A Versatile Heuristic Approach for Generalized Hub Location Problems

J. Fabian Meier & Uwe Clausen
meier@itl.tu-dortmund.de
Uwe.Clausen@iml.fraunhofer.de
TU Dortmund
44227 Dortmund
Germany

Abstract

The usability of hub location models heavily depends on an appropriate modelling approach for the economies of scale. Realistic hub location models require more sophisticated transport cost structures than the traditional α-discount. We develop a general modelling scheme for such problems allowing the definition of complicated (non-linear) costs and constraints; its structure allows an abstract definition of *neighbourhood* to use in metaheuristic algorithms. We test the approach on three models for hub location problems, using self-generated data for gauging and the AP data set for testing. We see that even large instances can be solved, but the gap to (nearly) optimal solutions is still to close.

Keywords: Hub Location, Economies of Scale, Simulated Annealing, Time Expanded Network

The research is part of the project: CL 318/14-1 "Lenkung des Güterflusses in durch Gateways gekoppelten Logistik-Service-Netzwerken mittels quadratischer Optimierung" of Deutsche Forschungsgemeinschaft.

1 Introduction

Hub location problems formalize the old idea of economies of scale in the transportation business: Bundling shipments decreases unit transport costs. But bundling shipments also necessitates detours and expensive facilities for transhipment. The transport costs have to be balanced with the strategic costs for building and maintaining hubs to achieve minimal overall costs.

The hub location research has diverse sources but was standardized with the work of O'Kelly [15] and the following papers of Campbell [7, 8].

They introduced a family of related formulations of hub location problems as Mixed Integer Programs, which can be characterized as follows: We start from a complete directed graph and a shipment for each node-node pair. A solution is a selection of a subset of the nodes as *hubs* (by binary variables) and a routing for each of the shipments through the graph from their sources to their sinks, where all intermediate nodes on every route are hubs. In addition to the cost for establishing hubs unit costs for transport are assumed. To get a rough approximation to economies of scale, these unit costs are reduced by a fixed factor on hub-hub-connections because these connections usually carry more transport volume.

These formulations were appropriate at the time because the resulting MIP has only few non-continuous variables (the binary hub variables) and can be approached with Branch-and-Bound or Branch-and-Cut. Nevertheless, its viewpoint on economies of scale is rather artificial. In 2012, Campbell and O'Kelly stated [9]:

> Much hub location research addressed the fundamental problems or related variants that assumed a constant (flow-independent) discount for transportation on all hub arcs (i.e., α) and required full interconnectivity of the hubs by hub arcs. This may be imposed by the form of the objective or constraints, or implied by the notation, and it often makes sense a priori. These two ideas are related because the economies of scale from concentrating flows on the hub arcs are presumed to create the cost discount for the hub arcs. However, optimal solutions in hub location problems can often result in the hub arcs carrying much smaller flows than do some of the access arcs, yet by assumption the costs are discounted only between the hubs. This clearly represents a limitation of these models because it undermines the basic premise for economies of scale. [...] Given these recent successes in solving very large scale capacitated and uncapacitated versions of fundamental hub location problems, it is reasonable to ask how much more effort should be directed at solving idealized problems with fully interconnected hubs and flow-independent discounts on hub arcs. Practical transportation hub networks would seem rarely to be larger than those solvable with the current state-of-the-art[.]

It seems appropriate to look for other models of transport costs which give a better representation of real situations. We can distinguish two main approaches:

Concave cost functions: Sinking unit costs for increasing transport volumes can be approximated by a concave cost function. This cost function is usually chosen as piece-wise linear function which can be justified by practical

(a finite number of possible modes of transport) or model-theoretic reasons (because linearity eases solvability). Cunha and Silva [11] solve a hub location problem with concave piece-wise linear function by a genetic algorithm. de Camargo et al. [12] use a concave, piece-wise linear function on hub-hub-connections and linear costs for other connections (FLOWLOC model) and solve the problem by Benders' decomposition algorithm.

Vehicle-based costs: The costs for trucks, trains and planes mainly depend on the travelled distance while the filling quota plays a minor role. Hence it is sensible to measure costs *per vehicle* and not *per volume*. Song [17] was one of the first to define a vehicle-based hub location problem. He modelled the costs a step function for a problem in air transport. The work of van de Leensel [18] used a similar model for discrete capacities in a telecommunication network. In [16] Sender and Clausen present a heuristic approach to a similar problem in rail freight. Generally, as also noted by [13], these problems become very difficult MIP because the economies of scale are purely modelled by the integrality of the number of vehicles; the relaxed LP version has no sense of bundling and gives very weak lower bounds. Successful approaches with standard MIP solvers are only documented for simplified problems [14, 3].

Our aim is to define a graph-based modelling approach for generalized hub location problems which should fulfil two prerequisites:

1. It should allow for the inclusion of many practical constraints, e.g.: Different transport cost models, hub sizes, allocation policies, time restrictions, cyclic (weekday-based) networks and some kinds of stochastic influences.

2. It should include a concept of *neighbourhood* which allows attacking the problem with metaheuristics both efficiently and effectively.

We want to use the next section to develop our *Commodity Process Model* (abbreviated ComPM) theoretically. Section 3 describes three hub location models with increasing generality, both modelled as MIP and as ComPM. Section 4 describes the generation of benchmark data. In Sect. 5 we calibrate the constructed heuristic algorithm, i.e. we set the search parameters by statistical methods. Section 6 discusses the numerical results, while Sect. 7 gives an outlook for further applications of the model.

2 The Commodity Process Model

The fundamental ingredient of the model is a directed graph $G = (N, P)$, where N represents nodes in a possibly time-expanded network and P represents transport actions. Furthermore we have a set of *commodities* (we use

shipments synonymously) with source and sink in N. The goal of the optimization problem is to find a route for each shipment through the graph so that all constraints are fulfilled and the total costs are as low as possible.

The term *commodity-driven* combines the following two properties:

1. Only the routes for the commodities are used as independent variables. All other decisions (like hub locations and sizes) are derived from the chosen routes.

2. The admissibility of a route for one commodity does not depend on the routes for the other commodities. Constraints modelling the interaction between two possible routes for different commodities are expressed in the objective function.

In the next subsection we want to derive and define a mathematical model for ComPM. We apply this model to examples in Sect 3.

2.1 Mathematical Definition

As before, we denote by $G = (N, P)$ the graph of *nodes* and *processes*. L is the set of load types, which is necessary for the numerical information attached to every commodity. C is the set of commodities. We define a *change function*

$$a : L \times P \times \mathbb{Q} \to \mathbb{Q}$$

with the property

$$a_{lp}(q) \leq q \qquad \forall l \in L,\ p \in P,\ q \in \mathbb{Q}$$

Possible applications are using load types like "remaining transport time" or "remaining number of allowed transhipments" for commodities to model time or transhipment restrictions.

To every commodity $c \in C$, we assign a source, one or more sinks and a load value for every load type (i.e. volume, weight, remaining transport time or other information important for cost or routing). The necessity of several sinks normally arises in time expanded networks where different arrival times are acceptable.

$$C^{\mathrm{so}} : C \to N \qquad c \mapsto c^{\mathrm{so}}$$

$$C^{\mathrm{si}} : C \to \mathcal{P}(N) \qquad c \mapsto c^{\mathrm{si}}$$

$$C^{L} : L \times C \to \mathbb{Q} \qquad (l, c) \mapsto c^{l}$$

Here, $\mathcal{P}$ denotes the power set. If there is only one sink, we identify c^{si} with the associated element in N.

Let $\phi = (p_1, \ldots, p_n)$ be a circle-free path in the graph $G = (N, P)$. We call ϕ *admissible* for $c \in C$ if

$$n_A(p_1) = c^{\mathrm{so}} \tag{1}$$

$$n_\Omega(p_n) \in c^{\mathrm{si}} \tag{2}$$

$$\forall l \in L, 0 \leq i \leq n: \quad \underbrace{a_{lp_i} \circ a_{lp_{i-1}} \circ \cdots \circ a_{lp_1}}_{:=\phi_l^i}\left(c^l\right) \geq 0, \tag{3}$$

where n_A and n_Ω denote the head and tail node of each process. If we denote by Φ_G the set of circle-free paths in G, then we call a map

$$S : C \to \Phi_G$$

an *admissible solution*, if $\forall c \in C$ the path $S(c)$ is admissible. By $S(c)^i$ we denote the load vector after i steps of the path $S(c)$ (as in (3)).

We call a subset $\hat{P} \subset P$ *local*, if any admissible path can contain at most one process from $\hat{P}$. A single process is always local, but also the set of all incoming processes of a given node.

On $\mathbb{Q}$-vector spaces of the form $\mathbb{Q}^M$ for some set M, we assume the partial order given by component-wise comparison. Then a *local cost function* $\hat{f}$ is an order preserving map $\hat{f} : \mathbb{Q}^{\hat{P}\times L} \to \mathbb{Q}$ with $\hat{f}(0) = 0$, where $\hat{P} = P(\hat{f})$ is a local process set assigned to $\hat{f}$. Let F be a set of local cost functions. Then we denote by

$$\begin{aligned} f : \mathbb{Q}^{P\times L} &\to \mathbb{Q} \\ z &\mapsto \sum_{\hat{f}\in F} \hat{f}\big(\pi_{\hat{p}}(z)\big) \end{aligned} \tag{4}$$

the *total cost function* for F. Here, $\pi_{\hat{p}}$ denotes the projection onto $\mathbb{Q}^{\hat{P}\times L}$, where $\hat{P}$ depends on $\hat{f}$ as above. To calculate the costs, we have to compute the total loads on every process induced by all paths. Let $\phi = (p_1, \ldots, p_n)$ be an admissible path as above. Let $v_\phi \in \mathbb{Q}^{P\times L}$ be the vector given by ϕ_l^i at index position (p_i, l), $i = 1, \ldots, n$, $l \in L$ and 0 elsewhere. The *total cost* of an admissible function is then given by

$$f(S) = f_F\Big(\sum_{c\in C} v_{S(c)}\Big). \tag{5}$$

Definition 1. A ComPM is a 9-tuple

$$\eta = (N, P, C, L, a, C^L, C^{\mathrm{so}}, C^{\mathrm{si}}, F)$$

where the ingredients fulfil the properties detailed above.

2.2 The Algorithmic Aspects of ComPM

Throughout this subsection let $c_0 \in C$ be a fixed commodity and let $C_0 = C \setminus \{c_0\}$. We want to investigate the following question: For a given solution S, how do we find the cheapest solution T with $S|_{C_0} = T|_{C_0}$?

Instead of viewing $S(c_0)$ to be a path in G with attached load vectors, we can consider an extended graph G^L, where the nodes are given by $N^L = N \times \mathbb{Q}^L_{\geq 0}$. A node $(n_0, z_0) \in N^L$ is connected to the node (n_1, z_1) by (p, z_0) if $n_A(p) = n_0$, $n_\Omega(p) = n_1$ and $a_{lp}(z_0^l) = z_1^l$ for each component of the vectors z_0, z_1. Then the admissible paths $(p_1, \dots, p_n)$ from c_0^{so} to $n_0 \in c_0^{\mathrm{si}}$ are in bijective correspondence to paths $\Big((p_1, S(c)^0), (p_2, S(c)^1), \dots, (p_n, S(c)^{n-1})\Big)$ in G^L from $\big(c_0^{\mathrm{so}}, S(c)^0\big)$ to $\big(n_0, S(c)^n\big)$.

To every arc (p, z) of G^L, we assign the vector $w_{(p,z)} \in \mathbb{Q}^{P \times L}$ given by z^l at the index position (p, l), $l \in L$ and 0 elsewhere. Then we can attach to every such arc the cost

$$g^{c_0}(p, z) = f_F\Big(\sum_{c \in C_0} v_{S(c)} + w_{(p,z)}\Big) - f_F\Big(\sum_{c \in C_0} v_{S(c)}\Big) \tag{6}$$

which can be understood as the opportunity cost of using this particular arc when all commodities besides c_0 are fixed.

Lemma 1. The additional (opportunity) cost caused by any path for c_0 is given by summing over the arc costs, i.e.

$$f(S) - f(S|_{C_0}) = \sum_{i=1}^{n} g^{c_0}\big(p_i, S(c_0)^{i-1}\big) \tag{7}$$

where $S(c_0) = (p_1, \dots, p_n) \in \Phi_G$.

Proof. We abbreviate $\sum_{c \in C_0} v_{S(c)}$ by v_0. Then we can write

$$\begin{aligned} f(S) - f(S|_{C_0}) &= f_F\Big(\sum_{c \in C} v_{S(c)}\Big) - f_F(v_0) \\ &= \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}\Big(\sum_{c \in C} v_{S(c)}\Big) - \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}(v_0) \\ &= \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}\Big(v_0 + \sum_{j=1}^{n} w_{(p_j, S(c_0)^{j-1})}\Big) - \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}(v_0) \end{aligned}$$

As all $\hat{f} \in F$ are local, we can write F as disjoint union $\cup_{i=1}^{n} F_i$ with $p_i \in \hat{P} = P(\hat{f})$ implies $\hat{f} \in F_i$:

$$= \sum_{i=1}^{n} \sum_{\hat{f} \in F_i} \hat{f} \circ \pi_{\hat{p}}\Big(v_0 + \sum_{j=1}^{n} w_{(p_j, S(c_0)^{j-1})}\Big) - \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}(v_0)$$

$$= \sum_{i=1}^{n} \sum_{\hat{f} \in F_i} \hat{f} \circ \pi_{\hat{p}}\Big(v_0 + w_{(p_i, S(c_0)^{i-1})}\Big) - \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}(v_0)$$

We can change the summation in the second sum since all additional terms are zero:

$$= \sum_{i=1}^{n} \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}\Big(v_0 + w_{(p_i, S(c_0)^{i-1})}\Big) - \sum_{\hat{f} \in F} \hat{f} \circ \pi_{\hat{p}}(v_0)$$

$$= \sum_{i=1}^{n} f_F\Big(v_0 + w_{(p_i, S(c_0)^{i-1})}\Big) - f_F(v_0)$$

$$= \sum_{i=1}^{n} g^{c_0}\big(p_i, S(c_0)^{i-1}\big)$$

□

Lemma 1 implies that the opportunity costs of a path for c_0 (with all $c \in C_0$ fixed) is the length of that path in G^L with arc cost g^{c_0}. Therefore, the best solution T with $T|_{C_0} = S|_{C_0}$ can be determined by finding the shortest path for c_0 in G^L with distance function g^{c_0}.

To quickly find shortest paths it is essential to restrict the number of possible paths – especially in the given, possibly infinite graph.

Let us fix $l \in L$. For every process $p \in P$ we can define the *l-length* by

$$\lambda^l(p) = \min_{q \in \mathbb{Q}} q - a_{lp}(q) \geq 0 \text{ (from the assumption).}$$

Using this l-length we can define the l-distance $\lambda^l(n_1, n_2)$ for every pair of nodes n_1 and n_2 in N as the total length of a shortest l-path. The distance matrix can be efficiently computed by the Floyd-Warshall algorithm.

If we have for $(n_1, z), (n_2, u) \in G_\Theta$:

$$z^l < \lambda^l(n_1, n_2)$$

then we already know that there cannot exist a path in G^L from (n_1, z) to (n_2, u), because for any path $\theta = \big((p_1, z), (p_2, z_1), \ldots, (p_n, z_{n-1})\big)$ with $\phi = (p_1, \ldots, p_n)$ we have

$$\begin{aligned}
\phi_l^n(z^l) &= a_{lp_n} \circ \cdots \circ a_{lp_1}(z^l) \\
&= \phi_l^n(z^l) - \phi_l^{n-1}(z^l) + \phi_l^{n-1}(z^l) - \ldots + \phi_l^1(z^l) - z^l + z^l \\
&\geq -\lambda^l(p_n) - \lambda^l(p_{n-1}) - \ldots - \lambda^l(p_1) + z^l \\
&\geq -\lambda^l(n_1, n_2) + z^l < 0.
\end{aligned}$$

So when we search a path for c_0 and reach a node $(n_1, z) \in G^L$ for which there exits $l \in L$ so that all sinks c_0^{si} become unreachable, we know that this node cannot be part of an admissible path.

2.3 Neighbourhood Generation

The generation of a good neighbourhood lies at the heart of many metaheuristic search methods. The classical aim of neighbourhood generation is to get a good coverage of the search space. In our case it is not sensible to use "all admissible routes for all commodities" as search space because this space is huge and contains a lot of obviously bad solutions. Thus we aim for a greedy repair approach detailed below.

Definition 2. Let $\check{C}$ be a subset of C. Then we define the restricted problem $\eta|_{\check{C}}$ by restricting every function in η to $\check{C}$. The restriction $S|_{\check{C}}$ of an admissible solution is still admissible.

Lemma 2. For a given solution $S_0|_{\check{C}}$ for $\eta|_{\check{C}}$, we find an optimal solution S for $\eta|_{\check{C}\cup\{c\}}$ with $S_0|_{\check{C}} = S|_{\check{C}}$ by adding a shortest path from G^L as described in the previous section.

Following lemma 2, we can extend every solution from $\check{C}$ to C by successively adding shortest paths for all elements $c \in C \setminus \check{C}$. This algorithm can be used for two purposes:

1. To create an initial solution from $\check{C} = \emptyset$.
2. To generate a neighbouring solution by randomly choosing a subset $C \setminus \check{C}$, deleting all routes in it and successively repairing it in the fashion described above.

This neighbourhood generation scheme is able to produce strong improvements in a short period of time. But of course it is easy to come up with counterexamples that show that this kind of neighbourhood generation may be trapped in a small compartment of the solution space without ever reaching (near) optimal solutions. We therefore consider the following improved modus operandi:

1. Divide $C \setminus \check{C}$ into two disjoint sets $C_r \cup C_d$.
2. Choose "random" routes for the elements of C_r (with a probability distribution favouring good routes).
3. Choose locally optimal routes for the elements of C_d in the sense of Lemma 2.
4. Replace the routes of C_r by locally optimal ones.

The first step ensures that also routes can be chosen that do not minimize opportunity costs for one single commodity. The whole approach has a number of parameters (size of $\check{C}$, C_r, C_d, algorithm for choosing the subset etc.) which can only reasonable be determined by a proper experimental set-up (details in Sect. 5).

2.4 The Simulated Annealing Algorithm

The described neighbourhood generation procedure will be used to implement a Simulated Annealing algorithm. The search parameters for this algorithm will be set by the experimental set-up defined in Sect. 5. Of course, any other neighbourhood search algorithm can be combined with the procedure from subsection 2.3, and also combinations with genetic algorithms are possible. Our numerical results (Sect. 6) will serve as a benchmark for future explorations of different algorithms.

3 Generalized Hub Location Problems

We want to describe three models here – with increasing generality.

M1 We use the model NMCHLP model from [10], which is a hub location problem with different hub levels.

M2 We replace the linear, α-discounted transport costs by a vehicle-based step function combined with a small linear component (representing fuel usage and other weight-based costs).

M3 We introduce the time component: We derive a weekday based plan with time restrictions for each shipment. We assume a large-scale transport network so that we measure the time in days.

3.1 M1: Modelling Different Hub Levels

We start by considering the NMCHLP model from [10] which we call M1. It is a multi-allocation hub location problem with classical α-discount, no direct transport, one or two hubs on every commodity route and hubs of several possible sizes. [10] provide experimental data based upon the AP data set which we will extend and use for our experiments.

Let us restate the model with slightly adjusted notation: Let $G = (N, P)$ be a graph, where N is the disjoint union of the source-sink-set S and the possible hub set H. If a depot serves as source/sink and possible hub simultaneously, it is virtually split into two nodes with distance zero. The arc set P consists of $P^{SH} \cup P^{HH} \cup P^{HS}$ connecting all sources to hubs, hubs to hubs and hubs to sinks respectively. Furthermore, we have a set of commodities C which have a source c^{so} and a sink c^{si} in S and a transport

volume $c^\mu \in \mathbb{Q}$. We have a unit transport cost table d_{ij} with $i, j \in N$. If elements i, j from S and i_H, j_H from H represent the same physical points, then we assume that $d_{i_H j_H} < d_{i_H j}$ and $d_{i_H j_H} < d_{i j_H}$ (the so called α-discount which is usually defined by three constants to adjust the distance values for source-hub, hub-hub and hub-sink transports).

An admissible path for a commodity $c \in C$ is given by $c^{\mathrm{so}} \to l \to m \to c^{\mathrm{si}}$ where $l \neq m \in H$ or $c^{\mathrm{so}} \to l \to c^{\mathrm{si}}$ with $l \in H$. The choice of path is represented by the binary variable x_{clm}, $l \neq m \in H$ or x_{cl}, $l \in H$.

To every element l of H we can assign a hub level $q \in Q_l = \{1, \ldots, q_l\}$ of size ξ_{lq} and of cost k_{lq}. The size restricts the incoming flow from sources; the flow from other hubs is unbounded if at least one hub level is installed. The choice of hub level is encoded in the binary variable z_{lq}. The model is then:

$$\begin{aligned}\text{Min} \quad & \sum_{l \in H} \sum_{q \in Q_i} k_{lq} z_{lq} + \sum_{c \in C} \sum_{l \in H} c^\mu \left(d_{c^{\mathrm{so}} l} + d_{l c^{\mathrm{si}}} \right) x_{cl} \\ & + \sum_{c \in C} \sum_{l,m \in H}^{l \neq m} c^\mu \left(d_{c^{\mathrm{so}} l} + d_{lm} + d_{m c^{\mathrm{si}}} \right) x_{clm} \end{aligned} \tag{8}$$

$$\sum_{l \in H} x_{cl} + \sum_{l,m \in H}^{l \neq m} x_{clm} = 1 \qquad \forall c \in C \tag{9}$$

$$x_{cl} + \sum_{m \in H}^{m \neq l} x_{clm} + x_{cml} \leq \sum_{q \in Q_l} z_{lq} \qquad \forall l \in H, c \in C \tag{10}$$

$$\sum_{c \in C} c^\mu \left(x_{cl} + \sum_{m \in H} x_{clm} \right) \leq \sum_{q \in Q_l} \xi_{lq} z_{lq} \qquad \forall l \in H \tag{11}$$

$$\sum_{q \in Q_l} z_{lq} \leq 1 \qquad \forall l \in H \tag{12}$$

In (8) we sum up the hub level cost and the transport cost. The possible routes are restricted to one per commodity in (9). The equation (10) states that the intermediate nodes in a route have to be hubs (of any level) while (11) restricts the incoming flow from sources with respect to the chosen hub size. Furthermore, (12) states that at most one hub level is chosen.

We can now describe the model as ComPM specifying the 9-tuple

$$\eta = (N, P, C, L, a, C^L, C^{\mathrm{so}}, C^{\mathrm{si}}, F)$$

Here, N, P and C can be kept as defined before. We introduce two load types $L = \{\mu, \eta\}$ where μ represents the volume and η the restriction of the number of arcs on each route. We choose $a_{\mu p} = \mathrm{id}$ and $a_{\eta p}(z) = z - 1$ to

represent the consumption of arcs during the route. We set c^{μ} as described before, and set $c^{\eta} = 3$ so that each route may consist of at most three arcs. The values c^{so} and c^{si} were already discussed; for c^{si} we identify the element with one-element set.

For the cost calculation we need two sets of local cost functions. First, we define the transport cost:

$$\begin{aligned} \hat{f}_{ij} : \mathbb{Q}^{\{p_{ij}\}\times L} &\to \mathbb{Q} \\ (z_{\mu}, z_{\eta}) &\mapsto d_{ij} \cdot z_{\mu} \end{aligned} \tag{13}$$

For the hub cost, we use the local process sets $\hat{P}_j = \{p_{ij}, i \in N\}$ for $j \in H$. We define

$$\hat{f}_j(z) = \begin{cases} 0 & z_{i\mu} = 0 \quad \forall i \in N \\ k_{jq} & \xi_{jq-1} < \sum_{i\in S} z_{i\mu} \leq \xi_{jq} \\ 1000 \cdot k_{jq_j} & \sum_{i\in S} z_{i\mu} > \xi_{jq_j} \\ k_{j1} & \text{otherwise} \end{cases} \tag{14}$$

Here, we assume $\xi_{j0} = 0$ and use $z_{i\mu}$ for the vector component belonging to (p_{ij}, μ). The term $1000 \cdot k_{jq_j}$ is used as upper bound for the cost of a hub exceeding the maximal specified level. To make the problem formally equivalent to the MIP, we could have used ∞ and extend the possible cost values with that value, but it is easier to handle the problem with a finite upper bound. Now F is just defined as union of all defined $\hat{f}_{ij}$ and $\hat{f}_j$.

3.2 M2: Introducing Vehicle-based Costs

Instead of d_{ij}, we consider costs t_{ij} per truck and $\overline{d}_{ij}$ per volume, and define the truck size to be r_{ij} $(i, j \in N)$. We keep the binary variables z_{lq} for hub level choice and the route variables x_{clm}, $l \neq m \in H$ and x_{cl}, $l \in H$. Furthermore, we introduce the integer variable v_{ij} for the number of vehicles on each connection $i \to j$ contained in the arc set P.

$$\begin{aligned} \text{Min} \quad & \sum_{l\in H} \sum_{q\in Q_l} k_{lq} z_{lq} + \sum_{c\in C} \sum_{l\in H} c^{\mu} \left(\overline{d}_{c^{\mathrm{so}}l} + \overline{d}_{lc^{\mathrm{si}}}\right) x_{cl} \\ + & \sum_{c\in C} \sum_{l,m\in H}^{l\neq m} c^{\mu} \left(\overline{d}_{c^{\mathrm{so}}l} + \overline{d}_{lm} + \overline{d}_{mc^{\mathrm{si}}}\right) x_{clm} + \sum_{(i,j)\in P} t_{ij} v_{ij} \end{aligned} \tag{15}$$

$$\sum_{l \in H} x_{cl} + \sum_{l,m \in H}^{l \neq m} x_{clm} = 1 \qquad \forall c \in C \qquad (16)$$

$$x_{cl} + \sum_{m \in H}^{m \neq l} x_{clm} + x_{cml} \leq \sum_{q \in Q_l} z_{lq} \qquad \forall l \in H, c \in C \qquad (17)$$

$$\sum_{c \in C} c^{\mu} \left(x_{cl} + \sum_{m \in H} x_{clm} \right) \leq \sum_{q \in Q_l} \xi_{lq} z_{lq} \qquad \forall l \in H \qquad (18)$$

$$\sum_{q \in Q_l} z_{lq} \leq 1 \qquad \forall l \in H \qquad (19)$$

$$\sum_{c \in C, c^{\mathrm{so}} = i} c^{\mu} \left(x_{cl} + \sum_{m \in H} x_{clm} \right) \leq r_{il} \cdot v_{il} \qquad \forall i \in S, l \in H \qquad (20)$$

$$\sum_{c \in C} c^{\mu} x_{clm} \leq r_{lm} \cdot v_{lm} \qquad \forall l \neq m \in H \qquad (21)$$

$$\sum_{c \in C, c^{\mathrm{si}} = j} c^{\mu} \left(x_{cm} + \sum_{l \in H} x_{clm} \right) \leq r_{mj} \cdot v_{mj} \qquad \forall m \in H, j \in S \qquad (22)$$

In (15) we have the adjusted cost function with the extra cost term per vehicle. Equations (16), (17), (18) and (19) stay unchanged. In (20), (21), (22) we compute the total transport volume on the arc $i \to j$ and calculate the number of vehicles that is necessary for that volume.

We will now transform the change into the ComPM model. This is easy because we only change the calculation of the transport costs. The local cost functions $\hat{f}_{ij}$ are replaced by

$$\begin{aligned} \hat{f}_{ij} : \mathbb{Q}^{\{p_{ij}\} \times L} &\to \mathbb{Q} \\ (z_{\mu}, z_{\eta}) &\mapsto t_{ij} \cdot \left\lceil \frac{z_{\mu}}{r_{ij}} \right\rceil + \overline{d}_{ij} \cdot z_{\mu} \end{aligned} \qquad (23)$$

3.3 M3: Using a Time Expanded Network

To get to model M3, we introduce travel times τ_{ij} in days for every arc $i \to j$. We consider a week with five working days and derive a cyclic plan (as we are on the strategic planning level, we ignore the special problems regarding stops on weekends). The extended model has integer variables v_{ij}^{τ}, for $\tau \in W = \{\text{Mo, Tu, We, Th, Fr}\}$, where τ denotes the starting time of the vehicle. W is equipped with the canonical $\mathbb{Z}$-action, e.g. $\text{Mo} + 2 = \text{We}$, $\text{Mo} - 3 = \text{We}$. Furthermore, we introduce a maximal transport time $c^{\theta} \in \mathbb{N}$ and starting day $\tau_c \in W$ for each $c \in C$. These can be used for

preprocessing by setting x_{clm} to zero if $\tau_{c^{\text{so}}l} + \tau_{lm} + \tau_{mc^{\text{si}}} > c^\theta$ and x_{lc} to zero if $\tau_{c^{\text{so}}l} + \tau_{lc^{\text{si}}} > c^\theta$. The hub levels are independent of the weekday, the capacity is considered as capacity per day. The adjusted model is:

$$\begin{aligned}\text{Min} \quad & \sum_{l\in H}\sum_{q\in Q_i} k_{lq} z_{lq} + \sum_{c\in C}\sum_{l\in H} c^\mu \left(\overline{d}_{c^{\text{so}}l} + \overline{d}_{lc^{\text{si}}}\right) x_{cl} \\ & + \sum_{c\in C}\sum_{l,m\in H}^{l\neq m} c^\mu \left(\overline{d}_{c^{\text{so}}l} + \overline{d}_{lm} + \overline{d}_{mc^{\text{si}}}\right) x_{clm} + \sum_{(i,j)\in P}\sum_{\tau\in W} t_{ij} v_{ij}^\tau \qquad (24)\end{aligned}$$

$$\sum_{l\in H} x_{cl} + \sum_{l,m\in H}^{l\neq m} x_{clm} = 1 \qquad \forall c \in C \qquad (25)$$

$$x_{cl} + \sum_{m\in H}^{m\neq l} x_{clm} + x_{cml} \leq \sum_{q\in Q_l} z_{lq} \qquad \forall l \in H, c \in C \qquad (26)$$

$$\sum_{c\in C}^{\tau_c=\tau-\tau_{c^{\text{so}}l}} c^\mu \left(x_{cl} + \sum_{m\in H} x_{clm}\right) \leq \sum_{q\in Q_l} \xi_{lq} z_{lq} \qquad \forall l \in H, \tau \in W \qquad (27)$$

$$\sum_{q\in Q_l} z_{lq} \leq 1 \qquad \forall l \in H \qquad (28)$$

$$\sum_{c\in C, c^{\text{so}}=i}^{\tau_c=\tau} c^\mu \left(x_{cl} + \sum_{m\in H} x_{clm}\right) \leq r_{il} \cdot v_{il}^\tau \qquad \forall i \in S, l \in H, \tau \in W \qquad (29)$$

$$\sum_{c\in C}^{\tau_c=\tau-\tau_{c^{\text{so}}l}} c^\mu x_{clm} \leq r_{lm} \cdot v_{lm}^\tau \qquad \forall l \neq m \in H, \tau \in W \qquad (30)$$

$$\sum_{c\in C, c^{\text{si}}=j}^{\tau_c=\tau-\tau_{c^{\text{so}}l}} c^\mu x_{cm} + \sum_{l\in H}\sum_{c\in C, c^{\text{si}}=j}^{\tau_c=\tau-\tau_{c^{\text{so}}l}-\tau_{lm}} c^\mu x_{clm} \leq r_{mj} \cdot v_{mj}^\tau \qquad \forall m \in H, j \in S, \tau \in W \qquad (31)$$

The change in the equation only affects the addition of carefully chosen τ superscripts to always add the transport volume for the same day.

To model the changes in ComPM, we construct a cyclic, time-expanded graph $G^\theta = (N^\theta, P^\theta)$ where $N^\theta = N \times W$ and $(p_{ij}, \tau) \in P^\theta = P \times W$ connects (i, τ) with $(j, \tau + \tau_{ij})$, where we again consider W as cyclic. We write $c_\theta^{\text{so}} = (c^{\text{so}}, \tau_c)$ for the source and define the sink set as $c_\theta^{\text{si}} = \left\{(c^{\text{si}}, \tau) \,\middle|\, \tau \in W\right\}$ because the commodity is allowed to arrive at any weekday as long as the time restrictions are kept. The load set is supplemented by a further load type called θ representing the remaining time with the starting load c^θ. The change function a for the process (p_{ij}, τ) and the load θ is set to $a_{(p_{ij},\tau)\theta}(z) = z - \tau_{ij}$.

The cost functions are slightly changed, reflecting the tactical and strategic nature of the costs

$$\begin{aligned} \hat{f}_{ij}^{\tau} : \mathbb{Q}^{\{(p_{ij},\tau)\}\times L} &\to \mathbb{Q} \\ (z_\mu, z_\eta, z_\theta) &\mapsto t_{ij} \cdot \left\lceil \frac{z_\mu}{r_{ij}} \right\rceil + \overline{d}_{ij} \cdot z_\mu \end{aligned} \tag{32}$$

For the hub cost, we use the local process sets $\hat{P}_j = \{(p_{ij}, \tau), i \in N, \tau \in W\}$ for $j \in H$. We define

$$\hat{f}_j(z) = \begin{cases} 0 & z_{i\tau\mu} = 0 \; \forall i \in N \\ k_{jq} & \xi_{jq-1} < \max_{\tau \in W} \sum_{i \in S} z_{i\tau\mu} \leq \xi_{jq} \\ 1000 \cdot k_{jq_j} & \max_{\tau \in W} \sum_{i \in S} z_{i\tau\mu} > \xi_{jq_j} \\ k_{j1} & \text{otherwise} \end{cases} \tag{33}$$

where $z_{i\tau\mu}$ is the component for (p_{ij}, τ) and the load μ.

4 Generating Benchmark data

We split the benchmark data into one set for the algorithm tuning and one set for the actual test. All data sets will be provided upon request.

We start by looking at problem M1. We generate a set of 40 instances SGI_n^{M1} with $n = 10, \ldots, 49$ nodes (serving as sources/sinks and hubs each). These nodes are assigned random (equally distributed) coordinates in a 1×1 box. For $i \in S, j \in H$ or $i \in H, j \in S$ we define d_{ij} to be the Euclidean distance, for $i, j \in H$ we use half of the Euclidean distance. For each pair $(i, j) \in S \times S$, we generate a commodity c with c^μ drawn from a geometric distribution with expectation value 10. For each $j \in H$, we generate three numbers g_j^0, g_j^1 and g_j^2 from a geometric distribution with expectation value 1000 and draw three numbers ρ_j^0, ρ_j^1 and ρ_j^2 from a uniform distribution on the unit interval. We set $Q = \{0, 1, 2\}$ and $\xi_{j0} = g_j^0$, $\xi_{j1} = g_j^0 + g_j^1$, $\xi_{j2} = g_j^0 + g_j^1 + g_j^2$. Furthermore, we set $k_{j0} = g_j^0 \cdot \rho_j^0$, $k_{j1} = k_{j0} + g_j^1 \cdot \rho_j^1$ and $k_{j2} = k_{j1} + g_j^2 \cdot \rho_j^2$. SGI_n^{M1} is used for the algorithm tuning.

For the test, we use the AP data set [4]. We thank Ivan Contreras for providing us with the full data set for AP used in [10] for the problem M1. Note that the scaling of the hub costs is not done by the factor 0.9 as indicated in [10, p. 450] because they really used 0.98 for scaling as they confirmed by personal communication.

For M2, we have to set truck sizes and also split the costs into truck costs and linear costs. We set the vehicle size r_{ij} equal to $10 \cdot c_s^\mu$ where c_s^μ is the average commodity size (for all $i, j \in N$). Furthermore, we set $\overline{d}_{ij} = D_{ij}/10$

and $t_{ij} = 0.8 \cdot r_{ij} \cdot D_{ij}$. Here, D_{ij} always means the undiscounted cost, i.e. the Euclidean distance between the coordinates. For M3, we generate from every commodity one commodity per weekday, and set the size to 0, c^μ or $2c^\mu$ with equal probability, so that the average transport volume per day stays unchanged and the total number of non-zero commodities rises by a factor of 10/3. If $D = \max_{i \in N, j \in N} D_{ij}$, then we set $\tau_{ij} = 3$ if $D_{ij} > \frac{2}{3}D$, $\tau_{ij} = 2$ if $\frac{2}{3}D \geq D_{ij} > \frac{1}{3}D$ and $\tau_{ij} = 1$ otherwise ($i \in H, j \in N, i \neq j$). For $i \in S$, $j \in H$, we use $\tau_{ij} = 2$ if $D_{ij} > \frac{2}{3}D$, $\tau_{ij} = 1$ if $\frac{2}{3}D \geq D_{ij} > \frac{1}{3}D$ and $\tau_{ij} = 0$ otherwise. The maximal transport time for a commodity is set to $\tau_{c^{\text{so}} c^{\text{si}}} + 1$ or $\tau_{c^{\text{so}} c^{\text{si}}} + 2$ with equal probability.

5 Algorithm Tuning with `iRace`

Using the specifications of subsection 2.3, we implement a Simulated Annealing algorithm. This algorithm depends on the careful choice of several search parameters; some specify the generation of neighbours, some specify the properties of the algorithm (like the initial temperature, cooling scheme etc.). Often, such parameters are chosen by undocumented experiments or rules of thumb [1, 2] which is neither scientific nor does it fully use the potential of algorithm improvement.

As described in subsection 2.3, the neighbourhood generation scheme needs to choose a subset $\hat{C} = C \setminus \check{C}$ of the commodities, split it into C_r and C_d and assign new routes to them by iteratively finding shortest paths for the opportunity costs:

1. We choose $n_{\texttt{LC}}$ local cost functions from F at random, where $n_{\texttt{LC}}$ is drawn from a geometric distribution with expectation value `LC`. Then $\hat{C}$ is the subset of C consisting of all commodities whose routes in the solution influence one or more of the chosen cost functions. In this way we guarantee that the chosen commodities are related to each other.

2. We choose $n_{\texttt{RC}}$ commodities from $\hat{C}$ to form C_r. $n_{\texttt{RC}}$ is drawn from a geometric distribution with expectation value `RC`.

3. We follow the four steps described in subsection 2.3; for finding shortest paths we implement a Dijkstra-like algorithm with time limit.

The geometric distributions ensure that most of the neighbours are very similar to the original solutions while a few of them is totally different. `LC` and `RC` are parameters that should be chosen by experiment.

The Simulated Annealing algorithm itself depends on four search parameters:

Param.	Lower B.	Upper B.
LC	1	5
RC	1	5
IT	0	100
US	10	10000
RT	0	100
TH	100	10000

Table 1: Parameter ranges for the different search parameters.

1. *The initial temperature:* As the temperature is compared to the solution cost to determine the acceptance of solutions, its value depends on the scale used for the cost values; thus we choose our parameter scale-independent and scale it by a factor derived from a cost value: If S is the initial solution of the problem we define the initial temperature to be $f(S)/|C| \cdot \texttt{IT}$ where IT is a search parameter.

2. *Recovery temperature:* If the search was unsuccessful for US steps, then the temperature is reheated by $f(S)/|C| \cdot \texttt{RT}$.

3. *Cooling speed:* Cooling means multiplying by a fixed factor chosen to half the temperature in TH steps.

The six parameters are equipped with the initial ranges specified in table 1; they are chosen large enough to include all values that could be considered sensible. Hence we have now parametrized a six dimensional space of algorithms of which we want to find the best. Before we talk about statistical methods, we need to define what we mean by that.

First of all, we choose a subset $\mathcal{I}$ of all benchmark instances which should be representative. For that we choose the instances SGI for the models M1, M2 and M3. The instances AP are kept for testing because the gauging and testing instances should be unrelated. Now in theory, with infinite time, we can apply all algorithms (from a reasonable discretization of the six dimensional space) to all instances several times until we get an average cost value for each algorithm on each instance. Which algorithm is then "the best"?

Using the average or median cost value is useless as the instances come from different sources and are differently scaled; in any case, multiplying all cost functions with a constant value does not change the problem (or its difficulty) so that a proper measure of algorithm strength should be independent of scaling. We can define $A \geq B$ for algorithms A and B if A beats B on at least half of the instances (ignoring ties). Then we consider the transitive extension of this relation which makes sure that in any finite

set $\mathcal{A}$ of algorithms there exists at least one algorithm A with $A \geq B$ for all $B \in \mathcal{A}$. The notion should be treated with care because the transitive extension can lead to the situation that $A \geq B$ while A wins less instances than B (which in turn implies that $B \geq A$ and therefore A and B are treated as equally good algorithms). Particularly, the notion $A > B$ should be avoided. Other scale-independent measures could be achieved by applying logarithms to the cost values, but will not explore this possibility here.

`iRace` is a powerful tool to set search parameters of heuristic and exact algorithms [6, 5]. In every iteration, a finite set of algorithms $\mathcal{A}$ is chosen by random parameter choice from the given space. An iteration consists of several steps: In each step, all algorithms from $\mathcal{A}$ are applied to one instance and the results are ranked from 1 to $|\mathcal{A}|$. In this way, we get a growing table of algorithm rankings. We set the hypothesis that all algorithms are equally good; this hypothesis can be tested with the non-parametric Friedman test on the ranking table. If the hypothesis is rejected, we eliminate one or more algorithms by pairwise comparison with the leading algorithm. In this way, the number of algorithms in the race is (possibly) reduced in every step, leaving more computation time for the prospectively good candidates. After the set computing time for the iteration is used up, the leftover algorithms are used to sharpen the probability distribution on the space of algorithm. We pursue by choosing a new set $\mathcal{A}$ from the new distribution. After several iteration, we are left with a set of good parameter vectors.

We use the parameter ranges specified in the table to apply the `R` implementation the `iRace` (`http://iridia.ulb.ac.be/irace/`) to a `C#` implementation of our Simulated Annealing algorithm. We chose to use 1000 experiments of 15 minutes each (about 10 days in total), split into four iterations, using the standard values for all `iRace` parameters.

The first iteration used 41 randomly generated parameter vectors and 6 randomly chosen instances (of our pool SGI of 120), from which 3 parameter vectors survived. The second iteration produced 34 additional parameter vectors (from the adjusted probability distribution) and used 17 instances. 5 parameter vectors survived. The third iteration sampled 27 additional parameter vectors and applied them to 16 instances, having 7 surviving parameter vectors. The last iteration added 22 parameter vectors and 13 instances. The resulting four parameter vectors are shown in table 2. In total 124 parameter vectors were tested on 52 instances.

6 Numerical Results

We compare our Simulated Annealing approach with two commercial solvers: `LocalSolver 4.0`, which is a highly optimized neighbourhood search implementation and `CPLEX 12.2` (via `GAMS`). For the Simulated Annealing

position	LC	RC	IT	US	RT	TH
1	2	4	0.5601	3044	65.5877	7614
2	2	4	0.7807	6218	56.5317	8235
3	2	4	3.1608	5361	49.6615	7967
4	2	4	2.6122	4185	62.5820	9959

Table 2: The surviving four parameter vectors.

approach, we ran each of the four identified best parameter settings (table 2) for 15 minutes and report the best result. The other two solvers are run for 60 minutes. We used a 3.4 GHz computer with 16GB RAM.

For M1, we compare our results to the optimal results from [10] (table 3). We see that the heuristic is often far way from the optimal results for this problem.

For M2, we compare the results of all three solvers (table 4). `LocalSolver` reached a local optimum quickly and gave poor results, even after we reformulated some linear constraints into additional costs and logic constraints (as recommended by the solver description). We, therefore, did not use it for further tests. `CPLEX` solved all problems up to 25 nodes to near optimality, but did not find feasible solutions for 40 nodes. From 50 nodes on, our 16GB RAM were not enough to generate the model. The heuristic gives solutions about 10% of optimality for 10 and 25 nodes but performs less well for 20 nodes. For larger instances, we lack comparison values, but as all AP instances use the same total flow and and total distance, a rough comparison is possible and values in the range of 100,000 to 200,000 provide reasonable solutions.

For M3, we compare our heuristic to `CPLEX` (table 5). Comparing the matching instances from M2 and M3, the number of commodities is roughly multiplied by 3, the number of nodes is multiplied by 5, but as many routes are eliminated for time restrictions, the problem is not necessarily more difficult. We see that `CPLEX` is able to solve instances up to 25 nodes (still), but produces rather large gaps. Still `CPLEX` wins most of the instances up to 25 nodes, but cannot find feasible solutions for larger instances.

At last, we want to compare the chosen hubs and their levels for the different instances and models (table 6). Because the quality of the solutions of the large problems is unreliable, we only look at the instances up to 50 nodes. We see that there are significant differences. In M2, we see the effect of stronger bundling, leading to fewer hubs especially in the small instances. For M3, the time restrictions have an additional effect, because large detours for shipments are no longer possible.

Inst	Heuristic	Optimal	Inst	Heuristic	Optimal
10LL	234,150	215,560	75LL	317,012	234,659
10LT	267,225	240,965	75LT	403,903	252,554
10TL	328,630	251,642	75TL	422,995	292,768
10TT	264,838	254,595	75TT	576,206	333,946
20LL	259,534	228,753	90LL	311,741	
20LT	290,252	245,970	90LT	416,884	
20TL	329,671	264,140	90TL	397,244	
20TT	327,407	286,050	90TT	557,476	
25LL	319,745	233,103	100LL	585,966	240,526
25LT	356,400	263,767	100LT	625,913	248,955
25TL	369,184	304,057	100TL	650,233	347,446
25TT	413,012	333,237	100TT	720,032	447,035
40LL	379,413	237,898	125LL	391,596	234,781
40LT	427,271	261,552	125LT	446,919	248,258
40TL	396,013	296,194	125TL	304,717	241,107
40TT	558,996	339,667	125TT	522,505	281,780
50LL	537,651	233,892	150LL	549,019	230,016
50LT	579,340	263,945	150LT	402,988	244,448
50TL	743,397	312,374	150TL	429,546	254,975
50TT	778,357	390,817	150TT	560,997	303,594
60LL	343,866		175LL	343,771	224,340
60LT	464,845		175LT	416,079	242,931
60TL	422,489		175TL	467,819	240,497
60TT	521,732		175TT	375,020	302,206
70LL	403,793		200LL	445,918	226,416
70LT	374,920		200LT	531,110	
70TL	380,025		200TL	407,006	266,785
70TT	535,973		200TT	498,999	284,362

Table 3: Results for M1 (best of four 15 minute runs); all values rounded to integers. The optimal values are precise up to ± 1 and are taken from own calculations (for 10LL to 50LL) and from [10] (for multiples of 25).

Inst	Heuristic	LocalSolv.	CPLEX	CPL-LB	Inst	Heuristic
10LL	74,241	151,075	71,301	71,301	75LL	119,334
10LT	94,783	134,713	88,727	88,727	75LT	188,667
10TL	78,113	200,195	71,955	71,955	75TL	115,589
10TT	100,080	181,566	92,592	92,592	75TT	219,772
20LL	125,776	387,149	70,156	70,156	90LL	86,942
20LT	132,465	409,779	96,021	96,021	90LT	228,647
20TL	103,092	668,420	73,177	73,177	90TL	128,554
20TT	142,147	1,857,775	108,041	108,041	90TT	244,261
25LL	90,701	442,310	78,575	78,575	100LL	148,355
25LT	123,445	1,636,001	111,089	106,850	100LT	237,057
25TL	113,338	1,202,972	106,651	104,433	100TL	178,363
25TT	149,786		142,906	141,748	100TT	293,992
40LL	130,418				125LL	97,740
40LT	174,614				125LT	220,979
40TL	145,964				125TL	86,512
40TT	257,975				125TT	227,656
50LL	146,552				150LL	157,960
50LT	244,640				150LT	154,165
50TL	208,883				150TL	124,621
50TT	381,171				150TT	220,836
60LL	94,932				175LL	90,593
60LT	237,839				175LT	178,612
60TL	138,935				175TL	110,056
60TT	265,581				175TT	147,305
70LL	110,519				200LL	120,272
70LT	169,558				200LT	202,822
70TL	133,681				200TL	132,912
70TT	253,739				200TT	203,242

Table 4: Results for M2, one hour time limit; all values rounded to integers. The comparison values are from `CPLEX 12.2` and `LocalSolver 4.0`. Empty cells mean no solution in the given time limit.

Inst	Heuristic	CPLEX	CPL-LB	Inst	Heuristic
10LL	334,282	284,986	270,316	75LL	337,645
10LT	352,000	330,154	306,479	75LT	188,667
10TL	361,088	342,134	324,146	75TL	487,502
10TT	379,273	348,828	338,419	75TT	565,916
20LL	340,774	269,606	221,746	90LL	323,179
20LT	365,502	252,885	224,780	90LT	437,595
20TL	348,865	471,135	269,253	90TL	350,618
20TT	400,458	317,815	256,690	90TT	492,722
25LL	304,590	295,530	214,531	100LL	383,964
25LT	394,267	403,544	240,668	100LT	440,503
25TL	378,332	324,041	284,086	100TL	466,314
25TT	455,957			100TT	654,296
40LL	365,733			125LL	278,339
40LT	442,277			125LT	461,876
40TL	447,210			125TL	360,984
40TT	556,524			125TT	481,604
50LL	427,693			150LL	310,021
50LT	513,881			150LT	340,622
50TL	516,870			150TL	351,093
50TT	757,945			150TT	441,221
60LL	291,847			175LL	315,373
60LT	506,956			175LT	368,062
60TL	380,352			175TL	349,843
60TT	637,320			175TT	507,613
70LL	382,061			200LL	282,732
70LT	424,092			200LT	365,807
70TL	421,399			200TL	351,490
70TT	673,625			200TT	445,863

Table 5: Results for M3, one hour time limit; all values rounded to integers. The comparison values are from `CPLEX 12.2`. Empty cells mean no solution in the given time limit.

Inst	M1	M2	M3
10LL	1_1 4_1 7_3	5_5	1_1 4_3 7_3
10LT	1_1 4_2 5_2 10_4	5_5 6_4	1_1 4_3 5_4 6_3 9_1
10TL	4_1 5_1 10_2	5_5	1_1 4_1 5_4 6_1 9_1
10TT	4_1 5_3 10_4	4_4 5_5	1_1 4_4 5_5 9_4
20LL	7_2 14_4	10_5	6_2 11_3 14_4
20LT	6_5 8_4 14_5	10_5 14_5	1_4 10_5 14_5
20TL	7_2 19_4	10_5	1_1 8_2 9_4 11_1 17_1
20TT	1_2 10_5 19_5	5_5 10_5	1_2 8_1 10_5 19_5
25LL	8_4 18_5	17_5	6_1 7_3 18_5
25LT	9_5 12_5 19_5	9_4 12_5 14_5	6_2 8_4 12_5 14_5 19_5
25TL	9_3 23_4	11_4 14_5	5_1 6_2 14_5
25TT	12_5 14_5 24_5	10_5 14_5 21_5	5_1 6_4 9_5 10_1 14_5 21_1
40LL	14_4 29_5	1_5 14_5 22_2	1_1 14_5 19_1 22_5 33_3
40LT	14_5 26_5 30_5	1_5 6_5 14_4 19_4 22_1	1_1 6_5 14_5 19_5 22_5 24_1 33_2 35_2
40TL	14_2 19_5	1_5 14_5 22_3	1_1 6_1 14_5 19_1 22_4 33_2
40TT	14_5 25_5 38_5	1_5 6_5 14_5 22_5 24_5 33_1	1_1 6_3 14_5 19_5 22_4 24_4 33_5
50LL	15_4 35_5	20_5 27_5 41_5	1_1 9_1 20_5 21_5 27_5 41_1
50LT	6_4 26_5 32_4 48_5	1_5 17_5 20_5 21_5 27_5 41_5 48_5	1_1 9_1 17_5 20_5 21_5 26_4 27_5 41_3 48_5
50TL	3_2 24_5	1_5 20_4 41_5	1_1 9_1 20_5 27_5 41_1
50TT	6_5 26_5 41_4 48_5	1_5 2_5 9_5 20_5 21_5 27_5 41_4	1_1 3_5 9_1 17_5 20_5 21_5 22_5 25_5 27_5 41_3

Table 6: Chosen hubs and sizes (in numbering from $1, \ldots, |H|$ and $1, \ldots, |Q|$) for the best solutions known for each instance and model.

7 Conclusion and Outlook

Following table 6 we see that the transport cost structure has severe influences onto the network structure. Models as M2 and M3 offer more realistic solutions, but are much harder to solve.

The Simulated Annealing algorithm produces feasible, usable results even for large instances. Nevertheless, there is a long way to go to achieve (near) optimality, because the algorithm is often stuck in a small area of the search space. To explore the strength of local descent in a large search space, we can combine it with a genetic algorithm which chooses a restricted set H of possible hubs for each individual. We hope to improve our primal solutions in this way. Furthermore, we can find lower bounds for larger problems by using a Benders decomposition approach by relaxing the integrality for the x variables in of M2 and M3. This will also be explored in the future.

An important goal is to extend the model by stochastic influences, especially stochastic shipping volumes. By introducing a load type *variance* we can model normally distributed shipping volumes and use non-linear cost functions on them. To compare this approach with other approaches, we will use event-driven simulation.

References

[1] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: Sampling design and iterative refinement. In *Hybrid Metaheuristics*, pages 108–122. Springer, 2007.

[2] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. Resende, and W. R. Stewart Jr. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.

[3] M. N. Baumung and H. I. Gündüz. Consolidation of less-than-truckload shipments in long-haul transportation networks. In *8th Workshop on Logistics and SCM*, pages 32–33. Deutsche Post Lehrstuhl für Optimierung von Distributionsnetzwerken RWTH Aachen, 2013.

[4] J. E. Beasley. OR library, 2012.

[5] M. Birattari. *Tuning metaheuristics: a machine learning perspective*, volume 197. Springer, 2009.

[6] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-race and iterated F-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer, 2010.

[7] J. F. Campbell. Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72(2):387–405, 1994.

[8] J. F. Campbell. Hub location and the p-hub median problem. *Operations Research*, 44(6):923–935, 1996.

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

[10] I. Contreras, J.-F. Cordeau, and G. Laporte. Exact solution of large-scale hub location problems with multiple capacity levels. *Transportation Science*, 46(4), 2012.

[11] C. B. Cunha and M. R. Silva. A genetic algorithm for the problem of configuring a hub-and-spoke network for a ltl trucking company in brazil. *European Journal of Operational Research*, 179(3):747–758, 2007.

[12] R. S. de Camargo, G. de Miranda Jr, and H. P. L. Luna. Benders decomposition for hub location problems with economies of scale. *Transportation Science*, 43(1):86–97, 2009.

[13] B. Fleischmann. Transport- und Tourenplanung. In D. Arnold, H. Isermann, A. Kuhn, H. Tempelmeier, and K. Furmans, editors, *Handbuch Logistik*, volume 3, pages 137–152. Springer, 2008.

[14] K. Haase and M. Hoppe. Transportnetzgestaltung für Paketdienstleister. *Zeitschrift für Betriebswirtschaft*, 78(9):857–874, 2008.

[15] 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.

[16] J. Sender and U. Clausen. Hub location problems with choice of different hub capacities and vehicle types. In *Network Optimization*, pages 535–546. Springer, 2011.

[17] G. Song. *Integer programming models for airline network design problems*. PhD thesis, University of Texas at Austin, 1996.

[18] R. van de Leensel. *Models and Algorithms for Telecommunication Network Design*. PhD thesis, Maastricht University, 1999.