxation and tabu search heuristic for most configurations.

Contents

1 Introduction

Following the global growth in trade volumes, liner shipping hub-and-spoke networks have emerged. In a liner shipping hub-and-spoke network, flow is concentrated on hub-to-hub connections. This has numerous advantages when the ships are sufficiently full. For example, larger, more efficient vessels can be used on hub-to-hub connections which are both cheaper per twenty-feet equivalent unit and possibly more environmentally friendly.

As mentioned by Gelareh et al. (2010), several alliances of larger liner shipping providers that emerged as a reaction to the current trend of global trade volumes have a dominant position. Therefore smaller, possibly newcomer liner shipping companies need to come up with ways to compete with the bigger alliances.

We will take a look at the same setting as in Gelareh et al. (2010). Here, it is investigated how a smaller newcomer liner shipping provider can compete with a bigger liner shipping provider that has already decided on the network it is going to use, and therefore already has established its fares and travel times between any pair of ports we consider in the network. We will adopt the same mathematical model and propose some heuristics to contribute to this problem. Data from this paper is however not publicly available. Therefore data for this thesis is taken from LINER-LIB [1].

In this paper, we start off with providing an overview of the existing literature on this topic. This will be covered in Section 2. We explain the dataset in Section 3. Then we will present a mixed-integer programming formulation (MIP) for our model in Section 4.1. The MIP is difficult to solve in practice, therefore we follow Gelareh et al. (2010) and apply a Lagrangian heuristic procedure in Section 4.2. We then propose two additional heuristics, namely a tabu search heuristic in Section 4.3 and a genetic algorithm in Section 4.4. The results are discussed in Section 5 and the thesis is concluded in Section 6.

[1] https://github.com/blof/LINERLIB

2 Literature review

In the past decades hub-and-spoke network design problems have emerged in all types of industries. Amongst others, airline passenger carriers, overnight package delivery services and rail sorting yards (O'Kelly, 1987). The first papers to address hub location problems are O'Kelly (1986a) and O'Kelly (1986b). Since then, hub location problems have been studied extensively. A paper that gives an excellent overview of the different hub location problems is Alumur and Kara (2008). Another discussion of the evolution of hub location problems can be found in Campbell and O'Kelly (2012). In the following, we will only focus on (general) hub location problems that include competition. However, to investigate different solution methods in Section 4 we also looked at hub location and facility location problems in different application areas.

One of the first papers on hub location problems in a competitive environment (passenger transportation) is Marianov et al. (1999), where customers were attracted to change company by offering a cheaper alternative. They proposed a tabu search to solve the problem. This work was later extended by Eiselt and Marianov (2009), using gravity-like utility functions.

Other papers also incorporate game theory in their models. Sasaki and Fukushima (2001) introduced a Stackelberg hub location problem, where a bigger company first decides on its hubs and other, smaller companies react on that. Lin and Lee (2010) also consider a game theoretic model using the Cournot-Nash equilibrium. Their results indicated that there are cooperative equilibra that can make all parties better off, but these equilibria are unstable in the non-cooperative game. Asgari et al. (2013) consider the competition and cooperation strategies amongst two major container hub ports and the shipping companies. They also use a game theoretic model and an interval branch-and-bound method to solve the models.

A paper that looks into a problem very closely related to our setting is Lüer-Villagra and Marianov (2013). This paper also looks into the situation where a big existing company already operates on a hub-and-spoke network, and tries to maximize profit for a newcomer company by choosing the best hub locations and pricing. They use a genetic algorithm to solve the model.

Sasaki et al. (2014) again propose a Stackelberg model. In this paper two firms compete for customers where the leader first locates its hubs to maximize revenue, the follower then locates its own hub arcs to maximize its own revenue. Mahmutogullari and Kara (2016) propose a mixed integer linear formulation for the competitive hub location model. They use an enumeration based solution approach.

3 Data

Most of the data from this paper is taken from LinerLib [1]. We use three instances. The first instance consists of 10 ports in East-Asia, whereas the other two consist of $n = 15$ and $n = 20$ randomly chosen ports. To compare our results with the results from Gelareh et al. (2010), we try to choose the same parameter values where known. We choose $(\beta^C_m, O^C_{ijm}(\beta^C_m)) \in \{(0.65, 0.4), (0.75, 0.3), (0.9, 0.2)\}$, $\forall$ $i, j \neq i, m \in \{1, ..., F^C)$ and $(\beta^t_{m'}, O^t_{ijm'}(\beta^t_{m'})) \in \{(0.25, 0.6), (0.50, 0.5), (0.75, 0.4)\}$.

To calculate the travel time between ports i and j, we first retrieved the distance D_{ij} between these ports from LinerLib, where we only selected routes that did not pass through the Panama or Suez canal. Note that the routes are symmetrical, i.e. the distance from i to j equals the distance from j to i. We assumed the average hourly speed of a ship equals 17 knots/hr. the travel time in days T_{ij} is therefore calculated as

$$T_{ij} = \frac{D_{ij}}{(24 \cdot 17)}$$

To calculate the costs from port i to j, we have added the fuel costs based on the distance from i to j and port call costs PC_j of port j. The port call costs for every port can be retrieved from LinerLib, as well as the number of bunker tons required per day for a ship. Based on this data, we assume a ship uses 50 bunker tons a day. The price of one bunker ton is retrieved online [2] and is set to 680, which seemed to be an average value in the months May and June 2018 when this research was conducted. The costs C_{ij} from port i to j can then be calculated as follows:

$$C_{ij} = PC_j + D_{ij} \cdot 680 \cdot 50$$

To simulate the competition between the two liner shipping providers, we first generated the costs and travel times as above. Then, for every pair of ports, we generated the costs and travel times of the two liner shipping providers, by multiplying C_{ij} and T_{ij} with a uniform random number between 0.7 and 1.3 for every pair of ports and every liner shipping provider. Lastly, we have used a holding cost of 15% per year.

[1] `https://github.com/blof/LINERLIB`

[2] `http://www.bunkerindex.com/prices/bixfree.php?priceindex_id=4`

4 Methodology

4.1 Mixed integer programming formulation

We adopt the same mathematical model as the MIP introduced in Gelareh et al. (2010). For simplicity, we also use the same mathematical notation for the sets, parameters and decision variables. In the following, $\mathscr{A}$ stands for the bigger liner shipping provider and $\mathscr{B}$ for the newcomer/smaller company. H denotes the set of ports and $C_{ij}^{\mathscr{A}}$ and $T_{ij}^{\mathscr{A}}$ $(i, j \in H)$ denote the rate of offered service and the travel time between ports i and j provided by liner shipping provider $\mathscr{A}$ respectively.

The objective of company $\mathscr{B}$ is to maximize its market shares. Therefore, we introduce an attraction function to measure the attraction for customers leaving from $\mathscr{A}$ to $\mathscr{B}$ when company $\mathscr{B}$ offers service against a certain cost (C_{ij}) or transportation time (T_{ij}). Note that we consider the offered service and travel time of company $\mathscr{A}$ between each pair of ports as given. Several different attraction functions can be proposed. In this paper we use the following function:

$$f(x) = e^{-\theta x^2}$$

where θ is a small, fixed constant. The exponential function is suitable because it touches the point (0,1) in the Cartesian plane, and decreases smoothly towards the y-axis. Additionally, the function never becomes negative. Note that the higher the costs or transportation time, the less people are attracted by company $\mathscr{B}$. To avoid non-linearity in our model we estimate the attraction function with a piecewise constant function (Gelareh et al., 2010). The idea behind this discretized function is that the number of attracted customers will change at certain threshold values, $0 < \beta_1^C < ... < \beta_{F^C}^C \leq 1$ and $0 < \beta_1^t < ... < \beta_{F^t}^t \leq 1$ for costs and transportation times respectively. Thus, if company $\mathscr{B}$ offers service against a certain cost or transportation time, it depends on the cost or transportation time offered by $\mathscr{A}$ to what extent customers are attracted by company $\mathscr{B}$.

We need the following decision variables for our model; x_{ijkl} equals 1 if the path from i to j uses the hub edge $k-l$ in the optimal solution. h_k equals 1 if location k is chosen as a hub. a_{ijk} equals 1 if the path (in the optimal solution) from i to j uses the spoke edge $i-k$ and i is not a hub, b_{ijk} equals 1 if the path in the optimal solution uses the spoke edge $k-j$ and j is not a hub. γ_{ijk} equals 1 if goods are transshipped at hub k in the optimal path from i to j. e_{ij} equals 1 if the path from i to j uses edge $i-j$ in the optimal solution and one of them is a hub, whereas s_{ij} equals 1 if there is a direct connection between i and j in the optimal solution and both are not hubs. Finally, δ_{ij}^m, $m \in 1, ..., F^C$ equals 1 if the transportation cost for connection $i-j$ offered by $\mathscr{B}$ lies in the interval $(\beta_{m-1}^C C_{ij}^{\mathscr{A}}, \beta_m^C C_{ij}^{\mathscr{A}}]$ and $\eta_{ij}^{m'}$, $m' \in 1, ..., F^t$ equals 1 if the service time for connection $i-j$ offered by $\mathscr{B}$ lies in the interval $(\beta_{m'-1}^t T_{ij}^{\mathscr{A}}, \beta_{m'}^t T_{ij}^{\mathscr{A}}]$.

We will now introduce the necessary parameters. Let T_{ijk}^{tr} denote the transit time for goods from i to j at hub port k, and C_k^h denotes the holding cost proportional to the length of time that goods are waiting at hub k. Furthermore, let $O^C(\beta_m^C)$ denote the percentage of market share that company

$\mathscr{B}$ gains if the cost lies in the interval $(\beta^C_{m-1}C^{\mathscr{A}}_{ij}, \beta^C_m C^{\mathscr{A}}_{ij}]$ and $O^t(\beta^t_{m'}))$ denotes the percentage of market share that company $\mathscr{B}$ gains if the service time lies in the interval $(\beta^t_{m'-1}T^{\mathscr{A}}_{ij}, \beta^t_{m'}T^{\mathscr{A}}_{ij}]$. t_{ij} denotes the travel time from port i to port j and C^{tr}_k denotes the transshipment cost at hub port k. The discount factor applied to hub-to-hub edges is denoted by α $(0 < \alpha < 1)$. Lastly, C_{ij} denotes the rate of offered service between ports i and j by liner shipping provider $\mathscr{B}$. We are now ready to present the MIP formulation of the model:

$$\text{Maximize} \quad \lambda \sum_i \sum_{j \neq i} \sum_{m=1}^{F^c} O^C_{ijm}(\beta^C_m)\delta^m_{ij} + (1-\lambda) \sum_i \sum_{j \neq i} \sum_{m'=1}^{F^t} O^t_{ijm'}(\beta^t_{m'})\eta^{m'}_{ij}, \tag{4.1}$$

Subject to

$$\sum_k h_k = p, \tag{4.2}$$

$$\sum_{l \neq i} x_{ijil} + \sum_{l \neq i,j} a_{ijl} + e_{ij} + s_{ij} = 1, \quad \forall i, j \neq i \in K, \tag{4.3}$$

$$\sum_{l \neq j} x_{ijlj} + \sum_{l \neq i,j} b_{ijl} + e_{ij} + s_{ij} = 1, \quad \forall i, j \neq i \in K, \tag{4.4}$$

$$\sum_{l \neq k,i} x_{ijkl} + b_{ijk} = \sum_{l \neq k,j} x_{ijlk} + a_{ijk}, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.5}$$

$$x_{ijkl} + x_{ijlk} \leq h_k, \quad \forall i, j \neq i, k, l > k \in K, \tag{4.6}$$

$$x_{ijkl} + x_{ijlk} \leq h_l, \quad \forall i, j \neq i, k, l > k \in K, \tag{4.7}$$

$$\sum_{l \neq k} x_{kjkl} \leq h_k, \quad \forall j, k \neq j \in K, \tag{4.8}$$

$$\sum_{k \neq l} x_{ilkl} \leq h_l, \quad \forall i, l \neq i \in K, \tag{4.9}$$

$$a_{ijk} + \sum_{l \neq j,k} x_{ijlk} \leq h_k, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.10}$$

$$b_{ijk} + \sum_{l \neq k,i} x_{ijkl} \leq h_k, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.11}$$

$$e_{ij} + 2x_{ijij} + \sum_{l \neq j,i} x_{ijil} + \sum_{l \neq i,j} x_{ijlj} \leq h_i + h_j, \quad \forall i, j \neq i \in K, \tag{4.12}$$

$$\sum_k \sum_{l \neq k} \alpha C_{kl} x_{ijkl} + \sum_{k \neq i,j} C_{ik} a_{ijk} + \sum_{k \neq i,j} C_{kj} b_{ijk} + C_{ij}(e_{ij} + s_{ij}) +$$
$$\sum_{k \neq i,j} (2C^{tr}_k + T^{tr}_{ijk} C^h_k)\gamma_{ijk} \leq C^{\mathscr{A}}_{ij} \sum_{m \in F^C} \beta^C_m \delta^m_{ij} + M\delta^{F^C+1}_{ij}, \quad \forall i, j \in K, \tag{4.13}$$

$$\sum_{m=1}^{F^C+1} \delta^m_{ij} = 1, \quad \forall i, j \in K, \tag{4.14}$$

$$\sum_k \sum_{l \neq k} t_{kl} x_{ijkl} + \sum_{k \neq i,j} t_{ik} a_{ijk} + \sum_{k \neq i,j} t_{kj} b_{ijk} + t_{ij}(e_{ij} + s_{ij}) +$$
$$\sum_{k \neq i,j} \gamma_{ijk} T^{tr}_{ijk} \leq T^{\mathscr{A}}_{ij} \sum_{m \in F^T} \beta^t_{m'} \eta^{m'}_{ij} + M\eta^{F^T+1}_{ij}, \quad \forall i, j \neq i \in K, \tag{4.15}$$

$$\sum_{m'=1}^{F^T+1} \eta_{ij}^{m'} = 1, \quad \forall i, j \neq i \in K, \tag{4.16}$$

$$\gamma_{ijk} \leq h_k, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.17}$$

$$\gamma_{ijk} \leq a_{ijk} + b_{ijk}, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.18}$$

$$\gamma_{ijk} \leq 1 - s_{ij}, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.19}$$

$$\gamma_{ijk} \leq 1 - e_{ij} \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.20}$$

$$\gamma_{ijk} \geq a_{ijk} \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.21}$$

$$\gamma_{ijk} \geq b_{ijk}, \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.22}$$

$$e_{ij} \leq 2 - (h_i + h_j), \quad \forall i, j \neq i \in K, \tag{4.23}$$

$$s_{ij} \leq 1 - h_i \quad \forall i, j \in K, \tag{4.24}$$

$$s_{ij} \leq 1 - h_j \quad \forall i, j \in K, \tag{4.25}$$

$$a_{ijk} \leq 1 - h_i \quad \forall i, j \neq i, k \neq i, j \in K, \tag{4.26}$$

$$b_{ijl} \leq 1 - h_j \quad \forall i, j \neq i, l \neq i, j \in K, \tag{4.27}$$

$$h_i, x_{ijkl}, a_{ijk}, b_{ijk}, e_{ij}, s_{ij}, \gamma_{ijk}, \delta_{ij}^m, \eta_{ij}^{m'} \in \{0, 1\}, \quad \forall i, j, k, l \in K. \tag{4.28}$$

Here, (4.1) maximizes the total percentage of market shares as a convex combination of costs and time. The parameter λ is application-specific and can be decided by the user. A company that wants to differentiate itself with high service levels can choose λ to be close to 0, whereas a company that wants to focus on cheap fares can choose λ close to 1. Constraints (4.2) ensure that exactly p ports are used as hubs. Constraints (4.3) − (4.5) enforce flow conservation. Constraints (4.6) − (4.7) make sure that both hubs are opened if 2 hubs are used along the route. Constraints (4.8) and (4.9) enforce that only demand with origin or destination in a hub can select a hub edge to leave from the origin or arrive at the destination. Constraints (4.10) − (4.11) make sure that if demand enters any other port (apart from origin and destination) along the route, this port is a hub node. Constraints (4.12) make sure the correct hub is allocated to a path. Constraints (4.13) choose the fare that company $\mathscr{B}$ offers. If the fare is too high to attract customers, the constraint becomes redundant. Constraints (4.14) make sure that either the fare is too expensive or the fare falls exactly between two boundaries of the attraction function. Constraints (4.15) − (4.16) do the same for the transportation time. Constraints (4.17) − (4.22) control the transits at any path. Constraints (4.23) − (4.27) control the destinations of any path. Lastly, constraints (4.28) ensure that all variables are binary.

4.2 Lagrangian heuristic

Solving the MIP exactly is rather difficult. Therefore we propose a Lagrangian heuristic that was introduced by Gelareh et al. (2010) in this section, a tabu search heuristic in 4.3 and a genetic

algorithm in 4.4. We explain the Lagrangian procedure by means of an example. We take a look at the following problem (D):

$$\begin{aligned} \max \quad & c^T x \\ \text{s.t.} \quad & Ax \leq b \\ & Dx \leq d \\ & x \geq 0 \end{aligned} \tag{4.29}$$

For this problem, we make the assumption that the problem without the difficult constraint set $Dx \leq d$ is relatively easy to solve. Hence, these constraints complicate the problem. The Lagrangian relaxation removes these restrictions and incorporates them in the objective function, assigning a Lagrangian multiplier vector λ to them. Violated constraints can be penalized in the objective by adjusting the Lagrangian multipliers. The Lagrangian relaxation (LR) is as follows:

$$\begin{aligned} \max \quad & c^T x + \lambda^T (d - Dx) \\ \text{s.t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{4.30}$$

Here, the function $\theta(\lambda) = \max(c^T x + \lambda^T (d - Dx))$, subject to the easy constraints $Ax \leq b$ and nonnegativity constraints $x \geq 0$ is called the Lagrangian function. The above problem is a relaxation of problem (D) and therefore provides upper bounds on the problem. The problem of seeking values for the Lagrangian multipliers that correspond to the lowest possible upper bound is called the Lagrangian dual problem and is defined as follows:

$$Z_d = \min(\theta(\lambda) : \lambda \geq 0)$$

The objective of this Lagrangian dual problem is called the the Lagrangian dual function and is an upperbound on the original problem (D).

In a Lagrangian heuristic procedure, a Lagrangian relaxation is solved to obtain an upper bound for the problem. A lower bound for the problem can be obtained by transforming the infeasible solution of the Lagrangian relaxation by making use of the interpretation of the problem. The Lagrangian dual function is non-smooth, and a possible way to determine the best λ corresponding with the tighest upper bound is subgradient optimization. In the following subsections, we will discuss the computation of a lower and upper bound and the heuristic procedure in detail.

4.2.1 Finding an upper bound

Gelareh et al. (2010) relax constraints (4.3), (4.4), (4.5), (4.8), (4.9), (4.10), (4.11) and (4.12), using the multipliers u^1_{ij}, u^2_{ij}, u^3_{ijk}, u^4_{ij}, u^5_{ij}, u^6_{ijk}, u^7_{ijk}, and u^8_{ij}. We then obtain the following problem:

$$\text{Maximize} \quad \lambda \sum_{i} \sum_{j \neq i} \sum_{m=1}^{F^c} O^C_{ijm}(\beta^C_m)\delta^m_{ij} + (1-\lambda) \sum_{i} \sum_{j \neq i} \sum_{m'=1}^{F^t} O^t_{ijm'}(\beta^t_{m'})\eta^{m'}_{ij} +$$
$$\sum_{i,j \neq i} -u^1_{ij}(\sum_{l \neq i} x_{ijil} + \sum_{l \neq i,j} a_{ijl} + e_{ij} + s_{ij} - 1) + \sum_{i,j \neq i} -u^2_{ij} \sum_{l \neq j} (x_{ijlj} + \sum_{l \neq i,j} b_{ijl} + e_{ij} + s_{ij} - 1) +$$
$$\sum_{i,j,k \neq i,j} -u^3_{ijk}(\sum_{l \neq k,i} x_{ijkl} + b_{ijk} - \sum_{l \neq k,j} x_{ijlk} + a_{ijk}) + \sum_{j,k \neq j} -u^4_{jk}(\sum_{l \neq k} x_{kjkl} - h_k) +$$
$$\sum_{i,l \neq i} -u^5_{il}(\sum_{k \neq l} x_{ilkl} - h_l) + \sum_{i,j,k \neq i,j} -u^6_{ijk}(a_{ijk} + \sum_{l \neq j,k} x_{ijlk} - h_k) +$$
$$\sum_{i,j,k \neq i,j} -u^7_{ijk}(b_{ijk} + \sum_{l \neq k,i} x_{ijkl} - h_k) + \sum_{i,j \neq i} -u^8_{ij}(e_{ij} + 2x_{ijij} + \sum_{l \neq j,i} x_{ijil} + \sum_{l \neq i,j} x_{ijlj} - h_i - h_j) \tag{4.31}$$

$$\text{s.t. } (4.2), (4.6), (4.7), (4.13), (4.14), (4.15), (4.16), (4.17), (4.18), (4.19), (4.20),$$
$$(4.21), (4.22), (4.23), (4.24), (4.25), (4.26) \text{ and } (4.27), (4.28)$$

When we solve this problem, regarding the Lagrangian multipliers as given, we obtain an upper bound on the optimal objective value of the MIP problem.

4.2.2 Finding a lower bound

After solving the Lagrangian relaxation and obtaining an optimal solution and corresponding upper bound, we make this solution feasible with a heuristic procedure (Gelareh et al., 2010). It turns out that once we know where to locate the hubs, the remaining problem is much easier to solve. Constraint 4.2 can be discarded and many of the remaining constraints can be decomposed for every pair of ports. The remaining problem is solved with CPLEX.

4.2.3 The complete Lagrangian heuristic procedure

Now we know how to find an upper and lower bound, we are ready to take a look at the optimization procedure. Different values of the Lagrangian multipliers can be chosen, and we try to find the value of these multipliers that corresponds to the lowest possible upper bound. This is done by a subgradient procedure. This procedure iteratively improves the value of the upper bound by updating the Lagrangian multipliers in every step as follows (Guta, 2003):

$$u^{q,n+1}_{ij(k)} = u^{q,n}_{ij(k)} + \theta^n \Delta^n_i$$

where, $q \in \{1, 2, 3, 4, 5, 6, 7, 8\}$. $u^{q,n}_{ij(k)}$ is the Lagrangian multiplier for iteration $n+1$, θ^n is the step size and Δ^n_i is the subgradient. Many forms of this optimization procedure exist. We will use

the step size as proposed by Guta (2003):

$$\theta^n = \frac{\mu^n(UB - LB)}{\|\Delta_i^n\|^2}$$

Here, θ^n is the step size for iteration n, UB is the best upper bound so far and LB is the best lower bound so far. μ^n is a step size parameter that can be chosen by the user.

The subgradient is different for every constraint. In particular, if the constraints are of the form $Ax \leq b$, then $\Delta_i^n = Ax^n - b$, where x^n is the solution of the Lagrangian relaxation. For example, for constraints 4.3 the subgradient becomes:

$$\Delta_{ij}^n = \sum_{l \neq i} x_{ijil} + \sum_{l \neq i,j} a_{ijl} + e_{ij} + s_{ij} - 1$$

As can be seen, the subgradient measures to what extent a constraint is violated. If a constraint is violated, the Lagrangian multiplier must be updated accordingly in order to heavily penalize the constraint violation.

The entire Lagrangian heuristic procedure is given in Algorithm 1. The algorithm has the parameters n, x and ϵ as inputs. n and ϵ determine the stopping conditions. The algorithm is terminated when a maximum of n iterations is reached, when the upper bound and lower bound coincide or when μ has become smaller than a preset value. x is chosen to determine the value of μ in equation 4.2.3. We will determine μ based on Han (2013), who compare different step size parameters for the subgradient method and the parameter $\mu = \frac{1}{2^{counter}}$, where *counter* indicates the number of times the procedure did not update the upper bound for x iterations seems suitable.

In every iteration, the heuristic first calculates an upper bound using the Lagrangian relaxation and transforms this solution into a feasible solution, obtaining a lower bound by considering the hub nodes as fixed. If the new upper (lower) bound is lower (larger) than the best upper (lower) bound found so far, the best upper (lower) bound is updated. Then, we calculate the subgradient and update the Lagrangian multipliers. We continue this process untill at least one of the stopping criteria is met.

4.3 Tabu search

Tabu search is a widely-used heuristic and has also been applied to network design problems. Calık et al. (2009) use a tabu search for the hub covering problem and found efficient solutions on a large data set. Abyazi-Sani and Ghanbari (2016) solved the uncapacitated single allocation hub location problem with a tabu search. They compared their results to known exact results and obtained the optimal solution for many instances in a short computation time.

Tabu search is a heuristic closely related to local search and uses a so-called tabu list to avoid getting stuck in a local optimum. It is necessary to define the neighbourhood of a solution, which is a set of solutions that can be reached by systematically changing a solution into a new solution by only changing some characteristics.

Algorithm 1: Subgradient procedure

Subgradient(n, x, ϵ)**;**
Initialization: $UB \longleftarrow \infty$**,** $LB \longleftarrow -\infty$**;**
$n_{cur} \longleftarrow 0$**,** $x_{cur} \longleftarrow 0$**,** $\mu_0 \longleftarrow 1, counter \longleftarrow 1,$ $\mathbf{u}_{ij(k)}^{q,0} \longleftarrow 0$ $\forall i, j, k$**;**
while $\mu \geq \epsilon$ *and* $n_{cur} \leq n$ **do**
    Obtain UB_{cur} **and** LR_{sol} **by solving (4.31) with** $u_{ij(k)}^{q,n}$**;**
    if $UB_{cur} \leq UB$ **then**
        $UB \longleftarrow UB_{cur}$ **;**
        $x_{cur} \longleftarrow 0$
    else
        $x_{cur} \longleftarrow x_{cur} + 1$
    Obtain LB_{cur} **by transforming** LR_{sol} **into a feasible solution;**
    if $LB_{cur} \geq LB$ **then**
        $LB \longleftarrow LB_{cur}$
    if $x_{cur} = x$ **then**
        $counter \longleftarrow counter + 1$ **;**
        $x_{cur} \longleftarrow 0$ **;**
        $\mu \longleftarrow \frac{1}{2^{counter}}$ **;**
    Calculate subgradient Δ_i^n**;**
    $u_{ij(k)}^{q,n+1} = u_{ij(k)}^{q,n} + \theta^n \Delta_i^n$**;**
    $n_{cur} \longleftarrow n_{cur} + 1$

The heuristic starts with an initial solution. Then the neighbourhood of that solution is explored and a new solution is chosen. Often this is the solution with the highest objective value. In the tabu search, we keep track of a tabu list, which is a list with solutions that cannot be used for a certain number of iterations θ. This list can for example contain the last k solutions, such that is it forbidden to go back to a past solution for θ iterations. The new solution that is chosen can of course not appear on the tabu list. If there is no solution that increases the objective value, we choose the solution that worsens the objective value the least.

The tabu search heuristic we propose consists of 2 parts. In the initialization, p hubs are chosen randomly and then a local search procedure is performed on the resulting solution with these hubs. The local search procedure is displayed in Algorithm 2. Given a certain solution, local search is performed as follows. We start by determining the neighbourhood of this solution. For every route from i to j, we construct with Allocate_I a direct route from i to j. With Allocate_II we construct a route that passes through exactly one hub node and with Allocate_III we construct a route that passes through exactly two hub nodes. We chose not to include the option of passing through more than 2 hub ports, because the additional port call costs and time for loading and unloading the ship might make this option less attractive and the computation time increases heavily. The 3 different allocation methods are summarized below.

- **Allocate_I:** Direct connections. In this method only direct connections from i to j are generated,
- **Allocate_II:** Connections that pass through a hub. Here, we establish connections of the type $i \longrightarrow k \longrightarrow j$, with k a random hub port,
- **Allocate_III:** Connections that pass through 2 hubs. Here, we establish connections of the type $i \longrightarrow k \longrightarrow l \longrightarrow j$, with k and j random hub ports.

After establishing the new routes we have determined the neighbourhood of an initial solution. If the objective value of this solution is higher than the best objective value found so far, we update the latter one. We continue this process for a fixed number of iterations and return the best solution found so far after these iterations.

The complete tabu search heuristic is shown in Algorithm 3. It has as inputs n and k. n determines the maximum number of iterations and k determines the maximum size of the tabu list. In the initial solution, hubs are chosen randomly and only direct paths from i to j are used. In every iteration, we perform a local search as explained above on the current solution. If we find a better solution than the best solution found so far, and this solution does not occur in the tabu list, we update the best solution. If the size of the tabu list exceeds the maximum size of the tabu list, we remove the first element of the tabu list (the solution that was stored in the list longest). Then, in the Swap_Hub method, a random hub node is switched with a random non-hub node. If there were existing routes through one of these nodes, we transform these routes to a direct route from i to j. The tabu search terminates when a pre-set number of maximum iterations n is reached.

Algorithm 2: Local search heuristic

```
1  LocalSearch(initialSolution);
2  Input: initialSolution;
3  Output: best solution after performing local search;
4  Initialization: best_soFar ⟵ initialSolution, current_Sol ⟵ initialSolution, counter
     ⟵ 0, n ⟵ 10;
5  while counter < n do
6      Compute neighbourhood of current_Sol with Allocate_I, Allocate_II and
        allocate_III;
7      current_Sol ⟵ solution with highest objective value;
8      if Objective of current_Sol > objective of best_soFar then
9          best_soFar ⟵ current_Sol
10 Return best_soFar
```

Algorithm 3: Tabu search heuristic

```
1  TabuSearch(n, k);
2  Initialization: best_soFar ⟵ initialSolution, current_Sol ⟵ initialSolution, counter
     ⟵ 0, TabuList ⟵ ∅;
3  while counter < n do
4      current_Sol ⟵ LocalSearch(current_Sol)
5      if Objective of current_Sol > objective of best_soFar and current_Sol ∉ TabuList then
6          best_soFar ⟵ current_Sol
7      Add current_Sol to TabuList;
8      if Size of TabuList > k then
9          Remove first element of TabuList
10     current_Sol ⟵ Swab_Hubs(current_Sol)
11     counter ⟵ counter + 1;
12     Return best_soFar
```

4.4 Genetic algorithm

Genetic algorithm is a meta-heuristic that can be used on a wide range of problems. The idea of the genetic algorithm originates from biology. We start with an initial population of n solutions and change this population by combining two solutions into one and mutating solutions, untill no better solutions are found. The population thus involves over time. The chromosomes correspond to solutions of the problem, and the fitness function that defines the quality of a chromosome (often this is the objective function). Solutions evolve through selecting two 'parents' and combining them into a child. Enough diversity in the chromosomes is maintained by the mutation operator.

Alp et al. (2003) propose a genetic algorithm for the p-median problem and many features of the genetic algorithm proposed in this paper are inspired by Alp et al. (2003). They do not implement a mutate procedure because they did not find much improvement after doing this, so we will also not adopt this procedure.

We define the chromosomes to be the chosen hub ports. We evaluate the fitness of our solutions with the objective function of the original MIP formulation. We aim to construct an initial population where all genes are present. The population should not be too big to slow down the algorithm, but a population that is too small might not exhibit enough genetic diversity. Alp et al. (2003) propose to choose the population size as a function of n, the number of ports, and p, the number of hub ports, as follows:

$$P(n,p) = \max\{2, [\frac{n}{100} \cdot \frac{ln(S)}{d}]\}d$$

where S is the binomial coefficient, the number of times we can choose p hub ports out of n ports. We thus have a total S of s possible unique hub choices. Furthermore, d is defined as $d = \left\lceil \frac{n}{p} \right\rceil$.

The complete genetic algorithm procedure is displayed in Algorithm 4. We will go through this algorithm step by step. First, we initialize the counter *iter* to 0, that keeps track of the total number of iterations the best objective value found so far did not change. We initialize a population of size $P(n,p)$ as follows. We want to have a population where all genes are present. We start with a solution that corresponds with hub ports $1, 2, ..., p$. For a second solution, we allocate the hub ports $p+1, p+2, ...2p$. When we have reached the total number of ports, we repeat the same process a little differently, as this time we allocate genes $1, 3, ..., 2p-1$. With this procedure, a lot of genetic variation is present in the initial population which is very desirable. The corresponding solutions are found by solving a MIP and considering the hub nodes as fixed. This can be done in reasonably faster time than solving the complete MIP as introduced in Section 4.1.

In every iteration, we randomly select 2 parents and combine them into a candidate solution. This is done as follows. First, if there are hub nodes that are the same in both parents, they are always selected as hub nodes in the new gene. Then, for every hub node that was present in either $parent1$ or $parent2$, we calculate (using the MIP with fixed hubs) the increase in fitness when adding that hub to the gene. We add the hubs that correspond with the highest increase till we again have p hubs.

After obtaining this candidate member, we check whether its fitness is higher than the best fitness found so far. If this is not the case, we update the counter *iter*. Then, we update the population with the replace procedure. This procedure checks whether the candidate solution from the combine algorithm is at least better than the worst solution in the population. If this is the case, the worst member of the population is replaced by the candidate member and the best solution found so far is updated. We repeat this process till we do not find an improvement in fitness for $\lceil n\sqrt{p} \rceil$ iterations, as Alp et al. (2003) propose.

Algorithm 4: Genetic algorithm

Initialization: $iter \longleftarrow 0$, initialize a population of size P(n,p), $bestSoFar \longleftarrow$ first member of the population, $candidate \longleftarrow$ null;
while $iter \leq \lceil n\sqrt{p} \rceil$ **do**
  randomly select parent1 and parent2 from the population;
  $candidate \longleftarrow combine(parent1, parent2)$
  if *fitness of candidate* $\leq fitnessofBestSoFar$ **then**
    $iter \longleftarrow iter + 1$;
  $bestSoFar \longleftarrow replace(candidate)$
  Return bestSoFar

5 Computational results

All results were obtained with a computer running Windows 10, Intel Core i7-6500U 2.5 GHz and 8 GB DDR3 with 2 cores. Where needed, we used the MIP solver CPLEX (version 12.8). All heuristics are implemented in Java (version 8, update 112). This section is organized as follows. We first discuss the MIP results. Then we perform a small sensitivity analysis for the Lagrangian heuristic and we end with a comparison of the different heuristics.

5.1 Mixed integer programming formulation

Table 5.1 gives an insight how big the instances are. Both the number of variables and the number of constraints grow heavily when the number of ports n increases. This implies that solving the problem for larger instances becomes harder and harder.

Table 5.1: Number of variables and number of constraints for the different instances.

n	Number of variables	Number of constraints
10	12439	18252
15	58334	80327
20	177662	232702

The results for the Asia instance with $n = 10$ for different configurations are displayed in Table 5.2. The same model with an equally sized instance instance is run for the same configurations in Gelareh et al. (2010). The running times in that paper are significantly higher than the running times obtained in this paper. This might be due to the more recent value of CPLEX and the higher computing power of the used laptop. However, as can be seen in Table 5.3, for larger sized instances of $n = 15$ and $n = 20$, the problem cannot be solved to optimality in reasonable time. Thus, the MIP can still not be solved exactly within reasonable timeframes for larger instances. One configuration, where $\lambda = 0$ seems to be much easier to solve than the other configurations and can even be solved to optimality in reasonable time for the larger instances. Recall that $\lambda = 0$ corresponds to the case where the objective only focuses on maximizing the market shares with respect to travel times. Because the travel times as taken from LinerLib are symmetric, and based on actual distances, they satisfy the triangle inequality. This does not necessarily need to hold for the fares, and with a problem where $\lambda \neq 0$ we may end up with a problem that does not satisfy the triangle inequality. It seems that this increases the complexity of the problem drastically.

To get an idea what an actual network might look like, we displayed the network for the instance with $n = 10$ ports in Asia in Figure 5.1. For the given settings, ports (2,6,7,9) are chosen as hubs. In the figure, only the routes leaving from port 4 going to all other ports are displayed. Consider for example the route from port 4 to 6. The distance from 4 to 6 is relatively large, and to save costs

Table 5.2: Results for the MIP model for the instance with $n = 10$ ports.

		p = 2			p = 4		
α	λ	Hubs	Objective	Time in seconds	Hubs	Objective	Time in seconds
0.6	0.0	(5,8)	27.00	1.14	(2,5,8,9)	27.37	0.66
	0.2	(5,7)	22.98	162.98	(2,6,7,9)	24.06	177.32
	0.4	(5,7)	18.96	194.14	(2,6,7,9)	20.27	240.14
	0.6	(5,7)	14.95	176,33	(2,6,7,9)	16.90	242.56
	0.8	(5,7)	10.93	210.21	(2,6,7,9)	13.54	800.09
	1.0	(5,7)	6.91	155.41	(2,6,7,9)	10.17	532.35
0.75	0.0	(5,8)	27.00	0.77	(2,5,8,9)	27.00	0.69
	0.2	(6,9)	22.90	164.72	(2,5,6,7)	23.24	223.35
	0.4	(6,9)	18.79	154.93	(2,5,6,7)	19.49	283.34
	0.6	(6,9)	14.69	183.56	(2,5,6,7)	15.73	218.89
	0.8	(6,9)	10.58	193.09	(2,5,6,7)	11.97	385.49
	1.0	(6,9)	6.48	223.62	(2,5,6,7)	8.215	279.96
0.9	0.0	(5,8)	27.00	0.74	(2,5,8,9)	27.00	0.77
	0.2	(6,9)	22.79	114.67	(3,5,8,9)	22.96	159.99
	0.4	(3,9)	18.58	217.72	(3,5,8,9)	18.92	231.13
	0.6	(3,9)	14.36	154.91	(3,5,8,9)	14.88	215.12
	0.8	(0,7)	10.15	213.62	(3,5,8,9)	10.84	229.92
	1.0	(6,9)	5.94	431.29	(3,5,8,9)	6.80	208.74

the route passes through hub 9, utilizing the hub-to-hub discount factor α between hubs 9 and 6. In contrast, if we look at the route from 4 to 8, we see that this route does not pass through hub 9. A possible reason for this is that the costs that would have been saved by using the hub edge 9 - 8 are not as big as the additional port call costs and holding costs that are involved when passing through hub 9.

5.2 Lagrangian heuristic

In the Lagrangian heuristic procedure described in Section 4.2, several parameters have to be specified, x, n and ϵ. Because quite some variation is possible in especially x, we perform a sensitivity analysis and try different values for x. If x is low, the μ will be updated faster and the stepsize θ will decrease faster than when compared to a higher value of x. Han (2013) propose a value for x of 5, but indicated they did not try different values. We use default values of $n = 500$ and $\epsilon = 0.000001$ in the following. Because of the long computation times for larger sized instances, we perform the sensitivity analysis for $n = 10$ and assume that larger instances follow the same behaviour. We

Figure 5.1: Resulting network from MIP model for an instance with n = 10 and p = 4, $\alpha = 0.6$ and $\lambda = 0.5$. In the figure, only the routes leaving from port 4 to all other ports are displayed.

consider the values $x = 1, 5, 10, 15$. The results are shown in Figure 5.2. It can be seen that the higher x, the longer the running time and the lower the best obtained upper bound. There is a trade-off between the quality of a solution and computation time.

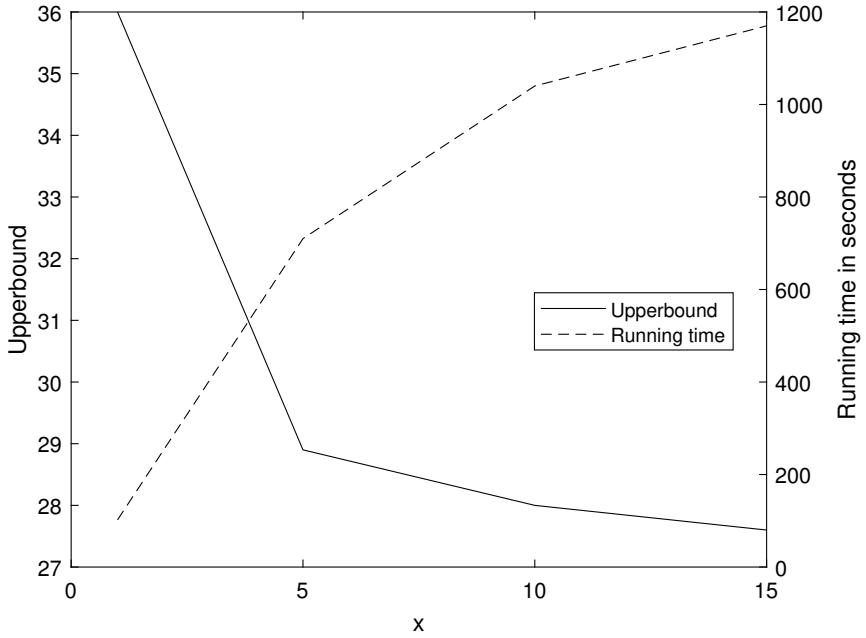

Figure 5.2: The lowest obtained upper bound (solid line) and the running time (dashed line) for various values of x.

Considering the results of the sensitivity analysis, we decided to use a value of $x = 5$, $n = 500$ and $\epsilon = 0.000001$. The Lagrangian heuristic is visualized in Figure 5.3. This shows a typical pattern for subgradient optimization. The lowerbound increases non-monotonically while the upper bound decreases closer and closer to the lowerbound. After some iterations, the upper and lower bound do not seem to get any more closer and it is advisable to terminate the heuristic there. It is remarkable that the lowerbound already obtains a relatively high value already in the first iterations. The solution quality of the lowerbound does not seem to improve after more iterations. In fact, it turns out that a good quality lower bound is often already found in the first few iterations.

It should be noted that, to speed up the Lagrangian heuristic procedure, the maximum time to solve the Lagrangian relaxation was set to 20 seconds as proposed by Gelareh et al. (2010). We found that this did not decrease the solution quality much, but assured much faster convergence of the algorithm.

We finally run the Lagrangian heuristic for instances with 10, 15 and 20 ports with different parameter configurations. The results are displayed in the left part of Table 5.3. The running times are often larger than the running times for a similar instance in Gelareh et al. (2010). This might be due to some implementation choices. For example, we chose a maximum number of iterations of 500 which seems rather conservative, as the total number of iterations in the aforementioned paper did never exceed 100. Because of these implementation issues, it is difficult to compare the results. The objective values found with the Lagrangian heuristic are close to the optimal values found in

the previous section, but not yet optimal. It should be noted that, as displayed in Figure 5.3, a good quality solution is often already obtained after the first few iterations. Therefore the high running times in Table 5.3 do not seem realistic as we can often find a good quality solution in the first few iterations (max 3 - 15 seconds for an instance of $n = 15$).

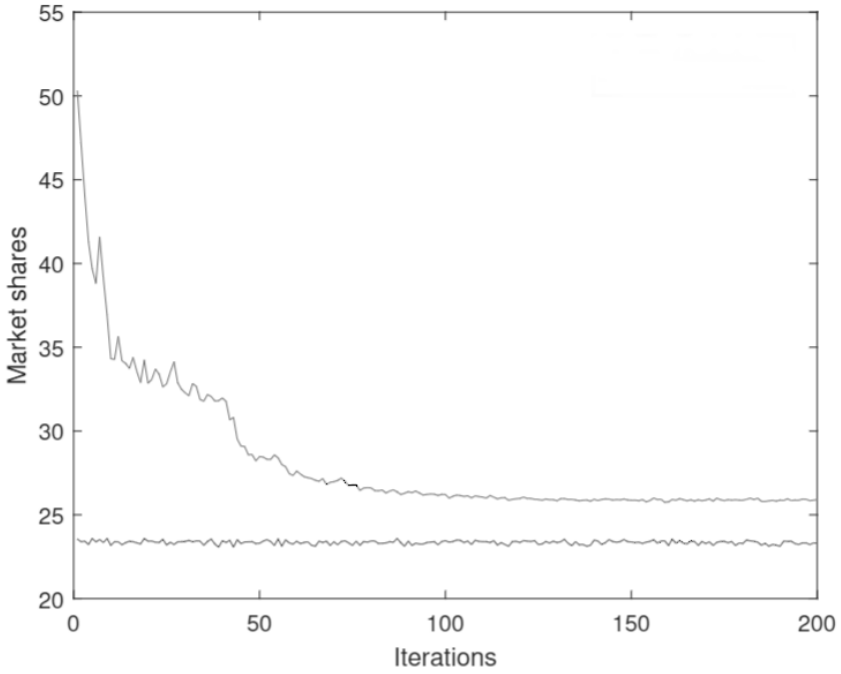

Figure 5.3: Typical development in subgradient optimisation. The graph was made applying the heuristic on an instance with $n = 10$ ports, $p = 4$ hubs, $\alpha = 0.6$ and $\lambda = 0.2$. The highest line corresponds to the upperbound and the lower line corresponds to the lowerbound.

5.3 Comparison of heuristics

The results for all heuristics and the MIP are displayed in Figure 5.3 for different instances. After a short analysis in the previous section, we found that a good quality solution can be found in the first iteration of the Lagrangian heuristic. It turns out that this is also the case for the tabu search and genetic algorithm. Also for these heuristics, a relatively high quality solution is already found in the first few iterations.

Initially, the neighbourhood of the tabu search heuristic was computed by considering the Allocate_I, Allocate_II and Allocate_III methods for every path from i to j. To speed up this process and to ensure enough variety in the solutions, we have altered this approach by only including every path with a certain probability. This reduces the running time significantly.

In Table 5.3, it can be seen that the Lagrangian heuristic provides good lower bounds for smaller

instances. As the instances grow however, the quality of the solution decreases slightly. The tabu search heuristic is outperformed by the Lagrangian heuristic for $n = 10$ and $n = 15$ several times, but outperforms the Lagrangian heuristic for $n = 20$ for all configurations. This suggests that the tabu search performs better on larger instances. The genetic algorithm is faster than both the Lagrangian heuristic and tabu search, and outperforms both in 13 out of 15 cases.

Table 5.3: Results for the MIP, Lagrangian heuristic (LR), tabu search (TS) and genetic algorithm (GA) for different instances and configurations. The time limit is set to 2 hours. The best obtained objective value(s) for the three heuristics are displayed in **bold**.

				MIP		LR		TS		GA	
n	α	λ	p	Objective	Time	Objective	Time	Objective	Time	Objective	Time
10	0.6	0.0	4	27.37	0.66	**27.37**	887.54	**27.37**	10.56	27.00	2.88
	0.6	0.2	4	24.06	177.32	23.84	2005.81	23.96	10.70	**24.06**	5.70
	0.6	1.0	4	10.17	532.35	9.68	2155.86	**10.13**	9.57	9.31	5.59
	0.75	0.4	4	19.49	283.34	19.37	1846.78	19.18	11.06	**19.49**	5.84
	0.9	0.6	4	14.88	215.12	14.80	1887.33	14.72	11.63	**14.88**	5.95
15	0.6	0.0	5	63.00	11.80	**63.00**	3857.58	**63.00**	28.07	**63.00**	18.95
	0.6	0.2	5	$55.24^{4.72\%}$	LIMIT	54.80	4107.64	54.14	45.06	**55.03**	23.66
	0.6	1.0	5	$22.81^{66.49\%}$	LIMIT	15.70	4758.47	18.71	38.80	**23.17**	24.46
	0.75	0.4	5	$45.27^{12.45\%}$	LIMIT	44.83	4758.56	44.07	44.19	**45.21**	23.43
	0.9	0.6	5	$34.17^{14.12\%}$	LIMIT	33.98	4974.47	32.86	49.18	**34.07**	36.83
20	0.6	0.0	5	64.20	100.52	**64.20**	6038.57	**64.20**	102.67	**64.20**	53.61
	0.6	0.2	5	$62.49^{6.48\%}$	LIMIT	54.31	6184.26	54.32	120.66	**55.08**	78.92
	0.6	1.0	5	$55.64^{15.32\%}$	LIMIT	14.66	LIMIT	14.80	123.16	**18.60**	68.88
	0.75	0.4	5	$60.78^{18.24\%}$	LIMIT	43.61	6582.42	43.98	139.33	**44.78**	69.40
	0.9	0.6	5	$59.06^{19.77\%}$	LIMIT	32.32	LIMIT	33.36	122.02	**33.92**	65.48

6 Conclusion

In this thesis we replicated the mixed integer programming model as proposed by Gelareh et al. (2010). The computation times obtained in this paper were considerably faster than the ones obtained in the aforementioned paper, possibly due to an improved version of CPLEX or improved computing power of the used computer. However, the computation times still became too large for larger sized instances. Therefore we proposed three heuristics and described them in detail.

For the Lagrangian heuristic, we have proposed a method to solve both the Lagrangian relaxation and a method how to obtain a feasible solution from this relaxation by making use of the simplified problem when considering the hubs as fixed. We have further proposed a method to solve the Lagrangian dual problem with subgradient optimization. Both the tabu search and genetic algorithm also made use of the simplified problem with fixed hubs that was proposed in Section 4.2.2.

Comparing the results of the three proposed heuristics, the genetic algorithm clearly outperforms both tabu search and Lagrangian relaxation in both solution quality and computation time. We think the genetic algorithm can still be improved by for example varying the size of the population. It would be interesting to look at this for further research.

We did not discuss one assumption that was implicitly made by Gelareh et al. (2010). In the proposed MIP model that we implemented from this paper, the percentage of market shares that can be obtained for every pair of ports i and j are maximized. With this approach, every origin-destination pair is given equal priority. This might however not be the most realistic. It can be of much higher importance for the smaller liner shipping provider to hold a high percentage of market share on busy routes with a lot of demand and/or high profits, whereas there are other routes that might not necessarily have a big influence on the profits of the smaller liner shipping provider. Incorporating this in the model, by for example applying a weighing function based on demand or profits that can be obtained from port i to j, can give different outcomes.

Bibliography

R. Abyazi-Sani and R. Ghanbari. An efficient tabu search for solving the uncapacitated single allocation hub location problem. *Computers & Industrial Engineering*, 93:99–109, 2016.

O. Alp, E. Erkut, and Z. Drezner. An efficient genetic algorithm for the p-median problem. *Annals of Operations research*, 122(1-4):21–42, 2003.

S. Alumur and B. Y. Kara. Network hub location problems: The state of the art. *European journal of operational research*, 190(1):1–21, 2008.

N. Asgari, R. Z. Farahani, and M. Goh. Network design approach for hub ports-shipping companies competition and cooperation. *Transportation Research Part A: Policy and Practice*, 48:1–18, 2013.

H. Calık, S. A. Alumur, B. Y. Kara, and O. E. Karasan. A tabu-search based heuristic for the hub covering problem over incomplete hub networks. *Computers & Operations Research*, 36(12): 3088–3096, 2009.

J. F. Campbell and M. E. O'Kelly. Twenty-five years of hub location research. *Transportation Science*, 46(2):153–169, 2012.

H. A. Eiselt and V. Marianov. A conditional p-hub location problem with attraction functions. *Computers & Operations Research*, 36(12):3128–3135, 2009.

S. Gelareh, S. Nickel, and D. Pisinger. Liner shipping hub network design in a competitive environment. *Transportation Research Part E: Logistics and Transportation Review*, 46(6):991–1004, 2010.

B. Guta. Subgradient optimization methods in integer programming with an application to a radiation therapy problem. pages PhD thesis, Technische Universität Kaiserlautern, 2003.

M. Han. Computational study of the step size parameter of the subgradient optimization method. 2013.

C.-C. Lin and S.-C. Lee. The competition game on hub network design. *Transportation Research Part B: Methodological*, 44(4):618–629, 2010.

A. Lüer-Villagra and V. Marianov. A competitive hub location and pricing problem. *European Journal of Operational Research*, 231(3):734–744, 2013.

A. I. Mahmutogullari and B. Y. Kara. Hub location under competition. *European Journal of Operational Research*, 250(1):214–225, 2016.

V. Marianov, D. Serra, and C. ReVelle. Location of hubs in a competitive environment. *European Journal of Operational Research*, 114(2):363–371, 1999.

M. E. O'Kelly. The location of interacting hub facilities. *Transportation science*, 20(2):92–106, 1986a.

M. E. O'Kelly. Activity levels at hub facilities in interacting networks. *Geographical Analysis*, 18 (4):343–356, 1986b.

M. E. O'Kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.

M. Sasaki and M. Fukushima. Stackelberg hub location problem. *Journal of the Operations Research Society of Japan*, 44(4):390–402, 2001.

M. Sasaki, J. F. Campbell, M. Krishnamoorthy, and A. T. Ernst. A stackelberg hub arc location model for a competitive environment. *Computers & operations research*, 47:27–41, 2014.