Computers & Operations Research 36 (2009) 3088 – 3096

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/cor

A tabu-search based heuristic for the hub covering problem over incomplete hub networks[☆]

Hatice Calık, Sibel A. Alumur*, Bahar Y. Kara, Oya E. Karasan

Department of Industrial Engineering, Bilkent University, 06800 Ankara, Turkey

ARTICLE INFO

Available online 24 December 2008

Keywords:
Hub location
Hub covering problem
Network design

ABSTRACT

Hub location problems deal with finding the location of hub facilities and with the allocation of demand nodes to these located hub facilities. In this paper, we study the single allocation hub covering problem over incomplete hub networks and propose an integer programming formulation to this end. The aim of our model is to find the location of hubs, the hub links to be established between the located hubs, and the allocation of non-hub nodes to the located hub nodes such that the travel time between any origin–destination pair is within a given time bound. We present an efficient heuristic based on tabu search and test the performance of our heuristic on the CAB data set and on the Turkish network.

1. Introduction

Hub facilities serve as accumulation and distribution points in many-to-many distribution networks. The flow is consolidated at the hub facilities in order take advantage of the economies of scale. The hub location problem includes the selection of the location of hub facilities and the allocation of the demand nodes to these located hub facilities. Such a location problem arises in airline, cargo delivery, and telecommunication systems.

Considering how non-hub nodes are allocated to the located hub nodes, two basic types of hub networks are defined in the literature: single and multiple allocation. In single allocation hub networks, each non-hub node is allocated to exactly one hub; in multiple allocation networks, a non-hub node can be allocated to more than one hub.

O'Kelly [1] originally introduced the hub location problem. Later, O'Kelly [2] provided the first quadratic formulation for the single allocation p-hub (median) location problem. The objective of his model was to minimize the total transportation cost of flow. In order to reflect the economies of scale in hub-to-hub connections, O'Kelly introduced a constant discount factor, $\alpha \in [0,1]$, for using inter-hub connections.

The rest of the literature on the hub location problem primarily focused on the linearization of the quadratic model proposed in O'Kelly [2], for example, Campbell [3], Ernst and Krishnamoorthy [4], O'Kelly et al. [5], and Skorin-Kapov et al. [6]. These studies introduce different mathematical formulations and solution procedures for the minimization of the total transportation cost. One can refer to Campbell et al. [7] and to Alumur and Kara [8] for recent surveys on hub location problems.

Campbell [9] introduced different hub location problems (p-hub center, hub covering) to the literature and considered different objective functions. In particular, the hub covering problem minimizes the total cost of establishing hub facilities, so that the cost (or travel time) between any origin–destination pair is within a given bound. Campbell [9] provided quadratic as well as linear formulations for both single and multiple allocation variants of the problem. The first attempt to provide computational results for the single allocation hub covering problem, however, was from Kara and Tansel [10]. They also suggested various linear formulations and proved the NP-hardness of the hub covering problem. Ernst et al. [11] proposed a better mathematical formulation for the hub covering problem using the "radius" idea.

Most studies on hub location problems assume a complete hub network, that is, every hub pair in the hub network is interconnected with a hub link. One could justify the need to design incomplete hub networks with various applications. For example, in cargo delivery systems, sending separate trucks from a distribution center (hub) to all other distribution centers is costly in terms of the investment in the total number of trucks. Instead, forcing some trucks to visit more than one distribution center, when there is enough capacity, may decrease the total investment cost considerably. Similarly, in airline applications, an airline company may want to schedule flights from an airport to a large number of destinations. Assigning a separate aircraft and separate air staff for each destination causes congestion in airports and air networks as well as high investment and

[☆]This research is supported by Turkish Academy of Science.

* Corresponding author. Tel.: +90 312 2901289; fax: +90 312 2664054.
E-mail address: alumur@bilkent.edu.tr (S.A. Alumur).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.11.023

operating costs to the company. Additionally, in telecommunication systems, connecting all terminals directly may become an unnecessary and expensive way of providing quality service to users. Therefore, incomplete hub networks are important in practice.

Few studies in the literature consider hub location problems over incomplete hub networks. For example, Nickel et al. [12] modeled a problem for urban public transportation networks. They introduced a fixed cost of locating hub arcs while minimizing the total transportation cost in their network and modeled the multiple allocation version of the problem. Later, Campbell et al. [13,14] introduced hub arc location problems. Instead of locating hub facilities, these problems locate hub arcs with reduced unit costs. The resulting hub arc network in these problems does not need to be connected.

Perhaps the most closely related study in the literature to our problem is by Alumur and Kara [15]. They also considered the hub covering problem over incomplete hub networks, while focusing on cargo applications. They modeled a special structure of the incomplete hub network design problem, in which they allowed for visiting at most three hubs on a route. The authors showed that even for the tightest service-time requirements, in some cases, there is no need for a complete hub network.

In this paper, we study the hub covering problem over incomplete hub networks. Our aim is to find the location of hubs, the hub links to be established between the located hubs, and the allocation of non-hub nodes to the located hub nodes such that the travel time between any origin–destination pair is within a given time bound. Similar to other hub location studies, we use a constant time discount factor $\alpha \in [0,1]$ to represent the economies of scale in hub-to-hub connections, and we do not allow direct connections between the non-hub nodes. Unlike Nickel et al. [12], we consider the hub covering version of the problem, and the objective of minimizing the cost of establishing the hub network alone instead of minimizing the total cost of flow, and we model the single allocation case of the problem. Unlike Campbell et al. [13,14] we do not locate a fixed number of hub arcs and we force the hub arc network to be connected. In contrast with Alumur and Kara [15], we do not impose any structure on the hub network other than connectivity and do not model the synchronization of trucks. Our integer programming formulation involves $O(n^4)$ decision variables and $O(n^4)$ constraints. In order to solve realistically sized instances, we present a heuristic for this problem. In contrast to other hub location problems, constructing feasible solutions for the hub covering problem, especially with tight time bounds, is a challenge.

Many studies in the literature apply different heuristic approaches to hub location problems, for example, the tabu search heuristics proposed by Klincewicz [16] and Skorin-Kapov and Skorin-Kapov [17], the simulated annealing heuristic by Ernst and Krishnamoorthy [18], and the Lagrangean relaxation based heuristic by Pirkul and Schilling [19] for p-hub median problems. Additional contributions include a shortest-path based heuristic by Ebery et al. [20], a genetic algorithm by Cunha and Silva [21], a hybrid heuristic by Chen [22], and a dual-ascent heuristic by Cánovas et al. [23] for hub location problems with fixed costs. Lastly, proposals for p-hub center problems include a tabu search based heuristic by Pamuk and Sepil [24] and a greedy heuristic by Ernst et al. [25]. The reader should note that, for hub location problems, nearest allocation strategy (assigning a non-hub node to its nearest hub) does not necessarily give optimum solutions for the hub location problem [1].

In this paper, we relax the complete hub network assumption in the hub covering problem and contribute a novel mathematical formulation to the hub location literature. To be able to handle real sized problems, we propose a tabu search based heuristic for our problem. As will be apparent with the computational studies, the heuristic behaves quite effectively. To the best of our knowledge, both the formulation and the heuristic are new for the hub covering problem.

We propose an integer programming formulation in the second section of this paper. In the third section, we present and explain our heuristic algorithm. The fourth section is dedicated to the computational analysis. We test and compare the performance of the heuristic with the optimization solver CPLEX 10.1 on the well-known CAB data set. Computational results of our heuristic on the Turkish network of 81 nodes are also presented. The last section is devoted to concluding remarks.

2. Mathematical formulation

We assume that there is a given node set N with n nodes and a potential hub set $H \subseteq N$ with h nodes. The mathematical model locates hubs from the potential hub set, constructs the hub network, and allocates the remaining nodes in set N to these hubs, such that the travel time between any origin–destination pair is less than a given time bound, T. The objective of our mathematical model is to minimize the total cost of establishing hubs and hub links.

The parameters of the model are as follows. The parameter f_{ij} is the fixed cost of opening a hub link between nodes i and j, fh_k is the fixed cost of opening a hub at node k, and t_{ij} is the travel time from node i to node j. The parameter T is the given time bound, and α is the time discount factor for hub-to-hub connections. The time discount factor, $\alpha \in [0,1]$, is different and most likely to be higher than the cost discount factor; it is expected to be a number close to 1.

We define the decision variables of the model as follows:

$X_{ik} = 1$ if node i is allocated to a hub at node k; 0 otherwise.
$Z_{ij} = 1$ if there is a hub link between hub i and hub j $(i < j)$; 0 otherwise.
${Y_{ij}}^{kl} = 1$ if the hub link $\{i,j\}$ is used on the path from hub k to hub l in the direction from i to j; 0 otherwise.
$r_k =$ radius of hub k.

More specifically, the allocation decisions are taken care by the classical X_{ik} variables. Consistent with the literature, if the variable $X_{kk} = 1$ for some $k \in H$, it means that node k is a hub node. The Z_{ij} variables indicate the existing links in the hub network to be designed. Finally, the ${Y_{ij}}^{kl}$ variables are used to construct a directed path from every hub node k, to every other hub node l using the existing hub links. Each available hub link $\{i,j\}$ can be used in either orientation ((i,j) or (j,i)) as part of this path. Note that Y and Z variables are defined only between the established hubs and Y variables are directed while Z variables are not.

The objective of our mathematical model is to minimize the total cost of establishing hubs and hub links. With the previously defined parameters and decision variables, the objective function is expressed as follows:

$$\text{Minimize} \sum_{i \in H} \sum_{j \in H: j > i} f_{ij} Z_{ij} + \sum_{k \in H} fh_k X_{kk} \tag{1}$$

In the objective function, in the first term, we sum the individual fixed costs of establishing hub links; in the second term, we calculate the total cost of establishing hubs.

We group and explain the constraints of our mathematical model as follows:

Standard single allocation hub constraints:

$$\sum_{k \in H} X_{ik} = 1 \quad \forall i \in N \tag{2}$$

$$X_{ik} \leqslant X_{kk} \quad \forall i \in N, \;\; k \in H \tag{3}$$

$$X_{ik} \in \{0,1\} \quad \forall i \in N, \;\; k \in H \tag{4}$$

Since we model the single allocation case of the problem, constraints (2) and (4) ensure that every node is allocated to exactly one hub node. Constraint (3) states that a node cannot be allocated to another node unless that node is a hub node.

Hub link decision constraints:

$$Z_{ij} \leqslant X_{ii} \quad \forall i,j \in H,\ \ i<j \tag{5}$$

$$Z_{ij} \leqslant X_{jj} \quad \forall i,j \in H,\ \ i<j \tag{6}$$

$$Y_{ij}^{kl} + Y_{ji}^{kl} \leqslant Z_{ij} \quad \forall i,j,k,l \in H,\ \ i<j,\ \ k \neq l \tag{7}$$

$$Y_{ij}^{kl} \in \{0,1\} \quad \forall i,j,k,l \in H,\ \ i \neq j,\ \ k \neq l \tag{8}$$

$$Z_{ij} \in \{0,1\} \quad \forall i,j \in H,\ \ i<j \tag{9}$$

In order to establish a hub link $\{i,j\}$, both end nodes of that link, i.e., nodes i and j, need to be hub nodes (constraints (5) and (6)). By constraint (7) if a hub link is to be used as a part of the path to be constructed for a given origin–destination hub pair, that hub link has to be established. Constraint (7) also ensures that at most one of $Y_{ij}{}^{kl}$ and $Y_{ji}{}^{kl}$ variables can be one because they need to provide a simple directed path. Since Y variables are directed, while travelling from any hub k to hub l, only one direction of link $\{i,j\}$ may be utilized. Constraints (8) and (9) force the path variables and the hub link decision variables to be binary.

Flow balance constraints:

$$\sum_{j \in H: j \neq i} Y_{ij}^{il} \geqslant X_{ii} + X_{ll} - 1 \quad \forall i,l \in H,\ \ i \neq l \tag{10}$$

$$\sum_{j \in H: j \neq i} Y_{ji}^{kl} - \sum_{j \in H: j \neq i} Y_{ij}^{kl} = 0 \quad \forall i,k,l \in H,\ \ i \neq k,\ \ i \neq l,\ \ k \neq l \tag{11}$$

$$\sum_{j \in H: j \neq i} Y_{ji}^{ki} \geqslant X_{ii} + X_{kk} - 1 \quad \forall i,k \in H,\ \ i \neq k \tag{12}$$

$$Y_{ij}^{kl} + Y_{ji}^{kl} \leqslant X_{kk} \quad \forall i,j,k,l \in H,\ \ i<j,\ \ k \neq l \tag{13}$$

$$Y_{ij}^{kl} + Y_{ji}^{kl} \leqslant X_{ll} \quad \forall i,j,k,l \in H,\ \ i<j,\ \ k \neq l \tag{14}$$

Constraints (10)–(12) are the flow balance constraints in the hub network. Via these constraints, every hub node sends and receives one unit of flow, and the connectivity in the hub network is established. By constraint (10), if both nodes i and l are hubs, then the origin hub i sends one unit of flow to the destination hub l in the hub network. By constraint (12), if nodes i and k are both hub nodes, then the destination hub i receives one unit of flow from the origin hub k in the hub network. When hub node i is neither the origin nor the destination, then the incoming flow must equal the outgoing flow, by constraint (11). We route the flow only in the hub network; thus, we ensure, by constraints (13) and (14), that the origin and destination nodes can only be hub nodes.

Time bound constraints:

$$r_k \geqslant t_{ik} X_{ik} \quad \forall i \in N,\ \ k \in H \tag{15}$$

$$\sum_{i \in H} \sum_{j \in H: i \neq j} \alpha t_{ij} Y_{ij}^{kl} + r_k + r_l \leqslant T \quad \forall k,l \in H \tag{16}$$

We define a decision variable, radius r, similarly to the one used in Ernst et al. [11]. For each hub, by constraint (15), the r variable calculates the maximum travel time between that hub and the nodes that are allocated to it. Constraint (16) is the time bound constraint. For each pair of hubs, the radii of these hubs plus the discounted total travel time of the path, chosen by the Y variables, between these hubs using the established hub links must not be greater than the given time bound, T.

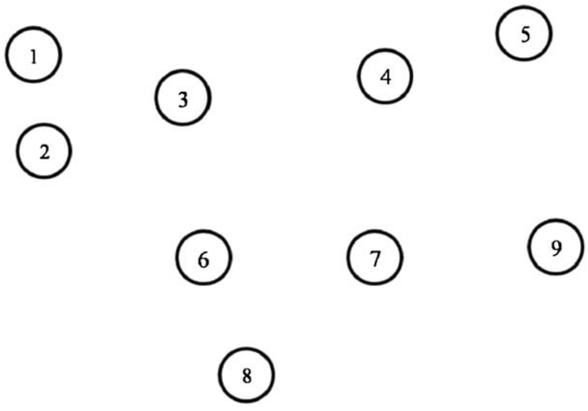

Fig. 1. The node set N.

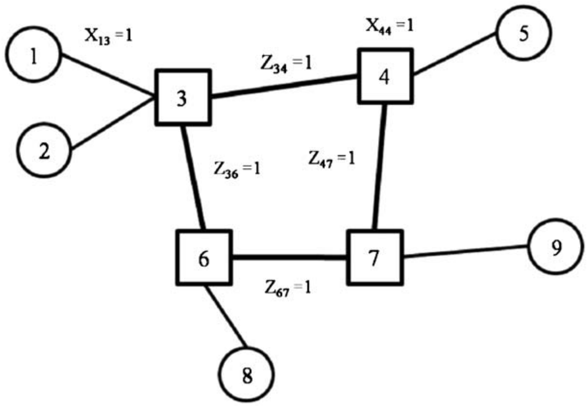

Fig. 2. The resulting network of a solution.

In order to explain our mathematical model thoroughly we present an example illustrated in Figs. 1 and 2.

Let Fig. 2 represent the resulting network of a potential solution regarding the node set in Fig. 1. According to this solution, $X_{33} = X_{44} = X_{66} = X_{77} = 1$ implying that nodes 3, 4, 6, and 7 are chosen as hub nodes. The variables $X_{13} = X_{23} = X_{54} = X_{86} = X_{97} = 1$ represent the allocations of the non-hub nodes to the hub nodes. Moreover, $Z_{34} = Z_{36} = Z_{47} = Z_{67} = 1$ indicate the constructed hub links. All other X and Z variables have zero values at this solution.

By constraints (13) and (14), all of the Y variables associated with the nodes 1, 2, 5, 8, and 9 are forced to be zero. By constraints (10)–(12) each of the hub nodes sends one unit of flow to all other hub nodes. Let us consider the flow from hub node 3 to hub node 7. By constraint (10), hub node 3 sends one unit of flow to hub node 7 in the network. Thus either $Y_{34}{}^{37}$ or $Y_{36}{}^{37}$ must be equal to 1. Similarly by constraint (12), hub node 7 must receive one unit of flow from hub node 3. Thus either $Y_{47}{}^{37}$ or $Y_{67}{}^{37}$ must be equal to 1. By constraint (11), the incoming flow must be equal to the outgoing flow for rest of the $Y_{ij}{}^{37}$ variables. Note that by constraint (7), for some hub link $\{i,j\}$, either one of $Y_{ij}{}^{37}$ and $Y_{ji}{}^{37}$ can take on the value one. Thus, there exist exactly two possible paths from hub node 3 to hub node 7. First one is by using hub arcs (3,6) and (6,7) and the second one is by using hub arcs (3,4) and (4,7). The model decides which path to choose by using the time bound constraint (16). Assume that

$\alpha t_{34} + \alpha t_{47} + r_3 + r_7 > T$ and $\alpha t_{36} + \alpha t_{67} + r_3 + r_7 \leqslant T$. Then the model lets $Y_{36}{}^{37} = Y_{67}{}^{37} = 1$ and all other $Y_{ij}{}^{37}$ variables to be zero.

Finally, the mathematical model we propose in this study consists of the objective function (1) and the constraints (2)–(16). In the worst case, $h = n$ and the model has $O(n^4)$ binary variables and $O(n^4)$ constraints. As expected, for greater values of n, the model results in a large number of variables and constraints which will render the solution to optimality extremely challenging. Consequently, to be able to answer this specific hub covering problem on realistic network dimensions, we introduce a tabu-based heuristic for our model in the next section.

3. Tabu search based heuristic algorithm

It is hard to solve most of the NP-complete problems to optimality for realistically sized instances. For the problem at hand, even finding a feasible solution is challenging. With this motivation, we decided to develop a heuristic algorithm for our problem. We used ideas from the well-known tabu search heuristic methodology in order to avoid getting stuck at a local optimum solution.

Our heuristic involves construction and improving phases. The algorithm starts with a set of initial hub locations determined with a procedure that we developed. With the given set of hub locations, the algorithm allocates the rest of the nodes to these hubs and constructs the hub network. In fact, the algorithm does not start with a complete solution but with a partial solution, i.e., a set of hubs, which does not guarantee a feasible allocation construction. During the solution construction phase, three different construction methods based on different allocation strategies are used. In each of these allocation strategies, feasible solutions are searched for initially over complete hub networks. When a feasible solution is found, in the improvement phase, hub links that do not lead to infeasibility are removed from the hub network to obtain better feasible solutions.

The construction phase is performed for a specified number (K) of random neighbors of a set of hub locations. The feasible neighbor with the best objective function value is selected for the next move. If no feasible solution is produced within $2{*}K$ neighbor traversals, a neighbor is selected randomly and moved to even if it is infeasible. Thus, our algorithm allows moving to infeasible neighbors of a solution.

The construction and improvement stages are performed at each move and the best of the feasible solutions found by the algorithm is reported. In summary, the algorithm seeks new solutions first by changing the hub set, then by constructing the allocations, and, finally, by reducing the hub connectivity.

In order to prevent cycling, we keep a list of tabu moves. In moving to a neighbor, both worse and better feasible solutions are accepted together with infeasible ones, unless they are tabu moves. By accepting worse feasible solutions, we may avoid getting stuck at a local optimum solution.

We provide formal and detailed descriptions of the steps of the algorithm below:

Solution: In a solution, all hub locations and allocations are determined, and the hub network is constructed, but its feasibility is not guaranteed.

Feasible solution: A solution is feasible if it guarantees that the shortest discounted travel time from each origin to each destination is within the time bound.

Move: At each iteration of the tabu search, a base hub set is chosen to detect the solutions that might be obtained from its neighborhoods. As a first step, this hub set is chosen from the initial hub locations set. During the following iterations, this hub set is chosen from the neighborhood of the base hub set of the previous iteration, with some criteria. The selection of the base hub set constitutes a move in the algorithm.

Initial hub locations: Finding an initial feasible solution for the hub covering problem is difficult for tight service time values. Therefore, instead of starting with a feasible solution, a base hub set is selected among several initial hub sets that are constructed with the following procedure, then, the feasible solutions are constructed by traversing neighbors of the base hub set.

In order to determine the initial hub sets, first an $n{\times}n$ cover matrix, $\mathbf{C} = [c_{ij}]$, is constructed as follows:

$$C_{ij} = \begin{cases} 1 & \text{if } t_{ij} \leqslant \dfrac{T}{2} \\ 0 & \text{otherwise} \end{cases}$$

where T is the time bound. For each node $i \in N$, the following steps are repeated: the node i is selected as a hub and the nodes that are not covered by node i are collected in a set. Among the remaining nodes, the node that covers the elements of this set the most is selected as another hub (if more than one such node exists, the remaining steps are followed for each case) and the covered elements are removed from the set. Until all nodes are covered, the node that most completely covers the uncovered set is selected as a hub and the elements covered by this hub are removed from the set. At the end of this process, at least one or more hub sets, which possibly contain different number of nodes, are constructed. Among the constructed hub sets, the one that contains the minimum number of hubs are selected and included in the initial hub locations set, say *locationsSet*, of the algorithm. Note that, for each of the elements of this set, although the allocations to hubs are within the time bound, the locations are not guaranteed to provide feasible solutions.

While the time limit is not exceeded, each hub set in *locationsSet* is chosen as the base hub set of the algorithm. If the tabu iteration limit is reached for a base hub set with no feasible solution, a randomly selected node is added to this base hub set, increasing the set size by one. The same steps are repeated until a feasible solution is obtained unless the time limit is exceeded. If all the elements of *locationsSet* are traversed before the time limit, the algorithm continues to select them for one more time.

Solution construction: When the locations of the hubs are determined, in order to construct feasible solutions with this given set of hubs, three different allocation strategies are performed: Types I, II, and III allocation. During all the allocation strategies, the feasibility of the solutions is determined by checking the time bound constraints of the problem. At the beginning of each allocation strategy, the hub network is assumed complete. As soon as a feasible solution is obtained at the end of any allocation strategy, the algorithm focuses on the hub network. To obtain better feasible solutions with the given allocations, hub links are removed randomly from the complete hub network. If the removal of a link leads to infeasibility, that link is added back to the solution, and another hub link is chosen, again randomly, to be removed. The three allocation strategies are described as follows:

(i) *Type* I *allocation*: In this strategy, as a starting point, the allocations are determined using the nearest allocation heuristic HEUR1 of O'Kelly [2], i.e., every non-hub node is allocated to its nearest hub node. We then concentrate on the hub with the largest radius, with respect to the nearest allocation strategy. All non-hub nodes are allocated to the hub with the largest radius, as long as this allocation does not increase the radius of this hub. In this way, the radii of some hubs may decrease, while the largest radius in the network stays constant, and the chance of reaching feasible solutions increases. If no feasible solution can be obtained with this procedure, two additional procedures are performed, respectively: Initially, the radii of the remaining hubs are calculated, and for each hub, the nodes allocated to it are distributed to other hubs, as long as their radii do not increase.

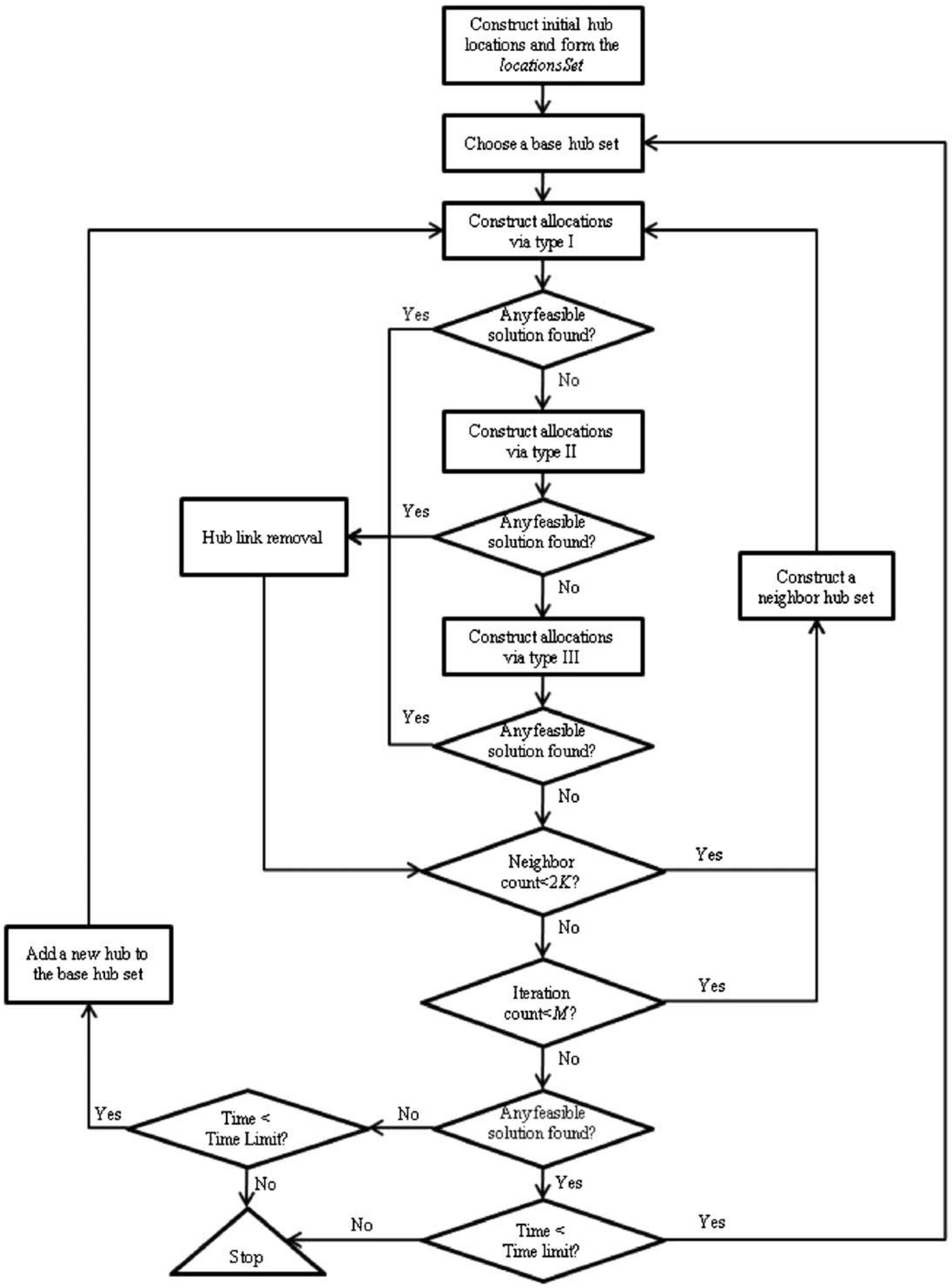
Construct initial hub locations and form the locationsSet
Choose a base hub set
Construct allocations via type I
Any feasible solution found?
Yes
No
Construct allocations via type II
Any feasible solution found?
Yes
No
Hub link removal
Construct allocations via type III
Any feasible solution found?
Yes
No
Construct a neighbor hub set
Neighbor count<2K?
Yes
No
Iteration count<M?
Yes
No
Add a new hub to the base hub set
Any feasible solution found?
No
Yes
Time < Time Limit?
Yes
No
Time < Time limit?
No
Yes
Stop

Fig. 3. The flowchart of the heuristic algorithm.

In the second procedure, all the nodes are first allocated to their nearest hubs, and the following procedure is repeated for each hub h. The node i which determines the radius value of hub h is discarded from the network. The feasibility of the network is checked and if the network is feasible without the node i, then node i is tried to be allocated to another hub without violating the feasibility. If the network is still infeasible without node i, then node j that now determines the radius value of hub h is discarded from the network together with node i. If discarding both of the nodes i and j does not yield a feasible solution, then another hub is selected for the same process. Otherwise, both of these nodes are tried to be allocated to different hubs without violating the feasibility. If neither of them can be allocated to new hubs, without violating the feasibility, then, another hub is chosen for the same process. If no feasible solution is found at the end of these processes in the Type I allocation strategy, the algorithm continues with Type II allocation strategy.

(ii) *Type* II *allocation*: In this strategy, first, a value named the potential radius is calculated for each hub. The potential radius is the maximum possible radius value for a hub node that will not exceed the given service time bound, T. In the beginning, without any allocations, since a complete hub network is constructed, the potential radius value of any hub is [$T-\alpha^*$ (the maximum travel time from that hub to another hub)]. Each non-hub node is allocated, starting from the non-hub node with the smallest index, to a randomly chosen hub to which the travel time is no more than the calculated potential radius. When a non-hub node is allocated to a hub, the potential radii of all other hubs are updated accordingly. If, at some point there is no feasible allocation for a non-hub node, we discontinue this strategy.

(iii) *Type* III *allocation*: In this strategy, all non-hub nodes are first allocated to only one hub, say h_1, in the hub set. If this allocation is not feasible, the non-hub node that determines the radius of h_1 is allocated to another hub, h_2. All non-hub nodes allocated to h_1 are selected one by one, in decreasing order of travel time to hub h_1, until the feasibility is reached or until all non-hub nodes are allocated to h_2. If feasibility is not achieved by any allocation from h_1 to h_2, allocations from h_1 to $h_3, h_4, \ldots$, from h_2 to $h_3, h_4, \ldots$ and all other combinations are checked. Note that the Type III allocation strategy restricts all allocations to, at most, two hubs.

The preliminary experimentations we performed on the individual allocation strategies showed that the least time-consuming strategy is Type I allocation, while the most time-consuming one is Type II allocation. However, Types II and III allocations may produce feasible solutions when no feasible solution can be obtained with Type I allocation. Therefore, in order to obtain good solutions in reasonable amounts of time, we primarily perform Type I allocation for each hub set (neighborhood). If no feasible solution is obtained by this strategy, Type II allocation is applied. The Type III allocation strategy is called for, only if feasibility is not achieved with the first two strategies.

Neighborhood: We define neighborhoods of solutions over the hub sets. A neighbor of a hub set H_i is another hub set H_j obtained by exchanging the role of exactly one of the hubs of H_i with a non-hub node. At each iteration of the algorithm, a specified number (*NeighIteration*) of random neighbors of the related hub set are generated as candidates for the next move.

Feasibility check: The feasibility of a constructed solution is checked as follows: initially a configuration matrix corresponding to the hub network of the solution is constructed. In this matrix, the indices corresponding to the links opened between the hubs have their time value multiplied by the discount factor, and the other indices have an infinite value. Then, the radius values of the hubs are calculated. Two conditions must be satisfied for a feasible solution: (i) traversing any radius twice should take no more than the time bound and (ii) for each hub pair h_i and h_j, the summation of $r(h_i)$, $r(h_j)$, and the shortest path between h_i and h_j times α should be no greater than the time bound T, where $r(h_i)$ is the radius value of hub h_i. The shortest path between h_i and h_i is calculated by using the configuration matrix as the distance matrix in Dijsktra's algorithm.

Tabu search iterations: The search starts with an initial hub set. At any iteration, the algorithm moves to a neighbor hub set with the best feasible solution. If no feasible solution is found within *NeighIteration* neighbors of a hub set, another subset of *NeighIteration* neighbors is generated and searched. If still no feasible solution is found within these neighbors, a random infeasible neighbor is chosen for the next hub set. In order to prevent cycling, the same node exchanges are avoided for a certain number of iterations, which is called tabu tenure in the literature. If no feasible solution is found within the specified number of tabu iterations, another hub is randomly added to the hub set, and the same steps are followed. The algorithm continues to randomly select base hub sets and traverse their neighbors until a specified time limit is reached.

We present a flow chart of the algorithm in Fig. 3.

4. Computational results

We first tested the performance of our model and heuristic on the CAB data set introduced by O'Kelly [2]. No real time data is provided for the CAB data set, thus, similar to other hub covering studies in the literature, we took $t_{ij} = d_{ij}$, where d_{ij} is the distance between nodes i and j. We took the fixed costs of opening hubs $fh_k = 100$ and fixed for all nodes [26].

Since we are building an incomplete hub network, we need fixed costs for hub links as well. For many applications of the hub location problem, this fixed cost value is dependent on both the travel distance and the flow between the nodes. Including flow in the fixed cost value also accounts for operational costs. In fact, this fixed cost value tends to be directly proportional to distance and inversely proportional to flow. In order to reflect this fact and to introduce a more realistic data set to the literature, we calculated and scaled the relative fixed costs of establishing hub links between the nodes of the network as follows:

$$f_{ij} = \frac{\mathrm{d}_{ij}/\mathrm{fl}_{ij}}{\max_{i,j}\{\mathrm{d}_{ij}/\mathrm{fl}_{ij}\}} \times 100 \quad \text{for all } i,j \neq i$$

where d_{ij} is the distance between nodes i and j, and fl_{ij} is the flow between nodes i and j.

For the CAB data set, we assumed that $H = N$ in all of the tested instances. For the rest of the parameters, we used the test bed shown in Table 1. In order to test the performance of our heuristic, we tested all possible α values reported in the literature for the hub covering problem, with the CAB data set.

The values in the last five columns in Table 1 present the time bounds, T. We tested both the tightest possible bounds for the given n and α values reported in [10] and the average values within these tightest bounds.

We solved our integer programming model by using CPLEX 10.1 on a personal computer with a 2.00 GHz Intel Core 2 Duo processor and 2 GB of RAM. We solved our model for all the CAB instances listed in Table 1. While solving the model, we limited the CPU time to two hours on CPLEX. In order to test the performance of our heuristic algorithm we applied it to the same CAB instances.

The size of the tabu list we used in our computations with the CAB data set was 5, the number of tabu iterations was 500, the number of neighborhoods to be detected at each iteration (*NeighIteration*) was 50 and the time limit was 100 seconds for 10 node instances and 600 seconds for 15 and 20 node instances. Tables 2 and 3 report and compare the results obtained with CPLEX and our heuristic algorithm with 10 nodes and 15 nodes, respectively.

Table 1
Test bed for the CAB data set.

n	α	T				
10	0.2	1425	1271.5	1118	975	832
	0.4	1627	1406	1185	1077.5	970
	0.6	1758	1572.5	1387	1267.5	1148
	0.8	1758	1673.5	1589	1523	1457
	1	1839	1815	1791	1778.5	1766
15	0.2	2004	1877	1750	1546	1342
	0.4	2162	1961	1760	1598	1436
	0.6	2214	2029	1844	1800	1756
	0.8	2424	2294.5	2165	2122.5	2080
	1	2611	2605.5	2600	2600	2600
20	0.2	1892	1720.5	1549	1452.5	1356
	0.4	2162	1961	1760	1616.5	1473
	0.6	2278	2137	1996	1915.5	1835
	0.8	2508	2386	2264	2209	2154
	1	2611	2605.5	2600	2600	2600

Table 2
Computational comparison of the IP model and the heuristic algorithm with $n = 10$.

Test bed			CPLEX		Heuristic				
n	α	T	Obj.	CPU time (s)	Obj.	CPU time (s)	Bcpu (s)	Gap (%)	Number of hub links
10	0.2	1425	206.374	1.892	206.374	100	0.141	0	1
	0.2	1271.5	306.631	2.422	306.631	100	0.752	0	2
	0.2	1118	308.195	1.070	308.195	100	0.327	0	2
	0.2	975	413.927	1.827	413.927	100	1.805	0	3
	0.2	832	423.804	2.380	423.804	100	0.248	0	4
	0.4	1627	206.374	3.725	206.374	100	0.147	0	1
	0.4	1406	306.631	2.576	306.631	100	2.152	0	2
	0.4	1185	317.359	1.746	317.359	100	0.189	0	2
	0.4	1077.5	413.927	1.836	413.927	100	3.431	0	3
	0.4	970	435.865	2.637	435.865	100	1.961	0	4
	0.6	1758	221.938	5.331	221.938	100	0.508	0	1
	0.6	1572.5	308.195	4.151	308.195	100	2.857	0	2
	0.6	1387	319.180	4.800	319.180	100	1.916	0	3
	0.6	1267.5	435.865	15.409	435.865	100	6.345	0	4
	0.6	1148	444.084	9.061	444.084	100	4.534	0	5
	0.8	1758	221.938	2.838	221.938	100	0.147	0	1
	0.8	1673.5	313.089	9.677	313.089	100	2.102	0	3
	0.8	1589	319.180	8.314	319.180	100	1.977	0	3
	0.8	1523	413.037	14.646	413.037	100	5.741	0	4
	0.8	1457	453.790	19.630	453.790	100	5.586	0	6
	1	1839	201.867	1.605	201.867	100	0.649	0	1
	1	1815	306.828	6.466	306.828	100	1.929	0	3
	1	1791	319.180	5.360	319.180	100	2.046	0	3
	1	1778.5	413.037	11.694	413.037	100	5.987	0	5
	1	1766	422.742	11.857	422.742	100	5.639	0	6
Average				5.879		100	2.365	0	
Maximum				19.630		100	6.345	0	

Table 3
Computational comparison of the IP model and the heuristic algorithm with $n = 15$.

Test bed			CPLEX		Heuristic				
n	α	T	Obj.	CPU time (s)	Obj.	CPU time (s)	Bcpu (s)	Gap (%)	Number of hub links
15	0.2	2004	221.938	46.122	221.938	600	0.251	0	1
	0.2	1877	301.698	29.151	301.698	600	0.736	0	2
	0.2	1750	307.941	48.007	307.941	600	4.264	0	2
	0.2	1546	405.821	144.628	406.225	600	3.921	0.10	3
	0.2	1342	413.026	108.052	413.026	600	57.378	0	3
	0.4	2162	210.057	43.626	210.057	600	0.176	0	1
	0.4	1961	301.265	26.515	301.265	600	5.263	0	2
	0.4	1760	313.450	55.393	313.450	600	2.402	0	2
	0.4	1598	408.281	202.490	408.281	600	3.185	0	3
	0.4	1436	424.826	524.280	424.826	600	1.426	0	3
	0.6	2214	210.057	39.990	210.057	600	0.179	0	1
	0.6	2029	301.698	38.293	301.698	600	3.299	0	2
	0.6	1844	323.578	728.059	323.578	600	1.079	0	3
	0.6	1800	417.946	1148.998	417.946	600	6.618	0	4
	0.6	1756	419.459	1118.715	423.162	600	6.433	0.88	5
	0.8	2424	209.334	168.332	209.334	600	0.191	0	1
	0.8	2294.5	311.362	223.915	311.362	600	3.290	0	3
	0.8	2165	332.650	690.407	332.650	600	3.085	0	3
	0.8	2122.5	424.225	2943.998	424.225	600	10.193	0	3
	0.8	2080	427.929	1904.888	427.929	600	20.465	0	4
	1	2611	202.814	49.559	202.814	600	0.184	0	1
	1	2605.5	304.110	607.270	304.110	600	3.982	0	3
	1	2600	304.110	1220.41	304.110	600	3.729	0	3
Average				526.569		600	6.162	0.04	
Maximum				2943.998		600	57.378	0.88	

In Tables 2 and 3, the columns under "CPLEX" present the optimum objective function value and the CPU time requirement in seconds, as reported by CPLEX. The columns under "Heuristic" report the objective function value, the CPU time requirement in seconds, and the gap of the heuristic. The column labeled Bcpu reports the CPU time when the best solution is obtained by the heuristic algorithm. The last column reports the number of hub links opened in the best solution found by the heuristic. We also listed the average and maximum CPU time requirements in seconds, for both CPLEX and our heuristic, in the last two rows of Tables 2 and 3.

Table 4
Computational comparison of the IP model and the heuristic algorithm with $n = 20$.

Test bed			CPLEX				Heuristic				
n	α	T	Obj.	CPU time (s)	Gap (%)	Lower bound	Obj.	CPU time (s)	Bcpu (s)	Gap (%)	Number of hub links
20	0.2	1892	236.454	532.03	0	236.454	236.454	600	0.213	0	1
	0.2	1720.5	338.609	515.991	0	338.609	338.609	600	5.580	0	2
	0.2	1549	365.989	3196.707	0	365.989	365.989	600	0.405	0	3
	0.2	1452.5	405.031	4281.201	0	405.031	408.281	600	47.784	0.80	3
	0.2	1356	513.635	7200	21.42	403.624	415.621	600	131.648	2.89*	3
	0.4	2162	247.777	2373.811	0	247.777	247.777	600	0.227	0	1
	0.4	1961	303.825	917.251	0	303.825	303.825	600	13.463	0	2
	0.4	1760	355.593	2494.244	0	355.593	355.593	600	2.516	0	2
	0.4	1616.5	619.423	7200	44.85	341.644	416.408	600	9.529	17.95*	4
	0.4	1473	858.353	7200	53.97	395.067	521.085	600	494.676	24.18*	6
	0.6	2278	252.748	4552.373	0	252.748	252.748	600	0.216	0	1
	0.6	2137	340.369	7200	22.40	264.115	340.369	600	7.208	22.40*	3
	0.6	1996	497.862	7200	50.60	245.938	355.593	600	6.224	30.84*	2
	0.6	1915.5	469.9047	7200	36.62	297.811	427.223	600	28.833	30.29*	2
	0.6	1835	614.8686	7200	51.21	300.000	427.946	600	27.645	29.90*	4
	0.8	2508	202.8139	2867.445	0	202.814	202.814	600	0.452	0	1
	0.8	2386	306.5502	7200	34.51	200.753	302.896	600	9.750	33.72*	2
	0.8	2264	560.239	7200	63.61	203.844	361.692	600	5.005	43.64*	3
	0.8	2209	N/A	7200	N/A	200.000	406.064	600	13.579	50.75*	5
	0.8	2154	N/A	7200	N/A	200.000	415.162	600	62.126	51.83*	4
	1	2611	202.814	6714.423	0	202.814	202.814	600	0.212	0	1
	1	2605.5	N/A	7200	N/A	200.000	304.110	600	5.266	34.23*	3
	1	2600	N/A	7200	N/A	200.000	304.110	600	5.660	34.23*	3
Average				5306.325	19.96			600	38.851	17.72	
Maximum				7200	63.61			600	494.676	51.83*	

Note from Table 2 that CPLEX solved all the instances with 10 nodes optimally in an average of a little less than 6 seconds of CPU time requirement. The maximum CPU time requirement by CPLEX for this network was less than 20 seconds. Our heuristic was able to solve all 10 node instances optimally. Observe from Table 2 that on the average the heuristic is able to find the optimal solutions in less than 3 seconds.

When the number of nodes becomes 15, the average CPU time requirement of CPLEX goes up to 9 minutes, with a maximum value of about 49 minutes. It is apparent from Table 3 that, even with a small number of nodes, the model is hard to solve to optimality. Our heuristic was able to obtain the optimal solutions at 21 out of the 23 instances with 15 nodes. At the other instances, in which our heuristic was not able to obtain the optimal solutions, the average gap of the heuristic was 0.49%.

We also tested the 20 node instances from the CAB data set. The results are reported in Table 4. CPLEX was not able to obtain optimum solutions in 2 hours of CPU time requirement in 13 out of the 23 instances with 20 nodes. In some of these instances, CPLEX could not even find an initial feasible solution. For Table 4, we also show the lower bound, the value of the best integer solution found, and its gap reported by CPLEX. In the instances when CPLEX could not find the optimum solution, we calculated the gap of our heuristic from this lower bound. These estimated gaps, reported with an asterisk (*) in Table 4, are naturally expected to be much higher than the actual optimality gaps of the heuristic.

Note from Table 4 that CPLEX was able to solve 10 of the 23 instances with 20 nodes optimally in 2 hours. Our heuristic was able to solve nine of these 10 instances optimally. In the other single instance, the gap of our heuristic from the optimal value was 0.80%. In the instances that we did not know the optimal solution, the average gap of the heuristic from the lower bound was 31.30%.

From Tables 2–4, observe that our heuristic was able to find optimal solutions for 55 of the 71 test instances. In three of the remaining instances in which CPLEX found the optimal solution, the average gap of our heuristic was 0.59%. In the remaining 13 instances we do not know if our heuristic was able to find the optimal solution or exactly how close it is to optimal.

Table 5
Computational results of the heuristic on the Turkish network.

T	CPU time (min)	Bcpu (min)	Number of hubs	Number of hub links
1880	10	1.065	2	1
1870	10	1.115	2	1
1860	30	16.554	3	3
1850	30	3.266	3	2
1840	30	8.899	3	2
1830	30	28.751	3	2
1820	30	1.825	3	3
1810	60	23.244	3	3
1800	60	4.753	4	6
1790	60	52.824	5	7
1780	90	84.073	5	10
1770	90	7.356	5	8
1760	90	91.828	7	15
Average CPU time (min)	47.692	25.043		

We have listed the number of hub links in the solutions in Table 2–4 to observe the incomplete hub network solutions. Excluding the cases for $p = 2$, where an incomplete hub network solution is not possible, we obtained incomplete hub networks at 39 of 71 instances. This result indicates that designing complete hub networks to provide service within a given service time bound is not cost effective, in many instances.

In order to observe the performance of our heuristic on larger networks, we tested the Turkish network. The Turkish network has 81 nodes, and we made all nodes candidate hub nodes. The time discount factor on the Turkish network with ground transportation was found to be 0.9 [27]. Thus, we took $\alpha = 0.9$ for all of the Turkish network instances. The fixed costs for opening hubs in the Turkish network were also obtained from [27].

For the Turkish network, the size of the tabu list is taken as 5, the number of tabu iterations as 200, and the number of neighborhoods to be detected at each iteration (*NeighIteration*) as 100. All other parameters of the test problems and the corresponding solutions are listed in Table 5.

From Table 5, observe that for tighter values of time bounds, we let the algorithm run longer since finding feasible solutions for the problem gets harder. Also note that tighter time bounds result in opening high number of hubs and hub links, but still the algorithm results in designing incomplete hub networks in most of the instances.

We were able to obtain solutions on the Turkish network in an average of 25 minutes. Even though we do not know the quality of our solutions on the Turkish network, this network is the largest data set that has been tested with incomplete hub network design problems. Obtaining optimal solutions even with complete hub networks is difficult on such a large network.

5. Conclusion

In this paper, we studied the single allocation hub covering problem over incomplete hub networks. We presented an $O(n^4)$ integer programming formulation of the problem. In order to solve realistically sized instances, we proposed a tabu-based heuristic algorithm. In contrast to other hub location problems, constructing feasible solutions for the hub covering problem, especially with tight time bounds, is challenging. Thus, we proposed and tested three different allocation strategies for constructing feasible solutions. To the best of the authors' knowledge, ours is the first heuristic in the literature that is proposed for the hub covering problem.

We tested our heuristic algorithm both on the CAB data set and the Turkish network. We compared the performance of our heuristic with CPLEX on the CAB data set and found that our heuristic obtained efficient solutions with less CPU time requirement than CPLEX. The computational times of our heuristic on the Turkish network were reasonable for such a large network, even with tight time bounds. The Turkish network, with 81 nodes, is the largest data set in the literature that is to be tested with incomplete hub network design problems.

References

[1] O'Kelly ME. The location of interacting hub facilities. Transportation Science 1986;20:92–105.
[2] O'Kelly ME. A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research 1987;32:393–404.
[3] Campbell JF. Hub location and the p-hub median problem. Operations Research 1996;44:923–35.
[4] Ernst AT, Krishnamoorthy M. Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science 1996;4:139–54.
[5] O'Kelly ME, Bryan D, Skorin-Kapov D, Skorin-Kapov J. Hub network design with single and multiple allocation: a computational study. Location Science 1996;4:125–38.
[6] Skorin-Kapov D, Skorin-Kapov J, O'Kelly M. Tight linear programming relaxations of uncapacitated p-hub median problems. European Journal of Operational Research 1996;94:582–93.
[7] Campbell JF, Ernst AT, Krishnamoorthy M. Hub location problems. In: Drezner Z, Hamacher HW, editors. Facility location: applications and theory. New York: Springer; 2002. p. 373–407.
[8] Alumur S, Kara BY. Network hub location problems: the state of the art. European Journal of Operational Research 2008;190:1–21.
[9] Campbell JF. Integer programming formulations of discrete hub location problems. European Journal of Operational Research 1994;72:387–405.
[10] Kara BY, Tansel B. The single assignment hub covering problem. Journal of the Operational Research Society 2003;54:59–64.
[11] Ernst AT, Jiang H, Krishnamoorthy M. Reformulations and computational results for uncapacitated single and multiple allocation hub covering problems. Unpublished Report, CSIRO Mathematical and Information Sciences, Australia; 2005.
[12] Nickel S, Schöbel A, Sonneborn T. Hub location problems in urban traffic networks. In: Niittymaki J, Pursula M, editors. Mathematics methods and optimization in transportation systems. Kluwer Academic Publishers; 2001. p. 1–12 [chapter 1].
[13] Campbell JF, Ernst AT, Krishnamoorthy M. Hub arc location problems: part I—introduction and results. Management Science 2005;51(10):1540–55.
[14] Campbell JF, Ernst AT, Krishnamoorthy M. Hub arc location problems: part II—formulations and optimal algorithms. Management Science 2005;51(10):1556–71.
[15] Alumur S, Kara BY. A hub covering network design problem for cargo applications in Turkey. Journal of the Operational Research Society 2008, doi:10.1057/jors.2008.92.
[16] Klincewicz JG. Avoiding local optima in the p-hub location problem using tabu search and GRASP. Annals of Operations Research 1992;40:283–302.
[17] Skorin-Kapov D, Skorin-Kapov J. On tabu search for the location of interacting hub facilities. European Journal of Operational Research 1994;73:502–9.
[18] Ernst AT, Krishnamoorthy M. Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science 1996;4(3):139–54.
[19] Pirkul H, Schilling DA. An efficient procedure for designing single allocation hub and spoke systems. Management Science 1998;44(12):235–42.
[20] Ebery J, Krishnamoorthy M, Ernst A, Boland N. The capacitated multiple allocation hub location problem: formulations and algorithms. European Journal of Operational Research 2000;120:614–31.
[21] Cunha CB, Silva MR. A genetic algorithm for the problem of configuring a hub-and-spoke network for a LTL trucking company in Brazil. European Journal of Operational Research 2007;179:747–58.
[22] Chen JF. A hybrid heuristic for the uncapacitated single allocation hub location problem. Omega 2007;35:211–20.
[23] Cánovas L, García S, Marín A. Solving the uncapacitated multiple allocation hub location problem by means of a dual-ascent technique. European Journal of Operational Research 2007;179:990–1007.
[24] Pamuk FS, Sepil C. A solution to the hub center problem via a single-relocation algorithm with tabu search. IIE Transactions 2001;33(5):399–411.
[25] Ernst A, Hamacher H, Jiang H, Krishnamoorthy M, Woeginger G. Uncapacitated single and multiple allocation p-hub center problems. Unpublished Report, CSIRO Mathematical and Information Sciences, Australia; 2002.
[26] O'Kelly ME. Hub facility location with fixed costs. Papers in Regional Science 1992;71(3):293–306.
[27] Tan PZ, Kara BY. A hub covering model for cargo delivery systems. Networks 2007;49:28–39.