LOUVAIN
School of Management

UNIVERSITE CATHOLIQUE DE LOUVAIN

LOUVAIN SCHOOL OF MANAGEMENT

An R script for single allocation hub location problems.

Supervisor : Agrell Per Joakim

Research Master Thesis submitted by
Berouayel Aladin

With a view of getting the degree
Master in Management (in Business Engineering)

ACADEMIC YEAR 2015-2016

Firstly and most importantly, I want to thank my supervisor, Per Joakim Agrell, for his advice, availability and support during the whole process that led to this thesis.

I also want to thank Alisson Moureau, Arnaud Havet and Sarah Wendt that all helped in one way or another during the writing of this thesis.

Finally, I thank my parents for their indefectible support during my formation in management engineering.

Table of contents

1. Introduction

1.1. Motivation

Logistic networks aim at transporting goods or passengers from their origin to their expected destination. When designing such networks, the most natural way to proceed is to use direct connections[1] between all the nodes[2]. In such a network, the goods are directly transported from their origin to their destination. It is true that using direct connections minimizes the travel distance as goods are transported on the shortest path possible, but the network it leads to can be suboptimal. As the number of nodes that needs to be connected increases, there can be a huge amount of connections in between them. Indeed, for a network with n nodes and with all the nodes connected to each other, the number of connections is $\frac{n(n-1)}{2}$ (Campbell and Krishnamoorthy, 2005, p 1540) so as n grows, the number of connections grows as $O(n^2)$[3]. This can have some downsides: firstly, if each connection is required to pay a fixed[4] cost to be allowed, a network with direct connections between all the nodes can be very costly. Secondly, as the total amounts of goods have to be transported on a huge number of connections, the quantity transported on each one of them can be very small. This can increase the cost of transportation per unit. In this situation, the minimization of the travel distance can be outbalanced by the increase of the transportation costs per unit of distance (Baumgartner, 2003, p 1).

Hub-and-spoke networks have been designed to compensate for these downsides. A hub-and-spoke network is a network in which nodes have been chosen to be used as transshipment and sorting points to help the transportation of goods or passengers between origin and destination pairs. In this setup, what has to be shipped between nodes i and j will not be directly transported between them. It will first be transported from i to a hub (or several

[1] Two nodes are said connected if there is something that is transported on the network between them that does not transit by another node on its way from the first to the second node.
[2] A node is a location that is the origin and/or the destination of a flow and/or that is a location where a hub can be located.
[3] For a definition of the order of a function, see (MIT, n.a.).
[4] A fixed cost is a cost that is not dependent on the flow.

hubs) and then from the final hub to j instead. A hub-and-spoke network with the same number of origins and destinations as a network with direct connections will have less connections. For a network with n points of which p (p<n) have been chosen to be hubs and where the hubs are all connected to each other, there are $\frac{p(p-1)}{2}$ connections between hubs (Campbell and Krishnamoorthy, 2005, p 1540) and n-p connections between non-hub nodes and hubs if we assume that each non-hub node is connected to a single hub. Figure 1 represents how a network with six nodes can be designed as a network with direct connections or as a hub-and-spoke network with two hubs and a single allocation of the non-hub nodes to the hubs (the black dots represent the nodes, the grey dots represent the hubs, the black lines represent the connections between a non-hub node and a hub and the grey line represents an inter-hub[1] connection). We can see that the number of connections diminishes drastically if hubs are located on the network (from 15 connections to 5 in this case).

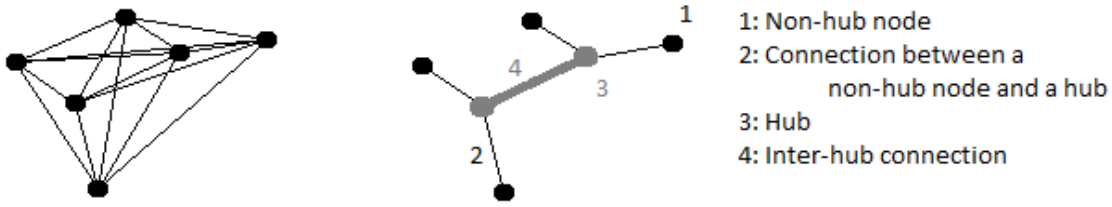

Figure 1: 6 nodes network with all the nodes connected to each other (left) or designed as a hub-and-spoke network with two hubs (right)

Therefore, if there is a considerable fixed cost to allow direct transportation between two nodes, this reduction of the number of links will reduce the total cost of the network. Moreover, the reduction of the number of connections leads to an aggregation of the flows: all the flows that have the same destination and the same origin will be aggregated on a single connection[2]. All the flows that are needed to be transported on the network will also be aggregated on the inter-hub connections. These aggregations can lead to a reduction of the unit flow cost[3] (for example, an airline company will more likely be able to use aircrafts loaded at full capacity). By locating hubs, it is also possible to take advantage of concentration of the sorting of the flows at the hubs instead of each node of the network (Bryan and O'Kelly, 1999, p 276, 277). With direct connections, the flows originating at each node are needed to be

[1] It is a connection between two hubs.
[2] For a single allocation of the non-hub nodes to the hubs.
[3] It is the cost to transport one unit of flow per unit of distance.

sorted at that node and directed to the correct destination. When hubs are used, sorting at the origin of the flows only needs to be done if the origin is connected to more than one hub. Instead, it needs to be done at each hub where the flows originating and destined at each allocated nodes need to be sorted.

The Hub-and-spoke architecture has been used to design networks in several fields. For example, most airlines have their proposed flights centered on a hub airport (or several hub airports). It allows them to propose more frequent flights and more destinations with the same fleet and to regroup passengers into bigger (and less costly) planes. Hub-and-spoke networks have also been widely used in the telecommunication industry. It is used for small-scale networks (the router in a local network) as well as large-scale (the internet). It is used because the installation of wires from each node to each node is costly in comparison to the transportation costs. Hubs are also used in postal delivery networks and many other fields (Baumgartner, 2003, pp 3-5).

In order to get the most out of the advantages of the hub-and-spoke networks, models that can help with the decisions regarding the hub locations and the non-hub nodes allocation to the hubs have been developed. This thesis proposes an R script that is an implementation of some of these models. R is an object-oriented programming language that is similar to the S language and environment. It is a free software under the GNU General Public License terms of the Free Software Foundation (The R foundation, n.a. (a)). That guarantees four freedoms to the R users. It allows the user to run the program for any utilization, to study its functioning and to adapt it to its needs, to dispense copies to anyone, and to improve it (and share these improvements with the community). In order to guarantee these four freedoms the source code of anything that is GPL licensed is freely accessible and modifiable (Free Software Foundation Inc, 2016). It has initially been developed as a data analysis and data visualization software for statisticians. Therefore, it provides a wide variety of functions for data manipulation, graphical display and calculation (The R foundation, n.a. (a)). It is supported by a community of users and developers that helps maintaining the language. As the R language is easily extensible, this community is also responsible for its permanent extensions through the implementation of packages. Packages are user-created sets of functions that extend the functionalities of R. These packages can be found, freely, on the CRAN website (The R

foundation, n.a. (b)). It is a network of web servers that store R code and documentation submitted by R users to share as authorized by the General Public License. The extensions provided by the R community has now made R adequate for other things than statistical analysis as the development of packages in several other fields has been started and is expending in finance, genomics and several other fields (Microsoft, n.a.).

This thesis is an attempt at contributing to this extension of the R language. To our best knowledge, there does not exist any package that focuses specifically on the hub location problems. After validation, the R script for the single allocation hub location problems can be embodied – possibly together with other network optimization functions– in a package and can be shared on the CRAN website. This will provide any person that wants to solve single allocation hub location problems the possibility to use the R language without having to write the algorithms themselves. The fact that the contribution of this thesis is an R package, will provide a free open-source alternative (as guaranteed by the GNU General Public License) to the existing optimization applications which is already designed to solve hub location problems. Because of the advantages of the hub-and-spoke network design, its modelization is studied in logistic formations and is used by small as well as big companies. Not in all these circumstances does the person that is involved have the willingness or the means to pay for IT implementations that optimize hub-and-spoke networks. It is therefore important for them to be able to find a free implementation of the problem, which is proposed by this thesis. This contribution is certainly not as powerful as the existing paying optimization software but it can be a starting point for future improvements from other contributors to the extension of the R language or for personal use as guaranteed by the GNU General Public License of R.

1.2. Research question

The research question of this thesis is an R script that solves to optimality hub location problems (Appendix 1). It is focused on the resolution of 4 major single allocation hub location problems[1]. These four problems are:

- The single allocation p-hub median problem

[1] The specificities of these hub location problems are explained in chapter 2.

- The single allocation p-hub center problem
- The single allocation hub covering problem
- The single allocation p-hub maximal covering problem.

The algorithms that solve these hub location problems all assume that the hubs can be located on any demand node, that all the hubs are connected and that there are no connections between non-hub nodes (Alumur and Kara, 2008, pp 1. 2). There are no special assumptions on the data of the problems they can solve except that the parameters of the problem to solve are required to be deterministic. Only the discount factor on the inter-hub connections are considered by the implementation.

The script allows the consideration of network characteristics and costs that are considered in the literature[1] about the hub location models like the hub capacities for example. It is also possible for the user not to enter the distance matrix required as argument by the script but rather to have it automatically searched for on the internet. In that case, the user just has to provide the names of the locations of the nodes.

The output of these models (if a feasible solution exists) is the optimal objective function value, and the matrix with the node allocation to the hubs (the hub locations can be found on the diagonal of this matrix[2]). As this matrix is not very visual, one special feature of the script is that it gives the user the possibility to display the optimal solution on a map (see 3.3.3.).

1.3. Limitations

There is an enormous amount of hub location problems considered in the literature. This thesis could not cover them all. We made the choice to only implement the hub location problems mentioned in the previous subsection as they are part of the basis of the hub location literature. The other important part of this basis, the multiple allocation hub location problems and the hub locations problems with fixed costs, are not covered. The hub location

[1] These characteristics and costs and the papers that consider them are discussed in the second chapter.
[2] It is the diagonal from the top left corner to the bottom right corner.

problems that consider specific aspects are not considered either. These problems include but are not limited to the stochastic models, the multi-periods models, the intermodal models or the models with competition between several firms[1]. The planar hub location problems are also out of scope of this thesis.

The other principal limitation of the implementation of this thesis is that it does not provide a heuristic algorithm[2] in case the problem exceeds the limitation of the arguments. The time and memory needed in order to solve to optimality grows rapidly with the augmentation of the number of nodes of the problem to solve. Therefore, for problems that can't be solved in a reasonable amount of time, it is recommended to use heuristic algorithms to find a sub-optimal solution faster which is not possible in the current state of the script.

1.4. Outline

The rest of this thesis is structured as follows:

- The **second chapter** ("Literature review") is a review of the literature that discuss hub location models. It is mainly focused on the models that have the same assumptions than the models implemented in this thesis. They have been extensively discussed and are the basis of the conception of many hub location models.

- The **third chapter** ("R implementation") focuses on the R script itself. It provides information about the choices that have been made regarding the implementation of the single allocation hub location models, the packages required or the different ways it displays the final solution to the user. It also explains further the different steps followed by the algorithms that solve these problems.

[1] Examples of papers that discuss these problems and their associated models can be found in subsection 2.5.
[2] ''A heuristic algorithm is one that is designed to solve a problem in a faster and more efficient fashion than traditional methods by sacrificing optimality'' (Kenny et al., 2014)

- The **fourth chapter** (“Computational Analysis”) focuses on the performance of the R functions. It also assesses the correctness of the R implementation by comparing the optimal solutions it calculated with optimal solutions of the same problems provided by the literature. For arguments of the functions which are not considered in the literature, it also analyses the effect of changes in the input arguments on two important performance indicators: the time and the maximum memory required by the different R functions to solve instances of hub location problems. They are critical limits of the script as the maximum memory used cannot exceed the memory available and will prohibit the resolution of some hub location problems. The speed at which a problem can be optimized will limit the number of different problems and networks that can be investigated in a predefined period of time.

- **Chapter five** (“Conclusion”) concludes this thesis by a critique about it and by proposing improvements and modifications that could be undertaken in future work on the subject. It also summarizes the theoretical and managerial contributions of the thesis.

2. Literature review

"Perhaps, Goldman (1969) is the first paper addressing the network hub location problem" (Alumur and Kara, 2008, p 2). However, the analytical research on the subject started when O'Kelly presented the first mathematical formulation of the problem in 1987 (Bryan and O'Kelly, 1999, p 1). Since then, many models, heuristics and algorithms have been developed.

This chapter of the thesis is a review of the models that have been formulated on the subject. It will have no interest in the algorithms and heuristics that solve them. As the number of papers formulating new models is too big in order to treat them all in this review, limited in its size, it will be focused on a subset of these models. Firstly, it is focused on the discrete[1] hub location models. This is because "the planar version of the hub location problem is underappreciated in the literature compared to the discrete version. There are only a few studies on the PHLP"[2] (Damgacioglu et al., 2015, p 225) and the implementations of this thesis is focused on discrete models. Most of the models reviewed here will be general models with no special assumptions on the parameters. These models are the foundation of the analytical research in the field and most of the existing models are a modified version of this basis to allow for the consideration of particular characteristics in the optimization (stochastic parameters, congestion into the hubs or onto the spokes, multimodal networks ...)[3]. In the literature and in this review, the discrete hub location models are classified in 4 categories[4]: the p-hub median problems, the hub location problems with fixed costs, the p-hub center problems and the hub covering problems (that include the p-hub maximal covering problems). We will review them in this order.

[1] In discrete hub location models, the hubs can only be located on a finite number of locations.
[2] PHLP: Planar Hub Location Models. In planar hub location models, the hubs can be located anywhere on the plane.
[3] For examples of papers that discuss these models, see 2.5.
[4] In Alumur and Kara (2008), for example.

Campbell (1994) says about discrete hub location problems:

> "Hub location problems can be viewed as embedded in an undirected network N = (V, A), where the set of nodes, or vertices, of the network V= {v_1, v_2, ..., v_q} correspond to origins/destinations and potential hub locations. Thus, hubs are restricted to be located at a subset of the vertices. Associated with link (a, b) ϵ A, which connects vertices v_a and v_b, is a non-negative weight d(a, b) = d(b, a) representing its length. This may correspond to travel distance, time, cost or some other attribute. Define C_{ab} to be the length of the shortest path between nodes a and b. The cost for movement on the path from origin i to destination j via hubs at nodes k and m, in that order, is $C_{ik} + \alpha C_{km} + C_{mj}$, where α is the discount factor for the inter-hub transportation. If k = m, then there is no inter-hub transportation. Associated with each o-d pair (i, j) is a non-negative weight representing the flow from i to j" (Campbell, 1994, p 388).

The optimization problems of locating hubs and allocating non-hub nodes to the hubs on such a network are proved to be NP-Hard[1]. For proofs on the problem's complexity, see Kara and Tansel (2000) (Ernst et al., 2009) for the single allocation p-hub center problem (multiple allocation) and Kara and Tansel (2003) for the single allocation hub covering problems, for example. Moreover, the allocation sub-problem of allocating the non-hub nodes to the hubs already located is also proven NP-Hard with some exceptions.

Alumur and Kara (2008) state that almost all the models, and therefore the models considered in this review, assume three things:

- *The hub network is complete*[2]
- *Economies of scale occur on the transportation between two hubs*
- *No direct connections between non-hub nodes are allowed* (Alumur and Kara, 2008, pp 1, 2).

For the models discussed in this literature review, these assumptions are supposed satisfied unless otherwise specified.

In the rest of this literature review, the indexes i and j are for nodes and k and m are for potential hubs. The names we will use for the models in the following will use a precise syntax

[1] "A problem (a language) is said to NP-hard if every problem in NP can be poly time reduced to it." (University of Pennsylvania, n.a., p 25) and "NP means verifiable in polynomial time" (University of Pennsylvania, n.a., p 11).

[2] i.e. every hub in the network is connected to every other hub.

adapted from the literature[1]. They are made of 4 different parts. The first letter tells whether the model is:

- capacitated (C-)
- uncapacitated (U-)
- a bi-criteria model (B-).

The following two letters separate:

- the single allocation models (-SA-)
- the multiple allocation models (-MA-).

The part of the name following these letters differentiates the 4 categories of models considered:

- -pHMP- for p-hub median problems
- -HLP- for the hub location problems with fixed costs
- -pHCP- for the p-hub center problems
- -SCP- for the hub covering problems (-pHMCV- stands for p-hub maximal covering problems)[2].

Finally, the letter(s) after the hyphen characterize the different models that would have the same name otherwise[3]. For example, the first model discussed in this review that has been proposed in O'Kelly (1987) is named USApHMP-Q, this means that it is an Uncapacitated Single Allocation P-Hub Median Problem. The Q stands for quadratic.

2.1. p-hub median models

The objective of the p-hub median models is to locate the hubs and allocate the non-hub nodes to the hubs in order to minimize the total cost of the network. The first model of a hub-and-spoke network with no restriction on the number of hubs ever formulated is a p-hub median in O'Kelly (1987). The paper formulates the problem (USApHMP-Q):

[1] It is close to the nomenclature of Ernst et al. (2009), for example.
[2] When the first letter is a lower case 'p', it means that the number of hubs to locate is exogenous. It is endogenous otherwise.
[3] The letters chosen for the name of each model are the letters used in the paper that presented the model, if the paper named it.

$$\min \sum_i \sum_j W_{ij}\left(\sum_k Z_{ik}C_{ik} + \sum_m Z_{jm}C_{jm} + \alpha \sum_k \sum_m Z_{ik}Z_{jm}C_{km}\right)$$

s.t.

$$(n-p+1)Z_{kk} - \sum_i Z_{ik} \geq 0 \quad for\ all\ k \quad (1)$$

$$\sum_k Z_{ik} = 1\ for\ all\ i \quad (2)$$

$$\sum_k Z_{kk} = p \quad (3)$$

$$Z_{ik} \in \{0,1\} \quad for\ all\ i,k \quad (4)$$

with binary variables Z_{ik}. The paper defines: Z_{ik} is equal to one if and only if node i is allocated to hub k and zero otherwise, Z_{kk} is equal to one if and only if k is a hub (O'Kelly, 1987, p 394). The paper defines the parameters of the model as: "W_{ij} is the number of units of flow between nodes i and j (exogenous), W_{ii}=0 by assumption, C_{ij} is the transportation cost of a unit of flow between i and j, C_{ii}=0 by assumption, p is the total number of hubs to be constructed (exogenous) and n is the total number of cities to be interconnected (exogenous)" (O'Kelly, 1987, p 394). "These inter-hub costs are multiplied by a parameter $\alpha \leq 1$ to reflect the scale effects in interfacility flows" (O'Kelly, 1987, p 394). "The inter-hub discount factor plays an important role in determining both the hub locations and the number of hubs to which each demand point is allocated [...]. As α decreases, hubs tend to spread farther apart and the number of spokes decreases [...]. For large α, hub interactions are expensive and hubs are drawn together to reduce the significant inter-hub transportation costs." (Campbell, 1994, p 391). O'Kelly (1987) also reformulates the objective function by defining O_i, the total amount of flow originating at node i and D_i, the total amount of flow that have node i as its destination. The objective function can then be formulated as:

$$\min\ \sum_i \sum_k Z_{ik}C_{ik}(O_i + D_i) + \sum_j \sum_m Z_{jm} \sum_i \sum_k Z_{ik}(\alpha W_{ij}C_{km})$$

This model is difficult to solve because both objective functions are quadratic. (O'Kelly, 1987, pp 394, 395). Therefore, several improvements and linearizations have been proposed in the literature.

2.1.1. Single allocation

The first linear formulation of the single allocation version of the p-hub median problem was introduced in Campbell (1994) with n^4+n^2+n variables and $3n^2+1$ linear constraints. The paper defines X_{ijkm} as the fraction of the flow from location i to location j that is routed through the hubs in k and m. Y_k is the binary variable that is equal to one if a hub is

opened in k and $C_{ijkm}=c_{ik}+\alpha c_{km}+c_{jm}$ is the cost to route one unit of flow from i to j by the hubs in k and m in that order. He formulates USApHMP-1L:

$$\min \sum_i \sum_j \sum_k \sum_m W_{ij} X_{ijkm} C_{ijkm}$$

s.t. (4) and

$$\sum_k Y_k = p \quad (5)$$

$$\sum_k \sum_m X_{ijkm} = 1 \ \ for\ all\ i,j \quad (6)$$

$$Z_{ik} \leq Y_k \ \ for\ all\ i,k \quad (7)$$

$$\sum_j \sum_m \left(W_{ij} X_{ijkm} + W_{ji} X_{jimk}\right) = \sum_j \left(W_{ij} + W_{ji}\right) Z_{ik} \quad for\ all\ i,k \quad (8)$$

$$Y_k \in \{0,1\} \ \ for\ all\ i,k \quad (9)$$

$$0 \leq X_{ijkm} \leq 1 \quad for\ all\ i,j,k,m \quad (10).$$

Constraint (5) guarantees that there are exactly p hubs that are opened; constraint (6) assures that all the flow is routed through the hubs. The constraint (7) allows an allocation of a node to a hub in k only if a hub is located in k and (8) guarantees that flow originating and arriving at i is only routed through the hub to which i is allocated to (single allocation constraint (Skorin-Kapov et al. 1996, p 586)). If X_{ijkm} are set to be binary variables, this last constraint (8) can be replaced by

$$Z_{ik} + Z_{jm} - 2X_{ijkm} \geq 0 \quad for\ all\ i,j,k,m \quad (11)$$ (Campbell, 1994, p 390).

As “the integrality restrictions imposed on a subset of variables, coupled with the large size of formulations (for a node network, the number of variables is $O(n^4)$) restricts the suitability of those formulations to small instances” (Skorin-Kapov et al., 1996, p 582) and “linear relaxation of Campbell's formulation […] are not tight and lead to fractional solutions” (Skorin-Kapov et al., 1996, p 586). Therefore, Skorin-Kapov et al. (1996) proposed new sets of constraints that get tighter LP relaxations than this version of the model. In this formulation, the hubs opened are no longer tracked by an independent variable (Y_k) but can rather be found in the allocation matrix ($Z_{kk}=Y_k=1$ only if a hub is opened in k) in the same way as in O'Kelly (1987). The new sets of constraints are (with the same objective function):

(2), (3), (4) and

$$Z_{ik} \leq Z_{kk} \quad for\ all\ i,k \quad (12)$$

$$\sum_m X_{ijkm} = Z_{ik} \quad for\ all\ i,j,k \quad (13)$$

$$\sum_k X_{ijkm} = Z_{jm} \quad for\ all\ i,j,m \quad (14)$$

$$X_{ijkm} \geq 0 \quad for\ all\ i,j,k,m \quad (15)$$

This formulation, called USApHMP-LP[1], now has n^2 binary variables (n less than the previous one), n^4 continuous variables and $1+n+n^2+2n^3$ constraints (Skorin-Kapov et al., 1996, p 587).

Sohn and Park (1998) further reduce the size of the formulation by assuming a symmetric unit flow cost. They also note that if the unit flow cost is proportional to the distance, $C_{ijkm}=C_{jimk}$ and the number of variables and constraints can be reduced. They define v_{ij} for i<j as $v_{ij}=W_{ij}+W_{ji}$ and reduce the formulation as follows:

$$\min \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\sum_{k}\sum_{m} v_{ij}C_{ijkm}X_{ijkm}$$

s.t. (2), (3), (4), (15) and

$$Z_{ik} \leq Z_{kk} \qquad for\ all\ i,k, i \neq k \quad (16)$$

$$\sum_{m} X_{ijkm} = Z_{ik} \qquad for\ all\ i = 1,\dots,n-1,\ \ j = i+1,\dots,n,\ \ k \quad (17)$$

$$\sum_{k} X_{ijkm} = Z_{jm} \qquad for\ all\ i = 1,\dots,n-1,\ \ j = i+1,\dots,n,\ \ k \quad (18)$$

resulting in a formulation with n^2 binary variables, $n^3(n-1)/2$ continuous variables and $1+n^3$ constraints. The LP relaxation will provide the same optimal solution as Skorin-Kapov et al. (1996) for symmetric problems (Sohn and Park, 1998, p 8). Although this model has the same form of constraints than in Skorin-Kapov et al. (1996), some of them are now on a subset of i,j,k and m.

Ernst and Krishnamoorthy (1996) follows a different path than the one resulting in the model of Sohn and Park (1998). It considers the flow starting at each node as a different commodity. They define Y_{ikm} as the total amount of flow of commodity i[2] that is routed through hubs in k and m and O_i and D_i similar as in O'Kelly (1987) and the model USApHMP-N:

$$\min \sum_{i}\sum_{k} C_{ik}Z_{ik}(\gamma O_i + \delta D_i) + \sum_{i}\sum_{k}\sum_{m} \alpha C_{km}Y_{km}^{i}$$

s.t. (2), (3), (4), (12) and

$$\sum_{m} Y_{km}^{i} - \sum_{m} Y_{mk}^{i} = O_i Z_{ik} - \sum_{j} W_{ij}Z_{jk} \quad for\ all\ i,k \quad (19)$$

$$Y_{km}^{i} \geq 0 \qquad for\ all\ i,k,m \quad (20)$$

in which (19) is the divergence constraint for each commodity i and each hub k. The model requires less variables and constraints than the models previously considered because it doesn't keep track of the flow between each pair of nodes separately as did the previous

[1] This is one of the models that have been implemented in the R script.
[2] i.e. the total flow originating at i

models. Therefore this model requires one index less (the two indexes i and j in the previous models are replaced by the commodity index i). It requires n^3+n^2 variables (n^2 are binary) and $1+n+2n^2$ linear constraints. Note that this model includes economies of scale on the transportation between non-hub nodes and hub nodes (included in the objective function by the parameters γ and δ). It is done to allow for the consideration of different discount factors between the costs of transportation from an origin node to a hub and from a hub to a destination node (Ernst and Krishnamoorthy, 1996, pp 8, 9).

2.1.2. Multiple allocation

The first linear model for the multiple allocation p-hub median problem (UMApHMP-1L) has been formulated in Campbell (1991) with $n+n^4$ variables and $1+n^2+2n^4$ constraints:

$\min \sum_i \sum_j \sum_k \sum_m W_{ij} X_{ijkm} C_{ijkm}$

s.t. (5), (6), (9),(10) and

$X_{ijkm} \leq Y_k \qquad for\ all\ i,j,k,m$ (21)

$X_{ijkm} \leq Y_m \qquad for\ all\ i,j,k,m$ (22) (Campbell, 1994, p 390).

This first multiple allocation model has then been improved by several papers. The first size reduction of this model can be found in Skorin-Kapov et al. (1996). They propose to replace constraint (21) and (22) by

$\sum_m X_{ijkm} \leq Y_k \qquad for\ all\ i,j,k$ (23)

$\sum_k X_{ijkm} \leq Y_m \qquad for\ all\ i,j,k$ (24)

like they did for the single allocation version. The resulting model (UMApHMP-LP) is equivalent to the one in Campbell (1994) but has less constraints ($2n^3+n^2+1$) and a tighter linear relaxation (Skorin-Kapov et al., 1996, p 585).

Another better proposition to replace these constraints comes from Hamacher et al. (2004). By developing rules on how to modify the facets from the uncapacitated facility location polyhedron to uncapacitated hub location they derive new inequalities that dominate the original ones (Hamacher et al., 2004, p 115) and obtain a better formulation by replacing constraints (23) and (24) by

$\sum_m X_{ijkm} + \sum_{m \neq k} X_{ijmk} \leq Y_k \quad \text{for all i, j, k}$ (25)

for a model (UMApHMP-FACET) (Hamacher et al., 2004, p 112) that has now n^3+n^2+1 linear constraints.

The final improvement of this model has been proposed in Marin et al. (2006). They replace the new constraints proposed by Skorin-Kapov et al. (1996) by the following sets of constraints:

$$Y_i + \sum_{u \neq i} \sum_s X_{ijus} \leq 1 \; for\ all\ i, j \neq i \quad (26)$$

$$Y_j + \sum_{u \neq j} \sum_s X_{ijsu} \leq 1 \; for\ all\ i, j \neq i \quad (27)$$

$$Y_i + \sum_{(u,s) \neq (i,i)} X_{iius} \leq 1 \; for\ all\ i \quad (28)$$

to obtain the model (UMApHMP-UM6) with $3n^2-n+1$ linear constraints that solves the cases with general assumptions. They also remove some variables associated with non-competitive costs from the formulation by setting their value to 0. To achieve that goal, they add to the formulation $X_{ijkm} = 0 \;\; if\ C_{ijkm} > \hat{c}_{ijkm} \;\; with\ \hat{c}_{ijkm} = min\{C_{ijkm}, C_{ijmk}, C_{ijkk}, C_{ijmm}\}$ (Marin et al., 2006, pp 281, 282).

Another model has been discussed in the literature involving 3 indexes. It has first been formulated in Ernst and Krishnamoorthy (1998) using the same commodity trick as in Ernst et al. (1996) for the single allocation version. They define X_{imj} to be the flow of commodity i that flows from hub m to node j for the following model (UMApHMP-N) (with V_{ik}: the amount of flow from i to k):

$$\min \sum_i (\sum_k \gamma C_{ik} V_{ik} + \sum_k \sum_m \alpha C_{km} Y^i_{km} + \sum_m \sum_j \delta C_{mj} X^i_{mj})$$

s.t. (5), (9), (20) and

$$\sum_k V_{ik} = O_i \qquad for\ all\ i \quad (29)$$

$$\sum_m X^i_{mj} = W_{ij} \qquad for\ all\ i, j \quad (30)$$

$$\sum_m Y^i_{km} + \sum_j X^i_{kj} - \sum_m Y^i_{mk} - V_{ik} = 0 \qquad for\ all\ i, k \quad (31)$$

$$V_{ik} \leq O_i Y_k \qquad for\ all\ i, k \quad (32)$$

$$X^i_{mj} \leq W_{ij} Y_m \qquad for\ all\ i, j, m \quad (33)$$

$$X^i_{mj} \geq 0 \qquad for\ all\ i, j, m \quad (34)$$

$$V_{ik} \geq 0 \qquad for\ all\ i, k \quad (35)$$

This model in Ernst and Krishnamoorthy (1998) has $2n^3+n^2+n$ variables and requires n^3+3n^2+n+1 constraints. The same paper proposes other models similar to UMApHMP-N, but

concludes that they are weaker than this model (Ernst et Krishnamoorthy, 1998, pp 104, 105).

2.1.3. Additional characteristics considered

Together with his multiple allocation model, Campbell (1994) proposes how to modify it in order to take flow thresholds into account. The flow threshold is defined as the minimum flow value needed to allow service on a link and is denoted T_{ik}, the flow thresholds for the link between nodes i and k (with $T_{ik} \leq \sum_j (W_{ij} + W_{ji})$) (Campbell, 1994, p 391). The addition of the following sets of constraints allows for the consideration of flow thresholds:

$$X_{ijkm} \leq Z_{ik} \quad for\ all\ i,j,k,m \quad (36)$$

$$X_{ijkm} \leq Z_{jm} \quad for\ all\ i,j,k,m \quad (37)$$

$$T_{ik} - \sum_j \sum_m (W_{ij} X_{ijkm} + W_{ji} X_{jimk}) \leq M(1 - Z_{ik}) \quad for\ all\ i,k \quad (38)$$

with M that can be set to the maximum possible flow on any spoke $M = \max_i \sum_j (W_{ij} + W_{ji})$. Campbell (1994) also considers the fixed cost for opening a spoke. He simply adds $+\sum_i \sum_k S_{ik} Z_{ik}$ into the objective function where S_{ik} is the fixed cost for opening a spoke between node i and hub k (Campbell, 1994, p 392).

2.2. Hub location models with fixed costs

This literature review will now focus on the hub location problems with fixed costs. This type of hub location problems is very similar to the p-hub median problem with two major differences:

- Most of the p-hub median formulations do not take into account the fixed costs for opening the hubs. They rather consider the number of hubs to open (incorporated into the formulation by a constraint setting this number of hubs). The hub locations problems with fixed costs do consider these costs in the objective function.
- Because these hub location problems consider these costs, the number of hubs is now a variable which is no longer imposed exogenously.

Because of this similitude, most of the formulations of the hub location problem with fixed costs can be obtained by modifying models of the p-hub median problem. The two differences between the p-hub median problems and the hub location problems with fixed costs lead to

two modifications to make to the previous models to obtain the following ones:

- The fixed costs have to be accounted for in the objective function. It is done by defining F_k, the fixed cost to open a hub in k and adding these costs in the objective function: $+\sum_k F_k Y_k$ or $+\sum_k F_k Z_{kk}$ depending on the variable used to assess if a hub is opened in k.
- As the number of hubs is no longer an exogenous parameter, the constraint on the number of hubs has to be removed from the formulation ((3) and (5))

For example the quadratic formulation (USAHLP-Q) in O'Kelly (1992) (O'Kelly, 1992, p 296) is the same as in O'Kelly (1987) (USApHMP-Q) (O'Kelly, 1987, pp 394, 395) for which we have to apply the two modifications mentioned above, the same goes for the formulations in Campbell (1994) that are the modifications of the formulations for the p-hub median problems of the same paper.

The tightest model in Boland et al. (2004) for the uncapacitated multiple allocation problem with fixed costs (UMAHLP-1, 2, 3) is based on the model UMApHMP-N of Ernst and Krishnamoorthy (1998) for the p-hub median problem (with the modifications mentioned above) but improves it with a new set of constraints. Note that this new set of constraints can be used to tighten the unmodified p-hub median model too, resulting in a model (UMApHMP-1,2,3,4,5) with $2n^3+n^2+n$ variables (n binary) and $5n^2+2n+1$ constraints. Their improvement is to remove equations (32) and (33) and to replace them by

$V_{kk} = O_k Y_k \quad for\ all\ k$ (39)

$X^i_{kk} = W_{ik} Y_k \quad for\ all\ i,k$ (40)

$V_{ik} \leq \sum_j W_{ij} Y_k \quad for\ all\ i,k, i \neq k$ (41)

$\sum_i X^i_{mj} \leq \sum_i W_{ij} Y_m \quad for\ all\ j,m \neq j$ (42)

In this case, equations (39) and (41) have the same purpose as (32) and the two others are the replacement of (33) (Boland et al., 2004, p 647).

Unlike the p-hub median problem, a great percentage of papers about hub location problems with fixed costs are interested in the capacitated version of the problem. The first article to assess this particular type of models is Campbell (1994). The capacitated hub location model is the same as the uncapacitated version with an additional constraint. He defines Cap_k,

the capacity of the hub that can be opened in k. The constraint to add in order to take the limited capacity into account is then:

$$\sum_i \sum_j W_{ij}(\sum_m [X_{ijkm} + X_{ijmk}] - X_{ijkk}) \leq Cap_k Y_k \quad for\ all\ k \quad (43)$$

for the multiple allocation version and

$$\sum_i \sum_j (W_{ij} + W_{ji}) Z_{ik} \leq Cap_k Y_k \quad for\ all\ k \quad (44)$$

for the single allocation (Campbell, 1994, pp 394, 395).

As for the uncapacitated version of the hub location problems with fixed costs most of the capacitated models are based on p-hub median models. Once again, the two modifications that are made for the uncapacitated version have to be applied. In addition, the models consider the capacity limitations by adding an additional constraint (as in Campbell (1994)) or by adding a new objective function making the model a bi-criteria model.

For example the models CSAHLP-LP[1] and CSAHLP-N in Ernst and Krishnamoorthy (1999) are based on the model USApHMP-LP of Skorin-Kapov et al. (1996) and the model USApHMP-N of Ernst and Krishnamoorthy (1996) respectively with the added constraint

$$\sum_i O_i Z_{ik} \leq Cap_k Z_{kk} \quad for\ all\ k \quad (45)$$[2] (Ernst and Krishnamoorthy, 1999, p 145).

da Graça Costa et al. (2008) considers a multi-objective model to take the capacity of the hubs into account. The models formulated are also based on the single allocation model in Ernst and Krishnamoorthy (1996). The paper proposes two versions (BSAHLP-1 and BSAHLP-2) with second objective functions:

$$\min \quad \sum_i \sum_k O_i T_k Z_{ik} + \sum_k P_k Z_{kk}$$

and

$$\min \quad \max_k \sum_i (O_i T_k Z_{ik} + P_k Z_{kk})$$

with T_k: the time the hub k takes to process one unit of flow and P_k: the fixed time to initiate the service at hub k (da Graça Costa et al., 2008, p 3674).

[1] A modified version of the capacity constraint of this model has been implemented in the R script.

[2] The capacitated models in this paper only consider flow arriving at the hub from a non-hub node. To include the flow arriving in the hub from other hubs, O_i has to be replaced by (O_i+ D_i).

The capacitated multiple allocation model CMAHLP-2 in Boland et al. (2004) is the same as the model the paper proposes for the uncapacitated version with the added constraint

$\sum_i V_{ik} \leq Cap_k Y_k \quad for\ all\ k$ (46) (Boland et al., 2004, p 647).

Ebery et al. (2000) proposes to add the equation

$\sum_i \sum_m Y_{km}^i \leq Cap_k Y_k \quad for\ all\ k$ (47)

to the model UMApHMP-N in Ernst and Krishnamoorthy (1998) (along with the two modifications to make it a hub location problem with fixed costs) in order to get a capacitated multiple allocation hub location problem with fixed costs (CMAHLP-N). They also remove equations (29) and (32) from CMAHLP-N and replace equations (33) by

$\sum_i \sum_k Y_{km}^i \leq \sum_i \sum_j W_{ij} Y_m\ for\ all\ m$ (48)

to formulate the model CMAHLP-F with $2n^2+2n$ constraints. They finally reduce the number of variables of the models (UMApHMP-N, CMAHLP-N and CMAHLP-F). They remove the variables V_{ik} by making the substitution $V_{ik} = \sum_m Y_{km}^i$ in the objective function and the constraints for final models with less n^2 variables. In this new model, the substitution allows for the simplification of equation (31) to

$\sum_k Y_{km}^i = \sum_j X_{mj}^i \quad for\ all\ i, m$ (49) (Ebery et al., 2000 , pp 617, 618).

2.3. p-hub center models

Another category of hub location problems is the p-hub center problem. It differs from the p-hub median problem and the hub location problem with fixed costs as its objective function is not a minisum but a minimax. Campbell (1994) defines 3 types of hub center problems. The first one aims at minimizing the maximum cost between any origin-destination pair. The second one minimizes the maximum cost on any connection (origin-hub, hub-hub and hub-destination) while the third type is only interested in the minimization of the maximum cost between any hub and any origin or destination. The paper then provides a basic formulation for the three objective functions of p-hub center problems both for the single and multiple allocation versions. The three objective functions they propose for both the single and multiple allocation models are:

- 1st type: $\min \max_{i,j,k,m} \{X_{ijkm}C_{ijkm}\}$
- 2nd type: $\min \max_{i,j,k,m} \{\max(C_{ik}, C_{mj}, \alpha C_{km})X_{ijkm}\}$
- 3rd type: $\min \max_{i,j,k,m} \{\max(C_{ik}, C_{mj})X_{ijkm}\}$[1]

The constraints for the single allocation model, USApHCP-1L, are exactly the same as for the single allocation p-hub median model (USApHMP-1L) in the same paper. The same goes for the multiple allocation model (UMApHCP-1L) that has the same sets of constraints than UMApHMP-1L (Campbell, 1994, p 396).

Several improvements, most of them very similar with improvements proposed for p-hub median problems, have been proposed to the models of Campbell (1994).

The first improvements come from Kara and Tansel (2000). They propose two new linear models based on Campbell (1994) (respectively USApHCP-LIN1 and USApHCP-LIN2) as well as a new model (USApHCP-LIN3), all for the single allocation version. The first linearization they propose is the same as the one that has been done by Skorin-Kapov et al. (1996) for the single allocation p-hub median problem. It consists in this case to replace the single allocation constraint (8) and constraint (6) by (13) and (14) and to add (2) to the formulation. The second linearization proposes to further replace (13) and (14) by

$$X_{ijkm} \geq Z_{ik} + Z_{jm} - 1 \quad for\ all\ i,j,k,m \quad (50)$$

and to relax integrality requirements on the X_{ijkm} to $X_{ijkm} \geq 0$. This last relaxation can be done because the new set of constraints together with the minimization of the objective function already guarantees that these variables will be binary (Kara and Tansel, 2000, pp 650, 651). They also propose a new model which they linearize. The final formulation is:

$$\min Z$$

s.t. (2), (3), (4), (12) and

$$Z \geq \sum_k [(C_{ik} + \alpha C_{km})Z_{ik}] + C_{jm}Z_{jm} \qquad for\ all\ i,j,m \quad (51)$$

with n^2 binary variables and n^3+n^2+n+1 constraints (Kara and Tansel, 2000, pp 654).

[1] The objective function of the third type is the same as for the second but with $\alpha = 0$.

Another new two-index formulation for the single allocation version is proposed in Ernst et al. (2009). It is based on the concept of the radius of the hub r_k which is the maximum transportation cost between the hub k and all the nodes that are allocated to it ($r_k \geq 0$). This model, called USApHCP-RAD[1], is formulated as:

$\min Z$

s.t. (2), (3), (4), (12) and

$r_k \geq C_{ik} Z_{ik} \quad for\ all\ i,k$ (52)

$Z \geq r_k + r_m + \alpha C_{km} \quad for\ k \leq m,\ m$ (53)

This last constraint (53) can be strengthened by replacing it by

$Z \geq r_k + r_m + \alpha C_{km} + (1-\alpha)(1 - Z_{kk}) min_{i:i\neq k} C_{ik} + (1-\alpha)(1 - Z_{mm}) min_{i:i\neq m} C_{im} \quad for\ all\ k,m$ (54)

This model has n^2+n+1 variables (n^2 binary) and $3n^2+n+1$ constraints (Ernst et al., 2009, pp 2231, 2232) and is considered the best model for the uncapacitated single allocation p-hub center problem to this day (Alumur and Kara, 2008, p 13)[2].

Ernst et al. (2009) also formulate a new multiple allocation p-hub center model involving three indexes. The paper defines binary variables $U_{.j}^{ik}$ as equal to 1 if -and only if- the node i is allocated to the hub in k and $V_{mj}^{i.}$ = 1 only if the node j is allocated to the hub m as well as $C_{max} = \max_{i,j} C_{ij}$. The model is then

$\min Z$

s.t. (5), (9) and

$U_{.j}^{ik} \leq Y_k \quad for\ all\ i,j,k$ (55)

$V_{mj}^{i.} \leq Y_m \quad for\ all\ i,j,m$ (56)

$\sum_k U_{.j}^{ik} = 1 \quad for\ all\ i,j$ (57)

$\sum_m V_{mj}^{i.} = 1 \quad for\ all\ i,j$ (58)

$Z \geq \sum_k (C_{ik} + \alpha C_{km}) U_{.j}^{ik} + \sum_n C_{nj} V_{nj}^{i.} - \alpha\left(1 - V_{mj}^{i.}\right) C_{max} \quad for\ all\ i \geq j, j, m$ (59)

$U_{.j}^{ik} \in \{0,1\} \quad for\ all\ i,k,j$ (60)

$V_{mj}^{i.} \in \{0,1\}\ for\ all\ i,j,m$ (61) (Ernst et al., 2009, p 2232).

---

[1] This is one of the models that have been implemented in the R script.

[2] Alumur and Kara (2008) is quite old but I also didn't find any paper released later that claims to have a model with a shorter computational time than Ernst et al. (2009).

2.4. Hub covering models

In terms of hub covering problems, we need to locate hubs to cover all demand and minimize the number of hubs located (or the costs for opening them). Like he did for the p-hub center problem, Campbell (1994) states 3 types of hub covering problems (which are analogous to the one for the p-hub center problem). The first type of covering problems they state is the one in which the origin-destination pair (i,j) is covered by the hubs in k and m if the total distance from i to j by the hubs k and m are smaller than a specified value ($C_{ijkm} \leq \omega_{ij}$). In the second type, an origin-destination pair is covered by the hubs in k and m if the cost on each link does not exceed a specified value ($\max\{C_{ik}, C_{mj}, \alpha C_{km}\} \leq \omega_{ij}$). The last one states that the origin-destination pair is covered if the hub-origin (i-k) and the hub-destination (j-m) links do not exceed specified different values (respectively ω_i and ω_j) ($C_{ik} \leq \omega_i$ and $C_{mj} \leq \omega_j$) (Campbell, 1994, p 399).

2.4.1. Single allocation

Campbell (1994) provides a first basic formulation for the single allocation hub covering problem. For this formulation he defines V_{ijkm}: binary parameter equal to 1 if the hubs k and m can cover the origin-destination pair (i,j) and F_k, the cost of opening a hub in k. The variables of the problem are X_{ijkm} and Z_{ik}, already defined in the models previously discussed. With these parameter and variables, their model, USASCP-1L, intends to minimize the total cost of opening the hubs

$\min \sum_k F_k Z_{kk}$

s.t. (2), (4), (8), (12) and

$\sum_k \sum_m V_{ijkm} X_{ijkm} \geq 1 \; for\ all\ i,j$ (62)

$X_{ijkm} \in \{0,1\} \; for\ all\ i,j,k,m$ (63).

This model has n^4+n^2 variables and $3n^2+n$ constraints (Campbell, 1994, p 401).

Kara and Tansel (2003) proposes to apply the same modifications to the hub covering model in Campbell (1994) as made by Skorin-Kapov et al. (1996) and Kara and Tansel (2000) for the p-hub median and p-hub center models. USASCP-LIN2 is constructed by replacing (8)

in USASCP-1L by (13) and (14). USASCP-LIN3 is constructed by further replacing these constraints by (6) and (50) and by relaxing the integrality constraints on the X_{ijkm} to (15). Both of these models have n^2+n^4 variables (n^4 continuous for USASCP-LIN3). USASCP-LIN2 counts $2n^3+2n^2+n$ constraints and USASCP-LIN3 has n^4+3n^2+n (Kara and Tansel, 2003, pp 60, 61).

Kara and Tansel (2003) also consider a two index formulation of the problem (derived from a quadratic formulation in Campbell (1994)), thus dropping the utilization of the X_{ijkm}. USASCP-LIN1 has the same objective function as the model in Campbell (1994) (although it doesn't consider the cost of opening the hubs but rather the number of hubs to locate (F_k is removed)) and is subject to the sets of constraints:

(2), (4), (12) and

$(C_{ik} + \alpha C_{km})Z_{ik} + C_{jm}Z_{jm} \leq \omega \quad for\ all\ i,j,k,m$ (64).

It does not consider the covering radius specific to each origin-destination pair (as was done in Campbell (1994)). They say that this last constraints (64) can be replaced by the strengthened constraints

$\sum_k (C_{ik} + \alpha C_{km})Z_{ik} + C_{jm}Z_{jm} \leq \omega \quad for\ all\ i,j,m$ (65)

to construct USASCP-KT (Kara and Tansel, 2003, p 60) with a number of constraints that grows as $O(N^3)$ instead of $O(N^4)$ (Ernst et al., 2011, p 4) [1].

Wagner (2008) further investigates the model USASCP-LIN1 in Kara and Tansel (2003) in order to improve it. The new strengthened model is obtained by the addition of three processes: the first is a reduction of the number of variables and the second is an abandonment of obsolete constraints. The two processes lead to a tighter model on which an aggregation of constraints is applied (third process). In order to reduce the number of variables, Wagner (2008) rules out some assignments. If the assignment of the node i to the hub k make it impossible to get from i to every j in time, it rules out the variables Z_{ik}. The set of every (i,k) valid assignments they consider is thus:

$VA = \{(i,k) | 2C_{ik} \leq \omega\ and\ C_{ik} + \min\ (1,\alpha)\ max_j C_{kj} \leq \omega\}$

Moreover, the number of constraints (64) is reduced (2nd process) as they are obsolete

[1] An authorization from the author is requested in order to quote this working paper. The authorization mail can be found in Appendix 5.

for $C_{ik} + \alpha C_{km} + C_{mj} \leq \omega$. If this is not the case and (i,k), (j,m) ϵ VA, the constraints can be tightened by prohibiting that Z_{ik} and Z_{jm} take the value 1 at the same time (Wagner, 2008, p 933). The set of incompatible assignments is then

$$IA = \{(i,j,k,m) | (i,k),(j,m) \in VA, i < j \; and \; C_{ik} + \alpha C_{km} + C_{mj} > \omega \}$$

It results in the model USASCP-W1. Its constraints are:

$$\sum_{k,(i,k) \in VA} Z_{ik} = 1 \quad for \; all \; i \quad (66)$$

$$Z_{ik} \leq Z_{kk} \quad for \; all \; (i,k) \in VA \quad (67)$$

$$Z_{ik} + Z_{jm} \leq 1 \quad (for \; all \; i,j,k,m) \in IA \quad (68)$$

$$Z_{ik} \in \{0,1\} \; for \; all \; (i,k) \in VA \quad (69)$$

The aggregation of the third process is made on the set of constraints (68). The aggregation algorithm can be found in Appendix 2. It is applied to any pair (i,j) with i<j and uses the fact that, for example, if $X_{12}+X_{34} \leq 1$ and $X_{12}+X_{35} \leq 1$, they can be aggregated into one constraint $X_{12}+X_{34}+X_{35} \leq 1$ (using the fact that the first constraint of the model implies $X_{34}+X_{35} \leq 1$). These three processes used together lead to USASCP-W2 that has a LP-relaxation with tighter bounds than the models previously considered (Wagner, 2008, pp 933, 934). Note that USASCP-W2 has a number of variables and constraints that depends on the threshold β (Ernst et al., 2011, p 5).

Ernst et al. (2011) propose a new model with $3n^2+n$ constraints based on the radius of the hubs as defined in the section on the p-hub center models. They define r_k , the radius of the node k, USASCP-RAD[1] can be formulated with the same objective function as the other hub covering models and subject to the constraints (2), (4), (12), (52), (53) (Ernst et al., 2011, p 5). This model seems to be the model that solves instances of the problem the most efficiently. It also resolves bigger instances of the problem than the models previously discussed (Ernst et al., 2011, p 15).

2.4.2. Multiple allocation

As for the single allocation hub covering problem, Campbell (1994) proposes a formulation of the multiple allocation problem. The formulation, UMASCP-1L, has the same

[1] This is one of the models that have been implemented in the R script.

objective function as the single allocation version and is subject to the constraints (9), (21), (22) and (62) for a total of $2n^4+n^2$ constraints (Campbell, 1994, p 400). Constraints (21) and (22) can be replaced in order to reduce the number of constraints like Skorin-Kapov et al. (1996) first proposed for the p-hub median problem (Wagner, 2008, p 935).

The size of this first multiple allocation model ($O(n^4)$) makes it impossible to solve large instances of the problem in a reasonable amount of time. In order to reduce it, Wagner (2008) defines the variables Q_{km}, the binary variables that take the value 1 if hubs are opened in k and m (as $Q_{km}=Q_{mk}$ and $Q_{kk}=Y_k$, only Q_{km} with k < m need to be considered (Wagner, 2008, p 935). He defines the valid routes sets $\mathrm{VR}_{\mathrm{ij}}^{1}$ (the routes that cover i and j with one hub (k)) and $\mathrm{VR}_{\mathrm{ij}}^{2}$ (the routes that cover i and j with two hubs (k and m)) as:

$$VR_{ij}^{1} = \{k | C_{ik} + C_{kj} \leq \omega\}$$

and

$$VR_{ij}^{2} = \{(k,m) | (k < m), \left(k, m \notin VR_{ij}^{1}\right) and\ (C_{ik} + \alpha C_{km} + C_{mj} \leq \omega\ or\ C_{im} + \alpha C_{mk} + C_{kj} \leq \omega).$$

The paper then formulates the problem as UMASCP-W with the same objective function subject to the sets of constraints:

(9) and

$$\sum_{k \in VR_{ij}^{1}} Y_k + \sum_{(k,m) \in VR_{ij}^{2}} Q_{km} \geq 1 \quad for\ all\ i,j \quad (70)$$

$$Q_{km} \leq Y_k \quad for\ all\ k < m, m \quad (71)$$

$$Q_{km} \leq Y_m \quad for\ all\ k < m, m \quad (72)$$

$$Q_{km} \geq 0\ for\ all\ k < m, m \quad (73)$$ (Wagner, 2008, pp 935, 936).

2.4.3. p-hub maximal covering models

p-hub maximal covering models do not intend to cover all nodes. In these models, the objective is to maximize the demand covered with a predefined maximal number of hubs whereas the hub covering models previously considered try to minimize the number of hubs that can cover all demand. Note that, in the literature, p-hub maximal covering models are classified as hub covering problems although their objective function and constraints are the same as in a p-hub median problem (Campbell, 1994, p 401).

Campbell (1994) provides the first p-hub maximal covering models. The model for multiple allocations (UMApHMCV-1L) is formulated as:

$$\max \sum_i \sum_j \sum_k \sum_m W_{ij} X_{ijkm} V_{ijkm}$$

s.t. (5), (6), (9), (10), (21) and (22)

Its model for single allocation (USApHMCV-1L) has the same objective function subject to the constraints (4), (5), (6), (7), (8), (9) and (10) (Campbell, 1994, pp 400, 401). Similar to what proposed Kara et Tansel (2003) for their model USApHCP-LIN2, Hwang and Lee (2012) propose to improve the model of Campbell (1994) to obtain a model with the same objective function as in USApHMCV-1L and the constraints (2), (3), (4), (11), (12), and (63) (Hwang and Lee, 2012, p 384).

Peker and Kara (2015) propose a new improved formulation that does not need to keep tracks of all the routes (and therefore has a reduced size). The model USApHMCV-P[1] is as follows:

$$\max \sum_i \sum_j W_{ij} F_{ij}$$

s.t. (2), (3), (4), (12) and

$$F_{ij} \leq \sum_k V_{ijkm} Z_{ik} + \partial_{ij}\left(1 - Z_{jm}\right) \ for\ all\ i, j, m \quad (74)$$

$$F_{ij} \geq 0\ for\ all\ i, j \quad (75)$$

with F_{ij} the fraction of the flow from i to j that is covered and $\partial_{ij} = \max_{k,m} V_{ijkm}$ that is used to tighten the constraint (Peker and Kara, 2015).

Peker and Kara (2015) also propose a parameter modification in order to take partial coverage into account. A partially covered node is a node that is at a distance of the hub that is greater than the cover radius of the hub but smaller than a different value. To take partial coverage into account the covering parameter (V_{ijkm}) is replaced by b_{ijkm}. The paper defines the new coverage parameter b_{ijkm} as:

$$b_{ijkm} = \begin{cases} 1 & if\ C_{ijkm} \leq \omega_{ij} \\ f\left(C_{ijkm}\right) & if\ \omega_{ij} \leq C_{ijkm} \leq \delta_{ij} \\ 0 & otherwize \end{cases}$$

[1] This is one of the models that have been implemented in the R script.

with f(.) being a non-increasing function and f(x) ϵ [0,1] (Peker and Kara, 2015, p 160).

2.5. Extensions and other models

The models previously reviewed are models based on general assumptions. They are the basis of the study of the hub location problems. As such they have been modified to allow for the study of several variations on the characteristics of the problem studied. The considerations of the maximum capacities at hub nodes, the flow thresholds as well as the fixed costs to locate the hubs and to connect the non-hub nodes to the hubs have already been discussed in the previous subsections. But this basis has also been modified to consider several other features. These new models have a similar general hub network structure than the models previously considered[1]. For example, a part of the literature focuses on stochastic parameters and the implication they have on the models: Contreras et al. (2011) considers the cases with stochastic demands (with demands that are independent random variables with known probability distributions), dependent stochastic transportation costs or independent stochastic transportation costs (Contreras et al., 2011). The three models are based on Hamacher et al. (2004) (p-hub median problem). Sim et al. (2009) studies the p-hub center problem with normally independently distributed travel times (Sim et al., 2009). Other papers consider multimodal networks (see Alumur et al. (2012)), hub congestion (see de Camargo et al. (2009)), competition between several decision makers (see Mahmutogullari and Kara (2016)) and decisions over multiple periods (see Gelareh et al. (2015)), for example. Literature on the hub location models is too vast to list all the variants of the models discussed. Another non-negligible part of the literature focuses on real-life application of the models in different fields with consideration of the particular characteristics of these fields. Airline passenger transportation is the most discussed one (see Bryan and O'Kelly (1999) for a review) but liners networks, telecommunication and rail-road transportation have also been considered to a lesser extent.

In the introduction of this literature review, we stated that most of the models discussed in the literature assume three things: that (1) the hub network is complete, (2)

[1] See the introduction of this literature for the assumptions.

economies of scale occur on the transportation between two hubs and (3) there are no direct connections between non-hub nodes (Alumur and Kara, 2008, pp 1, 2). However some models that do not satisfy one or several of these assumptions have been developed in the literature. The hub arcs location models and the star hub location models, for example, are two classes of models that do not satisfy the first assumption: each hub node is not connected to all the other hub nodes in the solution of these models.

Following Campbell and Krishnamoorthy (2005), the hub arc location problems see the hub location problems as network design problems[1] for which there are two decisions to make: where to locate hub arcs and where to locate the access arcs. The location of the hubs is determined by the location of the hub arcs (one hub at each end of a hub arc). The access arcs are the connections between non-hub nodes and hubs. As each arc has to be chosen independently, it is possible that the hub network is not complete. Hub arc location models have been developed to compensate for some inconsistencies of the hub locations models with complete hub networks. Firstly, with a complete hub network, it is possible that some inter-hub connections will only carry very little flow. In that situation, it can be more profitable not to allow any flow on that connection and to reroute it through different hubs. Secondly, most of the hub locations models assume a discount on the inter-hub connections due to the aggregation of the flows (assumption (2)). However, some optimal solutions of the models can have considerably larger flow between a non-hub node and a hub than on some connections between two hubs. In this case, “the basic assumption in hub median models that flow costs are discounted on hub arcs to reflect high volumes leads to a possible mismatch between the abstracted model and the underlying motivations of the model” (Campbell and Krishnamoorthy, 2005, p 1541). To avoid that mismatch, the hub arc location models also relax the second assumption earlier discussed (Campbell and Krishnamoorthy, 2005, pp 1540-1541).

Yaman (2008) introduces the specificities of another class of models that do not lead to a complete hub network: the star hub location models. Unlike the hub location models previously discussed, the star hub location models define two levels of hubs: a central hub that is located exogenously and endogenous secondary hubs. The secondary hubs are all connected

[1] In the literature, the hub location problems design have usually been seen and adapted from facility location problems.

to the central hub but not directly connected to each other. Therefore, the flow that has to be transported on the network will first travel from its origin to the secondary hub it is allocated to, from this secondary hub to the central hub. It will then go to the secondary hub the destination is allocated to and then to its destination. If the origin and the destination are both assigned to the same hub, it will only go from the origin to that hub and then from that hub to the destination. Secondary hubs are thus never connected with each other (Yaman, 2008, p 3009).

3. R implementation

The R script that solves the single allocation hub location problems is in Appendix 1.

3.1. Models implemented

The optimal solution is found using a different model for each of the problems solved. For the single allocation p-hub median problem, the model used is USApHMP-LP and has been first discussed in Skorin-Kapov et al. (1996) (see 2.1.1.). For the single allocation p-hub center problem, the model USApHCP-RAD (see 2.3.) in Ernst et al. (2009) has been implemented. The single allocation hub covering problem is modelized using USASCP-RAD (see 2.4.1.) from Ernst et al. (2011) and the single allocation p-hub maximal covering problem with USApHMCV-P (see 2.4.3.) in Peker and Kara (2015). The script for the p-hub median also allows the consideration of a maximum capacity for the hub candidates with the constraint of CSAHLP-LP formulated in Ernst and Krishnamoorthy (1999) (see 2.2.) modified to include the distribution of the flows to the destination nodes allocated to the hubs. It is also possible to consider the fixed costs to locate the hubs (see 2.2.) and the fixed costs to allocate the nodes from Campbell (1994) (see 2.1.3.) into the minimization of the total cost. For the hub covering problem, it is possible to choose between the minimization of the number of hubs and the minimization of the total cost to locate the hubs as both have been considered in the literature (see 2.4.)[1].

These models have been chosen because the computational studies of the papers that discuss them came to the conclusion that they are the most effective modelizations to solve problems that have the assumptions stated in the introduction of chapter 2 and we haven't found any paper released later that contradicted this assertion. For the single allocation p-hub median model, USApHMP-N would retrospectively have been a better choice of modelization as it reduces the number of variables and constraints which should lead to a shorter optimization time than the model currently implemented.

[1] See chapter 2 for the assumptions and the explanation of the syntax used to name the models.

3.2. Packages required

The R script requires the installation of several packages in order to run properly. The package 'stringr' (Wickham, 2015) that provides functions to manipulate strings. It regroups basic string operations functions as well as pattern matching functions that allows to replace, locate or extract part of a string and manipulate it (The R foundation, 2015). The script uses this package for the function that retrieves the distance matrix on the internet if required by the user. 'Rglpk' (Theussl et al., 2016a) provides an R interface to the GNU Linear Programming Kit (GLPK) (Theussl et al., 2016b). "GLPK package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP) and other related problems" (Makhorin, 2012). It is called by the mains functions to optimize the different hub location problems. The last package the script requires is the package 'googleVis' (Gesmann et al., 2016). It is an R interface to the 'Google Charts API', which is used to create the map that displays the optimal solution.

3.3. Functions

The R script consists of the main functions and secondary functions called for specific parts of the algorithms. The main functions are the functions that will be called by the user to solve the optimization of the single allocation hub location problems (see Appendix 1). There is one main function for the resolution of each of the four hub location problems solved:

- 'SApHMP(flow, cost, alpha, nHub, cap=NULL, hCost=NULL, sCost=NULL, dis=NULL, locNames=NULL, gVisDist=FALSE, gVisOutput=FALSE, gVisDisplayRegion=NULL)' optimizes the single allocation p-hub median problem.
- 'USApHCP(cost=NULL, alpha, nHub, locNames=NULL, gVisDist=FALSE, gVisOutput=FALSE, gVisDisplayRegion=NULL)' is for the single allocation p-hub center problem.
- 'USASCP(cost=NULL, alpha, bet, hCost=NULL, locNames=NULL, gVisDist=FALSE, gVisOutput=FALSE, gVisDisplayRegion=NULL)' for the single allocation hub covering problem.

- ‘USApHMCV(flow, cost=NULL, alpha, bet, nHub, locNames=NULL, gVisDist=FALSE, gVisOutput=FALSE, gVisDisplayRegion=NULL)’ for the single allocation p-hub maximal covering problem.

The names in brackets are the names of the arguments of the functions. Their signification is explained in the following subsubsection. When their default value is specified as the ‘NULL’ object, it means that their specification is not required for the algorithms of the functions to work properly[1].

3.3.1. Input arguments

The main functions require 2 categories of arguments: one that regroups arguments that specify the parameters of the problem to solve and one with options arguments.
The arguments that specify the parameters of the problem to solve are:

- **flow** (W_{ij} in the literature review) is the matrix that specify the flows from each origin to each destination for the p-hub median problem and the p-hub maximal covering problem. It needs to be a square matrix or a square ‘dataframe’ with numeric positive arguments. The diagonal of the matrix should be a vector of zeros as it is an assumption of the models used.
- **cost** (C_{ij} in the literature review) the matrix that specify the transportation costs between the nodes. The signification of the cost depends on the problem solved. For the p-hub median problem it is the transportation cost of one unit of flow on the total distance between the two nodes if the distance matrix is not specified or it is the transportation cost of one unit of flow per unit of distance if the distance matrix is specified. For the p-hub center problems, the hub covering problems and the p-hub maximal covering problems it is usually the distances between the nodes or the time it takes to transport the flow between them. The requirements for ‘cost’ are the same than for ‘flow’ except if ‘gVisDist’ is set to ‘TRUE’ for the p-hub center problem, the hub covering problem and the p-hub maximal covering problem as in that situation the matrix will be calculated by the script itself.

[1] When ‘cost=NULL’, the argument’s specification is not required only if ‘gVisDist=TRUE’. ‘LocNames’ is not required only if ‘gVisDist=gVisOutput=TRUE’.

- **alpha** (α in the literature review) is the discount factor that accounts for the reduced transportation costs on the inter-hub connections due to the aggregation of the flows. It needs to be a real number between 0 and 1 included.
- **bet**[1] (β in the literature review) is the maximal value the total cost to go from one node to another through their allocated hubs can take in order for the flow between the two nodes to be covered. It is only required for the two covering problems. It needs to be a real number greater than 0.
- **nHub** (p in the literature review) is the number of hubs to locate. This argument is not needed for the resolution of the single allocation hub covering problem as the number of hubs to locate is endogenous. It has to be a non-negative integer.
- **cap** (Cap_k in the literature review) is the maximal capacity of the hub candidates. It only has to be specified for the p-hub median problems that are capacitated. If it is specified, it needs to be a vector with non-negative numeric arguments.
- **hCost** (F_k in the literature review) is a vector specifying the fixed costs to locate the hubs for the p-hub median and the hub covering problems. It only needs to be specified if these fixed costs have to be considered. If it is the case, 'hCost' needs to be a numeric vector with positive values. The hub covering problem resolution will minimize the total cost to locate the hubs instead of the number of hubs to locate if 'hCost' is specified.
- **sCost** (S_{ik} in the literature review) is the matrix with the fixed costs to allocate the non-hub nodes to the hubs if they need to be considered for the p-hub median problems. If 'sCost' is specified, it has the same requirement as 'cost' and 'flow' (see above).
- **dis** is the distance matrix for the p-hub median problem. For this problem, it has been separated from the cost matrix to allow the user to either enter a cost matrix with arguments that are the total cost to transport one unit of flow on the total distance between the two nodes or that are the total cost per unit of flow per unit of distance. As for the cost matrix for the three other hub location problems solved by the script, the distance matrix of the p-hub median problem can be calculated by the script itself if the argument 'gVisDist' is set to 'TRUE'. The distance matrix do not have to be

[1] The term ''beta'' was already used by the R language so we could not name this parameter that way.

specified if it is the case. If it is specified by the user, the distance matrix has the same requirements as 'flow' (see above).

- **locNames** is the vector with the names of the nodes. It has three purposes. Firstly, if the distance or the cost matrix is calculated by the script, it uses these locations names to obtain the values from the internet. Secondly, it is used for the display of both the solution map and the solution matrices (see 3.3.3.). Finally, if the location names are specified, the solution displayed on the R console is modified for clarity purposes (see 3.3.3.) If 'gVisDist' or 'gVisOutput' are set to 'TRUE', the locations names have to be compatible with the 'Google Maps API' (Google, n.a. (b)).

Additional arguments have to be entered in order to specify the options:

- **gVisDist** is a boolean set to 'TRUE' (default to 'FALSE') if the script needs to request the cost matrix (distance matrix for the p-hub median problem) from the internet.
- **gVisOutput** is a boolean set to 'TRUE' (default to 'FALSE') if the user wants the optimal solution to be displayed on a map.
- **gVisDisplayRegion** is the name (compatible with the 'Google Maps API') of the region, continent or country that will be displayed on the map if 'gVisOutput' is set to 'TRUE'.

3.3.2. Algorithms of the R functions

The algorithms followed by the main functions can be break down into several steps that are followed sequentially to optimize the single allocation hub location problems. The functions start by checking that the input entered by the user has the conditions required by the script in order to work properly. This input is then modified to be entered as arguments for the solver to run the optimization. The final step is the display of the solution to the user.

The first step of the algorithm starts by verifying that the arguments entered for the function satisfy the requirements for the next steps to run properly. It first checks the conditions on each of the arguments of the function taken separately (see 3.3.1. for the requirements on each of the arguments). It was first intended for the script to allow the user to enter numbers in the arguments as characters. It would then convert any number entered as character into the corresponding numeric value. However, by doing this, the algorithm was

changing eventual letters into numbers whatever the method used. As this was not the way the algorithm was expected to perform, we decided to remove it from the script. It is now expected for the matrices, vectors and 'dataframes' to be formed only by numeric non-negative arguments. It then verifies the relative conditions on several arguments. For example, the matrices' sizes need to be the same and equal to the length of the vectors entered. If the solution is displayed on a map ('gVisOutput=TRUE') or if the distance matrix has to be constructed by the script and not entered by the user ('gVisDist=TRUE'), the location names need to be specified. If any of the arguments do not meet a condition, the execution of the script stops and an error message is displayed to the user. The only exceptions are the requirements on the diagonal of the matrices (the hub location models assume $C_{ii}=0$ and $W_{jj}=0$ for all i). If the diagonal of a matrix is not a vector of zeros, the script continues its execution and a warning message is displayed to the user. This first step of the algorithm also verifies the arguments in order to determine which characteristics of the problem need to be considered: for the p-hub median problem, the consideration of the hub capacities and the fixed costs to locate the hubs or to allocate the nodes to the hubs is checked. The script also verifies if the costs specified are the transportation costs per unit of flow or the transportation cost per unit of flow per unit of distance together with a distance matrix. The difference between the minimization of the number of hubs to locate and the minimization of the total cost to locate them is made for the hub covering problems. Finally, if the script needs to construct the distance matrix, this is done after the verification on the parameters concerned.

The following step of the algorithm concerns the optimization of the hub location problem. The arguments of the functions are first modified to construct the arguments of the solver used in order to optimize the hub location problems. These modifications are done because the parameters of the models as presented in the literature (see chapter 2) and as entered by the user for a correct utilization of the script are not compatible with the function used for the optimization. This is the case mainly because the parameters of the problem are entered as matrices (and the optimal value of each variable is presented in matrices) whereas the solver needs them to be presented as vectors.

The solver used has been chosen on the list of open source linear and mixed integer linear programs presented in "CRAN Task View: Optimization and Mathematical Programming"

(Theussl and Borchers, 2016). The two solvers present on that list that seemed the more suitable are 'Rglpk' and 'lpsolve' (Berkelaar et al., 2015). 'Rglpk' has been chosen over the package 'lpSolve' as it seems to be a more complete mixed integer linear programming solver. According to the specifications of the two solvers, 'lpsolve' uses the revised simplex together with the branch-and-bound method for the integer variables (Sourceforge, n.a.) and 'GLPK' includes a primal and dual simplex method, a primal and dual interior-point method and a branch-and-cut method among other things (Mokharin, 2012) which seemed more complete. The differences between the two solvers are slim nonetheless.

The function 'Rglpk_solve_LP' used to solve the problem is provided by the package 'Rglpk' (see: 3.2.). The arguments of this functions put together are the objective function, the constraints of the problem to solve and the type of the variables of the problem. The arguments specified by the script for the utilization of the function are:

- **obj**: a numeric vector with the coefficients of the objective function of the problem.
- **mat**: a numeric matrix that specify the coefficients of the constraints of the problem.
- **dir**: a vector with characters specifying the directions of the constraints.
- **rhs**: a numeric vector with the right hand side of the constraints. The arguments represent the part of the constraints that do not depend on the value of the variables of the problem.
- **types**: a vector that indicates if the variables are binary, continuous or integer.
- **max**: a logical set to 'TRUE' if it is a maximization problem and 'FALSE' if it is a minimization problem (Theussl et al., 2016a).

After these arguments are constructed, the solver's function is called and the solution is stored. The solution as returned by the solver is a list from which three components are used by the script (if an optimal solution has been found). The first component is a vector with the optimal value of each of the variables of the problem. The second is the value of the objective function at optimum. The third is the solution status: 0 if an optimal solution has been found non-zero otherwise (Theussl et al., 2016a).

The last step of the algorithm is in charge of the solution display to the user. It starts by converting the solution of the problem as returned by the solver into the solution as presented

in the literature. As explained above, the literature presents the variables into matrices whereas the solver uses a unique vector. If the solution display on a map is required by the user, the data is also converted into a 'dataframe' compatible with the 'googleVis' package in order to display the solution map.

3.3.3. Output and solution display

The script offers three different solution displays and outputs to the user. These outputs are similar whatever the hub location problem solved.

The first solution output is the display of the results on the screen using the R console. If the script has been able to find a solution to the problem, several sets of information are available. The first information displayed is the optimal value found. For the p-hub median, it is the minimal network cost and for the p-hub center, it is the greatest cost on any path on the optimal network. For the hub covering problem, the minimum number of hubs located or the minimum cost to locate them that cover all the nodes are displayed and for the p-hub maximal covering problem it is the maximum flow that can be covered with the specified number of hubs to locate. It also displays the number of hubs on the network (especially for the hub covering problem as it is endogenous), the names of the nodes that are selected to be hubs as well as the allocation of the non-hub nodes to the hubs. The inter-hub flows are also displayed for the p-hub median problem as well as the flows that are covered and not covered for the p-hub maximal covering problem. For convenience, the allocation matrix is displayed differently depending on the specification of the location names as showed in figure 2. They are both the optimal solution of a single allocation p-hub median problem on a 5 nodes network of the 'CAB data set' (see 4.1. for a presentation of this data set) with alpha=0.5, nHub=2.

```
An optimal solution has been found.
The optimal cost is: 150486463.8501
The 2 hub(s)are located at:
2, 5
The node allocation to the hubs is:
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    1
[2,]    0    1    0    0    0
[3,]    0    1    0    0    0
[4,]    0    0    0    0    1
[5,]    0    0    0    0    1
The inter-hub flows are:
      2     5
2 25998 72203
5 72203 87640
```

```
An optimal solution has been found.
The optimal cost is: 150486463.8501
The 2 hub(s)are located at:
 Baltimore  ,  Cincinnati
The node allocation to the hubs is:
                Baltimore    Cincinnati
 Atlanta                0             1
 Baltimore              1             0
 Boston                 1             0
 Chicago                0             1
 Cincinnati             0             1
The inter-hub flows are:
                Baltimore    Cincinnati
 Baltimore          25998         72203
 Cincinnati         72203         87640
```

Figure 2: Example of how the solution is displayed on the R console for a single allocation p-hub median problem with the nodes names not specified (left) and specified (right) ('CAB data set', 5 nodes, alpha=0.5, nHub=2).

Figure 3 shows the solution displayed on the R console for the single allocation p-hub center problem for the same network (5 nodes, alpha=0.5, nHub=2) when the nodes names are not specified. Figure 4 shows the optimal solution displayed for the single allocation hub covering problem for the same network, with bet=858 and alpha=0.5 and figure 5 shows the solution of a single allocation p-hub maximal covering problem on the same network with bet=858, alpha=0.5 and nHub =3.

```
An optimal solution has been found.
At optimum, the longest path of the network is: 858.2158
The 2 hub(s)are located at:
3, 5
The node allocation to the hubs is:
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    1
[2,]    0    0    0    0    1
[3,]    0    0    1    0    0
[4,]    0    0    0    0    1
[5,]    0    0    0    0    1
```

Figure 3: Solution displayed on the R console for a single allocation p-hub center problem with the nodes names not specified ('CAB data set', 5 nodes, alpha=0.5, nHub=2).

```
An optimal solution has been found.
The optimal number of hubs to locate is:  3
The 3 hub(s)are located at:
2, 3, 5
The node allocation to the hubs is:
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    1
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    0    1
[5,]    0    0    0    0    1
```

Figure 4: Solution displayed on the R console for a single allocation hub covering problem with the nodes names not specified ('CAB data set', 5 nodes, alpha=0.5, bet=858).

```
An optimal solution has been found.
The maximum quantity of flow that can be covered by  3  hubs is: 258044
The 3 hub(s)are located at:
1, 2, 5
The node allocation to the hubs is:
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    1    0    0    0
[4,]    0    0    0    0    1
[5,]    0    0    0    0    1
The flows covered are:
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    1    1    1    1
[2,]    1    0    1    1    1
[3,]    1    1    0    1    1
[4,]    1    1    1    0    1
[5,]    1    1    1    1    0
```

Figure 5: Solution displayed on the R console for a single allocation p-hub maximal covering problem with the nodes names not specified ('CAB data set', 5 nodes, alpha=0.5, nHub=3, bet=858).

If an optimal solution could not be found, the console warns the user that it is the case. For the p-hub median problem, the console also informs the user whether the solution could not be found because of the capacities of the hub candidates and for the hub covering problem, it displays the nodes combinations that cannot be covered whatever the hubs located.

If a solution has been found, the script also returns a list that can be stored or manipulated by the user. This list regroups three arguments (with one additional argument for the p-hub median problem). The first argument is the optimal numeric value found after

resolution. The second argument is a vector with the names of the hubs (or their number if no names have been specified) and the third is a matrix with the nodes allocations to the hubs. The additional argument returned to the user for the p-hub median problem is a matrix with the inter-hub flows. If no solution has been found, the script returns the 'NULL' object.

Finally, if the user requested it, the script displays the optimal hub location and node allocation to the hubs using the 'Google Chart Tool' (Google, n.a. (a)). The solution is displayed on a 'GeoChart'. 'GeoCharts' are maps that schematically display a region of the world and show information about the different areas displayed or about discrete points placed on the map. It is automatically displayed on the screen using an internet browser (Google, 2016). Figure 6 provides an example of how the optimal solution can be displayed on a map. This figure shows the optimal solution of a p-hub median on a 10 nodes network from the 'CAB data set' (Appendix 3) with α=0.5, nHub=4 and gVisDisplayRegion="US". The biggest points are the hubs and the smallest the non-hub nodes. The color of each point shows to which hub it is allocated. The text displayed when hovering over a node is also showed.

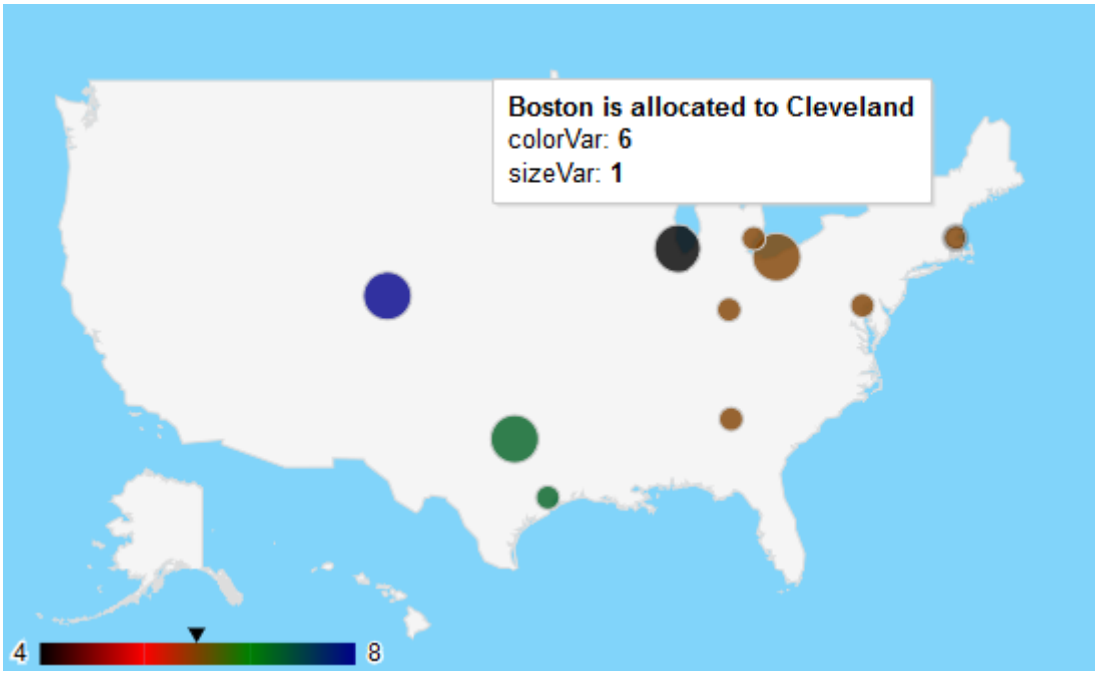

Figure 6: example of a solution map from the 'CAB data set' displayed on a 'geoChart' ('CAB data set', 5 nodes, alpha=0.5, nHub=4).

3.3.4. Sub-functions

In order to avoid rewriting the same piece of code once for each of the functions that solve hub location problems, these functions make use of sub-functions responsible for precise parts of the algorithm that are similar whatever the hub location problem solved. Five sub-functions that are not provided by packages or the original R installation are used by the algorithms solving the hub location problems. 'maxVectN(vect, n)' takes a numeric vector and a natural number n as arguments and returns the sum of the highest n numbers. It is used for the p-hub median when no solution has been found. It calculates the maximum capacity that can be processed by the number of hubs that can be located. If it is less than twice[1] the total amount of flow on the network it informs the user that this is why no feasible solution could be found. 'mapSolution(locNames, sizeVar, colorVar, hoverVar, gVisDisplayRegion)' creates the solution map to be displayed if 'gVisOutput=TRUE' and if an optimal solution has been found. The arguments of this function are the vectors that are part of the 'dataframe' used by 'googleVis' to create the map and the region to be displayed. It returns the constructed 'geoChart' that then needs to be plotted. The last three functions are used together to request the distance matrix from the internet if 'gVisDist=TRUE'. 'vectorizedDist(locVector)' returns a matrix with the distances between all the locations specified in the vector argument. It calls the function 'distance_google(adresse_depart, adresse_arrivee)' on the $n(n-1)$ combinations of two different nodes (the distance returned by the 'Google Maps API' are not symmetric, it is therefore not possible to reduce the number of calls by half by duplicating the coefficients of the bottom left triangular submatrix into the top right triangular submatrix) . This last function requests the distance between the two locations specified as arguments by using the 'Google Maps API'. The function 'distance_google' has been written by Delplace (2011). 'Interrupt(x)' is used to put a delay between two distances requests as the 'Google Maps API' only allows one request per second or it will return an 'OVER_QUERY_LIMIT' error message. Its argument is the number of seconds the script needs to be interrupted. The code of this function is the basic interruption method and comes from the R documentation (ETH Zurïch, n.a.).

[1] The capacity constraint counts the flows that have to be sorted at each hub which means that the total flow that arrives at each hub from its allocated origin nodes and the flow that goes out of each hub to each allocated destination node. Each flow is therefore "counted" twice.

4. Computational analysis

Two critical performance indicators of a script are its execution time and the memory it uses during its execution. The execution time limits the number of different problems that can be solved in a predetermined amount of time. The maximum memory it will allocate during its execution prohibits the resolution of some problems depending on the architecture. If the memory required exceeds the total memory available on the architecture, the execution will not be able to finish properly. Of course, it goes without saying, the correctness of the solution outputted by the script is even more critical. For these reasons the computational study of this thesis consists of calls of the four functions that solve the single allocation hub location problems with different sets of arguments. For each call, the execution time, the maximum memory used and/or the optimal solution returned will be recorded. The analysis of the data collected this way will then depend on the arguments studied: for arguments that are considered in the literature about models with the same assumptions than the R implementation, computational results are available and will help benchmark our script. If provided in the benchmark papers, the evolution of the execution time in function of the parameters of the problem[1] and the feasibility will be compared. The optimal solutions returned by the R implementation will also be compared with solutions available in the literature in order to assess its correctness. For arguments that are not considered in the literature on the subject, this computational analysis will try to determine if their modification has an effect on the performances indicators considered or not.

The execution time of the different calls of the functions will be recorded with the function 'system.time' (ETH Zurïch, 2016) or with the function 'profvis' (Github, n.a.) depending on the test. The memory used by the different functions will be recorded using the function 'profvis' from the package of the same name. 'profvis' is a function that summarizes

[1] As the R implementation and the implementations considered in the literature are executed on different architectures, it is impossible to compare their execution time directly. The rapid evolution of computers would induce a bias in the comparison. It is however possible to compare the marginal evolution of each implementation in function of the parameters of the problem solved.

the data collected by ‘Rprof’ (Github, n.a.). ‘Rprof’ is a memory and time profiling tool that is available in the original R installation. It is important to note that because ‘Rprof’ “[…] works by sampling, the result isn’t deterministic. Each time you profile your code, the result will be slightly different.” (Github, n.a.). The ‘profvis’ function will always be called with the argument ‘interval=0.005’. This argument sets the sampling interval at 5 milliseconds in comparison to the 10 milliseconds of the default settings. This will provide more reliable values, especially for execution times that are close to 10 milliseconds.

The network on which the effect of the different parameters will be tested is the ‘CAB data set’ (see 4.1.). It provides the nodes names, flows and the distances on a network of 25 nodes.

All the functions calls are done on a 3.10 GHz processor with 8 Go of RAM. It is run on the 64 bit version of the release 3.3.1 of R with the versions 1.0.0 of ‘stringr’, 0.6.2 of ‘Rglpk’, and the versions 0.3.2 of ‘profVis’ and 0.6.0 of ‘googleVis’.

We will start with a benchmark of the four functions of the R implementation with the computational results of papers that focus on models that have the same set of assumptions than our script. We will then perform additional computations for arguments of the R functions that, to our best knowledge, are not considered in the literature. The following subsection presents the data set used for our computations.

4.1. Data set presentation

For this computational study, a data set is used in order to specify the parameters of the networks on which the hubs will be located. Two data sets have been extensively used in the hub location literature: the ‘CAB data set’ (Civil Aeronautics Board) and the ‘AP data set’ (Australian Post). For this computational analysis, the ‘CAB data set’ has been chosen for several reasons. Firstly, it is the data set that has been used the most in the literature and in most of the papers that first discussed the models implemented. Secondly, unlike for the ‘AP data set’, the nodes names are provided and compatible with the ‘Google Maps API’ which will allow the analysis of the effect of the display of the solution map and the construction of

the distance matrix on the performance indicators. Finally, the ‘AP data set’ considers discount factors on the arcs between non-hub nodes and hubs which our R implementation doesn’t.

The ‘CAB data set’ (Appendix 3) is a survey of the Civil Aeronautics Board performed in 1970 (Kara and Tansel, 2000) that provides data about the passenger flows and distances between 25 American cities (the names of the 25 cities are also provided). This data set has first been used in O’Kelly (1987) and has since then been used for the computational studies of most of the literature about hub location problems (Alumur and Kara, 2008, p 3). The matrices of this data set are square matrices of 25 columns and 25 rows which diagonal is a vector constituted only with zeros, as assumed by the hub location problems. These matrices have the particularity of being both symmetric. However, this is not a condition required for the script to run properly. We assume that the flow matrix specifies the number of passengers to transport and the distance matrix is in kilometers. If a problem with less than 25 nodes has to be considered, the top left sub-matrices of the correct size are selected to constitute the data of the problem, as has been done in the literature.

4.2. Benchmark of the R implementation with the literature

The flow matrix, the cost matrix, α, β and p are the parameters of the problem that are considered in the papers discussing the models that have been implemented. As such, resolution time and optimal solutions of several instances of the different problems are available in the literature. Therefore, the performance of the R implementation will be tested by comparison with the results obtained by several papers for different sets of parameters. These parameters are ‘flow’, ‘cost’, ‘alpha’, ‘bet’ and ‘nHub’ in the arguments of the functions. The effect of their variation on the execution time and the differences in feasibility will be benchmarked if the relevant data is provided in the literature. The design of this part of the computational analysis will thus be very similar to the one of the papers its results are compared to. For papers that also use the ‘CAB data set’, this will also allow the verification of the correctness of the algorithms of the R implementation by comparison of the solutions calculated by the R implementation with the solutions provided in the papers. For these computations, ‘cap=hCost=sCost=dis=locNames=gVisDisplayRegion=NULL’ and ‘gVisDist=gVisOutput=FALSE’ as parameters specified by these arguments are not considered

in the comparison papers. For these tests, the execution times of the functions are recorded with the function ‘system.time’.

4.2.1. Benchmark of the function that solves single allocation p-hub median problems

The R implementation of the p-hub median problem will be compared to Skorin-Kapov et al. (1996) which first discussed the model implemented. The paper reports the optimal hub locations and the corresponding objective function values of instances of the ‘CAB data set’ with 5, 10, 15, 20 and 25 nodes. The numbers of hubs to locate that are considered are 2, 3 and 4 and the values of α are 0.2, 0.4, 0.6, 0.8 and 1.0. As they use the same data set as in this computational analysis, the comparison of the objective function values and the optimal solutions of the different instances will assess the correctness of the R implementation. Note that this paper reports the execution times, optimal hub locations and optimal objective function values found by the resolution of the LP relaxation of the problem. This LP relaxation provided a solution with integral variables for 57 out of the 60 instances solved. For the three instances with fractional solutions, an integral solution is also provided. It is obtained by adding integrality constraints to the fractional variables (Skorin-Kapov et al., 1996, pp 588-591). The paper only provides rough resolution times for the instances of the problem solved: “CPU time was dimension dependent: around a minute for n=10, few minutes for n=15, 2-3 hours for n=20 and 13-15 hours for n=25” (Skorin-Kapov et al., 1996, p 589) which will make the comparison of the evolution of the execution times less precise.

Table 5 shows the optimal objective function values and hub locations calculated by the R function ‘SApHMP’ for ‘CAB data set’ problems with the same size, α, β and p as in Skorin-Kapov et al. (1996).

The computational results of Skorin-Kapov et al. (1996) can be found in table 6 and table 7, Appendix 4.1. The solutions in table 6 are solutions of the LP relaxation of the problem. For instances of the LP relaxations that obtained non-integer variables values, table 7 shows the integer solution found by adding partial integrality constraints.

Table 5: Optimal solutions of the single allocation p-hub median problem calculated by the R function 'SApHMP' for instances of the 'CAB data set'.

n	p		α				
			0.2	0.4	0.6	0.8	1
10	2	IP_{obj} (x10^5)	615.39	673.651	731.912	790.172	834.999
		hubs	7;9	7;9	7;9	7;9	4;7
	3		491.455	567.36	643.264	716.284	775.928
			4;6;7	4;6;7	4;6;7	4;7;9	4;7;9
	4		394.746	493.313	577.268	660.771	735.542
			3;4;6;7	4;6;7;8	4;6;7;8	4;7;8;9	1;4;7;9
15	2		2320.672	2513.05	2705.428	2816.1	2889.773
			4;12	4;12	4;12	4;11	4;11
	3		1891.885	2140.5	2388.419	2600.271	2763.855
			4;7;12	4;7;12	4;7;12	4;7;8	4;7;8
	4		1513.032	1843.973	2152.586	2427.664	2644.539
			4;7;12;14	4;7;12;14	1;4;7;12	1;4;7;8	1;4;7;8

'IP_{obj}' is the optimal objective function value of the problem. 'Hubs' indicates the number of the nodes where the hubs are located (Skorin-Kapov et al., 1996, p 590, modified for our table). The other arguments of the function are 'cap=hCost=sCost=dis=locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'.

We first note that the computational study of Skorin-Kapov et al. (1996) could solve problems with a greater size than the R implementation. The R implementation could not solve the instances with 20 and 25 nodes. This is due to the fact that the number of variables grows as $O(n^4)$ and the number of constraints as $O(n^3)$. This causes the memory needed by the MILP solver that solves the single allocation p-hub median problem to exceed the memory available for relatively small instances of the problem. Second we note that for every single instances that have been solved with both implementations, the hub locations are exactly the same (for the instance which had a non-integer solution with the LP relaxation of Skorin-Kapov et al. (1996), the solution from the R implementation provided the same hub location as calculated in the paper with a partial set of integrality constraints). However the optimal objective function values are not the same. For instances of the problem with 10 nodes, the objective function values in Skorin-Kapov et al. (1996) are all between 1.00097 and 1.00098 time the objective function values of the solution of the R function in millions (except for the instance for which the solution of the LP relaxation was non-integer). It is probable that the values in Skorin-Kapov et al. (1996) are expressed in millions (or that their data from the 'CAB data set' were expressed in different units). If it is the case, the differences in values between the two implementations are less than 0.1% and can possibly be attributed to the rounding of the parameters of the problem. For instances with 15 nodes, the optimal objective function values seem different. The objective function values of the R function expressed in millions are

however all between 2.3649 and 2.3679 time the objective function values of Skorin-Kapov et al. (1996). Even if they do not mention it expressly, we think that it is likely that the data of the problem used by Skorin-Kapov et al. (1996) has been scaled for clarity purposes as has been done in the computational study of Peker and Kara (2015) (Peker and Kara, 2015, p 38). This is confirmed by the fact that, for example, the optimal value for n=15, p=2 and α=0.2 is greater than for n=20, p=2 and α=0.2 which shouldn't be the case. We add 5 more nodes that have a similar range of flows and costs to the network so the total cost of the network should increase and not decrease. With that in mind, the fact that all the hub locations of the optimal solutions are exactly the same and that the optimal objective function values are consistent with each other as explained before, we think that we can conclude that the R implementation for the single allocation hub location problem finds the correct solution for all the instances considered or at least finds the same solutions than the implementation of Skorin-Kapov et al. (1996). The size of the instances that can be solved by the R implementation is however lower than in the paper it is compared to.

The execution times will be more difficult to compare because they can vary quite much from one call of the function to another and Skorin-Kapov et al. (1996) only provides wide ranges of their resolution times. Moreover, they provide the execution time of the optimization of the LP relaxation of the problem. They say:" […] around a minute for n=10, few minutes for n=15 […]" (Skorin-Kapov et al., 1996, p 589). The approximated execution times of the R function were around 1.5 seconds to solve the instances with 10 nodes and 130-140 seconds for instances with 15 nodes. We can see that the resolution time of the R implementation is a lot lower than the resolution time of Skorin-Kapov et al. (1996) for n=10 but the difference seems to reduce for n=15 which would imply that the execution time of the R function increases faster with the size of the problem than the implementation of Skorin-Kapov et al. (1996).

4.2.2. Benchmark of the function that solves single allocation p-hub center problems

The R function that solves the single allocation p-hub center problem will be benchmarked with the computational results of Ernst et al. (2009). The paper provides the optimal objective function values for instances of size 10, 15, 20 and 25 of the 'CAB data set'

with α:={2; 3; 4} and p:={0.2; 0.4; 0.6; 0.8; 1.0} (Ernst et al., pp 2236, 2237). Like for the comparison of the p-hub median implementations, the comparison of these values will assess the correctness of the R algorithm. In this paper, the CPU time for the resolution of these instances is also available in the computational analysis and its evolution in function of the parameters of the problem will therefore be compared to the execution times of the function of the R implementation.

The optimal objective function values of the computational results of Ernst et al. (2009) can be found in table 8, Appendix 4.2. Unlike Skorin-Kapov et al. (1996), they do not provide the optimal hub location so only the optimal objective function values will be compared. Table 9 shows the optimal objective function values calculated by the R implementation of the single allocation p-hub center problem.

Table 9: Optimal objective function values of the single allocation p-hub center problem calculated by the R function 'USApHCP' for instances of the 'CAB data set'.

n	p	α				
		0.2	0.4	0.6	0.8	1
10	2	1425.581	1627.519	1759.129	1759.129	1839.65
	3	1119.535	1185.065	1387.003	1588.941	1790.548
	4	830.25	968.199	1146.186	1454.44	1764.791
15	2	2005.018	2160.748	2214.091	2423.801	2609.175
	3	1749.036	1760.146	1844.922	2166.538	2600.078
	4	1340.96	1434.38	1754.511	2080.062	2600.078*
20	2	1892.991	2160.748	2274.67	2501.924	2609.175
	3	1551.25	1760.146	1997.793	2263.544	2600.078
	4	1355.413	1472.707	1834.831	2152.997	2600.078
25	2	2131.198	2402.55	2558.736	2714.926	2827.158
	3	1923.118	2100.564	2340.25	2554.131	2758.394
	4	1619.483	1884.844	2182.491	2454.349	2726.281

The other arguments of the function are 'locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'.
*Optimal objective function value that is different to the optimal objective function value of Ernst et al. (2009).

The R implementation has been able to solve all the instances of the 'CAB data set' considered. The optimal objective function values are all the same than in Ernst et al. (2009) except for one: the instance with 15 nodes, 4 hubs and α=1. Ernst et al. (2009) reported a maximal distance between two nodes of 2166.54 whereas the R implementation calculated

2600.78. Unfortunately, as the optimal hub locations are not provided by Ernst et al. (2009), it is difficult to analyze the two solutions in order to determine the causes of that difference. Moreover, the optimal function values found by both implementations could be consistent with the values calculated for the problems with similar parameters: they are both greater than the optimal objective function value for n=15, p=4 and α=0.8, greater or equal to the optimal objective function value for n=10, p=4 and α=1 and less or equal to the optimal objective function values of n=15, p=3 and α=1 and n=20, p=4 and α=1. None of them can thus be eliminated by comparing it to the other values found by both implementations. Nevertheless, the R implementation that solves the single allocation p-hub center problem found the same solution for 59 of the 60 instances of the 'CAB data set' considered.

Table 10 shows the execution times of the R function 'USApHCP' to solve the different instances of the 'CAB data set' considered. The CPU time to solve the same instances of the problem in Ernst et al. (2009) can be found in table 11, Appendix 4.2.

Table 10: Execution times of the R function 'USApHCP' to solve single allocation p-hub center problems for instances of the 'CAB data set'.

n	p	α				
		0.2	0.4	0.6	0.8	1
10	2	.04	.05	.04	.03	.03
	3	.07	.08	.05	.05	.02
	4	.08	.06	.05	.05	.02
15	2	.26	.21	.2	.13	.05
	3	.29	.32	.36	.31	.1
	4	.55	.37	.16	.08	.05
20	2	.86	1.86	.45	.7	.57
	3	1.85	2.28	1.84	1.06	.93
	4	2.14	2.27	2.55	2.04	.67
25	2	6.74	5.47	3.24	4.28	1.
	3	5.54	7.08	6.23	5.06	3.62
	4	7.91	9.	6.25	3.74	19.45

The values are expressed in seconds. The other arguments of the function are 'locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'. The execution times are measured with 'system.time'.

The execution time of the R implementation and in Ernst et al. (2009) both increase when the size of the problem increases. It increases more with the size of the R function than

with the size of the implementation in Ernst et al. (2009) which makes the R implementation less powerful for the resolution of bigger instances of the single allocation p-hub center problems involving more nodes.

4.2.3. Benchmark of the function that solves single allocation hub covering problems

The benchmark paper for the resolution of the single allocation hub covering problem is Wagner (2008). It has been chosen over Ernst et al. (2011) as its computational analysis considers the 'CAB data set' whereas Ernst et al. (2011) solves instances of the 'AP data set' and the 'Turkish airline data set'. Wagner (2008) provides the optimal solution values and execution times for an instance of the 'CAB data set' with 25 nodes and several values of α and β. The values of β considered are the optimal objective functions values of the p-hub center with the same parameters rounded up and rounded down. The computation times to solve these hub covering problems are reported for several models. We will only compare and report the computation times of the model they considered that performs best, it is USASCP-W2 (see 2.1.1.) slightly modified. They also use preprocessing algorithms to reduce the computation time of the resolution of the problem (Wagner, 2008, pp 936, 937).

The results of Wagner (2008) on the single allocation hub covering problem are shown in table 12 and table 13, Appendix 4.3. The values of β considered in table 12 are the solutions of the single allocation p-hub center problem with the same parameters rounded down. For table 13, these same solutions are rounded up. Table 14 and table 15 shows the optimal number of hubs to locate and the execution time of the R function 'USASCP' with the same parameters as considered in Wager (2008).

Table 14: Execution times and optimal objective function values of the R function 'USASCP' for single allocation hub covering problems for instances of the 'CAB data set' with β that are objective function values of the single allocation p-hub center problem for the same instances rounded down.

α	β	p	time
0.2	2131	3	.47
0.6	2558	3	1.44
1	2827	3	.34
0.2	1923	4	1.32
0.6	2340	4	6.43
1	2758	4	.3
0.2	1619	5	1.
0.6	2182	5	1.75
1	2726	5	.28

The execution times are expressed in seconds. The other arguments of the function are 'hCost=locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'. The optimal numbers of hubs to locate 'p' are the optimal objective function values for the 25 nodes network of the 'CAB data set'. The execution times are measured with 'system.time'.

Table 15: Execution times and optimal objective function values of the function 'USASCP' for single allocation hub covering problems of the 'CAB data set' with β that are objective function values of the single allocation p-hub center problem for the same instances rounded up.

α	β	p	time
0.2	2132	2	.85
0.6	2559	2	1.17
1	2828	2	.31
0.2	1924	3	.69
0.6	2341	3	3.31
1	2759	3	.27
0.2	1620	4	.35
0.6	2183	4	.38
1	2727	4	.27

The execution times are expressed in seconds. The other arguments of the function are 'hCost=locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'. The optimal numbers of hubs to locate 'p' are the optimal objective function values for the 25 nodes network of the 'CAB data set'. The execution times are measured with 'system.time'.

The computational results of the two implementations for the single allocation hub covering problem calculated the same optimal objective function values for all the instances of the problem considered. The best implementation of Wagner (2008) performed however better overall with its computation times smaller for 16 out of the 18 instances considered that are consistently reduced when the value of β are rounded up in comparison to when they are rounded down which doesn't happen with the R implementation.

4.2.4. Benchmark of the function that solves single allocation p-hub maximal covering problems

The last function to benchmark against computational results from the literature is the function that solves the single allocation p-hub maximal covering problem. It will be compared with Hwang and Lee (2012) that provides optimal objective function values and optimization times for networks of 10 and 15 nodes of the 'CAB data set' and for several different values of α, β and p. The computational results of Hwang and Lee (2012) for the networks of 10 and 15 nodes can be found in table 16 and table 17, Appendix 4.4. They also provide suboptimal objective function values returned by the CPLEX after 5 hours of execution for problems of size 20 and 25 (Table 18, Appendix 4.4). The values of β considered depend on the value of α, p and n. They are shown in table 19, Appendix 4.4. (Hwang and Lee, 2012, p 386-388). Table 20 presents the optimal objective function values as returned by the R function for instances of the 'CAB data set' of sizes 10, 15, 20 and 25, and for several values of α, β and p.

Table 20: Optimal objective function values of single allocation p-hub maximal covering problems calculated by the R function 'USApHMCV' for instances of the 'CAB data set'.

n	p	α				
		0.2	0.4	0.6	0.8	1
10	2	994540	994540	987490	999026	984836
	3	999026	990542	984530	999026	999026
	4	999026	999026	999026	999026	999026
	5	991270	999026	999026	999026	999026
15	2	2358068	2364942	2364942	2364942	2364942
	3	2358068	2364942	2304218	2320434	2364942
	4	2364942	2364942	2364942	2364942	2364942
	5	2353712	2364942	2320434	2320434	2320434
20	2	5747720	5737094	5748824	5754594	5754594
	3	5743058	5739610	5719090	5754594	5754594
	4	5754594	5754594	5754594	5754594	5754594
	5	5722742	5754594	5754594	5752254	5710086
25	2	8540006	8536326	8536326	8536326	8527758
	3	8566986	8540006	8540006	8536326	8540006
	4	8533986	8517004	8540006	8540006	8536326
	5	8540006	8526490	8524146	8490176	8536326

The other arguments of the function are 'locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'.

As both Hwang and Lee (2012) and the R function that solves the single allocation p-hub maximal covering problem returned the optimal objective function value for problems of size 10 and 15 of the 'CAB data set', we can compare them both to assess the correctness of the R algorithm. We can see that both implementation returned the same optimal values for all the instances considered except for two. For n=15, α=0.8, p=5, β=2080 and for n=15, α=1, p=5, β=2600, the implementation of Hwang and Lee (2012) did not find a solution because of excess memory issues (Hwang and Lee, 2012, p 387). The R implementation, on its end, found an optimal solution for these two instances. Both computational analysis confirm the values found by each other for the problems they solved both. For instances of the problem with 20 and 25 nodes the benchmark paper only provides suboptimal solutions calculated with CPLEX in five hours where the R implementation found the optimal solution of all the instances considered. For these instances of the problem, the solutions provided by the R implementation are all better as they guarantee that more flow can be covered by the hubs located. To our best knowledge, the optimal objective function values of these instances of the 'CAB data set' with these values of α, β, p for the single allocation p-hub maximal covering problem were not provided by the literature on the subject.

Table 21 shows the time the R implementation of the single allocation p-hub maximal covering problem needed to return the optimal solution for all the instances of the 'CAB data set' considered.

Table 21: Execution times of the R function 'USApHMCV' to solve single allocation p-hub maximal covering problems for instances of the 'CAB data set'.

n	p	α						
		0.2	0.4	0.6	0.8	1	mean	maximum
10	2	1.34	1.26	.5	.2	.25		
	3	.31	.33	.3	.2	.22		
	4	.2	.2	.19	.17	.2	.3465	1.34
	5	.21	.23	.22	.2	.2		
15	2	15.14	1.98	1.05	1.07	1.09		
	3	3.23	1.06	5.	2.87	1.04		
	4	.98	1.07	1.11	.95	1.15	2.314	15.14
	5	3.26	1.08	.99	1.11	1.05		
20	2	40.59	14.41	27.49	3.53	3.58		
	3	28.15	11.37	33.99	3.26	3.38		
	4	3.29	19.11	3.24	3.3	3.54	12.803	40.59
	5	37.77	5.67	3.62	3.35	3.42		
25	2	385.96	183.2	51.55	48.97	9.22		
	3	979.01	40.05	89.31	23.52	9.26		
	4	89.58	641.23	44.21	9.14	9.72	188.492	979.01
	5	11.79	298.27	286.48	550.4	8.97		

The values are expressed in seconds. The other arguments of the function are 'locNames=gVisDisplayRegion=NULL' and 'gVisDist=gVisOutput=FALSE'. The execution times are measured with 'system.time'.

When comparing these execution times with what is reported in Hwang and Lee (2012) (see Table 17, Appendix 4.4) we can directly see that the execution time of the R implementation increases slower than what is reported in Hwang and Lee (2012). The mean execution time are around 315 time less for instances of size 10 and around 47000 for instances with 15 nodes. For instances of size 20 and 25, the CPLEX could not find the optimal solution for the implementation in Hwang and Lee (2012) in less than 5 hours whereas the R implementation found the optimal solution in around 13 seconds for n=20 and around 188 seconds for n=25 on average. These big differences in the evolution of the execution times with the size of the problem can be explained by the fact that the R function and Hwang and Lee (2012) are not the implementation of the same model. The R function is an implementation of a model of Peker and Kara (2015) that is supposed to perform better (see 2.4.3.). This last paper only provides solutions for instances of the 'CAB data set' of 25 nodes, with the same values of α and p but with lower values of β. It has not been possible to compare the computational results of Peker and Kara (2015) with the R implementation because its execution time grows very rapidly with a diminution of the parameter β (or an augmentation depending on the range of β and on the value of the other parameters). With the values

considered in Peker and Kara (2015), the R implementation of the problem could not find an optimal solution in a reasonable amount of time. The comparison of the optimal objective function values was therefore impossible and would not have allowed the assessment of the correctness of the R algorithm. The resolution time of the p-hub maximal covering problem by the R script fluctuates a lot with the value of β. For example, for the problem involving 25 nodes with α=0.2 and p=5, when β=1300 the execution time of the function is around 143 seconds. If β=1400, the execution time of the function is close to 39 seconds, if β=1500, it is close to 10 seconds and if β=1600, it is close to 11 seconds (when β=1800, the function could not find the optimal solution in less than 20 minutes). These variations are solely due to the time the solver 'Rglpk' takes to optimize the problem and we couldn't find any consistent connections between the values of β and the execution times of the solver. We think that these significant variations are due to the degeneration of the MILP. During the resolution of a degenerated formulation, "the solution may stay the same after a pivot" (MIT, 2013, p 17) and "the simplex method may cycle and be infinite" (MIT, 2013, p 17). We think that this explains why the R function could not solve the instances of the 'CAB data set' of the computational results in Peker and Kara (2015) in a reasonable amount of time. The values of β they consider are in a range for which the MILP degenerates.

4.3. Additional computations

4.3.1. Parameters not considered in the benchmark papers

The R implementation of the p-hub median problem can consider additional parameters than the one studied in the previous subsection. These parameters are the maximum capacities at the potential hub locations ('cap'), the fixed costs for locating the hubs ('hCost') and the fixed costs for connecting the non-hub nodes to the hubs (sCost). It is also possible to specify a distance matrix ('dis') together with a unit flow cost per unit of distance matrix. All the functions can also take an argument with the nodes names ('locNames') that changes the way the solution matrix is displayed and the resolution of the hub covering problem can take the fixed costs to locate the hubs into account. As they are not considered in the papers that discuss the models implemented the algorithms they are involved into can't be benchmarked. As the function can work properly without these arguments being specified,

we will try to determine whether their specification do have an effect on the time and maximum memory required by the different functions or not. In order to do so, samples of then calls of each function with and without the specification of one of the arguments will be compared. As the vectors and matrices considered in this subsection are not provided by the 'CAB data set', they are constructed for this computational analysis as follows:

- The capacity vector will be constructed so that the sum of any three[1] following hub candidates ({1;2;3}, {2;3;4}, …) capacities is equal to 2.1 times[2] the total flow of the network ($\sum_{y=x}^{y=x+2} Cap_y = 2.1 \sum_{i=1}^{i=size} \sum_{j=1}^{j=size} flow_{ij} \quad for\ all\ i,j,x \leq size - 3$). The capacity of each hub andidate is chosen so that the capacity constraint is binding for at least some hub locations combinations. For hub candidates that have a number k such that $\frac{k-1}{3} \in N$, $cap_k = 2.1 \frac{\sum_{i=1}^{i=size} \sum_{j=1}^{j=size} flow_{ij}}{3}$. The hub candidates that have a number k such that $\frac{k-2}{3} \in N$ have a capacity $cap_k = 2.1 \frac{11 \sum_{i=1}^{i=size} \sum_{j=1}^{j=size} flow_{ij}}{30}$ and the hub candidates that have a number k such that $\frac{k}{3} \in N$, have a capacity $cap_k = 2.1 \frac{9 \sum_{i=1}^{i=size} \sum_{j=1}^{j=size} flow_{ij}}{30}$.
- The fixed costs for locating the hubs ('hCost') will be set at 10^6 each for 'SApHMP'.
- The fixed costs for allocating the non-hub nodes to the hubs ('sCost') are set at 10^5 each except for the coefficients of the diagonal of the matrix which are equal to zero.
- When 'dis' is specified, it is the distance matrix of the 'CAB data set' (otherwise the distances is specified as 'cost'). In that situation, the cost matrix that is specified is the unit flow cost per unit of distance (instead of the unit flow cost on the total distance). This unit flow cost per unit of distance is set at one for all i,j such that i≠j and zero otherwise.
- The vector with the location names comes from the 'CAB data set'.

[1] For this test, the number of hubs to locate is three.

[2] It is two time the total flow on the network because the capacity constraint counts arriving flows from allocated origin nodes and departure flows from allocated destination nodes at each hub so each flow is counted twice. 5% is added to that capacity because they are not a multiple of the individual flows and as it is a single allocation problem, these flows can't be split in order to completely fill the available capacities at each hub.

The function that solves the p-hub median problem will be called for a problem of size 10 from the ‘CAB data set’ with alpha=0.5 and nHub=3. The effect of the specification of these parameters will be tested with ‘gVisDist’ and ‘gVisOutput’ both set to ‘FALSE’ as the algorithms they enable by being set to ‘TRUE’ only depend on the size of the problem (the effect on the performance indicators of the specification of the additional arguments is not impacted by the specification of ‘gVisDist’ and ‘gVisOutput’). The algorithm linked to ‘gVisDist’ determines the distance from each of the nodes to each other whatever the other arguments are and the one linked to ‘gVisOutput’ creates a map using a ‘dataframe’ which length only depends on the size of the problem. As the implications on the algorithms of the specification of ‘locNames’ for all the functions and ‘hCost’ for the p-hub median and the hub covering problems are the same, conclusions about the absolute effect on the performance indicators of the p-hub median problem can be extended to the other functions of the R implementation. Both performance indicators will be collected using the function ‘profvis’.

Table 22 and table 23 show the execution time and memory allocated for ten calls of ‘SApHMP’ with none of these arguments specified and with one of them specified at the time for a problem of the ‘CAB data set’ with 10 nodes.

Table 22: execution times of ‘SApHMP’ to solve single allocation p-hub median problems for instances of the ‘CAB data set’ with none of the additional arguments specified or with these arguments specified one at a time.

arguments specified	Number of the call										
	1	2	3	4	5	6	7	8	9	10	mean
none	1.76	1.545	1.535	1.66	1.59	1.655	1.67	1.545	1.74	1.51	1.621
cap	3.11	3.075	3.245	3.15	3.335	3.06	3.22	3.175	3.285	3.195	3.185
hCost	1.585	1.815	1.645	1.595	1.605	1.49	1.61	1.555	1.54	1.63	1.607
sCost	1.535	1.5	1.565	1.52	1.625	1.64	1.595	1.565	1.58	1.555	1.568
dis	1.635	1.625	1.635	1.57	1.625	1.595	1.64	1.585	1.66	1.62	1.619
locNames	1.565	1.51	1.565	1.475	1.655	1.61	1.665	1.505	1.59	1.775	1.592

The execution times are expressed in seconds. The other arguments of the function are alpha=0.5, nHub=3, gVisDist=gVisOutput=FALSE and gVisDisplayRegion=NULL. The execution times are measured with ‘profvis’ with a sampling interval of 0.005 seconds.

Table 23: memory used by ‘SApHMP’ to solve single allocation p-hub median problems for instances of the ‘CAB data set’ with none of the additional arguments specified or with these arguments specified one at a time.

arguments specified	Number of the call										
	1	2	3	4	5	6	7	8	9	10	mean
none	491.6	491.8	491.2	491.	491.8	492.	492.2	491.1	491.	491.	491.47
cap	494.5	493.5	494.3	493.7	493.2	493.6	494.2	495.3	496.8	494.	494.31
hCost	491.9	491.1	493.	491.7	492.9	491.	491.1	491.8	492.7	491.1	491.83
sCost	492.9	492.4	491.1	491.9	491.8	492.5	491.2	491.1	492.2	491.8	491.89
dis	493.	492.1	492.	494.5	491.2	491.1	491.2	491.8	491.7	491.1	491.97
locNames	493.8	493.7	490.1	491.	491.	491.3	491.	490.9	490.4	490.9	491.01

The memory is expressed in MB. The other arguments of the function are alpha=0.5, nHub=3, gVisDist=gVisOutput=FALSE and gVisDisplayRegion=NULL. The memory used is measured with profvis with a sampling interval of 0.005 seconds.

The data shows that only the specification of the maximum capacities of the potential hub locations significantly changes the execution time of the function. For this instance of the 'CAB data set' with 10 nodes, it almost doubles the mean execution time of the function. The 10 executions of the function with one of the other parameters specified have a lower mean execution time than with no parameter specified. For these parameters, the mean executions time are very close to each other with a maximum difference of 3.27%. Same conclusions can be drawn from the memory used by the different calls of the function with the difference that the consideration of the capacities of the potential hub locations now have very little effect (less than 0.6%) on the memory used by the function for these samples.

These observations are confirmed by an analysis of the script. What changes in the algorithm of the function with the specification of all the arguments considered in this section is, firstly, that it verifies that each of them meets its requirements as specified in subsubsection 3.3.1. These operations take a negligible amount of time in comparison to the rest of the algorithm and do not need to allocate a lot of memory. Secondly, except for 'locNames', the arguments which have a negligible effect on both the execution time and the memory used by the function only change values in the vector that specify the coefficients of the objective function. 'hCost' (respectively 'sCost') changes the coefficients of the Z_{kk} (respectively Z_{ik}) and 'dis' changes the way the C_{ijkm} are calculated. When 'dis' is specified $C_{ijkm}=dis_{ij}cost_{ij}+ \alpha dis_{km}cost_{km}+dis_{mj}cost_{mj}$, when it isn't, $C_{ijkm}=cost_{ij}+ \alpha cost_{km}+cost_{mj}$. These modifications are unlikely to have a significant effect on the performances indicators analyzed for the resolution of the problem as they only add a few non-zero coefficients to the formulation for 'hCost' and 'sCost' in the p-hub median problem and only changes non-zero coefficients for 'dis' in the p-hub median problem and 'hCost' in the hub covering problem (Ernst et al., 2011, p 9). The specification of 'locNames', only changes slightly the algorithm that is in charge of the display of the allocation matrix on the R console. When specified, it is displayed as an n x p matrix instead of n x n (see 3.3.3.) which has a negligible effect on the performance indicators as it requires a few simple operations. The specification of these arguments has therefore a negligible effect on the execution time of the function or the memory it allocates. The specification of the maximum capacities of the potential hubs does have a significant effect on the execution time. When it is specified, this argument adds n additional constraints to the model which significantly changes the number of non-zero coefficients in the constraints

matrix. This has an adverse effect on the execution time of the optimizer as would an increase of the size of the problem to solve (Ernst et al., 2011, p 9).

4.3.2. Options arguments

As the algorithms these arguments enable are exactly the same for all the functions, the conclusions for 'USApHMP' can be extended to the 3 other functions of the R script. Both performance indicators will be collected using the function 'profvis'.

'gVisDist' and 'gVisOutput' are arguments that specify if specific parts of the algorithms have to be called. When 'gVisDist=TRUE', the distance matrix of the problem do not have to be specified by the user but is rather constructed with the distances obtained on the internet with the 'Google Maps API'. When 'gVisOutput=TRUE', the solution map is created and displayed. These two parameters will be tested in a similar way as the parameters considered in the previous subsubsection. We will try to see by executing the script with different specifications if either of these arguments have a significant effect on the execution time or the memory allocated. Then, by analyzing the algorithms of the function, we will try to determine why these arguments do or do not have an effect on the performance indicators considered. These parts of the algorithms are exactly the same for all the functions. Therefore, conclusions on the absolute effect of their specification on the performance indicators about one of the functions can be applied to the remaining ones and only one function has to be analyzed. By looking at the script of the function, it is easy to see that the only other argument that may have an influence on the performance of these two algorithm is the size of the problem. The distance matrix is constructed by n^2 distances. $n(n-1)$ of them are taken on the internet and this algorithm consists thus of $n(n-1)$ calls of the function 'distance_google'. The creation of the map needs to construct a 'dataframe' constituted of 3 vectors which length equal to n. The effect on the performance indicators will thus only be studied for one function ('SApHMP') and for different sizes. The execution time and memory used by the function 'USApHMP' will be recorded for 'gVisDist=gVisOutput=FALSE' and when each of them is set to 'TRUE' separately for problems of size 5 and 10.

Tables 24 and table 25 show the execution times and memory allocation of 10 calls of the function that solves the single allocation p-hub median problem for instances of size 5 and 10. The function has been called with the arguments both set to 'FALSE' and with each of them as 'TRUE' independently. The two algorithms are run sequentially at different parts of the functions. It is thus possible to predict the effect on the performance indicators of the specification of both of them as 'TRUE' by adding their individual effect together.

Table 24: Execution times of 'SApHMP' to solve single allocation p-hub median problems for instances of the 'CAB data set' with different specifications of 'gVisDist' and 'gVisOutput'.

size	arguments specification	Number of the call										
		1	2	3	4	5	6	7	8	9	10	mean
5												
	gVisDist=gVisOutput=FALSE	.04	.04	.045	.035	.04	.04	.045	.045	.045	.035	.041
	gVisDist=TRUE	21.895	21.91	21.9	21.915	21.845	21.885	21.86	21.905	21.865	21.865	21.885
	gVisOutput=TRUE	.17	.195	.145	.105	.11	.12	.09	.11	.085	.095	.123
10												
	gVisDist=gVisOutput=FALSE	1.615	1.545	1.685	1.56	1.615	2.01	1.675	1.6	1.89	1.85	1.705
	gVisDist=TRUE	101.52	100.525	100.55	100.56	100.58	100.63	100.94	100.465	100.485	100.685	100.694
	gVisOutput=TRUE	1.7	2.065	1.73	1.065	1.69	2.18	1.62	1.865	1.67	1.95	1.754

The execution times are expressed in seconds. The other arguments of the function are alpha=0.5, nHub=3 and cap=hCost=sCost=dis=gVisDisplayRegion=NULL. The execution times are measured with profvis with a sampling interval of 0.005 seconds.

Table 25: Memory used by 'SApHMP' to solve single allocation p-hub median problems for instances of the 'CAB data set' with different specifications of 'gVisDist' and 'gVisOutput'.

size	arguments specification	Number of the call										
		1	2	3	4	5	6	7	8	9	10	mean
5												
	gVisDist=gVisOutput=FALSE	3.2	3.2	3.2	3.2	3.2	3.2	3.2	4.6	3.2	3.2	3.34
	gVisDist=TRUE	2.1	3.5	3.5	4.9	3.5	3.4	3.5	3.4	3.4	3.4	3.46
	gVisOutput=TRUE	5.1	3.4	3.4	3.4	4.8	3.4	3.4	3.3	4.8	3.4	3.84
10												
	gVisDist=gVisOutput=FALSE	491.7	493.1	492.	491.	491.1	492.4	491.9	492.	491.	492.2	491.84
	gVisDist=TRUE	494.6	494.1	492.4	492.4	492.3	493.6	492.3	494.	494.1	494.1	493.39
	gVisOutput=TRUE	493.	491.2	492.	491.5	493.8	489.2	492.1	491.4	491.3	491.9	491.74

The memory is expressed in MB. The other arguments of the function are alpha=0.5, nHub=3 and cap=hCost=sCost=dis=gVisDisplayRegion=NULL. The memory used is measured with profvis with a sampling interval of 0.005 seconds.

Table 24 shows that only 'gVisDist' has a significant effect on the execution time of the function in absolute values. The data in the tables can be confirmed by looking at the algorithm of the R function. When 'gVisOutput=TRUE', two objects are created: a 'dataframe' with 5 arguments of which 4 are numeric vectors of length n, 1 is a character vector of size n and one is a unique character (the 4 numeric vectors are also allocated in the memory separately). The 'geoMap' object is also created (together with two characters that store the color and size configurations of the map). These objects require a negligible amount of memory in comparison to the memory allocated for the solver (these objects grow as O(n) at most whereas the constraint matrices of the different problems grow at least as $O(n^2)$). The memory allocated to create the solution map and the time these operations take will thus become insignificant in comparison to the rest of the algorithm when the size of the problem grows.

The part of the algorithm that constructs the distance matrix is only constituted by a loop of one call of the function ‘distance_google’ (see 3.3.4.) for each of the $n(n-1)$ pairs of two different nodes. The differences in mean execution time between the samples with ‘gVisDist=gVisOutput=FALSE’ and with ‘gVisDist=TRUE’ are 21.84 seconds for n=5 and 98.9895 seconds for n=10. That means that if we assume that the whole difference is caused by the algorithm that constructs the distance matrix, each loop (90 for n=10 and 20 for n =5) has an execution time of 1.01 seconds for n=10 and 1.092 for n=5. The fact that the mean execution time of one call of the function ‘distance_google’ found for n=5 and n=10 are very close supports the fact that its effect on the execution time of the function is close to $1.01\ n(n-1)$ seconds whatever the size of the problem. Most of this time (1 seconds) is spent waiting because the ‘Google Maps API’ does not allow more than one request per second or it will send an ‘OVER_QUERRY_LIMIT’ message without the distance requested. If we withdraw this idle time from the total execution time of the function, we find that it takes, on average, about 0.01 seconds to request the distance between two nodes from the internet. These conclusions about ‘USApHMP’ can be extended to the tree other main functions of the R implementation.

5. Conclusion

5.1. Theoretical contribution

The contribution of this thesis is an R algorithm that solves to optimality hub location problems (chapter 3). In its current state, it can solve single allocation p-hub median problems, single allocation p-hub center problems, single allocation hub covering problems and single allocation p-hub maximal covering problems with general assumptions[1]. The script can solve capacitated and uncapacitated p-hub median problems and can include the fixed costs to locate the hubs and to allocate the non-hub nodes to the hubs in the minimization of the total cost of the network. The optimization of the hub covering problem can minimize the number of hubs to locate or the cost to locate them.

This thesis also provides a review of the literature about hub location models (See chapter 2). It is mainly focused on the models that have the same assumptions than the models implemented. They are the basis of the literature about hub location problems as many models are modified versions of this basis that take specific aspects of the problem into consideration.

The executions of the R implementation of single allocation hub location models confirmed the optimal solutions for all the instances of the 'CAB data set'[2] that could be solved reported in the benchmark papers except for one (subsection 4.2.). For one out of the 60 instances of the single allocation p-hub center problem (15 nodes, 4 hubs and α=1), Ernst et al. (2009) reported an optimal objective function value of 2166.54 whereas the R implementation calculated 2600.78 (subsubsection 4.2.2.). As Ernst et al. (2009) did not provide the optimal hub location that leads to that optimal value, the two solutions could not be analyzed to determine which is correct. The two optimal objective function values were also both consistent with the optimal values calculated for similar problems and could not be ruled out that way. To our best knowledge, our computational analysis also provided optimal solutions for instances of the 'CAB data set' with 20 and 25 nodes and several values of α, β

[1] The assumptions of these models are explained in the second chapter.
[2] A presentation of the 'CAB data set' is available in subsection 4.1.

and p[1] that only had sub-optimal solutions reported in the literature for the p-hub maximal covering problem.

5.2. Managerial contribution

This thesis provides an R script that solves several hub location problems by providing the optimal solution as explained in the previous subsection. As such, it is a useful tool to help with the managerial decisions regarding the configuration of hub-and-spoke networks. The solution matrix can be displayed in two different ways depending on the specification of the nodes names which improves the readability of the optimal solution when they are specified. If a solution has been found, the R script also helps with the presentation of the optimal network configuration by offering the possibility to display the optimal solution on a map (see 3.3.3.). In order to ease the gathering of the parameters of the problem to solve, it is possible to have the distance matrix automatically requested from the internet by the algorithm. Finally, if no solution has been found, the script warns the user about possible reasons why no feasible solution could be found: for the capacitated p-hub median problem, the fact that there are not enough capacities in the potential hub locations is considered. For the hub covering problem, if two nodes can't be covered whatever the hub locations and nodes allocation, the script returns the nodes numbers or names.

These implementations solved most of the instances of the 'CAB data set' (25 or less nodes) considered in the computational results of benchmark papers (subsection 4.2.). The main exception is the R implementation of the p-hub median problem that could not solve the instances with 20 and 25 nodes. The size of the formulation implemented for that problem grows rapidly with the size of the problem to solve and exceeded the memory available for these instances. The script could also not solve the instances of the p-hub maximal covering problem considered in Peker and Kara (2015) in a reasonable amount of time. This is because the optimization time by the script changes a lot with variations of the value of β and the values they considered are in a range for which the execution time of the function sky-

[1] It is the number of hubs to locate.

rocketed. This increase of the time to solve the problem is solely due to the MILP[1] solver the script uses. For the p-hub center problem, the hub covering problem and the p-hub maximal covering problem, we could not assess the feasibility of instances involving more than 25 nodes as it is the maximum size that can be considered with the data set used. Our computational analysis provided evidences that support the fact that the algorithms of the functions were correct for the parts that involve the arguments specifying the flow matrix, the cost matrix, α, β and p as they calculated the same objective function values than in the benchmark papers (except for one out of the 60 instances considered for the p-hub center problem). The other parameters that can be considered in the resolution of the hub location problems by the R implementation were not considered by the benchmark papers and the verification of the correctness of the algorithms they are involved in was therefore not possible. For these parameters, the computations showed that only the consideration of the hubs maximum capacities for the single allocation p-hub median problem had a significant effect on the execution time of the function. We also concluded that the only option considered by the R functions that significantly increases the execution time is when 'gVisDist=TRUE'[2]. If it is the case, the resolution time of the problems considered involving n nodes increased by a little more than $n(n-1)$ seconds for the request of the distance matrix on the internet. The specification of these additional parameters and options did not have a significant effect on the memory allocated for the execution of the different functions (subsection 4.3.).

We think that the different parameters that can be considered and the different things that the script can do in order to facilitate its utilization make it a good tool to help with the configuration of small size hub networks in several situations. Of course this script only accounts for the mathematical side of the problem. When making a final decision, more parameters that cannot be taken into account by the mathematical implementation should be considered.

[1] Mixed Integer Linear Program

[2] 'gVisDist=TRUE' sets the option to request the distance matrix from the internet instead of being specified by the user.

5.4. Critique

The R implementation of the p-hub median problem for instances of the 'CAB data set' performed poorly in comparison to the benchmark paper (subsubsection 4.2.1.). It couldn't solve instances of the problem with 20 and 25 nodes. This is due to the number of variables of the model implemented that grows as $O(n^4)$ (Skorin-Kapov et al., 1996, p 587) which causes high memory requirements even to solve small problems. The size of the formulation could be reduced by implementing the model USApHMP-N of Ernst and Krishnamoorthy (1996) (see 2.1.1.). This last model has a number of variables that grows as $O(n^3)$ (Ernst and Krishnamoorthy, 1996, pp 8, 9) and less constraints than the model in Skorin-Kapov et al. (1996) and would therefore need less memory. The R implementation would therefore be able to solve larger problems.

The R function that solves single allocation p-hub median problems could have been slightly modified in order to solve single allocation hub location problems with fixed costs as explained in the "Further work" subsection. These modifications of the function should have been done for this thesis as the R implementation could then solve all the single allocation hub location problems of general assumptions (see 2. For the assumptions).

The consideration of different discount factors for the transportation costs on the connections between the non-hub nodes and the hubs like in the model UMApHMP-N of Ernst and Krishnamoorthy (1998) (see 2.1.2.) would have improved the computational analysis. The current state of the implementation only considers the discount factor on the inter-hub connections. By adding these two parameters, the computational results of papers that solve instances of the 'AP data set' could have been compared to the R script. This would have allowed to assess the performance of the R implementation for problems involving more than 25 nodes. For the p-hub center, the hub covering and the p-hub maximal covering problems, we could only conclude that the R implementation could solve all the instances of the problems considered (except for Peker and Kara (2015) which could solve problems with values of β that could not be solved by the R function 'USApHMCV'[1]).

[1] Function that solves the single allocation p-hub maximal covering problem.

Finally, the additional computations that are not linked to the benchmark of the script (subsection 4.3.1.) only consider one instance of the problem. In order to gather more evidence about the effect of the parameters considered on the performance indicators, it should have considered more instances of the problem. We could only conclude that the problem considered shows evidences that only the consideration of the hubs capacities had an effect on the memory allocation and the execution time of the function that solves the single allocation p-hub median problem.

5.3. Further work

Most of the suggestions for further work are linked to the current limitations of the R implementation. Firstly, the R script could be modified to allow for the consideration of more types of hub location problems. The function that solves the single allocation p-hub median problem can be modified to consider the hub location problems with fixed costs. Only few modifications would be required. In the current state of the function, the number of hubs to locate has to be specified and the vector with the capacities of the hub candidates does not. In order to consider the two types of hub location problems, both arguments would not have to be specified in all the situations. The function would then have to check that if 'nHub'[1] is not specified, 'cap'[2] needs to be entered by the user. If 'nHub' is defined, the fixed costs to locate the hubs can both be considered or not. The construction of the constraints matrix, the direction vector and the right hand side vector of the solver would also have to be modified by removing the constraint that sets the number of hubs to locate if 'nHub' is not specified. With these two modifications, the function 'SApHMP'[3] could consider both the single allocation p-hub median problem and the single allocation hub location problem with fixed costs.

This thesis only considers single allocation hub location problems. Further work can therefore include the implementation of the multiple allocation versions of the problems. It is

[1] It is the argument of the function that specify the exogenous number of hubs to locate.
[2] It is the argument of the function that specify the maximum capacities of the potential hub locations.
[3] It is the function that solves the single allocation p-hub median problem.

also possible to implement models that do not satisfy the same assumptions than the models implemented in this thesis like star hub location models or hub arcs location models (See 2.5.). The R implementation could also be modified to consider different parameters of the problems to solve like multi-period parameters, multimodal networks or hub disruptions. The possibility to choose only a subset of the nodes to be potential hub locations would be another improvement of the implementation.

Secondly, most of the time the different functions need to solve large instances problems is due to the MILP solver 'Rglpk' (Theussl et al., 2016a). The most effective way to improve the performance of the implementation is then to change the solver with one that performs better for the resolution of hub location problems or to write a new specific solver. A heuristic algorithm could also be proposed to solve instances of the problem that can't be solved in a reasonable amount of time. The rest of the implementation can also be improved even if it would have less marginal effect on the overall execution time of the functions for bigger instances of the problem.

Appendices

Appendix 1: R script for single allocation hub location problems

What follows is the R script that solves single allocation hub location problems. It can be copy-pasted into the R console after the installation of the packages it requires: stringr (Wickham, 2015), Rglpk (Theussl et al., 2016a) and googleVis (Gesmann et al., 2016).

Starting pages of the functions:


```
library(stringr)
library(slam)
library(Rglpk)
suppressPackageStartupMessages(library(googleVis))
library(googleVis)

#interrupts the execution of the script for x seconds
#source:https://stat.ethz.ch/R-manual/R-devel/library/base/html/Sys.sleep.html
interrupt <- function(x)
{
   p1 <- proc.time()
   Sys.sleep(x)
   proc.time() - p1
}


#returns the distance between adresse_depart and adresse_arrivée
#source: Delplace (2011)
distance_google <- function(adresse_depart, adresse_arrivee){
  # here we use the distance in m and convert to km
 distance <- ""
 adresse_depart <-as.character(adresse_depart)
 adresse_arrivee <-as.character(adresse_arrivee)
```

```
		if(adresse_depart != adresse_arrivee){
				adresse_depart <- str_replace_all(adresse_depart," ", "+")
				adresse_arrivee <- str_replace_all(adresse_arrivee, " ", "+")
				url <-
str_c("http://maps.googleapis.com/maps/api/distancematrix/xml?origins=", adresse_depart,
str_c("&destinations=", str_c(adresse_arrivee,"&units=metric&mode=driving&sensor=false")))
				net <- paste(url,sep="\n")
				myfile <- file(net)
				code_source <- readLines(myfile)
				pos <- 14 #length(code_source)-15
				pos1 <- str_locate(code_source[pos],"<value>")[2]
				pos2 <- str_locate(code_source[pos],"</value>")[1]
				distance <- str_replace(str_sub(code_source[pos],pos1+1,pos2-
1),",",".")
			}
		else {
			distance = "0" }
return(round(as.numeric(distance)/1000,0))
}


#returns the distances between all the locations in a vector
vectorizedDist<-function(locVector){
	distMat<-matrix(0,ncol=length(locVector),nrow=length(locVector))
	for(i in 1:length(locVector)){
		for(j in 1:length(locVector)){
			if(i!=j){
				distance<-distance_google(locVector[i],locVector[j])
				if(length(distance)==1){
					distMat[i,j]<-distance
					}
				else{
					cat("Error: too many Google Map API request")
					stop
					}
				interrupt(1)
				}
			}
		}
	return(distMat)
	}


#returns the sum of the N greatest arguments of a vector
maxVectN<-function(vect, n){
	total<-0
	index<-1
	currentMax<-0
	previousMax<-Inf
	while(n>0){
		for(i in 1:length(vect)){
			if(vect[i]==currentMax && n>1){
```

```
                                index<-index+1
                                }
                        if(vect[i]>currentMax && vect[i]<previousMax){
                                currentMax<-vect[i]
                                index<-1
                                }
                }
                total<-total+index*currentMax
                previousMax<-currentMax
                n<-n-index
                currentMax<-0
                index<-1
                }
        return(total)
}


#constructs the solution map with googlevis
mapSolution<-function(locNames, sizeVar, colorVar, hoverVar, gVisDisplayRegion){
                gVisDataFrame<-data.frame(locNames, sizeVar, colorVar, hoverVar)
                colo <- '[ "BLACK", "RED", "GREEN", "DARKBLUE" ]'
                size <- '{ minValue: 1, minSize: 6, maxValue: 2, maxSize: 12 }'
                gVisMap<-gvisGeoChart(gVisDataFrame, locationvar="locNames",
colorvar="colorVar", sizevar="sizeVar", hovervar="hoverVar",options=list(region=gVisDisplayRegion,
displayMode='markers', sizeAxis=size, colors=colo, backgroundColor="#81d4fa",
datalessRegionColor="#f5f5f5",  markerOpacity=0.8))
                return(gVisMap)
                }


#solves the single allocation p-hub median problem
#model: USApHMP-LP from Skorin-Kapov et al. (1996)
                #hub capacity from Ernst et Krishnamoorthy (1999) modified
                #fixed cost for spoke from Campbell (1994)
                #fixed cost for hub
SApHMP<-function(flow, cost, alpha, nHub, cap=NULL, hCost=NULL, sCost=NULL, dis=NULL,
locNames=NULL, gVisDist=FALSE, gVisOutput=FALSE, gVisDisplayRegion=NULL){
#input conditions check
        #flow
        if(!is.matrix(flow) && !is.data.frame(flow)){
                cat("Error: flow needs to be a matrix or a data.frame.", sep="/n", fill=TRUE)
                stop
                }
        flow<-as.matrix(flow)
        if(!is.numeric(flow)){
                cat("Error: flow needs to be numeric.", sep="/n", fill=TRUE)
                stop
                }
        if(ncol(flow)!=nrow(flow)){
                cat("Error: flow cannot be converted into a square matrix.")
                stop
                }
```

```
	diagCheck<-0
	for(i in 1:ncol(flow)){
		if(diag(flow)[i]!=0){
			diagCheck<-1
			}
		}
	if(diagCheck==1){
		cat("WARNING: The diagonal of the flow matrix has at least one non-zero argument.",
sep="/n", fill=TRUE)
		#stop
		}
	#cost
	if(!is.matrix(cost) && !is.data.frame(cost)){
		cat("Error: the unit flow costs (cost) specified needs to be a matrix or a data.frame.",
sep="/n", fill=TRUE)
		stop
		}
	cost<-as.matrix(cost)
	if(!is.numeric(cost)){
		cat("Error: the unit flow cost matrix (cost) need to be numeric.", sep="/n", fill=TRUE)
		stop
		}
	if(ncol(cost)!=nrow(cost)){
		cat("Error: the unit flow cost matrix (cost) cannot be converted into a square matrix.",
sep="/n", fill=TRUE)
		stop
		}
	diagCheck<-0
	for(i in 1:ncol(cost)){
		if(diag(cost)[i]!=0){
			diagCheck<-1
			}
		}
	if(diagCheck==1){
		cat("WARNING: The diagonal of the unit flow cost matrix has at least one non-zero
argument.", sep="/n", fill=TRUE)
		#stop
		}
	#alpha
	if(!is.numeric(alpha)){
		cat("Error: alpha is not a number.", sep="/n", fill=TRUE)
		stop
		}
	if(alpha<0 || alpha>1){
		cat("Error: Alpha is less than 0 or greater than 1.", sep="/n", fill=TRUE)
		stop
		}
	#nHub
	if(!is.numeric(nHub)){
		cat("Error: nHub is not a number.", sep="/n", fill=TRUE)
		stop
		}
```

```
	if(nHub!=floor(nHub)){
		cat("Error: The number of hubs to locate is not a natural number.", sep="/n",
fill=TRUE)
		stop
		}
	if(nHub<=0){
		cat("Error: The number of hubs to locate is 0 or less than zero.", sep="/n", fill=TRUE)
		stop
		}
	#cap
	if(!is.vector(cap) && !is.data.frame(cap) && !is.null(cap)){
		cat("Error: The maximum hub capacities (cap) specified need to be a vector or a
data.frame.", sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(cap)){
		cap<-as.vector(cap)
		if(!is.numeric(cap)){
			cat("Error: the maximum hub capacity needs to be numeric.", sep="/n",
fill=TRUE)
			stop
			}
		}
	#hCost
	if(!is.vector(hCost) && !is.data.frame(hCost) && !is.null(hCost)){
		cat("Error: the fixed costs to locate the hubs (hCost) specified need to be a vector or a
data.frame.", sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(hCost)){
		hCost<-as.vector(hCost)
		if(!is.numeric(hCost)){
			cat("Error: The fixed costs to locate the hubs (hCost) need to be numeric.",
sep="/n", fill=TRUE)
			stop
			}
		}
	#sCost
	if(!is.matrix(sCost) && !is.data.frame(sCost) && !is.null(sCost)){
		cat("Error: The fixed costs to allow service on the spokes (sCost) specified need to be
a matrix or a data.frame.", sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(sCost)){
		sCost<-as.matrix(sCost)
		if(!is.numeric(sCost)){
			cat("Error: The fixed costs to allow service on the spokes (sCost) need to be
numeric.", sep="/n", fill=TRUE)
			stop
			}
		if(ncol(sCost)!=nrow(sCost)){
			cat("Error: The fixed costs to allow service on the spokes (sCost) cannot be
```

```
converted into a square matrix.", sep="/n", fill=TRUE)
                        stop
                        }
                diagCheck<-0
                for(i in 1:ncol(sCost)){
                        if(diag(sCost)[i]!=0){
                                diagCheck<-1
                                }
                        }
                if(diagCheck==1){
                        cat("WARNING: The diagonal of the spokes fixed costs matrix (sCost) has at
least one non-zero argument.", sep="/n", fill=TRUE)
                        #stop
                        }
                }
        #dis
        if(gVisDist==FALSE){
                if(!is.matrix(dis) && !is.data.frame(dis) && !is.null(dis)){
                        cat("Error: The distances specified (dis) need to be a matrix or a data.frame.",
sep="/n", fill=TRUE)
                        stop
                        }
                if(!is.null(dis)){
                        dis<-as.matrix(dis)
                        if(!is.numeric(dis)){
                                cat("Error: the distances specified (dis) need to be a numeric matrix
or data.frame.", sep="/n", fill=TRUE)
                                stop
                                }
                        if(ncol(dis)!=nrow(dis)){
                                cat("Error: the distances specified (dis) cannot be converted into a
square matrix.", sep="/n", fill=TRUE)
                                stop
                                }
                        diagCheck<-0
                        for(i in 1:ncol(dis)){
                                if(diag(dis)[i]!=0){
                                        diagCheck<-1
                                        }
                                }
                        if(diagCheck==1){
                                cat("WARNING: The diagonal of the distances matrix (dis) has at least
one non-zero argument.", sep="/n", fill=TRUE)
                                #stop
                                }
                        }
                }
        #locNames
        if(!is.vector(locNames) && !is.data.frame(locNames) && !is.null(locNames)){
                cat("Error: The nodes names specified need to be a vector or a data.frame.",
sep="/n", fill=TRUE)
                stop
```

```
		}
	if(!is.null(locNames)){
		locNames<-as.vector(locNames)
		}
	#relative parameter characteristics
	size<-ncol(flow)
	size2<-size*size
	size3<-size*size*size
	size4<-size*size*size*size
	#cost
	if(ncol(cost)!=size){
		cat("Error: the unit flow cost matrix (cost) does not have the same size as the flow
matrix.", sep="/n", fill=TRUE)
		stop
		}
	#cap
	isCap<-0
	if(!is.null(cap)){
		isCap<-1
		if(length(cap)!=size){
			cat(paste(" Error: The length of the hub capacity vector is not", size, "."),
sep="/n", fill=TRUE)
			stop
			}
		}
	#hCost
	ishCost<-0
	if(!is.null(hCost)){
		ishCost<-1
		if(length(hCost)!=size){
			cat(paste(" Error: The length of the fixed costs to locate the hubs vector is
not", size, "."), sep="/n", fill=TRUE)
			stop
			}
		}
	#sCost
	issCost<-0
	if(!is.null(sCost)){
		issCost<-1
		if(ncol(sCost)!=size){
			cat(paste(" Error: The size of the fixed costs to allow service on the spokes
matrix is not", size, "."), sep="/n", fill=TRUE)
			stop
			}
		}
	#dis
	isDist<-0
	if(gVisDist==FALSE){
		if(!is.null(dis)){
			isDist<-1
			if(ncol(dis)!=size){
				cat(paste(" Error: The size of the distance matrix (dis) is not ", size,
```

```
"."), sep="/n", fill=TRUE)
                                        stop
                                        }
                                }
                        }
        #locNames
        isLocNames<-0
        if(!is.null(locNames)){
                isLocNames<-1
                if(length(locNames)!=size){
                        cat(paste(" Error: The size of the vector with the nodes names specified is not
", size, "."), sep="/n", fill=TRUE)
                        stop
                        }
                }
        #gVisDist
        if(gVisDist==TRUE){
                isDist<-1
                if(isLocNames==0){
                        cat("Error: GoogleVis cannot calculate the distances between the nodes as
the nodes names have not been provided.", sep="/n", fill=TRUE)
                        }
                dis<-vectorizedDist(locNames)
                }
        #gVisOutput
        if(gVisOutput==TRUE){
                if(!is.character(gVisDisplayRegion)&&!is.null(gVisDisplayRegion)){
                        cat("Error: The region to display in googleVis specified needs to be a string.",
sep="/n", fill=TRUE)
                        stop
                        }
                if(isLocNames==0){
                        cat("Error: No location names (locNames) have been specified for googleVis.",
sep="/n", fill=TRUE)
                        stop
                        }
                }
#Rglpk input creation
        #obj
        obj<-vector(length=(size4+size2))
                #min sumisumjsumksumm WijCijkmXijkm (*disijkm)
        for(i in 1:size){
                for(j in 1:size){
                        for(k in 1:size){
                                for(m in 1:size){
                                        if(isDist==0){
                                                obj[(i-1)*size3+(j-1)*size2+(k-1)*size+m]<-
(cost[i,k]+alpha*cost[k,m]+cost[m,j])*flow[i,j]
                                                }
                                        if(isDist==1){
                                                obj[(i-1)*size3+(j-1)*size2+(k-1)*size+m]<-
(cost[i,k]*dis[i,k]+alpha*cost[k,m]*dis[k,m]+cost[m,j]*dis[m,j])*flow[i,j]
```

```
                                        }
                                    }
                                }
                            }
                        }
        for(i in 1:size){
                for(k in 1:size){
                        obj[size4+(i-1)*size+k]<-0
                        }
                }
                #+sumk FkZkk
        if(ishCost==1){
                for(k in 1:size){
                        obj[size4+(k-1)*size+k]<-hCost[k]
                        }
                }
                #+sumisumk SikZik
        if(issCost==1){
                for(i in 1:size){
                        for(k in 1:size){
                                obj[size4+(i-1)*size+k]<-obj[size4+(i-1)*size+k]+sCost[i,k]
                                }
                        }
                }
        #mat
        mat<-matrix(0, ncol=(size4+size2), nrow=(1+size+size2+2*size3+isCap*size))
                #Sumk Zkk=p
        for(k in 1:size){
                mat[1,size4+(k-1)*size+k]<-1
                }
                #Sumk Zik=1 for all i
        for (i in 1:size){
                for (k in 1:size){
                        mat[1+i,size4+(i-1)*size+k]<-1
                        }
                }
                #Zkk-Zik>=0 for all i,k
        for (i in 1:size){
                for (k in 1:size){
                        mat[1+size+(i-1)*size+k,size4+(i-1)*size+k]<-mat[1+size+(i-1)*size+k,size4+(i-
1)*size+k]-1
                        mat[1+size+(i-1)*size+k,size4+(k-1)*size+k]<-mat[1+size+(i-1)*size+k,size4+(k-
1)*size+k]+1
                        }
                }
                #Summ Xijkm-Zik=0 for all i,j,k
        for(i in 1:size){
                for(j in 1:size){
                        for(k in 1:size){
                                mat[1+size+size2+(i-1)*size2+(j-1)*size+k,size4+(i-1)*size+k]<--1
                                for(m in 1:size){
                                        mat[1+size+size2+(i-1)*size2+(j-1)*size+k,(i-1)*size3+(j-
```

```
1)*size2+(k-1)*size+m]<-1
                                        }
                                }
                        }
                }
                #Sumk Xijkm-Zjm=0 for all i,j,m
        for(i in 1:size){
                for(j in 1:size){
                        for(m in 1:size){
                                mat[1+size+size2+size3+(i-1)*size2+(j-1)*size+m,size4+(j-1)*size+m]<-
-1
                                for(k in 1:size){
                                        mat[1+size+size2+size3+(i-1)*size2+(j-1)*size+m,(i-
1)*size3+(j-1)*size2+(k-1)*size+m]<-1
                                        }
                                }
                        }
                }
                #CapkZkk-Sumi(Oi+Di)Zik>=0 for all k
        if(isCap==1){
                oiDi<-vector(length=size)
                for(i in 1:size){
                        oiDi[i]<-0
                        for(j in 1:size){
                                oiDi[i]<-oiDi[i]+flow[i,j]+flow[j,i]
                                }
                        }
                for(k in 1:size){
                        mat[1+size+size2+2*size3+k,size4+(k-1)*size+k]<-
mat[1+size+size2+2*size3+k,size4+(k-1)*size+k]+cap[k]
                        for (i in 1:size){
                                mat[1+size+size2+2*size3+k,size4+(i-1)*size+k]<-
mat[1+size+size2+2*size3+k,size4+(i-1)*size+k]-oiDi[i]
                                }
                        }
                }
        #dir
        dir1<-vector(length=1+size+size2+2*size3+isCap*size)
        for(b in 1:(size+1)){
                dir1[b]<-"=="
                }
        for (b in 1:(size2)){
                dir1[1+size+b]<-">="
                }
        for(b in 1:(2*size3)){
                dir1[1+size+size2+b]<-"=="
                }
        if(isCap==1){
                for(b in 1:size){
                        dir1[1+size+size2+2*size3+b]<-">="
                        }
                }
```

```
	#rhs
	rhs<-vector(length=1+size+size2+2*size3+isCap*size)
	rhs[1]<-nHub
	for(b in 1:size){
		rhs[1+b]<-1
		}
	for(b in 1:(size2+2*size3+isCap*size)){
		rhs[1+size+b]<-0
		}
	#types
	types<-vector(length=size4+size2)
	for(b in 1:(size4+size2)){
		types[b]<-"B"
		}
	#max
	max1<-FALSE

#solver launch
	sol<-Rglpk_solve_LP(obj=obj, mat=mat, dir=dir1, rhs=rhs, types=types, max=max1)

#solution output
	solValue<-as.numeric(unlist(sol[1]))
	solMatrix<-as.vector(unlist(sol[2]))
	solStatus<-sol[3]
	#output if optimum found
	if(solStatus==0){
		#hub location output
		outputHubs<-vector(length=nHub)
		hubNumbers<-vector(length=nHub)
		incr<-1
		for(k in 1:size){
			if(solMatrix[size4+(k-1)*size+k]==1){
				hubNumbers[incr]<-k
				if(isLocNames==1){
					outputHubs[incr]<-locNames[k]
					}
				if(isLocNames==0){
					outputHubs[incr]<-k
					}
				incr<-incr+1
				}
			}
		#allocation output
		outputSpoke<-matrix(0,ncol=1,nrow=1)
		if(isLocNames==1){
			outputSpoke<-matrix(0,ncol=nHub,nrow=size)
			rownames(outputSpoke)<-locNames
			colnames(outputSpoke)<-outputHubs
			for(i in 1:size){
				for(j in 1:nHub){
					outputSpoke[i,j]<-solMatrix[size4+(i-1)*size+hubNumbers[j]]
					}
```

```
                        }
                    }
                if(isLocNames==0){
                    outputSpoke<-matrix(0,ncol=size,nrow=size)
                    for(i in 1:size){
                        for(j in 1:size){
                            outputSpoke[i,j]<-solMatrix[size4+(i-1)*size+j]
                            }
                        }
                    }
                #interhub flows output
                interHub<-matrix(0, nrow=nHub, ncol=nHub)
                rownames(interHub)<-outputHubs
                colnames(interHub)<-outputHubs
                for(i in 1:size){
                    for(j in 1:size){
                        for(k in 1:nHub){
                            for(m in 1:nHub){
                                #interHub[k,m]= sumi sumj (flow(i,j)*X(i,j,k,m))
                                interHub[k,m]<-interHub[k,m]+flow[i,j]*solMatrix[(i-
1)*size3+(j-1)*size2+(hubNumbers[k]-1)*size+hubNumbers[m]]
                                }
                            }
                        }
                }
                cat("An optimal solution has been found.", sep="/n", fill=TRUE)
                cat(paste("The optimal cost is:",solValue), sep="/n", fill=TRUE)
                cat(paste("The",nHub, "hub(s)are located at:"), sep="/n", fill=TRUE)
                cat(paste(outputHubs), sep=", ",fill=TRUE)
                cat("The node allocation to the hubs is:", sep="/n", fill=TRUE)
                print(outputSpoke)
                cat("The inter-hub flows are:", sep="/n", fill=TRUE)
                print(interHub)
                }
        #output if optimum not found
        if(solStatus!=0){
                cat("An optimal solution could not be found.", sep="/n", fill=TRUE)
                if(isCap==1){
                    totalFlow<-0
                    totalFlowI<-vector(length=size)
                    totalHubCapacity<-maxVectN(cap, nHub)
                    minFlowI<-Inf
                    #2*sum k 1->p (max capk)>=sumi sumj Wij
                    for(i in 1:size){
                        for(j in 1:size){
                            totalFlow<-totalFlow+flow[i,j]
                            }
                        }
                    if(2*totalFlow>totalHubCapacity){
                        cat(paste("It could not be found because the total flow to transport
on the network (=", totalFlow,") is greater than the sum of the",nHub," greatest hub capacities (=",
totalHubCapacity,")."), sep="/n", fill=TRUE)
```

```
                }
            }
            #capk>= sumj Wij for all i,k
            for(i in 1:size){
                totalFlowI[i]<-0
                for(j in 1:size){
                    totalFlowI[i]<-totalFlowI[i]+flow[i,j]+flow[j,i]
                    }
                }
            for(i in 1:size){
                if(totalFlowI[i]<minFlowI){
                    minFlowI<-totalFlowI[i]
                    }
            }
            for(k in 1:size){
                if(cap[k]<minFlowI){
                    cat(paste("Warning: the hub capacity of node ", k, " is smaller
than the total flow involving the node with the smallest flows. Therefore, no node can be allocated to
it."), sep="/n", fill=TRUE)
                    }
                }
            }
    #gVis output
    if(solStatus==0 && gVisOutput==TRUE){
        sizeVar<-vector(length=size)
        incr<-1
        for(i in 1:size){
            if(hubNumbers[incr]==i){
                sizeVar[i]<-2
                if(incr<nHub){
                    incr<-incr+1
                    }
                }
            else{
                sizeVar[i]<-1
                }
            }
        colorVar<-vector(length=size)
        for(i in 1:size){
            for(k in 1:size){
                if(solMatrix[size4+(i-1)*size+k]==1){
                    colorVar[i]<-k
                    }
                }
            }
        hoverVar<-vector(length=size)
        for(i in 1:size){
            for(k in 1:size){
                if(solMatrix[size4+(i-1)*size+k]==1){
                    if(i==k){
                        hoverVar[i]<-paste(locNames[i], "is a hub.")
                        }
```

```
                                        else{
                                                hoverVar[i]<-paste(locNames[i], " is allocated to
",locNames[k])
                                                }
                                        }
                                }
                        }
                geoMap<-mapSolution(locNames=locNames, sizeVar=sizeVar, colorVar=colorVar,
hoverVar=hoverVar,gVisDisplayRegion=gVisDisplayRegion)
                plot(geoMap)
                }
        #returns the solution
        if(solStatus==0){
                returnList<-list(solValue,outputHubs,outputSpoke,interHub)
                return(returnList)
                }
        if(solStatus!=0){
                return(NULL)
                }
        }



#solves the single allocation p-hub center problem
#model: USApHCP-rad from Ernst et al (2009)
USApHCP<-function(cost=NULL, alpha, nHub, locNames=NULL, gVisDist=FALSE, gVisOutput=FALSE,
gVisDisplayRegion=NULL){
#input conditions check
#cost
        if(gVisDist==FALSE){
                if(!is.matrix(cost) && !is.data.frame(cost)){
                        cat("Error: the costs (cost) specified need to be a matrix or a data.frame.",
sep="/n", fill=TRUE)
                        stop
                        }
                cost<-as.matrix(cost)
                if(!is.numeric(cost)){
                        cat("Error: the cost matrix (cost) specified need to be numeric.", sep="/n",
fill=TRUE)
                        stop
                        }
                if(ncol(cost)!=nrow(cost)){
                        cat("Error: the cost matrix (cost) cannot be converted into a square matrix.",
sep="/n", fill=TRUE)
                        stop
                        }
                diagCheck<-0
                for(i in 1:ncol(cost)){
                        if(diag(cost)[i]!=0){
                                diagCheck<-1
                                }
                        }
```

```
		if(diagCheck==1){
			cat("WARNING: The diagonal of the cost matrix has at least one non-zero
argument.", sep="/n", fill=TRUE)
			#stop
			}
		}
	#alpha
	if(!is.numeric(alpha)){
		cat("Error: alpha is not a number.", sep="/n", fill=TRUE)
		stop
		}
	if(alpha<0 || alpha>1){
		cat("Error: Alpha is less than 0 or greater than 1.", sep="/n", fill=TRUE)
		stop
		}
	#nHub
	if(!is.numeric(nHub)){
		cat("Error: nHub is not a number.", sep="/n", fill=TRUE)
		stop
		}
	if(nHub!=floor(nHub)){
		cat("Error: The number of hubs to locate is not a natural number.", sep="/n",
fill=TRUE)
		stop
		}
	if(nHub<=0){
		cat("Error: The number of hubs to locate is 0 or less than zero.", sep="/n", fill=TRUE)
		stop
		}
	#locNames
	if(!is.vector(locNames) && !is.data.frame(locNames) && !is.null(locNames)){
		cat("Error: The nodes names specified need to be a vector or a data.frame.",
sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(locNames)){
		locNames<-as.vector(locNames)
		}
	#relative parameter characteristics
	size<-ncol(cost)
	size2<-size*size
	#locNames
	isLocNames<-0
	if(!is.null(locNames)){
		isLocNames<-1
		if(length(locNames)!=size){
			cat(paste(" Error: The size of the vector with the nodes names specified is not
", size, "."), sep="/n", fill=TRUE)
			stop
			}
		}
	#gVisDist
```

```
	if(gVisDist==TRUE){
		if(isLocNames==0){
			cat("Error: GoogleVis cannot calculate the distances between the nodes as no
location names have been provided.", sep="/n", fill=TRUE)
			stop
			}
		cost<-vectorizedDist(locNames)
		}
	#gVisOutput
	if(gVisOutput==TRUE){
		if(!is.character(gVisDisplayRegion)&&!is.null(gVisDisplayRegion)){
			cat("Error: The region to display in googleVis specified needs to be a string.",
sep="/n", fill=TRUE)
			stop
			}
		if(isLocNames==0){
			cat("Error: No location names (locNames) have been specified for googleVis.",
sep="/n", fill=TRUE)
			stop
			}
		}
#Rglpk input creation
	#obj
	obj<-vector(length=(1+size+size2))
		#min Z
	obj[1]<-1
	for(i in 1:(size+size2)){
		obj[i+1]<-0
		}
	#mat
	mat<-matrix(0, ncol=(1+size+size2), nrow=(1+size+3*size2))
		#sumk Zkk=p
	for(k in 1:size){
		mat[1,1+size+(k-1)*size+k]<-1
		}
		#sumk Zik=1 for all i
	for(i in 1:size){
		for(k in 1:size){
			mat[1+i,1+size+(i-1)*size+k]<-1
			}
		}
		#Zik-Zkk<=0 for all i,k
	for(i in 1:size){
		for(k in 1:size){
			mat[1+size+(i-1)*size+k,1+size+(i-1)*size+k]<-mat[1+size+(i-
1)*size+k,1+size+(i-1)*size+k]+1
			mat[1+size+(i-1)*size+k,1+size+(k-1)*size+k]<-mat[1+size+(i-
1)*size+k,1+size+(k-1)*size+k]-1
			}
		}
		#rk-CikZik>=0 for all i,k
	for(i in 1:size){
```

```
		for(k in 1:size){
			mat[1+size+size2+(i-1)*size+k,1+k]<-1
			mat[1+size+size2+(i-1)*size+k,1+size+(i-1)*size+k]<--cost[i,k]
			}
		}
		#Z-rk-rm>=alpha Ckm for all k,m
	for(k in 1:size){
		for(m in 1:size){
			mat[1+size+2*size2+(k-1)*size+m,1]<-1
			mat[1+size+2*size2+(k-1)*size+m,1+k]<-mat[1+size+2*size2+(k-
1)*size+m,1+k]-1
			mat[1+size+2*size2+(k-1)*size+m,1+m]<-mat[1+size+2*size2+(k-
1)*size+m,1+m]-1
			}
		}
	#dir
	dir1<-vector(length=1+size+3*size2)
	for(i in 1:(size+1)){
		dir1[i]<-"=="
		}
	for(i in 1:size2){
		dir1[1+size+i]<-"<="
		}
	for(i in 1:(2*size2)){
		dir1[1+size+size2+i]<-">="
		}
	#rhs
	rhs<-vector(length=1+size+3*size2)
	rhs[1]<-nHub
	for(i in 1:size){
		rhs[1+i]<-1
		}
	for(i in 1:(2*size2)){
		rhs[1+size+i]<-0
		}
	for(k in 1:size){
		for(m in 1:size){
			rhs[1+size+2*size2+(k-1)*size+m]<-alpha*cost[k,m]
			}
		}
	#types
	types<-vector(length=1+size+size2)
	for(b in 1:(1+size)){
		types[b]<-"C"
		}
	for(b in 1:(size2)){
		types[1+size+b]<-"B"
		}
	#max
	max1<-FALSE

#solver launch
```

```
	sol<-Rglpk_solve_LP(obj=obj, mat=mat, dir=dir1, rhs=rhs, types=types, max=max1)
#solution output
	solValue<-as.numeric(unlist(sol[1]))
	solMatrix<-as.vector(unlist(sol[2]))
	solStatus<-sol[3]
	#output if optimum found
	if(solStatus==0){
		#hub location output
		outputHubs<-vector(length=nHub)
		hubNumbers<-vector(length=nHub)
		incr<-1
		for(k in 1:size){
			if(solMatrix[1+size+(k-1)*size+k]==1){
				hubNumbers[incr]<-k
				if(isLocNames==1){
					outputHubs[incr]<-locNames[k]
					}
				if(isLocNames==0){
					outputHubs[incr]<-k
					}
				incr<-incr+1
				}
			}
		#allocation output
		outputSpoke<-matrix(0,ncol=1,nrow=1)
		if(isLocNames==1){
			outputSpoke<-matrix(0,ncol=nHub,nrow=size)
			rownames(outputSpoke)<-locNames
			colnames(outputSpoke)<-outputHubs
			for(i in 1:size){
				for(j in 1:nHub){
					outputSpoke[i,j]<-solMatrix[1+size+(i-1)*size+hubNumbers[j]]
					}
				}
			}
		if(isLocNames==0){
			outputSpoke<-matrix(0,ncol=size,nrow=size)
			for(i in 1:size){
				for(j in 1:size){
					outputSpoke[i,j]<-solMatrix[1+size+(i-1)*size+j]
					}
				}
			}
		cat("An optimal solution has been found.", sep="/n", fill=TRUE)
		cat(paste("At optimum, the longest path of the network is:",solValue), sep="/n",
fill=TRUE)
		cat(paste("The",nHub, "hub(s)are located at:"), sep="/n", fill=TRUE)
		cat(paste(outputHubs), sep=", ",fill=TRUE)
		cat("The node allocation to the hubs is:", sep="/n", fill=TRUE)
		print(outputSpoke)
		}
	#output if optimum not found
```

```
	if(solStatus!=0){
		cat("An optimal solution could not be found.", sep="/n", fill=TRUE)
		}
	#gVis output
	if(solStatus==0 && gVisOutput==TRUE){
		sizeVar<-vector(length=size)
		incr<-1
		for(i in 1:size){
			if(hubNumbers[incr]==i){
				sizeVar[i]<-2
				if(incr<nHub){
					incr<-incr+1
					}
				}
			else{
				sizeVar[i]<-1
				}
			}
		colorVar<-vector(length=size)
		for(i in 1:size){
			for(k in 1:size){
				if(solMatrix[1+size+(i-1)*size+k]==1){
					colorVar[i]<-k
					}
				}
			}
		hoverVar<-vector(length=size)
		for(i in 1:size){
			for(k in 1:size){
				if(solMatrix[1+size+(i-1)*size+k]==1){
					if(i==k){
						hoverVar[i]<-paste(locNames[i], "is a hub.")
						}
					else{
						hoverVar[i]<-paste(locNames[i], " is allocated to
",locNames[k])
						}
					}
				}
			}
		geoMap<-mapSolution(locNames=locNames, sizeVar=sizeVar, colorVar=colorVar,
hoverVar=hoverVar, gVisDisplayRegion=gVisDisplayRegion)
		plot(geoMap)
		}
	#returns the solution
	if(solStatus==0){
		returnList<-list(solValue,outputHubs,outputSpoke)
		return(returnList)
		}
	if(solStatus!=0){
		return(NULL)
		}
```

```
	}


#solves the single allocation hub covering problem
#model: USApSCP-rad from Ernst et al. (2011)
USASCP<-function(cost=NULL, alpha, bet, hCost=NULL, locNames=NULL, gVisDist=FALSE,
gVisOutput=FALSE, gVisDisplayRegion=NULL){
#input conditions check
	#cost
	if(gVisDist==FALSE){
		if(!is.matrix(cost) && !is.data.frame(cost)){
			cat("Error: the costs (cost) specified need to be a matrix or a data.frame.",
sep="/n", fill=TRUE)
			stop
			}
		cost<-as.matrix(cost)
		if(!is.numeric(cost)){
			cat("Error: the cost matrix (cost) specified need to be numeric.", sep="/n",
fill=TRUE)
			stop
			}
		if(ncol(cost)!=nrow(cost)){
			cat("Error: the cost matrix (cost) cannot be converted into a square matrix.",
sep="/n", fill=TRUE)
			stop
			}
		diagCheck<-0
		for(i in 1:ncol(cost)){
			if(diag(cost)[i]!=0){
				diagCheck<-1
				}
			}
		if(diagCheck==1){
			cat("WARNING: The diagonal of the cost matrix has at least one non-zero
argument.", sep="/n", fill=TRUE)
			#stop
			}
		}
	#alpha
	if(!is.numeric(alpha)){
		cat("Error: alpha is not a number.", sep="/n", fill=TRUE)
		stop
		}
	if(alpha<0 || alpha>1){
		cat("Error: Alpha is less than 0 or greater than 1.", sep="/n", fill=TRUE)
		stop
		}
	#bet
	if(!is.numeric(bet)){
		print("Error: beta is not a number.")
		stop
```

```
		}
	if(bet<0){
		print("WARNING: beta is less than 0")
		stop
		}
	#hCost
	if(!is.vector(hCost) && !is.data.frame(hCost) && !is.null(hCost)){
		cat("Error: the fixed costs to locate the hubs (hCost) specified need to be a vector or a
data.frame.", sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(hCost)){
		hCost<-as.vector(hCost)
		if(!is.numeric(hCost)){
			cat("Error: The fixed costs to locate the hubs (hCost) need to be numeric.",
sep="/n", fill=TRUE)
			stop
			}
		}
	#locNames
	if(!is.vector(locNames) && !is.data.frame(locNames) && !is.null(locNames)){
		cat("Error: The nodes names specified need to be a vector or a data.frame.",
sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(locNames)){
		locNames<-as.vector(locNames)
		}
	#relative parameter characteristics
	#gVisDist
	if(gVisDist==TRUE){
		if(isLocNames==0){
			cat("Error: GoogleVis cannot calculate the distances between the nodes as
the nodes names have not been provided.", sep="/n", fill=TRUE)
			}
		cost<-vectorizedDist(locNames)
		}
	size<-ncol(cost)
	size2<-size*size
	#hCost
	ishCost<-0
	if(!is.null(hCost)){
		ishCost<-1
		if(length(hCost)!=size){
			cat(paste(" Error: The length of the fixed costs to locate the hubs vector is
not", size, "."), sep="/n", fill=TRUE)
			stop
			}
		}
	#locNames
	isLocNames<-0
	if(!is.null(locNames)){
```

```
		isLocNames<-1
		if(length(locNames)!=size){
			cat(paste(" Error: The size of the vector with the nodes names specified is not
", size, "."), sep="/n", fill=TRUE)
			stop
			}
		}
	#gVisOutput
	if(gVisOutput==TRUE){
		if(!is.character(gVisDisplayRegion)&&!is.null(gVisDisplayRegion)){
			cat("Error: The region to display in googleVis specified needs to be a string.",
sep="/n", fill=TRUE)
			stop
			}
		if(isLocNames==0){
			cat("Error: No location names (locNames) have been specified for googleVis.",
sep="/n", fill=TRUE)
			stop
			}
		}
#Rglpk input creation
	#obj
	obj<-vector(length=(size+size2))
	for(i in 1:(size+size2)){
		obj[i]<-0
		}
	if(ishCost==1){
	#min sumk FkZkk
		for(k in 1:size){
			obj[size+(k-1)*size+k]<-hCost[k]
			}
		}
	if(ishCost==0){
	#min sumk Zkk
		for(k in 1:size){
			obj[size+(k-1)*size+k]<-1
			}
		}
	#mat
	mat<-matrix(0, ncol=(size+size2), nrow=(size+3*size2))
	#sumk Zik=1 for all i
	for(i in 1:size){
		for(k in 1:size){
			mat[i,size+(i-1)*size+k]<-1
			}
		}
	#Zik-Zkk<=0 for all i,k
	for(i in 1:size){
		for(k in 1:size){
			mat[size+(i-1)*size+k,size+(i-1)*size+k]<-mat[size+(i-1)*size+k,size+(i-
1)*size+k]+1
			mat[size+(i-1)*size+k,size+(k-1)*size+k]<-mat[size+(i-1)*size+k,size+(k-
```

```
1)*size+k]-1
                        }
                }
        #rk-CikZik>=0 for all i,k
        for(i in 1:size){
                for(k in 1:size){
                        mat[size+size2+(i-1)*size+k,k]<-1
                        mat[size+size2+(i-1)*size+k,size+(i-1)*size+k]<--cost[i,k]
                        }
                }
        #rk+rm<=beta-alpha Ckm for all k,m
        for(k in 1:size){
                for(m in 1:size){
                        mat[size+2*size2+(k-1)*size+m,k]<-mat[size+2*size2+(k-1)*size+m,k]+1
                        mat[size+2*size2+(k-1)*size+m,m]<-mat[size+2*size2+(k-1)*size+m,m]+1
                        }
                }
        #dir
        dir1<-vector(length=size+3*size2)
        for(i in 1:size){
                dir1[i]<-"=="
                }
        for(i in 1:size2){
                dir1[size+i]<-"<="
                }
        for(i in 1:size2){
                dir1[size+size2+i]<-">="
                }
        for(i in 1:size2){
                dir1[size+2*size2+i]<-"<="
                }
        #rhs
        rhs<-vector(length=size+3*size2)
        for(i in 1:size){
                rhs[i]<-1
                }
        for(i in 1:(2*size2)){
                rhs[size+i]<-0
                }
        for(k in 1:size){
                for(m in 1:size){
                        rhs[size+2*size2+(k-1)*size+m]<-bet-alpha*cost[k,m]
                        }
                }
        #types
        types<-vector(length=size+size2)
        for(i in 1:size){
                types[i]<-"C"
                }
        for(i in 1:size2){
                types[size+i]<-"B"
                }
```

```
	#max
	max1<-FALSE

#solver launch
	sol<-Rglpk_solve_LP(obj=obj, mat=mat, dir=dir1, rhs=rhs, types=types, max=max1)
#solution output
	solValue<-as.numeric(unlist(sol[1]))
	solMatrix<-as.vector(unlist(sol[2]))
	solStatus<-sol[3]
	#output if optimum found
	if(solStatus==0){
		#hub location output
		nHub<-0
		for(k in 1:size){
			nHub<-nHub+solMatrix[size+(k-1)*size+k]
			}
		outputHubs<-vector(length=nHub)
		hubNumbers<-vector(length=nHub)
		incr<-1
		for(k in 1:size){
			if(solMatrix[size+(k-1)*size+k]==1){
				hubNumbers[incr]<-k
				if(isLocNames==1){
					outputHubs[incr]<-locNames[k]
					}
				if(isLocNames==0){
					outputHubs[incr]<-k
					}
				incr<-incr+1
				}
			}
		#allocation output
		outputSpoke<-matrix(0,ncol=1,nrow=1)
		if(isLocNames==1){
			outputSpoke<-matrix(0,ncol=nHub,nrow=size)
			rownames(outputSpoke)<-locNames
			colnames(outputSpoke)<-outputHubs
			for(i in 1:size){
				for(j in 1:nHub){
					outputSpoke[i,j]<-solMatrix[size+(i-1)*size+hubNumbers[j]]
					}
				}
			}
		if(isLocNames==0){
			outputSpoke<-matrix(0,ncol=size,nrow=size)
			for(i in 1:size){
				for(j in 1:size){
					outputSpoke[i,j]<-solMatrix[size+(i-1)*size+j]
					}
				}
			}
		cat("An optimal solution has been found.", sep="/n", fill=TRUE)
```

```
		if(ishCost==1){
			cat(paste("The optimal cost to locate the hubs is: ",solValue), sep="/n",
fill=TRUE)
			}
		if(ishCost==0){
			cat(paste("The optimal number of hubs to locate is: ",solValue), sep="/n",
fill=TRUE)
			}
		cat(paste("The",nHub, "hub(s)are located at:"), sep="/n", fill=TRUE)
		cat(paste(outputHubs), sep=", ",fill=TRUE)
		cat("The node allocation to the hubs is:", sep="/n", fill=TRUE)
		print(outputSpoke)
		}
	#output if optimum not found
	if(solStatus!=0){
		cat("An optimal solution could not be found.", sep="/n", fill=TRUE)
		covImpoIJ<-matrix(0, ncol=size,nrow=size)
		for(i in 1:size){
			for(j in 1:size){
				for(k in 1:size){
					for(m in 1:size){
						if(cost[i,k]+alpha*cost[k,m]+cost[m,j]<=bet){
							covImpoIJ[i,j]<-1
							}
						}
					}
				}
			}
		for(i in 1:size){
			for(j in 1:size){
				if(covImpoIJ[i,j]==0){
					cat(paste("It could not be found because the flow from ",i, "
to ", j, " cannot be covered by any combination of hubs."), sep="/n", fill=TRUE)
					}
				}
			}
		}
	#gVis output
	if(solStatus==0 && gVisOutput==TRUE){
		sizeVar<-vector(length=size)
		incr<-1
		for(i in 1:size){
			if(hubNumbers[incr]==i){
				sizeVar[i]<-2
				if(incr<nHub){
					incr<-incr+1
					}
				}
			else{
				sizeVar[i]<-1
				}
			}
```

```
		colorVar<-vector(length=size)
		for(i in 1:size){
			for(k in 1:size){
				if(solMatrix[size+(i-1)*size+k]==1){
					colorVar[i]<-k
					}
				}
			}
		hoverVar<-vector(length=size)
		for(i in 1:size){
			for(k in 1:size){
				if(solMatrix[size+(i-1)*size+k]==1){
					if(i==k){
						hoverVar[i]<-paste(locNames[i], "is a hub.")
						}
					else{
						hoverVar[i]<-paste(locNames[i], " is allocated to
",locNames[k])
						}
					}
				}
			}
		geoMap<-mapSolution(locNames=locNames, sizeVar=sizeVar, colorVar=colorVar,
hoverVar=hoverVar, gVisDisplayRegion=gVisDisplayRegion)
		plot(geoMap)
		}
	#returns the solution
	if(solStatus==0){
		returnList<-list(solValue,outputHubs,outputSpoke)
		return(returnList)
		}
	if(solStatus!=0){
		return(NULL)
		}
	}




#solves the single allocation p-hub maximal covering problem
#model: USApHMCV-P from Peker and Kara (2015)
USApHMCV<-function(flow, cost=NULL, alpha, bet, nHub, locNames=NULL, gVisDist=FALSE,
gVisOutput=FALSE, gVisDisplayRegion=NULL){
#input conditions check
	#flow
	if(!is.matrix(flow) && !is.data.frame(flow)){
		cat("Error: flow needs to be a matrix or a data.frame.", sep="/n", fill=TRUE)
		stop
		}
	flow<-as.matrix(flow)
	if(!is.numeric(flow)){
```

```
		cat("Error: flow needs to be numeric.", sep="/n", fill=TRUE)
		stop
		}
	if(ncol(flow)!=nrow(flow)){
		cat("Error: flow cannot be converted into a square matrix.")
		stop
		}
	diagCheck<-0
	for(i in 1:ncol(flow)){
		if(diag(flow)[i]!=0){
			diagCheck<-1
			}
		}
	if(diagCheck==1){
		cat("WARNING: The diagonal of the flow matrix has at least one non-zero argument.",
sep="/n", fill=TRUE)
		#stop
		}
	#cost
	if(gVisDist==FALSE){
		if(!is.matrix(cost) && !is.data.frame(cost)){
			cat("Error: the unit flow costs (cost) specified needs to be a matrix or a
data.frame.", sep="/n", fill=TRUE)
			stop
			}
		cost<-as.matrix(cost)
		if(!is.numeric(cost)){
			cat("Error: the unit flow costs matrix (cost) needs to be numeric.", sep="/n",
fill=TRUE)
			stop
			}
		if(ncol(cost)!=nrow(cost)){
			cat("Error: the unit flow cost matrix (cost) cannot be converted into a square
matrix.", sep="/n", fill=TRUE)
			stop
			}
		diagCheck<-0
		for(i in 1:ncol(cost)){
			if(diag(cost)[i]!=0){
				diagCheck<-1
				}
			}
		if(diagCheck==1){
			cat("WARNING: The diagonal of the unit flow cost matrix has at least one
non-zero argument.", sep="/n", fill=TRUE)
			#stop
			}
		}
	#alpha
	if(!is.numeric(alpha)){
		cat("Error: alpha is not a number.", sep="/n", fill=TRUE)
		stop
```

```
		}
	if(alpha<0 || alpha>1){
		cat("Error: Alpha is less than 0 or greater than 1.", sep="/n", fill=TRUE)
		stop
		}
	#bet
	if(!is.numeric(bet)){
		print("Error: beta is not a number.")
		stop
		}
	if(bet<0){
		print("WARNING: beta is less than 0")
		stop
		}
	#nHub
	if(!is.numeric(nHub)){
		cat("Error: nHub is not a number.", sep="/n", fill=TRUE)
		stop
		}
	if(nHub!=floor(nHub)){
		cat("Error: The number of hubs to locate is not a natural number.", sep="/n",
fill=TRUE)
		stop
		}
	if(nHub<=0){
		cat("Error: The number of hubs to locate is 0 or less than zero.", sep="/n", fill=TRUE)
		stop
		}
	#locNames
	if(!is.vector(locNames) && !is.data.frame(locNames) && !is.null(locNames)){
		cat("Error: The nodes names specified need to be a vector or a data.frame.",
sep="/n", fill=TRUE)
		stop
		}
	if(!is.null(locNames)){
		locNames<-as.vector(locNames)
		}
	#relative parameter characteristics
	size<-ncol(flow)
	size2<-size*size
	size3<-size*size*size
	#cost
	if(gVisDist==FALSE){
		if(ncol(cost)!=size){
			cat("Error: the unit flow cost matrix (cost) does not have the same size as the
flow matrix.", sep="/n", fill=TRUE)
			stop
			}
		}
	#locNames
	isLocNames<-0
	if(!is.null(locNames)){
```

```
		isLocNames<-1
		if(length(locNames)!=size){
			cat(paste(" Error: The size of the vector with the nodes names specified is not
", size, "."), sep="/n", fill=TRUE)
			stop
			}
		}
	#gVisDist
	if(gVisDist==TRUE){
		if(isLocNames==0){
			cat("Error: GoogleVis cannot calculate the distances between the nodes as
the nodes names have not been provided.", sep="/n", fill=TRUE)
			}
		cost<-vectorizedDist(locNames)
		}
	#gVisOutput
	if(gVisOutput==TRUE){
		if(!is.character(gVisDisplayRegion)&&!is.null(gVisDisplayRegion)){
			cat("Error: The region to display in googleVis specified needs to be a string.",
sep="/n", fill=TRUE)
			stop
			}
		if(isLocNames==0){
			cat("Error: No location names (locNames) have been specified for googleVis.",
sep="/n", fill=TRUE)
			stop
			}
		}
#Rglpk input creation
	#covering parameter
	cover<-array(0, dim=c(size,size,size,size))
	for(i in 1:size){
		for(j in 1:size){
			for(k in 1:size){
				for(m in 1:size){
					if((cost[i,k]+alpha*cost[k,m]+cost[m,j])<=bet){
						cover[i,j,k,m]<-1
						}
					else{
						cover[i,j,k,m]<-0
						}
					}
				}
			}
		}
	lamb<-matrix(0, nrow=size, ncol=size)
	for(i in 1:size){
		for(j in 1:size){
			for(k in 1:size){
				for(m in 1:size){
					if(cover[i,j,k,m]==1){
						lamb[i,j]<-1
```

```
                                                }
                                        }
                                }
                        }
                }
        #obj
        obj<-vector(length=(2*size2))
        for(i in 1:size){
                for(j in 1:size){
                        #sumisumj WijFij
                        obj[(i-1)*size+j]<-0
                        obj[size2+(i-1)*size+j]<-flow[i,j]
                        }
                }
        #mat
        mat<-matrix(0, ncol=(2*size2), nrow=(1+size+size2+size3))
        #Sumk Zkk=p
        for(k in 1:size){
                mat[1,(k-1)*size+k]<-1
                }
        #sumk Zik=1 for all i
        for(i in 1:size){
                for(k in 1:size){
                        mat[1+i,(i-1)*size+k]<-1
                        }
                }
        #Zik-Zkk<=0 for all i,k
        for(i in 1:size){
                for(k in 1:size){
                        mat[1+size+(i-1)*size+k,(i-1)*size+k]<-mat[1+size+(i-1)*size+k,(i-1)*size+k]+1
                        mat[1+size+(i-1)*size+k,(k-1)*size+k]<-mat[1+size+(i-1)*size+k,(k-1)*size+k]-1
                        }
                }
        #Fij-sumk cover(ijkm)Zik+lamb(ij)Zjm<=lamb(ij) for all i,j,m
        for(i in 1:size){
                for(j in 1:size){
                        for(m in 1:size){
                                mat[1+size+size2+(i-1)*size2+(j-1)*size+m,size2+(i-1)*size+j]<-1
                                mat[1+size+size2+(i-1)*size2+(j-1)*size+m,(j-1)*size+m]<-
mat[1+size+size2+(i-1)*size2+(j-1)*size+m,(j-1)*size+m]+lamb[i,j]
                                for(k in 1:size){
                                        mat[1+size+size2+(i-1)*size2+(j-1)*size+m,(i-1)*size+k]<-
mat[1+size+size2+(i-1)*size2+(j-1)*size+m,(i-1)*size+k]-cover[i,j,k,m]
                                        }
                                }
                        }
                }
        #dir
        dir1<-vector(length=1+size+size2+size3)
        for(i in 1:(size+1)){
                dir1[i]<-"=="
                }
```

```
	for(i in 1:(size2+size3)){
		dir1[1+size+i]<-"<="
		}
	#rhs
	rhs<-vector(length=1+size+size2+size3)
	rhs[1]<-nHub
	for(i in 1:size){
		rhs[1+i]<-1
		}
	for(i in 1:size2){
		rhs[1+size+i]<-0
		}
	for(i in 1:size){
		for(j in 1:size){
			for(m in 1:size){
				rhs[1+size+size2+(i-1)*size2+(j-1)*size+m]<-lamb[i,j]
				}
			}
		}
	#types
	types<-vector(length=2*size2)
	for(i in 1:size2){
		types[size2+i]<-"C"
		types[i]<-"B"
		}
	#max
	max1<-TRUE

#solver launch
	sol<-Rglpk_solve_LP(obj=obj, mat=mat, dir=dir1, rhs=rhs, types=types, max=max1)
#solution output
	solValue<-as.numeric(unlist(sol[1]))
	solMatrix<-as.vector(unlist(sol[2]))
	solStatus<-sol[3]
	#output if optimum found
	if(solStatus==0){
		#hub location output
		outputHubs<-vector(length=nHub)
		hubNumbers<-vector(length=nHub)
		incr<-1
		for(k in 1:size){
			if(solMatrix[(k-1)*size+k]==1){
				hubNumbers[incr]<-k
				if(isLocNames==1){
					outputHubs[incr]<-locNames[k]
					}
				if(isLocNames==0){
					outputHubs[incr]<-k
					}
				incr<-incr+1
				}
			}
```

```
		#allocation output
		outputSpoke<-matrix(0,ncol=1,nrow=1)
		if(isLocNames==1){
			outputSpoke<-matrix(0,ncol=nHub,nrow=size)
			rownames(outputSpoke)<-locNames
			colnames(outputSpoke)<-outputHubs
			for(i in 1:size){
				for(j in 1:nHub){
					outputSpoke[i,j]<-solMatrix[(i-1)*size+hubNumbers[j]]
					}
				}
			}
		if(isLocNames==0){
			outputSpoke<-matrix(0,ncol=size,nrow=size)
			for(i in 1:size){
				for(j in 1:size){
					outputSpoke[i,j]<-solMatrix[(i-1)*size+j]
					}
				}
			}
		#covered flow output
		outputCovered<-matrix(0,ncol=size,nrow=size)
		for(i in 1:size){
			for(j in 1:size){
				outputCovered[i,j]<-solMatrix[size2+(i-1)*size+j]
				}
			}
		cat("An optimal solution has been found.", sep="/n", fill=TRUE)
		cat(paste("The maximum quantity of flow that can be covered by ", nHub, " hubs
is:",solValue), sep="/n", fill=TRUE)
		cat(paste("The",nHub, "hub(s)are located at:"), sep="/n", fill=TRUE)
		cat(paste(outputHubs), sep=", ",fill=TRUE)
		cat("The node allocation to the hubs is:", sep="/n", fill=TRUE)
		print(outputSpoke)
		cat("The flows covered are: ", sep="/n", fill=TRUE)
		print(outputCovered)
		}
	#output if optimum not found
	if(solStatus!=0){
		cat("An optimal solution could not be found.", sep="/n", fill=TRUE)
		}
	#gVis output
	if(solStatus==0 && gVisOutput==TRUE){
		sizeVar<-vector(length=size)
		incr<-1
		for(i in 1:size){
			if(hubNumbers[incr]==i){
				sizeVar[i]<-2
				if(incr<nHub){
					incr<-incr+1
					}
				}
```

```
                else{
                        sizeVar[i]<-1
                        }
                }
        colorVar<-vector(length=size)
        for(i in 1:size){
                for(k in 1:size){
                        if(solMatrix[(i-1)*size+k]==1){
                                colorVar[i]<-k
                                }
                        }
                }
        hoverVar<-vector(length=size)
        for(i in 1:size){
                for(k in 1:size){
                        if(solMatrix[(i-1)*size+k]==1){
                                if(i==k){
                                        hoverVar[i]<-paste(locNames[i], "is a hub.")
                                        }
                                else{
                                        hoverVar[i]<-paste(locNames[i], " is allocated to
",locNames[k])
                                        }
                                }
                        }
                }
        geoMap<-mapSolution(locNames=locNames, sizeVar=sizeVar, colorVar=colorVar,
hoverVar=hoverVar, gVisDisplayRegion=gVisDisplayRegion)
        plot(geoMap)
        }
#returns the solution
if(solStatus==0){
        returnList<-list(solValue,outputHubs,outputSpoke,outputCovered)
        return(returnList)
        }
if(solStatus!=0){
        return(NULL)
        }
}
```

Appendix 2: Aggregation algorithm of Wagner (2008)

The following is the aggregation algorithm of Wagner (2008) for the hub covering problem (see 2.4.1.). It is applied on any pair (i,j) such that i<j :

"1. $M = \{(k,m)|(i,k,j,m) \in IA\}$

2. if $M = \emptyset$ $terminate$

3. $k^{max} = \arg\max_{\mathrm{k}} |\{(k,m)|(k,m) \in M\}|$

   $m^{max} = \arg\max_{\mathrm{m}} |\{(k,m)|(k,m) \in M\}|$

4. if $|\{m|(k^{max},m) \in M\}| < |\{k|(k,m^{max}) \in M\}|$ go to Step 7

5. add constraint '$Z_{ik^{max}} + \sum_{m\ with\ (k^{max},m) \in M} Z_{jm} \leq 1$';

   $M = M \setminus \{(k^{max},m)|(k^{max},m) \in M\}$

6. go to step 2

7. add constraint '$Z_{jm^{max}} + \sum_{k\ with\ (k,m^{max}) \in M} Z_{ik} \leq 1$';

   $M = M \setminus \{(k,m^{max})|(k,m^{max}) \in M\}$

8. go to step 2" (Wagner, 2008, p 934).

Appendix 3: The ‘CAB data set’ (see 4.1.)

Table 1: Distance matrix of the ‘CAB data set’ (part 1). (Brunel University London, n.a.)

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	577.	946.5	597.6	373.8	559.8	709.	1208.3	603.6	695.2	680.7	1936.6
2	577.	0.0	369.5	613.	429.1	312.9	1196.5	1502.1	405.9	1242.	960.3	2318.1
3	946.5	369.5	0.0	858.3	749.6	556.1	1541.3	1764.8	621.3	1603.2	1251.	2600.1
4	597.6	613.	858.3	0.0	255.	311.3	790.1	907.4	237.1	932.2	406.3	1741.9
5	373.8	429.1	749.6	255.	0.0	225.9	794.2	1080.4	238.9	879.6	533.2	1889.5
6	559.8	312.9	556.1	311.3	225.9	0.0	1009.7	1216.9	94.3	1104.6	694.9	2047.1
7	709.	1196.5	1541.3	790.1	794.2	1009.7	0.0	663.9	982.7	221.4	447.8	1249.8
8	1208.3	1502.1	1764.8	907.4	1080.4	1216.9	663.9	0.0	1143.8	874.5	551.6	841.6
9	603.6	405.9	621.3	237.1	238.9	94.3	982.7	1143.8	0.0	1094.9	636.9	1978.9
10	695.2	1242.	1603.2	932.2	879.6	1104.6	221.4	874.5	1094.9	0.0	642.2	1375.6
11	680.7	960.3	1251.	406.3	533.2	694.9	447.8	551.6	636.9	642.2	0.0	1358.2
12	1936.6	2318.1	2600.1	1741.9	1889.5	2047.1	1249.8	841.6	1978.9	1375.6	1358.2	0.0
13	332.5	786.6	1137.3	485.6	402.3	627.1	411.1	880.1	620.5	477.5	378.6	1608.1
14	592.6	949.6	1266.9	1186.9	947.3	1084.5	1097.6	1714.7	1151.9	963.7	1236.2	2335.8
15	908.8	938.7	1124.8	345.9	598.5	626.2	851.8	694.	535.	1046.1	405.1	1530.6
16	426.2	999.5	1368.3	830.4	700.4	922.3	423.7	1066.6	936.3	305.3	674.5	1661.8
17	756.2	179.2	190.3	720.5	578.3	409.4	1362.9	1625.9	489.6	1417.1	1096.7	2453.4
18	672.6	96.3	274.3	675.3	512.4	365.7	1289.	1574.8	453.3	1337.6	1038.6	2396.8
19	1590.2	1999.6	2299.4	1447.1	1570.7	1743.4	895.1	593.4	1682.5	1017.3	1048.5	358.4
20	527.3	210.8	494.2	403.9	255.7	104.6	1049.3	1301.5	198.9	1125.	768.2	2125.5
21	483.5	736.4	1043.5	255.9	307.3	491.1	537.6	781.	450.3	677.1	229.5	1582.4
22	2141.	2456.3	2703.4	1853.6	2036.1	2164.9	1493.8	955.8	2086.8	1649.6	1506.5	361.5
23	2184.4	2339.5	2503.8	1733.1	1967.3	2027.3	1686.7	1024.6	1936.3	1891.2	1503.8	986.8
24	408.2	844.2	1188.5	1005.8	775.2	933.2	912.2	1519.2	992.3	795.2	1038.6	2157.5
25	540.7	36.5	405.8	592.	399.2	298.8	1161.7	1475.5	392.9	1205.7	931.7	2288.7

Table 2: Distance matrix of the ‘CAB data set’ (part 2). (Brunel University London, n.a.)

	13	14	15	16	17	18	19	20	21	22	23	24	25
1	332.5	592.6	908.8	426.2	756.2	672.6	1590.2	527.3	483.5	2141.	2184.4	408.2	540.7
2	786.6	949.6	938.7	999.5	179.2	96.3	1999.6	210.8	736.4	2456.3	2339.5	844.2	36.5
3	1137.3	1266.9	1124.8	1368.3	190.3	274.3	2299.4	494.2	1043.5	2703.4	2503.8	1188.5	405.8
4	485.6	1186.9	345.9	830.4	720.5	675.3	1447.1	403.9	255.9	1853.6	1733.1	1005.8	592.
5	402.3	947.3	598.5	700.4	578.3	512.4	1570.7	255.7	307.3	2036.1	1967.3	775.2	399.2
6	627.1	1084.5	626.2	922.3	409.4	365.7	1743.4	104.6	491.1	2164.9	2027.3	933.2	298.8
7	411.1	1097.6	851.8	423.7	1362.9	1289.	895.1	1049.3	537.6	1493.8	1686.7	912.2	1161.7
8	880.1	1714.7	694.	1066.6	1625.9	1574.8	593.4	1301.5	781.	955.8	1024.6	1519.2	1475.5
9	620.5	1151.9	535.	936.3	489.6	453.3	1682.5	198.9	450.3	2086.8	1936.3	992.3	392.9
10	477.5	963.7	1046.1	305.3	1417.1	1337.6	1017.3	1125.	677.1	1649.6	1891.2	795.2	1205.7
11	378.6	1236.2	405.1	674.5	1096.7	1038.6	1048.5	768.2	229.5	1506.5	1503.8	1038.6	931.7
12	1608.1	2335.8	1530.6	1661.8	2453.4	2396.8	358.4	2125.5	1582.4	361.5	986.8	2157.5	2288.7
13	0.0	858.3	700.8	348.3	955.6	880.	1265.6	651.1	255.	1808.5	1872.7	660.5	751.5
14	858.3	0.0	1500.8	675.8	1098.3	1021.6	1977.6	1015.2	1065.6	2591.4	2725.8	197.8	923.2
15	700.8	1500.8	0.0	1039.8	1018.4	987.9	1280.7	728.4	450.4	1589.8	1401.3	1311.2	922.3
16	348.3	675.8	1039.8	0.0	1178.4	1095.7	1304.	918.6	602.	1916.6	2090.1	496.4	963.
17	955.6	1098.3	1018.4	1178.4	0.0	84.3	2143.6	328.8	880.5	2574.1	2415.5	1008.2	215.6
18	880.	1021.6	987.9	1095.7	84.3	0.0	2082.3	273.4	818.1	2526.6	2388.7	926.6	132.8
19	1265.6	1977.6	1280.7	1304.	2143.6	2082.3	0.0	1814.8	1264.2	661.7	1129.3	1800.1	1968.7
20	651.1	1015.2	728.4	918.6	328.8	273.4	1814.8	0.0	552.4	2253.2	2128.8	875.3	194.6
21	255.	1065.6	450.4	602.	880.5	818.1	1264.2	552.4	0.0	1735.9	1712.1	871.6	706.5
22	1808.5	2591.4	1589.8	1916.6	2574.1	2526.6	661.7	2253.2	1735.9	0.0	694.9	2404.8	2430.3
23	1872.7	2725.8	1401.3	2090.1	2415.5	2388.7	1129.3	2128.8	1712.1	694.9	0.0	2528.5	2321.9
24	660.5	197.8	1311.2	496.4	1008.2	926.6	1800.1	875.3	871.6	2404.8	2528.5	0.0	813.6
25	751.5	923.2	922.3	963.	215.6	132.8	1968.7	194.6	706.5	2430.3	2321.9	813.6	0.0

Table 3: Flow matrix of the 'CAB data set' (part 1). (Brunel University London, n.a.)

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	6469	7629	20036	4690	6194	11688	2243	8857	7248	3559	9221
2	6469	0	12999	13692	3322	5576	3878	3202	6699	4198	2454	7975
3	7629	12999	0	35135	5956	14121	5951	5768	16578	4242	3365	22254
4	20036	13692	35135	0	19094	35119	21423	27342	51341	15826	28537	65387
5	4690	3322	5956	19094	0	7284	3102	1562	7180	1917	2253	5951
6	6194	5576	14121	35119	7284	0	5023	3512	10419	3543	2752	14412
7	11688	3878	5951	21423	3102	5023	0	11557	6479	34261	10134	27350
8	2243	3202	5768	27342	1562	3512	11557	0	5615	7095	10753	30362
9	8857	6699	16578	51341	7180	10419	6479	5615	0	4448	5076	22463
10	7248	4198	4242	15826	1917	3543	34261	7095	4448	0	4370	17267
11	3559	2454	3365	28537	2253	2752	10134	10753	5076	4370	0	15287
12	9221	7975	22254	65387	5951	14412	27350	30362	22463	17267	15287	0
13	10099	1186	1841	12980	1890	2043	6929	1783	4783	3929	3083	5454
14	22866	7443	23665	44097	7097	15642	7961	3437	24609	8602	4092	15011
15	3388	1162	6517	51525	2009	5014	4678	8897	9969	2753	7701	17714
16	9986	5105	3541	14354	1340	2016	13511	2509	4224	20013	2809	10037
17	46618	24817	205088	172895	25303	62034	29801	23273	79945	28080	17291	105507
18	11639	6532	37669	37305	6031	15385	7549	5160	20001	5971	4462	20040
19	1380	806	2885	15418	1041	2957	5550	8750	4291	2131	3239	31780
20	5261	8184	13200	26221	4128	5035	3089	2583	10604	3579	2309	10822
21	5985	3896	7116	42303	5452	7482	9958	7288	11925	6809	16003	16450
22	6731	7333	17165	35303	3344	6758	14110	17481	13091	8455	8381	92083
23	2704	3719	4284	13618	1067	2191	4911	7930	4172	2868	3033	32908
24	12250	2015	8085	17580	4608	6599	2722	1278	12891	2336	1755	3865
25	16132	565	51895	40708	7050	14181	10802	8447	19500	5616	7266	24583

Table 4: Flow matrix of the 'CAB data set' (part 2). (Brunel University London, n.a.)

	13	14	15	16	17	18	19	20	21	22	23	24	25
1	10099	22866	3388	9986	46618	11639	1380	5261	5985	6731	2704	12250	16132
2	1186	7443	1162	5105	24817	6532	806	8184	3896	7333	3719	2015	565
3	1841	23665	6517	3541	205088	37669	2885	13200	7116	17165	4284	8085	51895
4	12980	44097	51525	14354	172895	37305	15418	26221	42303	35303	13618	17580	40708
5	1890	7097	2009	1340	25303	6031	1041	4128	5452	3344	1067	4608	7050
6	2043	15642	5014	2016	62034	15385	2957	5035	7482	6758	2191	6599	14181
7	6929	7961	4678	13511	29801	7549	5550	3089	9958	14110	4911	2722	10802
8	1783	3437	8897	2509	23273	5160	8750	2583	7288	17481	7930	1278	8447
9	4783	24609	9969	4224	79945	20001	4291	10604	11925	13091	4172	12891	19500
10	3929	8602	2753	20013	28080	5971	2131	3579	6809	8455	2868	2336	5616
11	3083	4092	7701	2809	17291	4462	3239	2309	16003	8381	3033	1755	7266
12	5454	15011	17714	10037	105507	20040	31780	10822	16450	92083	32908	3865	24583
13	0	3251	1126	5926	10653	3062	759	1255	6173	2974	1056	1504	4588
14	3251	0	5550	9473	169397	25073	1170	14272	8543	8064	1840	20618	20937
15	1126	5550	0	2152	26816	6931	4947	2676	8033	12692	6157	3065	12044
16	5926	9473	2152	0	21806	4519	886	1742	4782	6453	2022	3546	5065
17	10653	169397	26816	21806	0	9040	11139	63153	34092	70935	14957	28398	166694
18	3062	25073	6931	4519	9040	0	2802	30224	7982	14964	4589	6227	12359
19	759	1170	4947	886	11139	2802	0	1869	3716	11510	3519	569	3520
20	1255	14272	2676	1742	63153	30224	1869	0	5020	6610	2139	5431	13541
21	6173	8543	8033	4782	34092	7982	3716	5020	0	9942	3276	3820	11799
22	2974	8064	12692	6453	70935	14964	11510	6610	9942	0	35285	2566	19926
23	1056	1840	6157	2022	14957	4589	3519	2139	3276	35285	0	940	4951
24	1504	20618	3065	3546	28398	6227	569	5431	3820	2566	940	0	6237
25	4588	20937	12044	5065	166694	12359	3520	13541	11799	19926	4951	6237	0

Nodes names of the ‘CAB data set’ (Brunel University London, n.a.):

1 Atlanta
2 Baltimore
3 Boston
4 Chicago
5 Cincinnati
6 Cleveland
7 Dallas-Fort Worth
8 Denver
9 Detroit
10 Houston
11 Kansas City
12 Los Angeles
13 Memphis
14 Miami
15 Minneapolis
16 New Orleans
17 New York
18 Philadelphia
19 Phoenix
20 Pittsburgh
21 St. Louis
22 San Francisco
23 Seattle
24 Tampa
25 Washington DC

R input of the ‘CAB data set’, 25 nodes network (Brunel University London, n.a.):

It can be copy-pasted in order to reproduce the results of our computational analysis.

```
cabflow<-matrix(0, 25, 25)
cabcost<-matrix(0, 25, 25)
cablocNames<-vector(length=25)

cabflow[ 1 , 1 ]<- 0
cabflow[ 1 , 2 ]<- 6469
cabflow[ 1 , 3 ]<- 7629
cabflow[ 1 , 4 ]<- 20036
cabflow[ 1 , 5 ]<- 4690
cabflow[ 1 , 6 ]<- 6194
cabflow[ 1 , 7 ]<- 11688
cabflow[ 1 , 8 ]<- 2243
cabflow[ 1 , 9 ]<- 8857
cabflow[ 1 , 10 ]<- 7248
cabflow[ 1 , 11 ]<- 3559
```

```
cabflow[ 1 , 12 ]<- 9221
cabflow[ 1 , 13 ]<- 10099
cabflow[ 1 , 14 ]<- 22866
cabflow[ 1 , 15 ]<- 3388
cabflow[ 1 , 16 ]<- 9986
cabflow[ 1 , 17 ]<- 46618
cabflow[ 1 , 18 ]<- 11639
cabflow[ 1 , 19 ]<- 1380
cabflow[ 1 , 20 ]<- 5261
cabflow[ 1 , 21 ]<- 5985
cabflow[ 1 , 22 ]<- 6731
cabflow[ 1 , 23 ]<- 2704
cabflow[ 1 , 24 ]<- 12250
cabflow[ 1 , 25 ]<- 16132
cabflow[ 2 , 1 ]<- 6469
cabflow[ 2 , 2 ]<- 0
cabflow[ 2 , 3 ]<- 12999
cabflow[ 2 , 4 ]<- 13692
cabflow[ 2 , 5 ]<- 3322
cabflow[ 2 , 6 ]<- 5576
cabflow[ 2 , 7 ]<- 3878
cabflow[ 2 , 8 ]<- 3202
cabflow[ 2 , 9 ]<- 6699
cabflow[ 2 , 10 ]<- 4198
cabflow[ 2 , 11 ]<- 2454
cabflow[ 2 , 12 ]<- 7975
cabflow[ 2 , 13 ]<- 1186
cabflow[ 2 , 14 ]<- 7443
cabflow[ 2 , 15 ]<- 1162
cabflow[ 2 , 16 ]<- 5105
cabflow[ 2 , 17 ]<- 24817
cabflow[ 2 , 18 ]<- 6532
cabflow[ 2 , 19 ]<- 806
cabflow[ 2 , 20 ]<- 8184
cabflow[ 2 , 21 ]<- 3896
cabflow[ 2 , 22 ]<- 7333
cabflow[ 2 , 23 ]<- 3719
cabflow[ 2 , 24 ]<- 2015
cabflow[ 2 , 25 ]<- 565
cabflow[ 3 , 1 ]<- 7629
cabflow[ 3 , 2 ]<- 12999
cabflow[ 3 , 3 ]<- 0
cabflow[ 3 , 4 ]<- 35135
cabflow[ 3 , 5 ]<- 5956
cabflow[ 3 , 6 ]<- 14121
cabflow[ 3 , 7 ]<- 5951
cabflow[ 3 , 8 ]<- 5768
cabflow[ 3 , 9 ]<- 16578
cabflow[ 3 , 10 ]<- 4242
cabflow[ 3 , 11 ]<- 3365
cabflow[ 3 , 12 ]<- 22254
cabflow[ 3 , 13 ]<- 1841
cabflow[ 3 , 14 ]<- 23665
cabflow[ 3 , 15 ]<- 6517
cabflow[ 3 , 16 ]<- 3541
cabflow[ 3 , 17 ]<- 205088
cabflow[ 3 , 18 ]<- 37669
```

```
cabflow[ 3 , 19 ]<- 2885
cabflow[ 3 , 20 ]<- 13200
cabflow[ 3 , 21 ]<- 7116
cabflow[ 3 , 22 ]<- 17165
cabflow[ 3 , 23 ]<- 4284
cabflow[ 3 , 24 ]<- 8085
cabflow[ 3 , 25 ]<- 51895
cabflow[ 4 , 1 ]<- 20036
cabflow[ 4 , 2 ]<- 13692
cabflow[ 4 , 3 ]<- 35135
cabflow[ 4 , 4 ]<- 0
cabflow[ 4 , 5 ]<- 19094
cabflow[ 4 , 6 ]<- 35119
cabflow[ 4 , 7 ]<- 21423
cabflow[ 4 , 8 ]<- 27342
cabflow[ 4 , 9 ]<- 51341
cabflow[ 4 , 10 ]<- 15826
cabflow[ 4 , 11 ]<- 28537
cabflow[ 4 , 12 ]<- 65387
cabflow[ 4 , 13 ]<- 12980
cabflow[ 4 , 14 ]<- 44097
cabflow[ 4 , 15 ]<- 51525
cabflow[ 4 , 16 ]<- 14354
cabflow[ 4 , 17 ]<- 172895
cabflow[ 4 , 18 ]<- 37305
cabflow[ 4 , 19 ]<- 15418
cabflow[ 4 , 20 ]<- 26221
cabflow[ 4 , 21 ]<- 42303
cabflow[ 4 , 22 ]<- 35303
cabflow[ 4 , 23 ]<- 13618
cabflow[ 4 , 24 ]<- 17580
cabflow[ 4 , 25 ]<- 40708
cabflow[ 5 , 1 ]<- 4690
cabflow[ 5 , 2 ]<- 3322
cabflow[ 5 , 3 ]<- 5956
cabflow[ 5 , 4 ]<- 19094
cabflow[ 5 , 5 ]<- 0
cabflow[ 5 , 6 ]<- 7284
cabflow[ 5 , 7 ]<- 3102
cabflow[ 5 , 8 ]<- 1562
cabflow[ 5 , 9 ]<- 7180
cabflow[ 5 , 10 ]<- 1917
cabflow[ 5 , 11 ]<- 2253
cabflow[ 5 , 12 ]<- 5951
cabflow[ 5 , 13 ]<- 1890
cabflow[ 5 , 14 ]<- 7097
cabflow[ 5 , 15 ]<- 2009
cabflow[ 5 , 16 ]<- 1340
cabflow[ 5 , 17 ]<- 25303
cabflow[ 5 , 18 ]<- 6031
cabflow[ 5 , 19 ]<- 1041
cabflow[ 5 , 20 ]<- 4128
cabflow[ 5 , 21 ]<- 5452
cabflow[ 5 , 22 ]<- 3344
cabflow[ 5 , 23 ]<- 1067
cabflow[ 5 , 24 ]<- 4608
cabflow[ 5 , 25 ]<- 7050
```

```
cabflow[ 6 , 1 ]<- 6194
cabflow[ 6 , 2 ]<- 5576
cabflow[ 6 , 3 ]<- 14121
cabflow[ 6 , 4 ]<- 35119
cabflow[ 6 , 5 ]<- 7284
cabflow[ 6 , 6 ]<- 0
cabflow[ 6 , 7 ]<- 5023
cabflow[ 6 , 8 ]<- 3512
cabflow[ 6 , 9 ]<- 10419
cabflow[ 6 , 10 ]<- 3543
cabflow[ 6 , 11 ]<- 2752
cabflow[ 6 , 12 ]<- 14412
cabflow[ 6 , 13 ]<- 2043
cabflow[ 6 , 14 ]<- 15642
cabflow[ 6 , 15 ]<- 5014
cabflow[ 6 , 16 ]<- 2016
cabflow[ 6 , 17 ]<- 62034
cabflow[ 6 , 18 ]<- 15385
cabflow[ 6 , 19 ]<- 2957
cabflow[ 6 , 20 ]<- 5035
cabflow[ 6 , 21 ]<- 7482
cabflow[ 6 , 22 ]<- 6758
cabflow[ 6 , 23 ]<- 2191
cabflow[ 6 , 24 ]<- 6599
cabflow[ 6 , 25 ]<- 14181
cabflow[ 7 , 1 ]<- 11688
cabflow[ 7 , 2 ]<- 3878
cabflow[ 7 , 3 ]<- 5951
cabflow[ 7 , 4 ]<- 21423
cabflow[ 7 , 5 ]<- 3102
cabflow[ 7 , 6 ]<- 5023
cabflow[ 7 , 7 ]<- 0
cabflow[ 7 , 8 ]<- 11557
cabflow[ 7 , 9 ]<- 6479
cabflow[ 7 , 10 ]<- 34261
cabflow[ 7 , 11 ]<- 10134
cabflow[ 7 , 12 ]<- 27350
cabflow[ 7 , 13 ]<- 6929
cabflow[ 7 , 14 ]<- 7961
cabflow[ 7 , 15 ]<- 4678
cabflow[ 7 , 16 ]<- 13511
cabflow[ 7 , 17 ]<- 29801
cabflow[ 7 , 18 ]<- 7549
cabflow[ 7 , 19 ]<- 5550
cabflow[ 7 , 20 ]<- 3089
cabflow[ 7 , 21 ]<- 9958
cabflow[ 7 , 22 ]<- 14110
cabflow[ 7 , 23 ]<- 4911
cabflow[ 7 , 24 ]<- 2722
cabflow[ 7 , 25 ]<- 10802
cabflow[ 8 , 1 ]<- 2243
cabflow[ 8 , 2 ]<- 3202
cabflow[ 8 , 3 ]<- 5768
cabflow[ 8 , 4 ]<- 27342
cabflow[ 8 , 5 ]<- 1562
cabflow[ 8 , 6 ]<- 3512
cabflow[ 8 , 7 ]<- 11557
```

```
cabflow[ 8 , 8 ]<- 0
cabflow[ 8 , 9 ]<- 5615
cabflow[ 8 , 10 ]<- 7095
cabflow[ 8 , 11 ]<- 10753
cabflow[ 8 , 12 ]<- 30362
cabflow[ 8 , 13 ]<- 1783
cabflow[ 8 , 14 ]<- 3437
cabflow[ 8 , 15 ]<- 8897
cabflow[ 8 , 16 ]<- 2509
cabflow[ 8 , 17 ]<- 23273
cabflow[ 8 , 18 ]<- 5160
cabflow[ 8 , 19 ]<- 8750
cabflow[ 8 , 20 ]<- 2583
cabflow[ 8 , 21 ]<- 7288
cabflow[ 8 , 22 ]<- 17481
cabflow[ 8 , 23 ]<- 7930
cabflow[ 8 , 24 ]<- 1278
cabflow[ 8 , 25 ]<- 8447
cabflow[ 9 , 1 ]<- 8857
cabflow[ 9 , 2 ]<- 6699
cabflow[ 9 , 3 ]<- 16578
cabflow[ 9 , 4 ]<- 51341
cabflow[ 9 , 5 ]<- 7180
cabflow[ 9 , 6 ]<- 10419
cabflow[ 9 , 7 ]<- 6479
cabflow[ 9 , 8 ]<- 5615
cabflow[ 9 , 9 ]<- 0
cabflow[ 9 , 10 ]<- 4448
cabflow[ 9 , 11 ]<- 5076
cabflow[ 9 , 12 ]<- 22463
cabflow[ 9 , 13 ]<- 4783
cabflow[ 9 , 14 ]<- 24609
cabflow[ 9 , 15 ]<- 9969
cabflow[ 9 , 16 ]<- 4224
cabflow[ 9 , 17 ]<- 79945
cabflow[ 9 , 18 ]<- 20001
cabflow[ 9 , 19 ]<- 4291
cabflow[ 9 , 20 ]<- 10604
cabflow[ 9 , 21 ]<- 11925
cabflow[ 9 , 22 ]<- 13091
cabflow[ 9 , 23 ]<- 4172
cabflow[ 9 , 24 ]<- 12891
cabflow[ 9 , 25 ]<- 19500
cabflow[ 10 , 1 ]<- 7248
cabflow[ 10 , 2 ]<- 4198
cabflow[ 10 , 3 ]<- 4242
cabflow[ 10 , 4 ]<- 15826
cabflow[ 10 , 5 ]<- 1917
cabflow[ 10 , 6 ]<- 3543
cabflow[ 10 , 7 ]<- 34261
cabflow[ 10 , 8 ]<- 7095
cabflow[ 10 , 9 ]<- 4448
cabflow[ 10 , 10 ]<- 0
cabflow[ 10 , 11 ]<- 4370
cabflow[ 10 , 12 ]<- 17267
cabflow[ 10 , 13 ]<- 3929
cabflow[ 10 , 14 ]<- 8602
```

```
cabflow[ 10 , 15 ]<- 2753
cabflow[ 10 , 16 ]<- 20013
cabflow[ 10 , 17 ]<- 28080
cabflow[ 10 , 18 ]<- 5971
cabflow[ 10 , 19 ]<- 2131
cabflow[ 10 , 20 ]<- 3579
cabflow[ 10 , 21 ]<- 6809
cabflow[ 10 , 22 ]<- 8455
cabflow[ 10 , 23 ]<- 2868
cabflow[ 10 , 24 ]<- 2336
cabflow[ 10 , 25 ]<- 5616
cabflow[ 11 , 1 ]<- 3559
cabflow[ 11 , 2 ]<- 2454
cabflow[ 11 , 3 ]<- 3365
cabflow[ 11 , 4 ]<- 28537
cabflow[ 11 , 5 ]<- 2253
cabflow[ 11 , 6 ]<- 2752
cabflow[ 11 , 7 ]<- 10134
cabflow[ 11 , 8 ]<- 10753
cabflow[ 11 , 9 ]<- 5076
cabflow[ 11 , 10 ]<- 4370
cabflow[ 11 , 11 ]<- 0
cabflow[ 11 , 12 ]<- 15287
cabflow[ 11 , 13 ]<- 3083
cabflow[ 11 , 14 ]<- 4092
cabflow[ 11 , 15 ]<- 7701
cabflow[ 11 , 16 ]<- 2809
cabflow[ 11 , 17 ]<- 17291
cabflow[ 11 , 18 ]<- 4462
cabflow[ 11 , 19 ]<- 3239
cabflow[ 11 , 20 ]<- 2309
cabflow[ 11 , 21 ]<- 16003
cabflow[ 11 , 22 ]<- 8381
cabflow[ 11 , 23 ]<- 3033
cabflow[ 11 , 24 ]<- 1755
cabflow[ 11 , 25 ]<- 7266
cabflow[ 12 , 1 ]<- 9221
cabflow[ 12 , 2 ]<- 7975
cabflow[ 12 , 3 ]<- 22254
cabflow[ 12 , 4 ]<- 65387
cabflow[ 12 , 5 ]<- 5951
cabflow[ 12 , 6 ]<- 14412
cabflow[ 12 , 7 ]<- 27350
cabflow[ 12 , 8 ]<- 30362
cabflow[ 12 , 9 ]<- 22463
cabflow[ 12 , 10 ]<- 17267
cabflow[ 12 , 11 ]<- 15287
cabflow[ 12 , 12 ]<- 0
cabflow[ 12 , 13 ]<- 5454
cabflow[ 12 , 14 ]<- 15011
cabflow[ 12 , 15 ]<- 17714
cabflow[ 12 , 16 ]<- 10037
cabflow[ 12 , 17 ]<- 105507
cabflow[ 12 , 18 ]<- 20040
cabflow[ 12 , 19 ]<- 31780
cabflow[ 12 , 20 ]<- 10822
cabflow[ 12 , 21 ]<- 16450
```

```
cabflow[ 12 , 22 ]<- 92083
cabflow[ 12 , 23 ]<- 32908
cabflow[ 12 , 24 ]<- 3865
cabflow[ 12 , 25 ]<- 24583
cabflow[ 13 , 1 ]<- 10099
cabflow[ 13 , 2 ]<- 1186
cabflow[ 13 , 3 ]<- 1841
cabflow[ 13 , 4 ]<- 12980
cabflow[ 13 , 5 ]<- 1890
cabflow[ 13 , 6 ]<- 2043
cabflow[ 13 , 7 ]<- 6929
cabflow[ 13 , 8 ]<- 1783
cabflow[ 13 , 9 ]<- 4783
cabflow[ 13 , 10 ]<- 3929
cabflow[ 13 , 11 ]<- 3083
cabflow[ 13 , 12 ]<- 5454
cabflow[ 13 , 13 ]<- 0
cabflow[ 13 , 14 ]<- 3251
cabflow[ 13 , 15 ]<- 1126
cabflow[ 13 , 16 ]<- 5926
cabflow[ 13 , 17 ]<- 10653
cabflow[ 13 , 18 ]<- 3062
cabflow[ 13 , 19 ]<- 759
cabflow[ 13 , 20 ]<- 1255
cabflow[ 13 , 21 ]<- 6173
cabflow[ 13 , 22 ]<- 2974
cabflow[ 13 , 23 ]<- 1056
cabflow[ 13 , 24 ]<- 1504
cabflow[ 13 , 25 ]<- 4588
cabflow[ 14 , 1 ]<- 22866
cabflow[ 14 , 2 ]<- 7443
cabflow[ 14 , 3 ]<- 23665
cabflow[ 14 , 4 ]<- 44097
cabflow[ 14 , 5 ]<- 7097
cabflow[ 14 , 6 ]<- 15642
cabflow[ 14 , 7 ]<- 7961
cabflow[ 14 , 8 ]<- 3437
cabflow[ 14 , 9 ]<- 24609
cabflow[ 14 , 10 ]<- 8602
cabflow[ 14 , 11 ]<- 4092
cabflow[ 14 , 12 ]<- 15011
cabflow[ 14 , 13 ]<- 3251
cabflow[ 14 , 14 ]<- 0
cabflow[ 14 , 15 ]<- 5550
cabflow[ 14 , 16 ]<- 9473
cabflow[ 14 , 17 ]<- 169397
cabflow[ 14 , 18 ]<- 25073
cabflow[ 14 , 19 ]<- 1170
cabflow[ 14 , 20 ]<- 14272
cabflow[ 14 , 21 ]<- 8543
cabflow[ 14 , 22 ]<- 8064
cabflow[ 14 , 23 ]<- 1840
cabflow[ 14 , 24 ]<- 20618
cabflow[ 14 , 25 ]<- 20937
cabflow[ 15 , 1 ]<- 3388
cabflow[ 15 , 2 ]<- 1162
cabflow[ 15 , 3 ]<- 6517
```

```
cabflow[ 15 , 4 ]<- 51525
cabflow[ 15 , 5 ]<- 2009
cabflow[ 15 , 6 ]<- 5014
cabflow[ 15 , 7 ]<- 4678
cabflow[ 15 , 8 ]<- 8897
cabflow[ 15 , 9 ]<- 9969
cabflow[ 15 , 10 ]<- 2753
cabflow[ 15 , 11 ]<- 7701
cabflow[ 15 , 12 ]<- 17714
cabflow[ 15 , 13 ]<- 1126
cabflow[ 15 , 14 ]<- 5550
cabflow[ 15 , 15 ]<- 0
cabflow[ 15 , 16 ]<- 2152
cabflow[ 15 , 17 ]<- 26816
cabflow[ 15 , 18 ]<- 6931
cabflow[ 15 , 19 ]<- 4947
cabflow[ 15 , 20 ]<- 2676
cabflow[ 15 , 21 ]<- 8033
cabflow[ 15 , 22 ]<- 12692
cabflow[ 15 , 23 ]<- 6157
cabflow[ 15 , 24 ]<- 3065
cabflow[ 15 , 25 ]<- 12044
cabflow[ 16 , 1 ]<- 9986
cabflow[ 16 , 2 ]<- 5105
cabflow[ 16 , 3 ]<- 3541
cabflow[ 16 , 4 ]<- 14354
cabflow[ 16 , 5 ]<- 1340
cabflow[ 16 , 6 ]<- 2016
cabflow[ 16 , 7 ]<- 13511
cabflow[ 16 , 8 ]<- 2509
cabflow[ 16 , 9 ]<- 4224
cabflow[ 16 , 10 ]<- 20013
cabflow[ 16 , 11 ]<- 2809
cabflow[ 16 , 12 ]<- 10037
cabflow[ 16 , 13 ]<- 5926
cabflow[ 16 , 14 ]<- 9473
cabflow[ 16 , 15 ]<- 2152
cabflow[ 16 , 16 ]<- 0
cabflow[ 16 , 17 ]<- 21806
cabflow[ 16 , 18 ]<- 4519
cabflow[ 16 , 19 ]<- 886
cabflow[ 16 , 20 ]<- 1742
cabflow[ 16 , 21 ]<- 4782
cabflow[ 16 , 22 ]<- 6453
cabflow[ 16 , 23 ]<- 2022
cabflow[ 16 , 24 ]<- 3546
cabflow[ 16 , 25 ]<- 5065
cabflow[ 17 , 1 ]<- 46618
cabflow[ 17 , 2 ]<- 24817
cabflow[ 17 , 3 ]<- 205088
cabflow[ 17 , 4 ]<- 172895
cabflow[ 17 , 5 ]<- 25303
cabflow[ 17 , 6 ]<- 62034
cabflow[ 17 , 7 ]<- 29801
cabflow[ 17 , 8 ]<- 23273
cabflow[ 17 , 9 ]<- 79945
cabflow[ 17 , 10 ]<- 28080
```

```
cabflow[ 17 , 11 ]<- 17291
cabflow[ 17 , 12 ]<- 105507
cabflow[ 17 , 13 ]<- 10653
cabflow[ 17 , 14 ]<- 169397
cabflow[ 17 , 15 ]<- 26816
cabflow[ 17 , 16 ]<- 21806
cabflow[ 17 , 17 ]<- 0
cabflow[ 17 , 18 ]<- 9040
cabflow[ 17 , 19 ]<- 11139
cabflow[ 17 , 20 ]<- 63153
cabflow[ 17 , 21 ]<- 34092
cabflow[ 17 , 22 ]<- 70935
cabflow[ 17 , 23 ]<- 14957
cabflow[ 17 , 24 ]<- 28398
cabflow[ 17 , 25 ]<- 166694
cabflow[ 18 , 1 ]<- 11639
cabflow[ 18 , 2 ]<- 6532
cabflow[ 18 , 3 ]<- 37669
cabflow[ 18 , 4 ]<- 37305
cabflow[ 18 , 5 ]<- 6031
cabflow[ 18 , 6 ]<- 15385
cabflow[ 18 , 7 ]<- 7549
cabflow[ 18 , 8 ]<- 5160
cabflow[ 18 , 9 ]<- 20001
cabflow[ 18 , 10 ]<- 5971
cabflow[ 18 , 11 ]<- 4462
cabflow[ 18 , 12 ]<- 20040
cabflow[ 18 , 13 ]<- 3062
cabflow[ 18 , 14 ]<- 25073
cabflow[ 18 , 15 ]<- 6931
cabflow[ 18 , 16 ]<- 4519
cabflow[ 18 , 17 ]<- 9040
cabflow[ 18 , 18 ]<- 0
cabflow[ 18 , 19 ]<- 2802
cabflow[ 18 , 20 ]<- 30224
cabflow[ 18 , 21 ]<- 7982
cabflow[ 18 , 22 ]<- 14964
cabflow[ 18 , 23 ]<- 4589
cabflow[ 18 , 24 ]<- 6227
cabflow[ 18 , 25 ]<- 12359
cabflow[ 19 , 1 ]<- 1380
cabflow[ 19 , 2 ]<- 806
cabflow[ 19 , 3 ]<- 2885
cabflow[ 19 , 4 ]<- 15418
cabflow[ 19 , 5 ]<- 1041
cabflow[ 19 , 6 ]<- 2957
cabflow[ 19 , 7 ]<- 5550
cabflow[ 19 , 8 ]<- 8750
cabflow[ 19 , 9 ]<- 4291
cabflow[ 19 , 10 ]<- 2131
cabflow[ 19 , 11 ]<- 3239
cabflow[ 19 , 12 ]<- 31780
cabflow[ 19 , 13 ]<- 759
cabflow[ 19 , 14 ]<- 1170
cabflow[ 19 , 15 ]<- 4947
cabflow[ 19 , 16 ]<- 886
cabflow[ 19 , 17 ]<- 11139
```

```
cabflow[ 19 , 18 ]<- 2802
cabflow[ 19 , 19 ]<- 0
cabflow[ 19 , 20 ]<- 1869
cabflow[ 19 , 21 ]<- 3716
cabflow[ 19 , 22 ]<- 11510
cabflow[ 19 , 23 ]<- 3519
cabflow[ 19 , 24 ]<- 569
cabflow[ 19 , 25 ]<- 3520
cabflow[ 20 , 1 ]<- 5261
cabflow[ 20 , 2 ]<- 8184
cabflow[ 20 , 3 ]<- 13200
cabflow[ 20 , 4 ]<- 26221
cabflow[ 20 , 5 ]<- 4128
cabflow[ 20 , 6 ]<- 5035
cabflow[ 20 , 7 ]<- 3089
cabflow[ 20 , 8 ]<- 2583
cabflow[ 20 , 9 ]<- 10604
cabflow[ 20 , 10 ]<- 3579
cabflow[ 20 , 11 ]<- 2309
cabflow[ 20 , 12 ]<- 10822
cabflow[ 20 , 13 ]<- 1255
cabflow[ 20 , 14 ]<- 14272
cabflow[ 20 , 15 ]<- 2676
cabflow[ 20 , 16 ]<- 1742
cabflow[ 20 , 17 ]<- 63153
cabflow[ 20 , 18 ]<- 30224
cabflow[ 20 , 19 ]<- 1869
cabflow[ 20 , 20 ]<- 0
cabflow[ 20 , 21 ]<- 5020
cabflow[ 20 , 22 ]<- 6610
cabflow[ 20 , 23 ]<- 2139
cabflow[ 20 , 24 ]<- 5431
cabflow[ 20 , 25 ]<- 13541
cabflow[ 21 , 1 ]<- 5985
cabflow[ 21 , 2 ]<- 3896
cabflow[ 21 , 3 ]<- 7116
cabflow[ 21 , 4 ]<- 42303
cabflow[ 21 , 5 ]<- 5452
cabflow[ 21 , 6 ]<- 7482
cabflow[ 21 , 7 ]<- 9958
cabflow[ 21 , 8 ]<- 7288
cabflow[ 21 , 9 ]<- 11925
cabflow[ 21 , 10 ]<- 6809
cabflow[ 21 , 11 ]<- 16003
cabflow[ 21 , 12 ]<- 16450
cabflow[ 21 , 13 ]<- 6173
cabflow[ 21 , 14 ]<- 8543
cabflow[ 21 , 15 ]<- 8033
cabflow[ 21 , 16 ]<- 4782
cabflow[ 21 , 17 ]<- 34092
cabflow[ 21 , 18 ]<- 7982
cabflow[ 21 , 19 ]<- 3716
cabflow[ 21 , 20 ]<- 5020
cabflow[ 21 , 21 ]<- 0
cabflow[ 21 , 22 ]<- 9942
cabflow[ 21 , 23 ]<- 3276
cabflow[ 21 , 24 ]<- 3820
```

```
cabflow[ 21 , 25 ]<- 11799
cabflow[ 22 , 1 ]<- 6731
cabflow[ 22 , 2 ]<- 7333
cabflow[ 22 , 3 ]<- 17165
cabflow[ 22 , 4 ]<- 35303
cabflow[ 22 , 5 ]<- 3344
cabflow[ 22 , 6 ]<- 6758
cabflow[ 22 , 7 ]<- 14110
cabflow[ 22 , 8 ]<- 17481
cabflow[ 22 , 9 ]<- 13091
cabflow[ 22 , 10 ]<- 8455
cabflow[ 22 , 11 ]<- 8381
cabflow[ 22 , 12 ]<- 92083
cabflow[ 22 , 13 ]<- 2974
cabflow[ 22 , 14 ]<- 8064
cabflow[ 22 , 15 ]<- 12692
cabflow[ 22 , 16 ]<- 6453
cabflow[ 22 , 17 ]<- 70935
cabflow[ 22 , 18 ]<- 14964
cabflow[ 22 , 19 ]<- 11510
cabflow[ 22 , 20 ]<- 6610
cabflow[ 22 , 21 ]<- 9942
cabflow[ 22 , 22 ]<- 0
cabflow[ 22 , 23 ]<- 35285
cabflow[ 22 , 24 ]<- 2566
cabflow[ 22 , 25 ]<- 19926
cabflow[ 23 , 1 ]<- 2704
cabflow[ 23 , 2 ]<- 3719
cabflow[ 23 , 3 ]<- 4284
cabflow[ 23 , 4 ]<- 13618
cabflow[ 23 , 5 ]<- 1067
cabflow[ 23 , 6 ]<- 2191
cabflow[ 23 , 7 ]<- 4911
cabflow[ 23 , 8 ]<- 7930
cabflow[ 23 , 9 ]<- 4172
cabflow[ 23 , 10 ]<- 2868
cabflow[ 23 , 11 ]<- 3033
cabflow[ 23 , 12 ]<- 32908
cabflow[ 23 , 13 ]<- 1056
cabflow[ 23 , 14 ]<- 1840
cabflow[ 23 , 15 ]<- 6157
cabflow[ 23 , 16 ]<- 2022
cabflow[ 23 , 17 ]<- 14957
cabflow[ 23 , 18 ]<- 4589
cabflow[ 23 , 19 ]<- 3519
cabflow[ 23 , 20 ]<- 2139
cabflow[ 23 , 21 ]<- 3276
cabflow[ 23 , 22 ]<- 35285
cabflow[ 23 , 23 ]<- 0
cabflow[ 23 , 24 ]<- 940
cabflow[ 23 , 25 ]<- 4951
cabflow[ 24 , 1 ]<- 12250
cabflow[ 24 , 2 ]<- 2015
cabflow[ 24 , 3 ]<- 8085
cabflow[ 24 , 4 ]<- 17580
cabflow[ 24 , 5 ]<- 4608
cabflow[ 24 , 6 ]<- 6599
```

```
cabflow[ 24 , 7 ]<- 2722
cabflow[ 24 , 8 ]<- 1278
cabflow[ 24 , 9 ]<- 12891
cabflow[ 24 , 10 ]<- 2336
cabflow[ 24 , 11 ]<- 1755
cabflow[ 24 , 12 ]<- 3865
cabflow[ 24 , 13 ]<- 1504
cabflow[ 24 , 14 ]<- 20618
cabflow[ 24 , 15 ]<- 3065
cabflow[ 24 , 16 ]<- 3546
cabflow[ 24 , 17 ]<- 28398
cabflow[ 24 , 18 ]<- 6227
cabflow[ 24 , 19 ]<- 569
cabflow[ 24 , 20 ]<- 5431
cabflow[ 24 , 21 ]<- 3820
cabflow[ 24 , 22 ]<- 2566
cabflow[ 24 , 23 ]<- 940
cabflow[ 24 , 24 ]<- 0
cabflow[ 24 , 25 ]<- 6237
cabflow[ 25 , 1 ]<- 16132
cabflow[ 25 , 2 ]<- 565
cabflow[ 25 , 3 ]<- 51895
cabflow[ 25 , 4 ]<- 40708
cabflow[ 25 , 5 ]<- 7050
cabflow[ 25 , 6 ]<- 14181
cabflow[ 25 , 7 ]<- 10802
cabflow[ 25 , 8 ]<- 8447
cabflow[ 25 , 9 ]<- 19500
cabflow[ 25 , 10 ]<- 5616
cabflow[ 25 , 11 ]<- 7266
cabflow[ 25 , 12 ]<- 24583
cabflow[ 25 , 13 ]<- 4588
cabflow[ 25 , 14 ]<- 20937
cabflow[ 25 , 15 ]<- 12044
cabflow[ 25 , 16 ]<- 5065
cabflow[ 25 , 17 ]<- 166694
cabflow[ 25 , 18 ]<- 12359
cabflow[ 25 , 19 ]<- 3520
cabflow[ 25 , 20 ]<- 13541
cabflow[ 25 , 21 ]<- 11799
cabflow[ 25 , 22 ]<- 19926
cabflow[ 25 , 23 ]<- 4951
cabflow[ 25 , 24 ]<- 6237
cabflow[ 25 , 25 ]<- 0
cabcost[ 1 , 1 ]<- 0
cabcost[ 1 , 2 ]<- 576.9631
cabcost[ 1 , 3 ]<- 946.4954
cabcost[ 1 , 4 ]<- 597.5972
cabcost[ 1 , 5 ]<- 373.8127
cabcost[ 1 , 6 ]<- 559.7673
cabcost[ 1 , 7 ]<- 709.0215
cabcost[ 1 , 8 ]<- 1208.328
cabcost[ 1 , 9 ]<- 603.6477
cabcost[ 1 , 10 ]<- 695.208
cabcost[ 1 , 11 ]<- 680.709
cabcost[ 1 , 12 ]<- 1936.572
cabcost[ 1 , 13 ]<- 332.4644
```

```
cabcost[ 1 , 14 ]<- 592.5679
cabcost[ 1 , 15 ]<- 908.7715
cabcost[ 1 , 16 ]<- 426.1877
cabcost[ 1 , 17 ]<- 756.1987
cabcost[ 1 , 18 ]<- 672.5906
cabcost[ 1 , 19 ]<- 1590.224
cabcost[ 1 , 20 ]<- 527.3008
cabcost[ 1 , 21 ]<- 483.4673
cabcost[ 1 , 22 ]<- 2140.978
cabcost[ 1 , 23 ]<- 2184.402
cabcost[ 1 , 24 ]<- 408.1648
cabcost[ 1 , 25 ]<- 540.7388
cabcost[ 2 , 1 ]<- 576.9631
cabcost[ 2 , 2 ]<- 0
cabcost[ 2 , 3 ]<- 369.5327
cabcost[ 2 , 4 ]<- 613.0386
cabcost[ 2 , 5 ]<- 429.1079
cabcost[ 2 , 6 ]<- 312.8831
cabcost[ 2 , 7 ]<- 1196.489
cabcost[ 2 , 8 ]<- 1502.14
cabcost[ 2 , 9 ]<- 405.8975
cabcost[ 2 , 10 ]<- 1241.961
cabcost[ 2 , 11 ]<- 960.3459
cabcost[ 2 , 12 ]<- 2318.076
cabcost[ 2 , 13 ]<- 786.5959
cabcost[ 2 , 14 ]<- 949.5669
cabcost[ 2 , 15 ]<- 938.7461
cabcost[ 2 , 16 ]<- 999.5005
cabcost[ 2 , 17 ]<- 179.2426
cabcost[ 2 , 18 ]<- 96.2744
cabcost[ 2 , 19 ]<- 1999.584
cabcost[ 2 , 20 ]<- 210.7656
cabcost[ 2 , 21 ]<- 736.3755
cabcost[ 2 , 22 ]<- 2456.263
cabcost[ 2 , 23 ]<- 2339.509
cabcost[ 2 , 24 ]<- 844.1663
cabcost[ 2 , 25 ]<- 36.4947
cabcost[ 3 , 1 ]<- 946.4954
cabcost[ 3 , 2 ]<- 369.5327
cabcost[ 3 , 3 ]<- 0
cabcost[ 3 , 4 ]<- 858.3308
cabcost[ 3 , 5 ]<- 749.6018
cabcost[ 3 , 6 ]<- 556.0706
cabcost[ 3 , 7 ]<- 1541.273
cabcost[ 3 , 8 ]<- 1764.791
cabcost[ 3 , 9 ]<- 621.3306
cabcost[ 3 , 10 ]<- 1603.165
cabcost[ 3 , 11 ]<- 1250.962
cabcost[ 3 , 12 ]<- 2600.078
cabcost[ 3 , 13 ]<- 1137.335
cabcost[ 3 , 14 ]<- 1266.851
cabcost[ 3 , 15 ]<- 1124.778
cabcost[ 3 , 16 ]<- 1368.267
cabcost[ 3 , 17 ]<- 190.3157
cabcost[ 3 , 18 ]<- 274.3105
cabcost[ 3 , 19 ]<- 2299.429
cabcost[ 3 , 20 ]<- 494.2224
```

```
cabcost[ 3 , 21 ]<- 1043.484
cabcost[ 3 , 22 ]<- 2703.402
cabcost[ 3 , 23 ]<- 2503.828
cabcost[ 3 , 24 ]<- 1188.549
cabcost[ 3 , 25 ]<- 405.7886
cabcost[ 4 , 1 ]<- 597.5972
cabcost[ 4 , 2 ]<- 613.0386
cabcost[ 4 , 3 ]<- 858.3308
cabcost[ 4 , 4 ]<- 0
cabcost[ 4 , 5 ]<- 255.0303
cabcost[ 4 , 6 ]<- 311.3071
cabcost[ 4 , 7 ]<- 790.1213
cabcost[ 4 , 8 ]<- 907.4331
cabcost[ 4 , 9 ]<- 237.0703
cabcost[ 4 , 10 ]<- 932.2173
cabcost[ 4 , 11 ]<- 406.3386
cabcost[ 4 , 12 ]<- 1741.873
cabcost[ 4 , 13 ]<- 485.5564
cabcost[ 4 , 14 ]<- 1186.858
cabcost[ 4 , 15 ]<- 345.8738
cabcost[ 4 , 16 ]<- 830.3635
cabcost[ 4 , 17 ]<- 720.4687
cabcost[ 4 , 18 ]<- 675.3437
cabcost[ 4 , 19 ]<- 1447.104
cabcost[ 4 , 20 ]<- 403.8657
cabcost[ 4 , 21 ]<- 255.8823
cabcost[ 4 , 22 ]<- 1853.617
cabcost[ 4 , 23 ]<- 1733.132
cabcost[ 4 , 24 ]<- 1005.761
cabcost[ 4 , 25 ]<- 592.0278
cabcost[ 5 , 1 ]<- 373.8127
cabcost[ 5 , 2 ]<- 429.1079
cabcost[ 5 , 3 ]<- 749.6018
cabcost[ 5 , 4 ]<- 255.0303
cabcost[ 5 , 5 ]<- 0
cabcost[ 5 , 6 ]<- 225.8954
cabcost[ 5 , 7 ]<- 794.1726
cabcost[ 5 , 8 ]<- 1080.374
cabcost[ 5 , 9 ]<- 238.944
cabcost[ 5 , 10 ]<- 879.5647
cabcost[ 5 , 11 ]<- 533.156
cabcost[ 5 , 12 ]<- 1889.528
cabcost[ 5 , 13 ]<- 402.3291
cabcost[ 5 , 14 ]<- 947.3188
cabcost[ 5 , 15 ]<- 598.541
cabcost[ 5 , 16 ]<- 700.4368
cabcost[ 5 , 17 ]<- 578.3286
cabcost[ 5 , 18 ]<- 512.3965
cabcost[ 5 , 19 ]<- 1570.725
cabcost[ 5 , 20 ]<- 255.6551
cabcost[ 5 , 21 ]<- 307.3289
cabcost[ 5 , 22 ]<- 2036.128
cabcost[ 5 , 23 ]<- 1967.256
cabcost[ 5 , 24 ]<- 775.239
cabcost[ 5 , 25 ]<- 399.2253
cabcost[ 6 , 1 ]<- 559.7673
cabcost[ 6 , 2 ]<- 312.8831
```

```
cabcost[ 6 , 3 ]<- 556.0706
cabcost[ 6 , 4 ]<- 311.3071
cabcost[ 6 , 5 ]<- 225.8954
cabcost[ 6 , 6 ]<- 0
cabcost[ 6 , 7 ]<- 1009.689
cabcost[ 6 , 8 ]<- 1216.868
cabcost[ 6 , 9 ]<- 94.2588
cabcost[ 6 , 10 ]<- 1104.574
cabcost[ 6 , 11 ]<- 694.9153
cabcost[ 6 , 12 ]<- 2047.122
cabcost[ 6 , 13 ]<- 627.115
cabcost[ 6 , 14 ]<- 1084.5
cabcost[ 6 , 15 ]<- 626.1548
cabcost[ 6 , 16 ]<- 922.3181
cabcost[ 6 , 17 ]<- 409.3542
cabcost[ 6 , 18 ]<- 365.6853
cabcost[ 6 , 19 ]<- 1743.432
cabcost[ 6 , 20 ]<- 104.6478
cabcost[ 6 , 21 ]<- 491.1125
cabcost[ 6 , 22 ]<- 2164.855
cabcost[ 6 , 23 ]<- 2027.319
cabcost[ 6 , 24 ]<- 933.196
cabcost[ 6 , 25 ]<- 298.8486
cabcost[ 7 , 1 ]<- 709.0215
cabcost[ 7 , 2 ]<- 1196.489
cabcost[ 7 , 3 ]<- 1541.273
cabcost[ 7 , 4 ]<- 790.1213
cabcost[ 7 , 5 ]<- 794.1726
cabcost[ 7 , 6 ]<- 1009.689
cabcost[ 7 , 7 ]<- 0
cabcost[ 7 , 8 ]<- 663.8762
cabcost[ 7 , 9 ]<- 982.7378
cabcost[ 7 , 10 ]<- 221.422
cabcost[ 7 , 11 ]<- 447.8044
cabcost[ 7 , 12 ]<- 1249.763
cabcost[ 7 , 13 ]<- 411.1133
cabcost[ 7 , 14 ]<- 1097.608
cabcost[ 7 , 15 ]<- 851.8228
cabcost[ 7 , 16 ]<- 423.7053
cabcost[ 7 , 17 ]<- 1362.874
cabcost[ 7 , 18 ]<- 1288.966
cabcost[ 7 , 19 ]<- 895.0908
cabcost[ 7 , 20 ]<- 1049.266
cabcost[ 7 , 21 ]<- 537.6206
cabcost[ 7 , 22 ]<- 1493.843
cabcost[ 7 , 23 ]<- 1686.675
cabcost[ 7 , 24 ]<- 912.2104
cabcost[ 7 , 25 ]<- 1161.676
cabcost[ 8 , 1 ]<- 1208.328
cabcost[ 8 , 2 ]<- 1502.14
cabcost[ 8 , 3 ]<- 1764.791
cabcost[ 8 , 4 ]<- 907.4331
cabcost[ 8 , 5 ]<- 1080.374
cabcost[ 8 , 6 ]<- 1216.868
cabcost[ 8 , 7 ]<- 663.8762
cabcost[ 8 , 8 ]<- 0
cabcost[ 8 , 9 ]<- 1143.791
```

```
cabcost[ 8 , 10 ]<- 874.5181
cabcost[ 8 , 11 ]<- 551.6299
cabcost[ 8 , 12 ]<- 841.624
cabcost[ 8 , 13 ]<- 880.0728
cabcost[ 8 , 14 ]<- 1714.651
cabcost[ 8 , 15 ]<- 694.0088
cabcost[ 8 , 16 ]<- 1066.563
cabcost[ 8 , 17 ]<- 1625.87
cabcost[ 8 , 18 ]<- 1574.822
cabcost[ 8 , 19 ]<- 593.4216
cabcost[ 8 , 20 ]<- 1301.511
cabcost[ 8 , 21 ]<- 780.9512
cabcost[ 8 , 22 ]<- 955.802
cabcost[ 8 , 23 ]<- 1024.566
cabcost[ 8 , 24 ]<- 1519.174
cabcost[ 8 , 25 ]<- 1475.479
cabcost[ 9 , 1 ]<- 603.6477
cabcost[ 9 , 2 ]<- 405.8975
cabcost[ 9 , 3 ]<- 621.3306
cabcost[ 9 , 4 ]<- 237.0703
cabcost[ 9 , 5 ]<- 238.944
cabcost[ 9 , 6 ]<- 94.2588
cabcost[ 9 , 7 ]<- 982.7378
cabcost[ 9 , 8 ]<- 1143.791
cabcost[ 9 , 9 ]<- 0
cabcost[ 9 , 10 ]<- 1094.906
cabcost[ 9 , 11 ]<- 636.9045
cabcost[ 9 , 12 ]<- 1978.943
cabcost[ 9 , 13 ]<- 620.488
cabcost[ 9 , 14 ]<- 1151.868
cabcost[ 9 , 15 ]<- 535.0244
cabcost[ 9 , 16 ]<- 936.2502
cabcost[ 9 , 17 ]<- 489.5645
cabcost[ 9 , 18 ]<- 453.2583
cabcost[ 9 , 19 ]<- 1682.489
cabcost[ 9 , 20 ]<- 198.9058
cabcost[ 9 , 21 ]<- 450.2585
cabcost[ 9 , 22 ]<- 2086.845
cabcost[ 9 , 23 ]<- 1936.304
cabcost[ 9 , 24 ]<- 992.3379
cabcost[ 9 , 25 ]<- 392.9045
cabcost[ 10 , 1 ]<- 695.208
cabcost[ 10 , 2 ]<- 1241.961
cabcost[ 10 , 3 ]<- 1603.165
cabcost[ 10 , 4 ]<- 932.2173
cabcost[ 10 , 5 ]<- 879.5647
cabcost[ 10 , 6 ]<- 1104.574
cabcost[ 10 , 7 ]<- 221.422
cabcost[ 10 , 8 ]<- 874.5181
cabcost[ 10 , 9 ]<- 1094.906
cabcost[ 10 , 10 ]<- 0
cabcost[ 10 , 11 ]<- 642.2092
cabcost[ 10 , 12 ]<- 1375.635
cabcost[ 10 , 13 ]<- 477.459
cabcost[ 10 , 14 ]<- 963.7202
cabcost[ 10 , 15 ]<- 1046.119
cabcost[ 10 , 16 ]<- 305.3132
```

```
cabcost[ 10 , 17 ]<- 1417.072
cabcost[ 10 , 18 ]<- 1337.648
cabcost[ 10 , 19 ]<- 1017.332
cabcost[ 10 , 20 ]<- 1125.041
cabcost[ 10 , 21 ]<- 677.0608
cabcost[ 10 , 22 ]<- 1649.619
cabcost[ 10 , 23 ]<- 1891.166
cabcost[ 10 , 24 ]<- 795.2136
cabcost[ 10 , 25 ]<- 1205.747
cabcost[ 11 , 1 ]<- 680.709
cabcost[ 11 , 2 ]<- 960.3459
cabcost[ 11 , 3 ]<- 1250.962
cabcost[ 11 , 4 ]<- 406.3386
cabcost[ 11 , 5 ]<- 533.156
cabcost[ 11 , 6 ]<- 694.9153
cabcost[ 11 , 7 ]<- 447.8044
cabcost[ 11 , 8 ]<- 551.6299
cabcost[ 11 , 9 ]<- 636.9045
cabcost[ 11 , 10 ]<- 642.2092
cabcost[ 11 , 11 ]<- 0
cabcost[ 11 , 12 ]<- 1358.213
cabcost[ 11 , 13 ]<- 378.5906
cabcost[ 11 , 14 ]<- 1236.192
cabcost[ 11 , 15 ]<- 405.0906
cabcost[ 11 , 16 ]<- 674.479
cabcost[ 11 , 17 ]<- 1096.712
cabcost[ 11 , 18 ]<- 1038.645
cabcost[ 11 , 19 ]<- 1048.539
cabcost[ 11 , 20 ]<- 768.1641
cabcost[ 11 , 21 ]<- 229.4867
cabcost[ 11 , 22 ]<- 1506.451
cabcost[ 11 , 23 ]<- 1503.794
cabcost[ 11 , 24 ]<- 1038.624
cabcost[ 11 , 25 ]<- 931.7148
cabcost[ 12 , 1 ]<- 1936.572
cabcost[ 12 , 2 ]<- 2318.076
cabcost[ 12 , 3 ]<- 2600.078
cabcost[ 12 , 4 ]<- 1741.873
cabcost[ 12 , 5 ]<- 1889.528
cabcost[ 12 , 6 ]<- 2047.122
cabcost[ 12 , 7 ]<- 1249.763
cabcost[ 12 , 8 ]<- 841.624
cabcost[ 12 , 9 ]<- 1978.943
cabcost[ 12 , 10 ]<- 1375.635
cabcost[ 12 , 11 ]<- 1358.213
cabcost[ 12 , 12 ]<- 0
cabcost[ 12 , 13 ]<- 1608.082
cabcost[ 12 , 14 ]<- 2335.816
cabcost[ 12 , 15 ]<- 1530.57
cabcost[ 12 , 16 ]<- 1661.778
cabcost[ 12 , 17 ]<- 2453.352
cabcost[ 12 , 18 ]<- 2396.794
cabcost[ 12 , 19 ]<- 358.3762
cabcost[ 12 , 20 ]<- 2125.512
cabcost[ 12 , 21 ]<- 1582.369
cabcost[ 12 , 22 ]<- 361.5388
cabcost[ 12 , 23 ]<- 986.8149
```

```
cabcost[ 12 , 24 ]<- 2157.517
cabcost[ 12 , 25 ]<- 2288.748
cabcost[ 13 , 1 ]<- 332.4644
cabcost[ 13 , 2 ]<- 786.5959
cabcost[ 13 , 3 ]<- 1137.335
cabcost[ 13 , 4 ]<- 485.5564
cabcost[ 13 , 5 ]<- 402.3291
cabcost[ 13 , 6 ]<- 627.115
cabcost[ 13 , 7 ]<- 411.1133
cabcost[ 13 , 8 ]<- 880.0728
cabcost[ 13 , 9 ]<- 620.488
cabcost[ 13 , 10 ]<- 477.459
cabcost[ 13 , 11 ]<- 378.5906
cabcost[ 13 , 12 ]<- 1608.082
cabcost[ 13 , 13 ]<- 0
cabcost[ 13 , 14 ]<- 858.251
cabcost[ 13 , 15 ]<- 700.8213
cabcost[ 13 , 16 ]<- 348.2725
cabcost[ 13 , 17 ]<- 955.6191
cabcost[ 13 , 18 ]<- 879.9795
cabcost[ 13 , 19 ]<- 1265.573
cabcost[ 13 , 20 ]<- 651.1179
cabcost[ 13 , 21 ]<- 254.9977
cabcost[ 13 , 22 ]<- 1808.52
cabcost[ 13 , 23 ]<- 1872.696
cabcost[ 13 , 24 ]<- 660.5173
cabcost[ 13 , 25 ]<- 751.4614
cabcost[ 14 , 1 ]<- 592.5679
cabcost[ 14 , 2 ]<- 949.5669
cabcost[ 14 , 3 ]<- 1266.851
cabcost[ 14 , 4 ]<- 1186.858
cabcost[ 14 , 5 ]<- 947.3188
cabcost[ 14 , 6 ]<- 1084.5
cabcost[ 14 , 7 ]<- 1097.608
cabcost[ 14 , 8 ]<- 1714.651
cabcost[ 14 , 9 ]<- 1151.868
cabcost[ 14 , 10 ]<- 963.7202
cabcost[ 14 , 11 ]<- 1236.192
cabcost[ 14 , 12 ]<- 2335.816
cabcost[ 14 , 13 ]<- 858.251
cabcost[ 14 , 14 ]<- 0
cabcost[ 14 , 15 ]<- 1500.774
cabcost[ 14 , 16 ]<- 675.7505
cabcost[ 14 , 17 ]<- 1098.282
cabcost[ 14 , 18 ]<- 1021.611
cabcost[ 14 , 19 ]<- 1977.613
cabcost[ 14 , 20 ]<- 1015.165
cabcost[ 14 , 21 ]<- 1065.599
cabcost[ 14 , 22 ]<- 2591.447
cabcost[ 14 , 23 ]<- 2725.79
cabcost[ 14 , 24 ]<- 197.8015
cabcost[ 14 , 25 ]<- 923.2229
cabcost[ 15 , 1 ]<- 908.7715
cabcost[ 15 , 2 ]<- 938.7461
cabcost[ 15 , 3 ]<- 1124.778
cabcost[ 15 , 4 ]<- 345.8738
cabcost[ 15 , 5 ]<- 598.541
```

```
cabcost[ 15 , 6 ]<- 626.1548
cabcost[ 15 , 7 ]<- 851.8228
cabcost[ 15 , 8 ]<- 694.0088
cabcost[ 15 , 9 ]<- 535.0244
cabcost[ 15 , 10 ]<- 1046.119
cabcost[ 15 , 11 ]<- 405.0906
cabcost[ 15 , 12 ]<- 1530.57
cabcost[ 15 , 13 ]<- 700.8213
cabcost[ 15 , 14 ]<- 1500.774
cabcost[ 15 , 15 ]<- 0
cabcost[ 15 , 16 ]<- 1039.77
cabcost[ 15 , 17 ]<- 1018.399
cabcost[ 15 , 18 ]<- 987.8645
cabcost[ 15 , 19 ]<- 1280.737
cabcost[ 15 , 20 ]<- 728.3743
cabcost[ 15 , 21 ]<- 450.3982
cabcost[ 15 , 22 ]<- 1589.835
cabcost[ 15 , 23 ]<- 1401.321
cabcost[ 15 , 24 ]<- 1311.21
cabcost[ 15 , 25 ]<- 922.3145
cabcost[ 16 , 1 ]<- 426.1877
cabcost[ 16 , 2 ]<- 999.5005
cabcost[ 16 , 3 ]<- 1368.267
cabcost[ 16 , 4 ]<- 830.3635
cabcost[ 16 , 5 ]<- 700.4368
cabcost[ 16 , 6 ]<- 922.3181
cabcost[ 16 , 7 ]<- 423.7053
cabcost[ 16 , 8 ]<- 1066.563
cabcost[ 16 , 9 ]<- 936.2502
cabcost[ 16 , 10 ]<- 305.3132
cabcost[ 16 , 11 ]<- 674.479
cabcost[ 16 , 12 ]<- 1661.778
cabcost[ 16 , 13 ]<- 348.2725
cabcost[ 16 , 14 ]<- 675.7505
cabcost[ 16 , 15 ]<- 1039.77
cabcost[ 16 , 16 ]<- 0
cabcost[ 16 , 17 ]<- 1178.439
cabcost[ 16 , 18 ]<- 1095.657
cabcost[ 16 , 19 ]<- 1304.043
cabcost[ 16 , 20 ]<- 918.5615
cabcost[ 16 , 21 ]<- 601.9917
cabcost[ 16 , 22 ]<- 1916.578
cabcost[ 16 , 23 ]<- 2090.089
cabcost[ 16 , 24 ]<- 496.4224
cabcost[ 16 , 25 ]<- 963.0435
cabcost[ 17 , 1 ]<- 756.1987
cabcost[ 17 , 2 ]<- 179.2426
cabcost[ 17 , 3 ]<- 190.3157
cabcost[ 17 , 4 ]<- 720.4687
cabcost[ 17 , 5 ]<- 578.3286
cabcost[ 17 , 6 ]<- 409.3542
cabcost[ 17 , 7 ]<- 1362.874
cabcost[ 17 , 8 ]<- 1625.87
cabcost[ 17 , 9 ]<- 489.5645
cabcost[ 17 , 10 ]<- 1417.072
cabcost[ 17 , 11 ]<- 1096.712
cabcost[ 17 , 12 ]<- 2453.352
```

```
cabcost[ 17 , 13 ]<- 955.6191
cabcost[ 17 , 14 ]<- 1098.282
cabcost[ 17 , 15 ]<- 1018.399
cabcost[ 17 , 16 ]<- 1178.439
cabcost[ 17 , 17 ]<- 0
cabcost[ 17 , 18 ]<- 84.3365
cabcost[ 17 , 19 ]<- 2143.565
cabcost[ 17 , 20 ]<- 328.7515
cabcost[ 17 , 21 ]<- 880.5469
cabcost[ 17 , 22 ]<- 2574.082
cabcost[ 17 , 23 ]<- 2415.489
cabcost[ 17 , 24 ]<- 1008.2
cabcost[ 17 , 25 ]<- 215.561
cabcost[ 18 , 1 ]<- 672.5906
cabcost[ 18 , 2 ]<- 96.2744
cabcost[ 18 , 3 ]<- 274.3105
cabcost[ 18 , 4 ]<- 675.3437
cabcost[ 18 , 5 ]<- 512.3965
cabcost[ 18 , 6 ]<- 365.6853
cabcost[ 18 , 7 ]<- 1288.966
cabcost[ 18 , 8 ]<- 1574.822
cabcost[ 18 , 9 ]<- 453.2583
cabcost[ 18 , 10 ]<- 1337.648
cabcost[ 18 , 11 ]<- 1038.645
cabcost[ 18 , 12 ]<- 2396.794
cabcost[ 18 , 13 ]<- 879.9795
cabcost[ 18 , 14 ]<- 1021.611
cabcost[ 18 , 15 ]<- 987.8645
cabcost[ 18 , 16 ]<- 1095.657
cabcost[ 18 , 17 ]<- 84.3365
cabcost[ 18 , 18 ]<- 0
cabcost[ 18 , 19 ]<- 2082.316
cabcost[ 18 , 20 ]<- 273.4106
cabcost[ 18 , 21 ]<- 818.1228
cabcost[ 18 , 22 ]<- 2526.562
cabcost[ 18 , 23 ]<- 2388.689
cabcost[ 18 , 24 ]<- 926.6267
cabcost[ 18 , 25 ]<- 132.7684
cabcost[ 19 , 1 ]<- 1590.224
cabcost[ 19 , 2 ]<- 1999.584
cabcost[ 19 , 3 ]<- 2299.429
cabcost[ 19 , 4 ]<- 1447.104
cabcost[ 19 , 5 ]<- 1570.725
cabcost[ 19 , 6 ]<- 1743.432
cabcost[ 19 , 7 ]<- 895.0908
cabcost[ 19 , 8 ]<- 593.4216
cabcost[ 19 , 9 ]<- 1682.489
cabcost[ 19 , 10 ]<- 1017.332
cabcost[ 19 , 11 ]<- 1048.539
cabcost[ 19 , 12 ]<- 358.3762
cabcost[ 19 , 13 ]<- 1265.573
cabcost[ 19 , 14 ]<- 1977.613
cabcost[ 19 , 15 ]<- 1280.737
cabcost[ 19 , 16 ]<- 1304.043
cabcost[ 19 , 17 ]<- 2143.565
cabcost[ 19 , 18 ]<- 2082.316
cabcost[ 19 , 19 ]<- 0
```

```
cabcost[ 19 , 20 ]<- 1814.83
cabcost[ 19 , 21 ]<- 1264.193
cabcost[ 19 , 22 ]<- 661.6543
cabcost[ 19 , 23 ]<- 1129.327
cabcost[ 19 , 24 ]<- 1800.098
cabcost[ 19 , 25 ]<- 1968.689
cabcost[ 20 , 1 ]<- 527.3008
cabcost[ 20 , 2 ]<- 210.7656
cabcost[ 20 , 3 ]<- 494.2224
cabcost[ 20 , 4 ]<- 403.8657
cabcost[ 20 , 5 ]<- 255.6551
cabcost[ 20 , 6 ]<- 104.6478
cabcost[ 20 , 7 ]<- 1049.266
cabcost[ 20 , 8 ]<- 1301.511
cabcost[ 20 , 9 ]<- 198.9058
cabcost[ 20 , 10 ]<- 1125.041
cabcost[ 20 , 11 ]<- 768.1641
cabcost[ 20 , 12 ]<- 2125.512
cabcost[ 20 , 13 ]<- 651.1179
cabcost[ 20 , 14 ]<- 1015.165
cabcost[ 20 , 15 ]<- 728.3743
cabcost[ 20 , 16 ]<- 918.5615
cabcost[ 20 , 17 ]<- 328.7515
cabcost[ 20 , 18 ]<- 273.4106
cabcost[ 20 , 19 ]<- 1814.83
cabcost[ 20 , 20 ]<- 0
cabcost[ 20 , 21 ]<- 552.4229
cabcost[ 20 , 22 ]<- 2253.211
cabcost[ 20 , 23 ]<- 2128.828
cabcost[ 20 , 24 ]<- 875.2542
cabcost[ 20 , 25 ]<- 194.5945
cabcost[ 21 , 1 ]<- 483.4673
cabcost[ 21 , 2 ]<- 736.3755
cabcost[ 21 , 3 ]<- 1043.484
cabcost[ 21 , 4 ]<- 255.8823
cabcost[ 21 , 5 ]<- 307.3289
cabcost[ 21 , 6 ]<- 491.1125
cabcost[ 21 , 7 ]<- 537.6206
cabcost[ 21 , 8 ]<- 780.9512
cabcost[ 21 , 9 ]<- 450.2585
cabcost[ 21 , 10 ]<- 677.0608
cabcost[ 21 , 11 ]<- 229.4867
cabcost[ 21 , 12 ]<- 1582.369
cabcost[ 21 , 13 ]<- 254.9977
cabcost[ 21 , 14 ]<- 1065.599
cabcost[ 21 , 15 ]<- 450.3982
cabcost[ 21 , 16 ]<- 601.9917
cabcost[ 21 , 17 ]<- 880.5469
cabcost[ 21 , 18 ]<- 818.1228
cabcost[ 21 , 19 ]<- 1264.193
cabcost[ 21 , 20 ]<- 552.4229
cabcost[ 21 , 21 ]<- 0
cabcost[ 21 , 22 ]<- 1735.937
cabcost[ 21 , 23 ]<- 1712.136
cabcost[ 21 , 24 ]<- 871.6396
cabcost[ 21 , 25 ]<- 706.5024
cabcost[ 22 , 1 ]<- 2140.978
```

```
cabcost[ 22 , 2 ]<- 2456.263
cabcost[ 22 , 3 ]<- 2703.402
cabcost[ 22 , 4 ]<- 1853.617
cabcost[ 22 , 5 ]<- 2036.128
cabcost[ 22 , 6 ]<- 2164.855
cabcost[ 22 , 7 ]<- 1493.843
cabcost[ 22 , 8 ]<- 955.802
cabcost[ 22 , 9 ]<- 2086.845
cabcost[ 22 , 10 ]<- 1649.619
cabcost[ 22 , 11 ]<- 1506.451
cabcost[ 22 , 12 ]<- 361.5388
cabcost[ 22 , 13 ]<- 1808.52
cabcost[ 22 , 14 ]<- 2591.447
cabcost[ 22 , 15 ]<- 1589.835
cabcost[ 22 , 16 ]<- 1916.578
cabcost[ 22 , 17 ]<- 2574.082
cabcost[ 22 , 18 ]<- 2526.562
cabcost[ 22 , 19 ]<- 661.6543
cabcost[ 22 , 20 ]<- 2253.211
cabcost[ 22 , 21 ]<- 1735.937
cabcost[ 22 , 22 ]<- 0
cabcost[ 22 , 23 ]<- 694.9363
cabcost[ 22 , 24 ]<- 2404.839
cabcost[ 22 , 25 ]<- 2430.269
cabcost[ 23 , 1 ]<- 2184.402
cabcost[ 23 , 2 ]<- 2339.509
cabcost[ 23 , 3 ]<- 2503.828
cabcost[ 23 , 4 ]<- 1733.132
cabcost[ 23 , 5 ]<- 1967.256
cabcost[ 23 , 6 ]<- 2027.319
cabcost[ 23 , 7 ]<- 1686.675
cabcost[ 23 , 8 ]<- 1024.566
cabcost[ 23 , 9 ]<- 1936.304
cabcost[ 23 , 10 ]<- 1891.166
cabcost[ 23 , 11 ]<- 1503.794
cabcost[ 23 , 12 ]<- 986.8149
cabcost[ 23 , 13 ]<- 1872.696
cabcost[ 23 , 14 ]<- 2725.79
cabcost[ 23 , 15 ]<- 1401.321
cabcost[ 23 , 16 ]<- 2090.089
cabcost[ 23 , 17 ]<- 2415.489
cabcost[ 23 , 18 ]<- 2388.689
cabcost[ 23 , 19 ]<- 1129.327
cabcost[ 23 , 20 ]<- 2128.828
cabcost[ 23 , 21 ]<- 1712.136
cabcost[ 23 , 22 ]<- 694.9363
cabcost[ 23 , 23 ]<- 0
cabcost[ 23 , 24 ]<- 2528.479
cabcost[ 23 , 25 ]<- 2321.873
cabcost[ 24 , 1 ]<- 408.1648
cabcost[ 24 , 2 ]<- 844.1663
cabcost[ 24 , 3 ]<- 1188.549
cabcost[ 24 , 4 ]<- 1005.761
cabcost[ 24 , 5 ]<- 775.239
cabcost[ 24 , 6 ]<- 933.196
cabcost[ 24 , 7 ]<- 912.2104
cabcost[ 24 , 8 ]<- 1519.174
```

```
cabcost[ 24 , 9 ]<- 992.3379
cabcost[ 24 , 10 ]<- 795.2136
cabcost[ 24 , 11 ]<- 1038.624
cabcost[ 24 , 12 ]<- 2157.517
cabcost[ 24 , 13 ]<- 660.5173
cabcost[ 24 , 14 ]<- 197.8015
cabcost[ 24 , 15 ]<- 1311.21
cabcost[ 24 , 16 ]<- 496.4224
cabcost[ 24 , 17 ]<- 1008.2
cabcost[ 24 , 18 ]<- 926.6267
cabcost[ 24 , 19 ]<- 1800.098
cabcost[ 24 , 20 ]<- 875.2542
cabcost[ 24 , 21 ]<- 871.6396
cabcost[ 24 , 22 ]<- 2404.839
cabcost[ 24 , 23 ]<- 2528.479
cabcost[ 24 , 24 ]<- 0
cabcost[ 24 , 25 ]<- 813.5513
cabcost[ 25 , 1 ]<- 540.7388
cabcost[ 25 , 2 ]<- 36.4947
cabcost[ 25 , 3 ]<- 405.7886
cabcost[ 25 , 4 ]<- 592.0278
cabcost[ 25 , 5 ]<- 399.2253
cabcost[ 25 , 6 ]<- 298.8486
cabcost[ 25 , 7 ]<- 1161.676
cabcost[ 25 , 8 ]<- 1475.479
cabcost[ 25 , 9 ]<- 392.9045
cabcost[ 25 , 10 ]<- 1205.747
cabcost[ 25 , 11 ]<- 931.7148
cabcost[ 25 , 12 ]<- 2288.748
cabcost[ 25 , 13 ]<- 751.4614
cabcost[ 25 , 14 ]<- 923.2229
cabcost[ 25 , 15 ]<- 922.3145
cabcost[ 25 , 16 ]<- 963.0435
cabcost[ 25 , 17 ]<- 215.561
cabcost[ 25 , 18 ]<- 132.7684
cabcost[ 25 , 19 ]<- 1968.689
cabcost[ 25 , 20 ]<- 194.5945
cabcost[ 25 , 21 ]<- 706.5024
cabcost[ 25 , 22 ]<- 2430.269
cabcost[ 25 , 23 ]<- 2321.873
cabcost[ 25 , 24 ]<- 813.5513
cabcost[ 25 , 25 ]<- 0
cablocNames[ 1 ]<- " Atlanta  "
cablocNames[ 2 ]<- " Baltimore  "
cablocNames[ 3 ]<- " Boston  "
cablocNames[ 4 ]<- " Chicago  "
cablocNames[ 5 ]<- " Cincinnati  "
cablocNames[ 6 ]<- " Cleveland  "
cablocNames[ 7 ]<- " Dallas-Fort Worth "
cablocNames[ 8 ]<- " Denver  "
cablocNames[ 9 ]<- " Detroit  "
cablocNames[ 10 ]<- " Houston  "
cablocNames[ 11 ]<- " Kansas City "
cablocNames[ 12 ]<- " Los Angeles "
cablocNames[ 13 ]<- " Memphis  "
cablocNames[ 14 ]<- " Miami  "
cablocNames[ 15 ]<- " Minneapolis  "
```

```
cablocNames[ 16 ]<- " New Orleans "
cablocNames[ 17 ]<- " New York "
cablocNames[ 18 ]<- " Philadelphia  "
cablocNames[ 19 ]<- " Phoenix  "
cablocNames[ 20 ]<- " Pittsburgh  "
cablocNames[ 21 ]<- " St. Louis "
cablocNames[ 22 ]<- " San Francisco "
cablocNames[ 23 ]<- " Seattle  "
cablocNames[ 24 ]<- " Tampa  "
cablocNames[ 25 ]<- " Washington DC "
```

Appendix 4: Computational results of the benchmark papers

Appendix 4.1: computational results for the single allocation p-hub median problem in Skorin-Kapov et al. (1996)

Table 6: Optimal solutions of the LP relaxation of the single allocation p-hub median problem for instances of the 'CAB data set' (Skorin-Kapov et al., 1996, p 590).

n	p		α				
			0.2	0.4	0.6	0.8	1
10	2	LP_{obj}	615.99	674.31	732.63	790.94	835.81
		hubs	7;9	7;9	7;9	7;9	4;7
	3		491.93	567.91	643.89	716.98	776.68
			4;6;7	4;6;7	4;6;7	4;7;9	4;7;9
	4		395.13	493.79	577.83	661.41	736.26
			3;4;6;7	4;6;7;8	4;6;7;8	4;7;8;9	1;4;7;9
15	2		981.28	1062.63	1143.97	1190.77	1221.92
			4;12	4;12	4;12	4;11	4;11
	3		799.97	905.1	1009.93	1099.51	1167.23
			4;7;12	4;7;12	4;7;12	4;7;8	NON INTEGER
	4		639.77	779.71	910.21	1026.52	1118.23
			4;7;12;14	4;7;12;14	1;4;7;12	1;4;7;8	1;4;7;8
20	2		979.09	1042.57	1106.04	1169.52	1210.08
			4;17	4;17	4;17	4;17	4;20
	3		724.54	847.77	970.99	1086.07	1156.07
			4;12;17	4;12;17	4;12;17	NON INTEGER	4;11;20
	4		477.62	727.1	869.16	1008.49	1107.92
			4;12;16;17	1;4;12;17	1;4;12;17	1;4;8;7	NON INTEGER
25	2		1000.91	1101.63	1201.21	1294.08	1359.19
			12;20	12;29	12;20	12;20	8;20
	3		767.35	901.7	1033.56	1158.83	1256.63
			4;12;17	4;12;18	2;4;12	2;4;12	4;8;20
	4		629.63	787.51	939.21	1087.66	1211.23
			4;12;17;24	1;4;12;17	1;4;12;17	1;4;12;18	4;7;8;20

"'LP_{obj}' indicates the objective function value of the linear programming relaxation of the p-HM-SA model; 'hubs' indicate hub locations whenever the LP solution is integer, otherwise it is stated that the solution is not integer" (Skorin-Kapov et al., 1996, p590).

Table 7: Integral optimal solutions of the single allocation p-hub median problem for instances of the 'CAB data set' for which the LP relaxation provided non integral solutions (Skorin-Kapov et al., 1996, p 590).

n	p	α	LPobj	IPobj	Hubs
15	3	1	1167.23	1168.68	4;7;8
20	4	1	1107.92	1111.01	4;8;13;20
20	3	0.8	1086.07	1091.05	4;8;17

Appendix 4.2: computational results for the single allocation p-hub center problem in Ernst et al. (2009)

Table 8: Optimal objective function values of the single allocation p-hub center problem for instances of the 'CAB data set' (Ernst et al., 2009, p 2237).

n	p	α				
		0.2	0.4	0.6	0.8	1
10	2	1425.58	1627.52	1759.13	1759.13	1839.65
	3	1119.54	1185.07	1387.	1588.94	1790.55
	4	830.25	968.2	1146.19	1454.44	1764.79
15	2	2005.02	2160.75	2214.09	2423.8	2609.18
	3	1749.04	1760.15	1844.92	2166.54	2600.08
	4	1340.96	1434.38	1754.51	2080.06	2166.54
20	2	1892.99	2160.75	2274.67	2501.93	2609.18
	3	1551.25	1760.15	1997.79	2263.54	2600.08
	4	1355.41	1472.71	1834.83	2153.	2600.08
25	2	2131.2	2402.55	2558.74	2714.93	2827.16
	3	1923.12	2100.47	2340.25	2554.13	2758.39
	4	1619.48	1884.84	2182.49	2454.35	2726.28

Table 11: CPU time of the single allocation p-hub center problem for instances of the 'CAB data set' (Ernst et al., 2009, pp 2236-2237).

n	p	α				
		0.2	0.4	0.6	0.8	1
10	2	.34	.12	.09	.06	.06
	3	.29	.07	.08	.04	.04
	4	.09	.04	.03	.04	.01
15	2	.77	.35	.32	.29	.21
	3	.4	.35	.22	.21	.11
	4	.43	.25	.16	.17	.21
20	2	.94	1.29	.81	.65	.48
	3	1.11	1.1	.85	.74	.32
	4	.99	1.09	.75	.79	.24
25	2	2.64	1.88	2.35	1.68	1.52
	3	2.08	.77	3.23	2.29	1.58
	4	1.49	2.59	3.03	2.54	1.32

The values are expressed in seconds.

Appendix 4.3: computational results for the single allocation hub covering problem in Wagner (2008)

Table 12: Optimal objective function values and CPU time of the single allocation hub covering problem for instances of the 'CAB data set' (with β rounded down) (Wagner, 2008, p 936).

α	β	p	time
0.2	2131	3	0.547
0.6	2558	3	0.344
1	2827	3	0.031
0.2	1923	4	1.281
0.6	2340	4	0.359
1	2758	4	0.031
0.2	1619	5	0.078
0.6	2182	5	3.875
1	2726	5	0.016

The CPU time is expressed in seconds and accounts for the CPU time of both CPLEX and the preprocessing algorithms. 'p' (the number of hubs to locate) is the objective function value.

Table 13: Optimal objective function values and CPU time of the single allocation hub covering problem for instances of the 'CAB data set' (with β rounded up) (Wagner, 2008, p 936).

α	β	p	time
0.2	2132	2	0.094
0.6	2559	2	0.141
1	2828	2	0.031
0.2	1924	3	0.141
0.6	2341	3	0.234
1	2759	3	0.031
0.2	1620	4	0.062
0.6	2183	4	0.109
1	2727	4	0.015

The CPU times are expressed in seconds and accounts for the CPU time of both CPLEX and the preprocessing algorithms. 'p' (the number of hubs to locate) is the objective function value.

Appendix 4.4: computational results for the single allocation p-hub maximal covering problem in Hwang and Lee (2012)

Table 16: Optimal objective function values of the single allocation p-hub maximal covering problem for instances of the 'CAB data set' (Hwang and Lee, 2012, p 387).

n	p	α				
		0.2	0.4	0.6	0.8	1
10	2	994540	994540	987490	999026	984836
	3	999026	990542	984530	999026	999026
	4	999026	999026	999026	999026	999026
	5	991270	999026	999026	999026	999026
15	2	2358068	2364942	2364942	2364942	2364942
	3	2358068	2364942	2304218	2320434	2364942
	4	2364942	2364942	2364942	2364942	2364942
	5	2353712	2364942	2320434	N/A	N/A

N/A: Could not find the solution because of the excessive memory required.

Table 17: CPU time of the single allocation p-hub maximal covering problem for instances of the 'CAB data set' (Hwang and Lee, 2012, p 387).

n	p	α						
		0.2	0.4	0.6	0.8	1	mean	maximum
10	2	105.5	166.66	176.58	145.27	140.08		
	3	96.09	72.26	147.2	166.08	194.95		
	4	13.38	32.27	40.67	116.94	171.92	109.22	238.39
	5	9.39	9.03	24.74	116.95	238.39		
15	2	69593.89	22992.23	28797.33	31358.06	79142.94		
	3	46564.87	44563.3	24947.25	152505.4	48012.04		
	4	11323.92	9143.45	97384.2	537666.2	726322.9	108848.6	726322.9
	5	2840.75	1689.92	24426.86	N/A	N/A		

The values are expressed in seconds.

Table 18: suboptimal objective function values of the single allocation p-hub maximal covering problem for instances of the 'CAB data set' after 5 hours of execution of CPLEX. (Hwang and Lee, 2012, p 388)

n	p	α				
		0.2	0.4	0.6	0.8	1
20	2	5730024	5641474	5724572	5651126	5680064
	3	5736620	5590496	5563324	5671362	5710086
	4	5714034	5539926	5661780	5635422	5710086
	5	5641474	5647056	5581418	5656732	5710086
25	2	8512014	8490416	8380482	8265190	8413908
	3	8520466	8454224	8393032	8331996	8527758
	4	8433028	8332814	8333976	8477300	8489150
	5	8387986	8348758	8397124	8322308	8489150

Table 19: Cover radii considered in the computational results of Kara and Tansel (2003) and Hwang and Lee (2012) (Hwang and Lee, 2012, p 386).

n	α				
	0.2	0.4	0.6	0.8	1
10	1425	1627	1671	1744	1839
	1117	1185	1387	1589	1791
	811	970	1148	1457	1770
	736	863	1079	1413	1766
15	2004	2019	2103	2424	2611
	1638	1741	1844	2165	2610
	1324	1436	1756	2100	2605
	1149	1287	1560	2080	2600
20	1851	2067	2255	2493	2611
	1549	1744	1996	2264	2605
	1356	1473	1835	2154	2601
	1162	1386	1663	2118	2600
25	2136	2401	2557	2713	2826
	1913	2099	2336	2552	2762
	1617	1881	2184	2457	2726
	1346	1597	2002	2307	2725

Appendix 5: Authorization mail of Dr. Houyuan Jiang for a quotation

Dear Berouayel,

Thank you very much for your interests in our research. Yes, I am happy for you to cite the paper. Good luck with your research.

Best regards, Houyuan Jiang

From: Aladin Berouayel [mailto:aladin.berouayel@student.uclouvain.be]
Sent: 25 July 2016 12:28
To: Houyuan Jiang <h.jiang@jbs.cam.ac.uk>
Subject: Quotation request for "Reformulations and Computational Results for the Uncapacitated Single Allocation Hub Covering Problem"

Dr. Houyuan Jiang,

I am a student in business engineering in the Louvain School of Management of the Université Catholique de Louvain. I'm currently writing my thesis that consists of R implementations of single allocation hub location models. I intend on implementing the model USAHCoP-r of the working paper "Reformulations and Computational Results for the Uncapacitated Single Allocation Hub Covering Problem" in the "Cambridge Judge Business School Working Papers". It would request an author's permission. Could you please tell me if I am authorized to quote your paper in my thesis?

Thank you very much for your time.

Berouayel Aladin

Universit Catholique de Louvain

References

Alumur, S., & Kara, B. Y. (2008), Network hub location problems: The state of the art. *European journal of Operational Research*, 190(1), 1-21.

Alumur, S. A., Kara, B. Y., & Karasan, O. E. (2012). Multimodal hub location and hub network design. *Omega*, *40*(6), 927-939.

Baumgartner, S. (2003). Polyhedral analysis of hub center problems. *Doctoral dissertation, Technische Universität Kaiserslautern.*

Berkelaar, M., & others (2015). *Package 'lpSolve'*. https://cran.r-project.org/web/packages/lpSolve/lpSolve.pdf (Retrieved on 04/08/2016) Reference manual of the package 'lpSolve'.

Boland, N., Krishnamoorthy, M., Ernst, A. T., & Ebery, J. (2004). Preprocessing and cutting for multiple allocation hub location problems. *European Journal Of Operational Research*, 155(3), 638-653.

Brunel University London (n.a.). *CAB data set.* http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/phub4.txt (Retrieved on 04/08/2016) Data set used for the computational analysis.

Bryan, D. L., & O'Kelly, M. E. (1999). Hub-and-spoke networks in air transportation: An analytical review. *Journal Of Regional Science*, 39(2), 275-295.

Campbell, J. F. (1994). Integer programming formulations of discrete hub location problems. *European Journal Of Operational Research*, 72(2), 387-405.

Campbell, J. F., Ernst, A. T., & Krishnamoorthy, M. (2005). Hub Arc Location Problems: Part I--Introduction and Results. *Management Science*, *51*(10), 1540-1555.

Contreras, I., Cordeau, J., & Laporte, G. (2011). Stochastic uncapacitated hub location. *European Journal Of Operational Research*, *212*(3), 518-528.

da Graça Costa, M., Captivo, M. E., & Clímaco, J. (2008). Capacitated single allocation hub location problem—A bi-criteria approach. *Computers & Operations Research*, 35(11), 3671-3695.

Damgacioglu, H., Dinler, D., Evin Ozdemirel, N., & Iyigun, C. (2015). A genetic algorithm for the uncapacitated single allocation planar hub location problem. *Computers & Operations Research*, 62, 224-236.

de Camargo, R. S., Miranda Jr, G., Ferreira, R. P. M., & Luna, H. P. (2009). Multiple allocation hub-and-spoke network design under hub congestion. *Computers & Operations Research*, *36*(12), 3097-3106.

Delplace, A. (2011). Modèles de localisation en R: théorie et appliquations. *TFE Master 60, Université Catholique de Louvain.*

Ebery, J., Krishnamoorthy, M., Ernst, A., & Boland, N. (2000). The capacitated multiple allocation hub location problem: Formulations and algorithms. *European Journal Of Operational Research*, 120(3), 614-631.

Ernst, A. T., Hamacher, H., Jiang, H., Krishnamoorthy, M., & Woeginger, G. (2009). Uncapacitated single and multiple allocation p-hub center problems. *Computers & Operations Research*, 36(7), 2230-2241.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Baatar, D. (2011). Reformulations and computational results for uncapacitated single and multiple allocation hub covering problems. *University of Cambridge, judge business school: working papers series*.

Ernst, A. T., & Krishnamoorthy, M. (1996), Efficient Algorithms for the Uncapacitated Single Allocation p-hub Median Problem. *Location Science*, 4 (3), 139–154.

Ernst, A. T., & Krishnamoorthy, M. (1998). Exact and heuristic algorithms for the uncapacitated multiple allocation p-hub median problem. *European Journal Of Operational Research*, 104(1), 100-112.

Ernst, A. T., & Krishnamoorthy, M. (1999). Solution algorithms for the capacitated single allocation hub location problem. *Annals Of Operations Research*, 86(1-4), 141-159.

ETH Zurïch (2016). *CPU Time Used.* https://stat.ethz.ch/R-manual/R-devel/library/base/html/system.time.html (Retrieved on 04/08/2016) R documentation on the function system.time().

ETH Zurïch (n.a.). *Suspend Execution for a Time Interval.* https://stat.ethz.ch/R-manual/R-devel/library/base/html/Sys.sleep.html (Retrieved on 04/08/2016) Documentation on how to suspend the execution of the execution of an R script.

Free Software Foundation Inc (2016), *Qu'est ce que GNU?* https://www.gnu.org/ (Retrieved on 30/06/2016) Summary of the implications of a GNU project license.

Gelareh, S., Monemi, R. N., & Nickel, S. (2015). Multi-period hub location problems in transportation. *Transportation Research Part E: Logistics and Transportation Review*, *75*, 67-94.

Gesmann, M., de Castillo, D., & Cheng, J. (2016), Package 'googleVis'. https://cran.r-project.org/web/packages/googleVis/googleVis.pdf (Retrieved on 03/08/2016) Reference manual of the package 'googleVis'.

Github (n.a.). *Profvis – Interactive Vizualizations for Profiling R Code.* https://rstudio.github.io/profvis/index.html (Retrieved on 04/08/2016) Introduction to the function and package 'profvis'.

Google (2016). Vizualization: GeoChart. https://developers.google.com/chart/interactive/docs/gallery/geochart (Retrieved on 04/08/2016). Introduction to the GeoChart of the Google chart tools.

Google (n.a. (a)). *About Google chart tools.* https://developers.google.com/chart/ (Retrieved on 04/08/2016) Home page of the Google chart tools.

Google (n.a. (b)). *Google Maps API.* https://developers.google.com/maps/ (Retrieved on 09/08/2016) Home page of the Google Maps API in French.

Hamacher, H. W., Labbé, M., Nickel, S., & Sonneborn, T. (2004), Adapting polyhedral properties from facility to hub location problems. *Discrete Applied Mathematics*, 145 (1), 104-116.

Hwang, Y. H., & Lee, Y. H. (2012). Uncapacitated single allocation p-hub maximal covering problem. *Computers & Industrial Engineering*, 63(2), 382-389.

Jinhyeon Sohn, J., & Sungsoo, P. (1998). Efficient solution procedure and reduced size formulations for p-hub location problems. *European Journal Of Operational Research*, 108(1), 118-126.

Kara, B. Y., & Tansel, B. Ç. (2000). On the single-assignment p-hub center problem. *European Journal Of Operational Research*, 125(3), 648-655.

Kara, B., & Tansel, B. (2003). The single-assignment hub covering problem: Models and linearizations. *Journal Of The Operational Research Society*, 54(1), 59-64.

Kenny, V., Nathal, M., & Saldana, S. (2014). *Heuristic algorithms.* https://optimization.mccormick.northwestern.edu/index.php/Heuristic_algorithms (Retrieved on 08/08/2016). Introduction to heuristic algorithms.

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

Makhorin, A. (2012). *GLPK (GNU Linear Programming Kit).* https://www.gnu.org/software/glpk/glpk.html (Retrieved on 03/08/2016) Introduction to the GNU Linear Programming Kit.

Marín, A., Cánovas, L., & Landete, M. (2006). New formulations for the uncapacitated multiple allocation hub location problem. *European Journal Of Operational Research*, 172(1), 274-292.

Microsoft (n.a.). *What is R?* https://mran.microsoft.com/documents/what-is-r/#rpackages (Retrieved on 03/08/2016) Introduction to R.

MIT (2013). *Degeneracy in Linear Programming.* http://ocw.mit.edu/courses/sloan-school-of-management/15-053-optimization-methods-in-management-science-spring-2013/tutorials/MIT15_053S13_tut07.pdf (Retrieved on 04/08/2016). MIT course on the degeneration of linear programs.

MIT (n.a.). *Big O notation.* http://web.mit.edu/16.070/www/lecture/big_o.pdf (Retrieved on 05/082016) Introduction to the complexity and to the big O notation.

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

O'Kelly, M. E. (1992). Hub facility location with fixed costs. *Papers in Regional Science*, 71(3), 293-306.

Peker, M., & Kara, B. Y. (2015). The P-Hub maximal covering problem and extensions for gradual decay functions. *Omega*, 54, 158-172.

R Core Team (2016). *Writing R Extensions.* https://cran.r-project.org/doc/manuals/R-exts.html (Retrieved on 04/08/2016) R manual on extensions writing.

Sim, T., Lowe, T. J., & Thomas, B. W. (2009). The stochastic -hub center problem with service-level constraints. *Computers & Operations Research*, *36*(12), 3166-3177.

Skorin-Kapov, D., Skorin-Kapov, J., & O'Kelly, M. (1996). Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal Of Operational Research*, 94(3), 582-593.

Sourceforge (n.a.). *Introduction to lp solve 5.5.2.3.* http://lpsolve.sourceforge.net/ (Retrieved on 04/08/2016). Introduction to the MILP solver 'lp-solve'.

The R foundation (2015). *Introduction to stringr.* https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html (Retrieved on 06/08/2016) Introduction to stringr.

The R foundation (n.a. (a)). *What is R?* https://www.r-project.org/about.html (Retrieved on 05/06/2016) Introduction to the R programming language and environment.

The R foundation (n.a. (b)). *The comprehensive R archive network.* https://cran.r-project.org/ (Retrieved on 05/06/2016) Home page of the CRAN network that shares R packages and other resources.

Theussl, S., Hornik, K., Buchta, C., & Schuchardt, H. (2016a). *Package 'Rglpk'.* https://cran.r-project.org/web/packages/Rglpk/Rglpk.pdf (Retrieved on 03/08/2016) Reference manual of the package 'Rglpk'.

Theussl, S., Hornik, K., Buchta, C., & Schuchardt, H. (2016b). *Rglpk: R/GNU Linear Programing Kit Interface.* https://cran.r-project.org/web/packages/Rglpk/index.html (Retrieved on 03/08/2016). General information about the package 'Rglpk'.

Theussl, S., & Borchers, H. W. (2016). *CRAN Task View: Optimization and Mathematical Programming.* https://cran.r-project.org/web/views/Optimization.html (Retrieved on 04/08/2016) List of packages that provide optimization solvers.

University of Pennsylvania (n.a.). *PNP-4.ppt.* www.seas.upenn.edu/~bhusnur4/cit596_spring2014/PNP.pptx *(Retrieved* on 04/08/2016) Introduction to the concepts P, NP and NP-Hard and introduction to complexity theory.

Wagner, B. (2008). Model formulations for hub covering problems. *Journal Of The Operational Research Society*, 59(7), 932-938.

Wickham, H. (2015). *Package 'stringr'.* https://cran.r-project.org/web/packages/stringr/stringr.pdf (Retrieved on 03/08/2016) Reference manual of the package 'stringr'.

Yaman, H. (2008). Star p-hub median problem with modular arc capacities. *Computers & Operations Research*, *35*(9), 3009-3019.
