Available online at www.sciencedirect.com

ScienceDirect

ELSEVIER

Computers & Operations Research 35 (2008) 3009–3019

computers & operations research

www.elsevier.com/locate/cor

Star p-hub median problem with modular arc capacities

Hande Yaman

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

Available online 7 February 2007

Abstract

We consider the hub location problem, where p hubs are chosen from a given set of nodes, each nonhub node is connected to exactly one hub and each hub is connected to a central hub. Links are installed on the arcs of the resulting network to route the traffic. The aim is to find the hub locations and the connections to minimize the link installation cost. We propose two formulations and a heuristic algorithm to solve this problem. The heuristic is based on Lagrangian relaxation and local search. We present computational results where formulations are compared and the quality of the heuristic solutions are tested.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: Hub location; p-Hub median; Star–star network; Lagrangian relaxation; Local search

1. Introduction

We consider a hub location problem that arises in the design of a star–star network. We are given a set of nodes which generate traffic. Among these nodes, p of them are chosen to be hubs. Each node is assigned to exactly one hub node and all the traffic originating and destinating at this node transits through this hub. Every hub node is assigned to itself. Direct links connect the node to its hub. There is a designated central hub and all the traffic that is consolidated at hub nodes goes through this central hub. All hub nodes are connected to the central hub by direct links. The resulting network is called a star–star network, as each network composed of a hub and the nodes assigned to it is a star and the network connecting the central hub to the remaining hubs is a star.

The traffic is routed as follows in this network. If two nodes i and m are assigned to the same hub, say j, the traffic from node i to node m goes from node i to hub j and from hub j to node m. If node i is assigned to hub j and node m is assigned to a different hub l, then the traffic from node i to node m first goes from node i to its hub j, from hub j to the central hub, from the central hub to hub l and finally from hub l to node m. So if node i is assigned to hub j, then all the traffic originating at node i travels to hub j and this is the exact amount of traffic to travel from node i to hub j. The amount of traffic that travels from hub j to node i is the amount of traffic with destination i. The amount of traffic that travels from a hub node j to the central hub depends on the assignment of nodes. This amount is equal to the traffic from the nodes assigned to hub j to nodes that are assigned to other hub nodes.

In Fig. 1, a star–star network is depicted. Node 0 is the central hub and nodes 1, 2, 3 and 4 are the other hubs. There is a unit traffic demand from every node to every other node in the network (including the central hub). As there are 15 nodes in the network, every node is the origin and destination of 14 units of traffic. Hence, the traffic on an arc connecting a nonhub node to its hub is 14 units. As no other node is assigned to hub 1, the traffic from this hub to the

E-mail address: hyaman@bilkent.edu.tr.

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.01.014

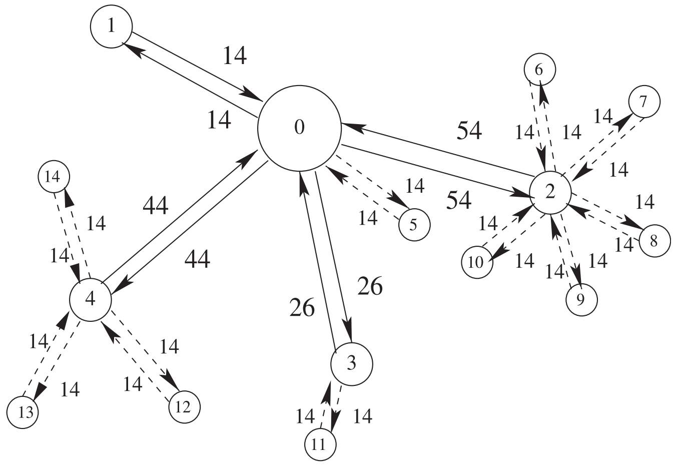

Fig. 1. The traffic on a star–star network.

central hub is the traffic originating at this node and so is 14 units. Six nodes (including the hub itself) are assigned to hub 2. So this hub sends $6 \times 9 = 54$ units of traffic to the central hub. The amount of traffic on each arc is given in the figure.

Links should be installed on the arcs of the network to route the traffic. Links can be installed only in integer amounts on each arc. We call the problem of choosing the locations of hubs and assigning the remaining nodes to hubs in order to minimize the cost of installing links as the *star p-hub median problem with modular arc capacities* (*SM*).

The problem *SM* has applications in telecommunications and transportation [1]. Our motivation to study this problem comes from the cargo delivery sector. Hub location problems have long been studied in the context of cargo delivery. But often in the literature, the cargo delivery network has been modeled as a complete-star network, i.e., it is assumed that the network connecting the hubs is complete. However, a recent study of the cargo delivery companies in Turkey has shown that this is not always the case [2]. One of the largest companies in Turkey has a star–star network where the central hub is located in Ankara. We consider the problem of designing a network for such a company where hub locations and assignments should be determined. Here the links to be installed on the arcs correspond to the vehicles which travel between cities and so the design cost is the total transportation cost.

If the links are assumed to be uncapacitated, then the problem becomes a fixed charge problem. In this case, *SM* can be solved as a facility location problem where the cost of locating a facility at a node includes the cost of installing a link to connect this facility to the central hub (see e.g., [3,4]). Hence the *p*-median problem is a special case of *SM*. As this facility location problem is NP-hard, *SM* is also NP-hard. For reviews on facility location problems, we refer the reader to e.g., Cornuéjols et al. [5,6], Krarup and Pruzan [7], Labbé et al. [8] and Sridharan [9].

Gavish [10] formulates the problem of designing a network where traffic requirements are only from and to a central hub. Nodes are connected to the hubs via multidrop links and hubs are connected to a central unit through a star network. The objective function involves the cost of establishing the links and installing the hubs. There are different types of links with different costs and capacities.

Chardaire et al. [11] present two integer programming formulations and a simulated annealing algorithm for the design of a network with two levels of hubs, i.e., each terminal is connected to a first level hub which is connected to a second level hub which is connected to a central unit. Here, traffic flows are not considered and fixed costs of connections and of installing facilities are minimized.

Helme and Magnanti [12] study the problem of designing a star–star satellite communication network. Each hub has a local switch and an earth station. Nodes assigned to the same hub use the local switch and nodes assigned to different hubs use the earth station and the central hub to communicate. The aim is to choose the location of hubs and assign nodes to hubs to minimize the cost of installing hubs, connecting nodes to hubs and using the capacities of the earth stations and the local switches. The authors propose a quadratic formulation and a linearization and report computational results with a branch and bound algorithm and greedy heuristics.

Labbé and Yaman [1] study a closely related problem to *SM* where there is a cost for routing the traffic in the network rather than a cost for installing links. In their problem, the number of hubs is not fixed and the cost of locating hubs is included in the total cost. The authors present formulations, polyhedral results and a Lagrangian relaxation heuristic. The heuristic performs very well and proves optimality for most of the instances. This motivated the heuristic in our paper.

In the classical setting, hub location problems assume that hubs are connected by a complete network. See Campbell et al. [13] for a recent survey on hub location problems. O'Kelly [14], Campbell [15], Skorin-Kapov et al. [16], Ernst and Krishnamoorthy [17], Ebery [18] and Labbé et al. [19] present formulations for the hub location problems. Sohn and Park [20] formulate the allocation problem in the case of two hubs as a linear programming (LP) problem. Ernst and Krishnamoorthy [17] present a branch and bound algorithm and an exact method based on shortest paths for the case where the number of hubs is fixed [21]. A Lagrangian relaxation heuristic is given in Pirkul and Schilling [22]. Hamacher et al. [23], Labbé and Yaman [24] and Labbé et al. [19] study the polyhedral properties of hub location problems.

Hub location problems with modular arc capacities are studied by Yaman [25] and Yaman and Carello [26]. In the first paper, polyhedral results are presented for the case where hubs are uncapacitated. In the second paper, the authors present a branch and cut algorithm for a version of the problem where hubs have capacities limiting the amount of traffic transiting through them. In both papers, the hubs are assumed to be connected by a complete network and the number of hubs is not fixed.

To the best of our knowledge, there is no previous study on *SM*. In this paper, we present two formulations for *SM*, compare their sizes and strengths. We present the outcomes of computational experiments to show the limits of these formulations. Then we present a heuristic based on Lagrangian relaxation and local search. We perform a computational study using data from the Turkish network to see the quality of solutions given by the heuristic.

The paper is organized as follows. In Section 2, we give the notation and two formulations. We compare the strength of the LP relaxations and the sizes of the formulations. We present a Lagrangian relaxation in Section 3 and compare the bound of the Lagrangian dual with those of the LP relaxations of the formulations. The heuristic algorithm is outlined in Section 4 and computational results are presented in Section 5.

2. Notation and formulations

Let I be the set of nodes and 0 be the central hub. Let t_{im} denote the amount of traffic to be routed from node $i \in I$ to node $m \in I$ and let d_{im} be the distance from node $i \in I$ to node $m \in I$. As the traffic from a node to itself does not travel on the arcs of the network, we assume that $t_{ii} = 0$ for all $i \in I$.

The traffic between a nonhub node and a hub node is carried on links of type 1 and the traffic between a hub node and the central hub is carried on links of type 2. We denote by Q_1 and Q_2 the capacity and by c_1 and c_2 the installation cost per unit distance of links of type 1 and type 2, respectively. The classical discount factor α in hub location problems can be incorporated into the model by taking $c_2 = \alpha c_1$. The cost of assigning node i to hub j, that we denote by C_{ij}, is equal to $c_1 d_{ij} \lceil \sum_{m \in I\setminus\{i\}} t_{im}/Q_1 \rceil + c_1 d_{ji} \lceil \sum_{m \in I\setminus\{i\}} t_{mi}/Q_1 \rceil$. If node i becomes a hub, then it is assigned to itself at no cost, i.e., $C_{ii} = 0$ for all $i \in I$.

In Fig. 2, we see the example network given in Fig. 1. Suppose that the capacity of a unit link connecting a hub to the central hub is 15 and the capacity of a unit link connecting a nonhub node to a hub is 10. The amount of capacity to be installed on each arc of the network is given in the figure. As the traffic on an arc connecting a nonhub node to its hub is 14, we need two links of type 1 on this arc. Whereas we need to install only one link of type 1 on the arc from hub 1 to the central hub as type 1 links have capacity 15. The number of links of type 1 to be installed on the arc from hub 2 to the central hub is four to enable the routing of 54 units of traffic.

To model *SM*, we define x_{ij} to be 1 if node $i \in I$ is assigned to hub node $j \in I$ and to be 0 otherwise. With this set of variables, we can obtain a nonlinear formulation of the problem:

$$\min \quad \sum_{i\in I}\sum_{j\in I} C_{ij}x_{ij} + c_2 \sum_{j\in I}\left(d_{j0}\left\lceil \sum_{i\in I}\sum_{m\in I\setminus\{i\}} \frac{t_{im}}{Q_2} x_{ij}(1-x_{mj})\right\rceil + d_{0j}\left\lceil \sum_{i\in I}\sum_{m\in I\setminus\{i\}} \frac{t_{mi}}{Q_2} x_{ij}(1-x_{mj})\right\rceil\right) \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in I} x_{ij} = 1 \quad \forall i \in I, \tag{2}$$

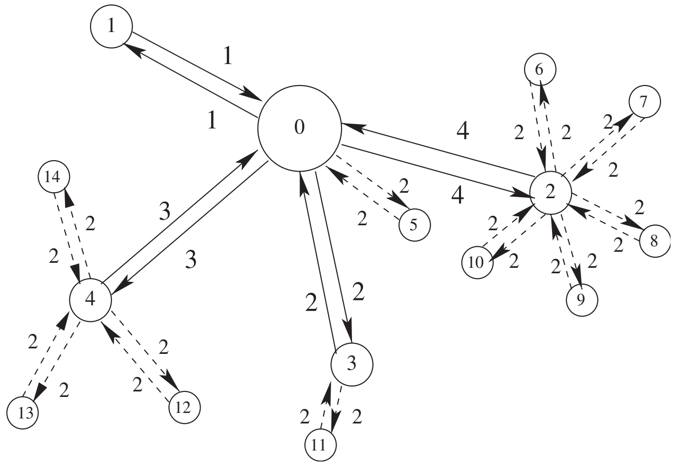

Fig. 2. The arc capacities.

$$\sum_{j\in I} x_{jj} = p, \tag{3}$$

$$x_{ij} \leqslant x_{jj} \quad \forall i \in I,\ \ j \in I, \tag{4}$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I,\ \ j \in I. \tag{5}$$

Constraints (2), (4) and (5) ensure that each node is assigned to exactly one hub node. Due to constraints (3), p nodes are assigned to themselves and so they are hubs.

The traffic from node i to node m uses the arc from hub node j to the central hub if node i is assigned to j and node m is not assigned to j, that is, if $x_{ij}(1-x_{mj})$ is one. So, for a given vector x, the traffic on the arc from a hub node j to the central node is equal to $\sum_{i\in I}\sum_{m\in I\setminus\{i\}} t_{im}x_{ij}(1-x_{mj})$. Consequently, the number of links to be installed on this arc is equal to $\lceil\sum_{i\in I}\sum_{m\in I\setminus\{i\}}(t_{im}/Q_2)x_{ij}(1-x_{mj})\rceil$. The objective function is the total cost of installing links.

It is easy to obtain a linear integer programming formulation using additional variables $u_{imj}=x_{ij}(1-x_{mj})$ for $i\in I$, $m\in I$ and $j\in I$, $z_j^{\text{out}}=\lceil\sum_{i\in I}\sum_{m\in I\setminus\{i\}}(t_{im}/Q_2)u_{imj}\rceil$ and $z_j^{\text{in}}=\lceil\sum_{i\in I}\sum_{m\in I\setminus\{i\}}(t_{mi}/Q_2)u_{imj}\rceil$. Here u_{imj} is a zero–one variable that takes value one if node i is assigned to hub j and node m is assigned to another hub and zero otherwise. The variables z_j^{out} and z_j^{in} are the numbers of links to be installed on the arc from hub j to the central hub and the arc from the central hub to hub j, respectively. We call the following formulation *SM*1:

$$\min \quad \sum_{i\in I}\sum_{j\in I} C_{ij}x_{ij} + c_2\sum_{j\in I}(d_{j0}z_j^{\text{out}} + d_{0j}z_j^{\text{in}}) \tag{6}$$

s.t. (2)–(5)

$$u_{imj} \geqslant x_{ij} - x_{mj} \quad \forall i \in I,\ \ m \in I,\ \ j \in I, \tag{7}$$

$$z_j^{\text{out}} \geqslant \sum_{i\in I}\sum_{m\in I\setminus\{i\}} \frac{t_{im}}{Q_2}u_{imj} \quad \forall j \in I, \tag{8}$$

$$z_j^{\text{in}} \geqslant \sum_{i\in I}\sum_{m\in I\setminus\{i\}} \frac{t_{mi}}{Q_2}u_{imj} \quad \forall j \in I, \tag{9}$$

$$u_{imj} \geqslant 0 \quad \forall i \in I,\ \ m \in I,\ \ j \in I, \tag{10}$$

$$z_j^{\text{out}}, z_j^{\text{in}} \in \mathbb{Z}_+ \quad \forall j \in I. \tag{11}$$

Due to constraints (7), u_{imj} is one if $x_{ij}-x_{mj}$ is one. Otherwise, u_{imj} can be zero or one. But, as we minimize the cost of installing links, an optimal solution satisfies $z_j^{\text{out}}=\lceil\sum_{i\in I}\sum_{m\in I\setminus\{i\}}(t_{im}/Q_2)(x_{ij}-x_{mj})^+\rceil$ and $z_j^{\text{in}}=\lceil\sum_{i\in I}\sum_{m\in I\setminus\{i\}}(t_{mi}/Q_2)(x_{ij}-x_{mj})^+\rceil$ for $j\in I$ whenever $d_{j0}>0$ and $d_{0j}>0$ and $x_{ij}(1-x_{mj})=(x_{ij}-x_{mj})^+$.

An alternative formulation, called *SM*2 can be obtained by defining variables $v_{ij}^{\text{out}} = \sum_{m\in I\setminus\{i\}} (t_{im}/Q_2)x_{ij}(1-x_{mj})$ and $v_{ij}^{\text{in}} = \sum_{m\in I\setminus\{i\}} (t_{mi}/Q_2)x_{ij}(1-x_{mj})$ for all $i \in I$ and $j \in I$ and by minimizing (6) subject to constraints (2)–(5), (11) and

$$v_{ij}^{\text{out}} \geqslant \sum_{m\in I\setminus\{i\}} \frac{t_{im}}{Q_2}(x_{ij}-x_{mj}) \quad \forall i \in I,\ \ j \in I, \tag{12}$$

$$v_{ij}^{\text{in}} \geqslant \sum_{m\in I\setminus\{i\}} \frac{t_{mi}}{Q_2}(x_{ij}-x_{mj}) \quad \forall i \in I,\ \ j \in I, \tag{13}$$

$$z_j^{\text{out}} \geqslant \sum_{i\in I} v_{ij}^{\text{out}} \quad \forall j \in I, \tag{14}$$

$$z_j^{\text{in}} \geqslant \sum_{i\in I} v_{ij}^{\text{in}} \quad \forall j \in I, \tag{15}$$

$$v_{ij}^{\text{out}}, v_{ij}^{\text{in}} \geqslant 0 \quad \forall i \in I,\ \ j \in I. \tag{16}$$

Here, v_{ij}^{out} is the amount of traffic originating at node *i* and that flows on the arc from hub node *j* to the central hub. Similarly, v_{ij}^{in} is the amount of traffic with node *i* as destination that flows on the arc from the central hub to hub node *j*. This formulation is inspired by the formulation proposed by Ebery [18] for the uncapacitated hub location problem where hubs are connected by a complete network.

In comparing formulations, we investigate the sizes and the strength of the LP relaxations. Formulation *SM*1 has $O(n^3)$ variables and *SM*2 has $O(n^2)$ variables where $n = |I|$. Notice that for given $\overline{x}$ which satisfies (2)–(4) and $0 \leqslant \overline{x}_{ij} \leqslant 1$ for $i \in I$ and $j \in I$, the vector $(\overline{x}, \overline{v}, \overline{z})$ where $\overline{v}_{ij}^{\text{out}} = (\sum_{m\in I\setminus\{i\}} (t_{im}/Q_2)(\overline{x}_{ij}-\overline{x}_{mj}))^+$ and $\overline{v}_{ij}^{\text{in}} = (\sum_{m\in I\setminus\{i\}} (t_{mi}/Q_2)(\overline{x}_{ij}-\overline{x}_{mj}))^+$ for all $i \in I$ and $j \in I$, $\overline{z}_j^{\text{out}} = \sum_{i\in I}(\sum_{m\in I\setminus\{i\}}(t_{im}/Q_2)(\overline{x}_{ij}-\overline{x}_{mj}))^+$ and $\overline{z}_j^{\text{in}} = \sum_{i\in I}(\sum_{m\in I\setminus\{i\}} (t_{mi}/Q_2)(\overline{x}_{ij}-\overline{x}_{mj}))^+$ for $j \in I$ is feasible for the LP relaxation of *SM*2.

In *SM*1, the constraints imply that $z_j^{\text{out}} \geqslant \sum_{i\in I}\sum_{m\in I\setminus\{i\}} (t_{im}/Q_2)(x_{ij}-x_{mj})^+$ and $z_j^{\text{in}} \geqslant \sum_{i\in I}\sum_{m\in I\setminus\{i\}}(t_{mi}/Q_2)(x_{ij}-x_{mj})^+$ for all $j \in I$. If there exists $j \in I$ such that

$$\sum_{i\in I}\sum_{m\in I\setminus\{i\}} \frac{t_{im}}{Q_2}(\overline{x}_{ij}-\overline{x}_{mj})^+ > \sum_{i\in I}\left(\sum_{m\in I\setminus\{i\}} \frac{t_{im}}{Q_2}(\overline{x}_{ij}-\overline{x}_{mj})\right)^+$$

or

$$\sum_{i\in I}\sum_{m\in I\setminus\{i\}} \frac{t_{mi}}{Q_2}(\overline{x}_{ij}-\overline{x}_{mj})^+ > \sum_{i\in I}\left(\sum_{m\in I\setminus\{i\}} \frac{t_{mi}}{Q_2}(\overline{x}_{ij}-\overline{x}_{mj})\right)^+,$$

then there does not exist any vector *u* such that $(\overline{x}, u, \overline{z})$ is feasible for the LP relaxation of *SM*1. So the LP relaxation of *SM*1 is stronger than the one of *SM*2.

As a result, we can expect the LP relaxation of *SM*1 to give a better lower bound at the expense of a longer solution time. It is important to see whether the quality of the lower bound decreases the size of the branch and bound tree and the number of LPs solved so that the total solution time is less. In Section 5, we report the results of a computational study to see this trade off between the size and the strength of these two formulations.

3. Lagrangian relaxation

Another possibility to compute a lower bound is to use Lagrangian relaxation. We relax the assignment constraints (2) in a Lagrangian manner in our nonlinear formulation. Let λ_i be the Lagrange multiplier for constraint (2). The relaxed problem is

$$\begin{aligned} LR(\lambda) = & \sum_{i\in I}\lambda_i + \min \sum_{i\in I}\sum_{j\in I}(C_{ij}-\lambda_i)x_{ij} \\ & + c_2 \sum_{j\in I}\left(d_{j0}\left[\sum_{i\in I}\sum_{m\in I\setminus\{i\}}\frac{t_{im}}{Q_2}x_{ij}(1-x_{mj})\right] + d_{0j}\left[\sum_{i\in I}\sum_{m\in I\setminus\{i\}}\frac{t_{mi}}{Q_2}x_{ij}(1-x_{mj})\right]\right) \\ \text{s.t.}\quad & (3)\text{–}(5) \end{aligned} \tag{17}$$

and can be solved by decomposing the problem for every node $j \in I$ as follows. Let

$$\begin{aligned} LR_j(\lambda) = & -\lambda_j + \min \sum_{i\in I\setminus\{j\}}(C_{ij}-\lambda_i)x_{ij} \\ & + c_2 d_{j0}\left[\sum_{m\in I\setminus\{j\}}\frac{t_{jm}}{Q_2}(1-x_{mj}) + \sum_{i\in I\setminus\{j\}}\sum_{m\in I\setminus\{i,j\}}\frac{t_{im}}{Q_2}x_{ij}(1-x_{mj})\right] \\ & + c_2 d_{0j}\left[\sum_{m\in I\setminus\{j\}}\frac{t_{mj}}{Q_2}(1-x_{mj}) + \sum_{i\in I\setminus\{j\}}\sum_{m\in I\setminus\{i,j\}}\frac{t_{mi}}{Q_2}x_{ij}(1-x_{mj})\right] \\ \text{s.t.}\quad & x_{ij}\in\{0,1\}\quad \forall i\in I\setminus\{j\}. \end{aligned} \tag{18}$$

Let π be an order on set I such that $LR_{\pi(1)}(\lambda)\leqslant LR_{\pi(2)}(\lambda)\leqslant\cdots\leqslant LR_{\pi(n)}(\lambda)$. Then $LR(\lambda)=\sum_{i\in I}\lambda_i+\sum_{l=1}^{p}LR_{\pi(l)}(\lambda)$ (we open the p hubs with smallest $LR_j(\lambda)$ values). Let $LD=\max_\lambda LR(\lambda)$.

We do not know any efficient way of computing $LR_j(\lambda)$. So we compute lower bounds on it as follows. For $j \in I$, define

$$\begin{aligned} LR^r_j(\lambda) = & -\lambda_j + \min \sum_{i\in I\setminus\{j\}}(C_{ij}-\lambda_i)x_{ij} \\ & + c_2 d_{j0}\sum_{m\in I\setminus\{j\}}\frac{t_{jm}}{Q_2}(1-x_{mj}) + c_2 d_{j0}\sum_{i\in I\setminus\{j\}}\sum_{m\in I\setminus\{i,j\}}\frac{t_{im}}{Q_2}x_{ij}(1-x_{mj}) \\ & + c_2 d_{0j}\sum_{m\in I\setminus\{j\}}\frac{t_{mj}}{Q_2}(1-x_{mj}) + c_2 d_{0j}\sum_{i\in I\setminus\{j\}}\sum_{m\in I\setminus\{i,j\}}\frac{t_{mi}}{Q_2}x_{ij}(1-x_{mj}) \\ \text{s.t.}\quad & x_{ij}\in\{0,1\}\quad \forall i\in I\setminus\{j\}. \end{aligned} \tag{19}$$

Let σ be an order on set I such that $LR^r_{\sigma(1)}(\lambda)\leqslant LR^r_{\sigma(2)}(\lambda)\leqslant\cdots\leqslant LR^r_{\sigma(n)}(\lambda)$, $LR^r(\lambda)=\sum_{i\in I}\lambda_i+\sum_{l=1}^{p}LR^r_{\sigma(l)}(\lambda)$ and $LD^r=\max_\lambda LR^r(\lambda)$. As $LR^r(\lambda)\leqslant LR(\lambda)$ for any λ, the dual bound $LD^r\leqslant LD$. The important question is how does LD^r compare to the bounds of the LP relaxations of $SM1$ and $SM2$. The proof of the below statement is similar to the proof of Proposition 2 in [1] and so is omitted.

Proposition 1. *LD^r is equal to the bound of the LP relaxation of $SM1$.*

For $j \in I$, we can rewrite (19) as

$$\begin{aligned} LR^r_j(\lambda) = & \gamma_j + \min \sum_{i\in I\setminus\{j\}}\alpha_{ij}x_{ij} + \sum_{i\in I\setminus\{j\}}\sum_{m\in I\setminus\{i,j\}}\beta_{imj}x_{ij}(1-x_{mj}) \\ \text{s.t.}\quad & x_{ij}\in\{0,1\}\quad \forall i\in I\setminus\{j\}, \end{aligned}$$

where $\gamma_j=-\lambda_j+\sum_{m\in I\setminus\{j\}}(c_2d_{j0}t_{jm}+c_2d_{0j}t_{mj})/Q_2$, $\alpha_{ij}=(C_{ij}-\lambda_i-(c_2d_{j0}t_{ji}+c_2d_{0j}t_{ij})/Q_2)$ for all $i\in I\setminus\{j\}$ and $\beta_{imj}=(c_2d_{j0}t_{im}+c_2d_{0j}t_{mi})/Q_2$ for all $i\in I\setminus\{j\}$ and $m\in I\setminus\{i,j\}$. Since $\beta_{imj}\geqslant 0$ for all $i\in I\setminus\{j\}$ and $m\in I\setminus\{i,j\}$, the value $LR^r_j(\lambda)$ can be computed by solving a mincut problem (see [27]). We construct the graph $G_j = (I \cup \{o, d\}, A_j)$ where o and d are dummy origin and destination nodes, respectively and $A_j = \{(o, i), (i, d) : i \in I\backslash\{j\}\} \cup \{(i, m) : i \in I\backslash\{j\}, m \in I\backslash\{i, j\}\}$. Let ω_{im} denote the capacity of arc $(i, m) \in A_j$. The capacities are as follows:

$$\omega_{im} = \beta_{imj} \quad \text{for } i \in I\backslash\{j\}, \quad m \in I\backslash\{i, j\},$$

$$\omega_{id} = (\alpha_{ij})^+ \quad \text{for } i \in I\backslash\{j\},$$

$$\omega_{oi} = (-\alpha_{ij})^+ \quad \text{for } i \in I\backslash\{j\}.$$

If S is a minimum cut with capacity $\omega(S)$, then $LR^r_j(\lambda) = \gamma + \omega(S) - \sum_{i\in I\backslash\{j\}}(-\alpha_{ij})^+$ (see [1] for a detailed explanation).

Hence $LR^r(\lambda)$ can be computed by solving n mincut problems. We compute LD^r using the subgradient method.

4. Heuristic algorithm

In this section, we present our heuristic algorithm. The algorithm has two basic parts. The first part is based on the Lagrangian relaxation of the previous section. We solve the Lagrangian dual problem with the subgradient algorithm and generate feasible solutions using the optimal solutions of the relaxations. The best solution of this part is input into the second part which is a local search algorithm. The heuristic is given in Algorithm 1.

Algorithm 1. Lagrangian heuristic and local search

```
$\sigma \leftarrow 2$, $\lambda \leftarrow 0$, $noimp \leftarrow 0$
$lb \leftarrow 0$ and $ub \leftarrow \infty$
while $\frac{ub-lb}{ub} * 100 > 10^{-4}$ and $\sigma > 10^{-4}$ do
    Compute $LR^r(\lambda)$ and let $x$ be the optimal solution
    $x^f \leftarrow Feasible(x)$
    if $Cost(x^f) < ub$ then
        $ub \leftarrow Cost(x^f)$
        $x^* \leftarrow x^f$
    if $LR^r(\lambda) > lb$ then
        $lb \leftarrow LR^r(\lambda)$
        $noimp \leftarrow 0$
    else
        increment $noimp$
    if $noimp > 15$ then
        $noimp \leftarrow 0$
        $\sigma \leftarrow \sigma/2$
    $s \leftarrow \sigma(ub - LR^r(\lambda))/\sum_{i\in I}(1 - \sum_{j\in I} x_{ij})^2$
    $\lambda_i \leftarrow \lambda_i - s(1 - \sum_{j\in I} x_{ij})$ for all $i \in I$
$x^* \leftarrow LocalAssignment(x^*)$
$ub \leftarrow Cost(x^*)$
$update \leftarrow 1$
while $update$ do
    $update \leftarrow 0$
    if $Cost(LocalHub(x^*)) < ub$ then
        $x^* \leftarrow LocalHub(x^*)$
        $ub \leftarrow Cost(x^*)$
        $update \leftarrow 1$
```

We denote by x^* the best solution, by lb the lower bound and by ub the upper bound. For a feasible assignment vector x, let $Cost(x)$ denote the corresponding link installation cost. The algorithm starts with Lagrange multipliers and the lower bound lb set to zero, the upper bound ub set to infinity and the parameter that multiplies the step size σ set to 2. While the percentage gap between the upper and lower bounds ($(ub - lb)/ub * 100$) and σ are both greater than

10^{-4}, we compute $LR^r(\lambda)$. If the optimal solution of the relaxation, say x, satisfies constraints (2), then let x^f be this solution. If x does not satisfy (2), then we construct a new solution x^f as follows. We keep the open hubs and assign the remaining nodes to these hubs. If node i is assigned to a single node j in x, then this assignment is kept. Otherwise, there are two cases. If i is assigned to several nodes, then we pick the cheapest one in terms of the assignment costs C_{ij}. If node i is not assigned at all, then it is assigned to hub j with the cheapest assignment cost C_{ij}. This procedure which checks whether x satisfies (2) and constructs solution x^f from x is called *Feasible*(x). If x^f improves upon the best solution, we update the upper bound and keep x^f as the best solution.

If the value of the relaxation, $LR^r(\lambda)$, is greater than the lower bound, then we update the lower bound. We keep track of the number of consecutive times we encounter a nonimproving $LR^r(\lambda)$ in *noimp*. If *noimp* exceeds 15, we divide σ by 2.

We update the Lagrange multipliers as $\lambda_i = \lambda_i - s(1 - \sum_{j\in I} x_{ij})$ for $i \in I$ where the step size s is computed as $s = \sigma(ub - LR^r(\lambda))/\sum_{i\in I}(1 - \sum_{j\in I} x_{ij})^2$.

When the Lagrangian dual is solved, the local search starts with the best solution found so far, x^*. The search heuristic first tries to improve the assignments of this solution. For each node which is not a hub itself, we evaluate all other solutions where this node is assigned to another hub and all other assignments are kept as they are. If there is an improving solution, we move to it. For a given solution x, this search procedure based on assignments is called *LocalAssignment*(x).

Then we consider a different neighborhood. We exchange a hub node, say j, with a nonhub node and assign the nodes that were previously assigned to node j to their closest hubs. If the resulting solution x' has a cost not superior to 1.1 times the cost of the best solution, then we check if it is possible to improve the assignments. Here we apply *LocalAssignment*(x'). If the final solution has a better cost than the best upper bound, then we update the best solution. For a given solution x, this procedure is called *LocalHub*(x). We continue this search as long as there is an update.

5. Computational study

In this section, we report the results of two computational experiments. The heuristic algorithm is implemented in C programming language. All computation is carried out on an AMD Opteron 252 processor (2.6 GHz) with 2 GB of RAM.

In the first experiment, we solve *SM* for varying values of n and p using the two formulations *SM*1 and *SM*2 and the heuristic algorithm. This experiment gives us an idea of the computational limits of the two formulations and the quality of the heuristic solutions. In the second experiment, we consider the whole Turkish network with its 81 cities. We solve *SM* using our heuristic, for $p = 1, \dots, 25$ to see the computational effort needed as well as the quality of the solutions.

In both experiments, we use data from the Turkish network. We consider the 81 cities in Turkey as the node set. To create a problem with n nodes for the first experiment, we take the first n cities. We take Ankara to be the central hub.

In our first experiment, we take $n = 20, 30, \dots, 70$ and 81 and $p = 5, 10, 15$. We solve the two formulations *SM*1 and *SM*2 using CPLEX 8.1.0 with default settings. We set a time limit of 1 h. In Table 1, we report the results of this experiment. For each problem defined by its number of nodes n and number of hubs p, and for formulation i, we report the best upper bound ub_i, the optimal value of the LP relaxation LP_i, the percentage duality gap dg_i computed using the best upper bound given by the two formulations, i.e., $(\min\{ub_1, ub_2\} - LP_i)/\min\{ub_1, ub_2\} * 100$, the number of nodes in the branch and bound tree $nodes_i$ and the cpu time cpu_i for $i = 1, 2$. If the solver stops with the time limit, then in the cpu time column, we report the remaining gap as given by the solver in parenthesis. If the solver cannot report a lower bound or an upper bound within the time limit, then the gap is infinite and we do not report it.

The results reported in Table 1 reveal that the LP relaxation of *SM*1 gives much better lower bounds compared to the LP relaxation of *SM*2 but is difficult to solve. For $n \geqslant 70$, the solver could not solve the LP relaxations of *SM*1 in 1 h. There is no such problem with *SM*2 but this formulation has a weak LP bound and the duality gaps are very high. The performance of this formulation gets worse with increasing p. Still, this formulation is preferable for medium and large size problems as it outputs a feasible solution within the time limit.

Next, we solve the same instances using our heuristic and report the results in Table 2. For each instance, we report the lower bound given by the Lagrangian dual lb, the objective function value of the best feasible solution before the local search starts ub_0, the number of iterations of the subgradient algorithm *iter*, the objective function value of the best feasible solution after local search ub, the percentage improvements compared to ub_0, $imp_0 = (ub_0 - ub)/ub_0 * 100$, the

Table 1
Comparison of the two formulations

n	p	$SM1$					$SM2$				
		ub_1	LP_1	dg_1	$nodes_1$	cpu_1	ub_2	LP_2	dg_2	$nodes_2$	cpu_2
20	5	25507.20	24500.05	3.95	82	8.34	25507.20	23653.80	7.27	595	1.59
30	5	57306.80	56415.60	1.56	3568	961.50	57306.80	52829.11	7.81	37 403	199.51
40	5	167590.00	166180.13	0.84	1503	1836.40	167590.00	159558.69	4.79	31 276	420.21
50	5	–	246506.80	0.65	86	–	248116.20	234421.83	5.52	133 404	(0.24%)
60	5	317314.40	315748.49	0.45	0	(0.49%)	317190.80	301977.07	4.80	41 021	(0.79%)
70	5	–	–	–	0	–	393931.40	373940.45	5.07	20 455	(2.17%)
81	5	–	–	–	0	–	438887.20	416512.15	5.10	8708	(2.76%)
20	10	24859.40	22140.97	10.94	62	11.99	24859.40	21057.20	15.29	7011	12.59
30	10	55089.20	52329.07	5.01	11 144	(1.13%)	55817.40	48935.08	11.17	676 045	(5.69%)
40	10	164306.60	159359.90	3.00	2987	(1.90%)	164294.60	150575.69	8.35	343 810	(3.29%)
50	10	278798.80	239063.07	2.31	789	(13.93%)	244725.60	222926.14	8.91	153 840	(4.27%)
60	10	–	305060.88	–	0	–	313348.80	286692.22	8.51	52 996	(5.25%)
70	10	–	–	–	0	–	388097.60	356395.45	8.17	32 734	(5.53%)
81	10	–	–	–	0	–	436204.00	396774.80	9.04	23 138	(7.05%)
20	15	25672.80	21263.90	17.17	4	3.83	25672.80	20266.60	21.06	134	0.60
30	15	55203.00	50893.36	7.81	8603	(1.78%)	55794.20	47142.05	14.60	446 408	(6.38%)
40	15	164868.80	156898.04	4.83	4278	(3.32%)	166648.60	146207.75	11.32	327 791	(7.38%)
50	15	482782.80	236462.27	4.64	486	(50.69%)	247962.40	218113.35	12.04	157 650	(7.99%)
60	15	–	301130.98	3.76	0	–	312891.20	280056.46	10.49	72 379	(7.45%)
70	15	–	–	–	0	–	387929.80	349141.79	10.00	44 321	(7.60%)
81	15	–	–		0	–	433425.40	388905.84	10.27	31 596	(8.36%)

Table 2
Results of the heuristic algorithm

n	p	lb	ub_0	$iter$	ub	imp_0%	imp%	cpu_s	cpu
20	5	24497.90	26112.80	106	25845.00	1.03	−1.32	0.01	0.38
30	5	56415.61	58200.80	399	58037.20	0.28	−1.27	0.04	13.37
40	5	166179.83	167952.00	197	167634.80	0.19	−0.03	0.64	23.88
50	5	246506.80	248177.20	271	248177.20	0.00	−0.02	0.10	107.85
60	5	315726.13	317314.41	289	317154.41	0.05	0.01	0.21	300.07
70	5	391607.78	392739.56	241	392524.78	0.05	0.36	0.37	564.91
81	5	437124.06	438513.00	308	438198.59	0.07	0.16	0.65	1579.88
20	10	22140.96	26293.20	803	24859.40	5.45	0.00	0.18	2.38
30	10	52324.96	56307.00	103	55567.80	1.31	−0.87	3.09	5.58
40	10	159348.06	165233.20	275	163076.39	1.31	0.74	12.99	38.59
50	10	239063.08	243599.00	283	242357.00	0.51	0.97	27.41	119.83
60	10	305047.88	310893.59	432	308927.22	0.63	1.41	65.44	387.21
70	10	381278.38	384855.41	400	383324.19	0.40	1.23	124.88	841.53
81	10	423919.06	428088.81	430	426245.41	0.43	2.28	271.49	1845.54
20	15	21263.90	26878.20	24	25672.80	4.48	0.00	0.18	0.23
30	15	50890.46	57937.60	141	55277.60	4.59	−0.14	6.44	9.18
40	15	156898.00	164668.78	1049	162087.59	1.57	1.69	25.37	115.77
50	15	236462.28	245160.42	381	242061.00	1.26	2.38	79.51	187.63
60	15	301107.59	308693.56	372	306285.16	0.78	2.11	197.71	440.40
70	15	377653.66	382756.81	444	380776.38	0.52	1.84	434.30	1159.39
81	15	419718.66	426167.63	567	423352.03	0.66	2.32	857.14	2677.62

percentage improvements compared to the best upper bound found using the two formulations, $imp = (\min\{ub_1, ub_2\} - ub)/ \min\{ub_1, ub_2\} * 100$, the cpu time of the search algorithm cpu_s and the total cpu time of the heuristic cpu, both in seconds.

Table 3
Results for 81 node instances

p	lb	ub_0	$iter$	ub	$imp_0\%$	$gap\%$	cpu_s	cpu
1	506443.91	506444.00	4	506444.00	0.00	0.00	0.54	36.41
2	472585.16	472902.00	37	472694.00	0.04	0.02	0.56	297.73
3	454629.63	455047.19	198	455006.81	0.01	0.08	0.60	1171.51
4	444254.50	445071.41	299	444863.41	0.05	0.14	0.62	1548.33
5	437124.06	438513.00	308	438198.59	0.07	0.25	0.65	1575.76
6	433329.41	436736.19	1684	436421.78	0.07	0.71	24.65	7938.60
7	429591.41	431647.00	1069	431428.59	0.05	0.43	25.32	4530.86
8	427414.56	429665.41	714	429076.59	0.14	0.39	74.92	2870.64
9	425439.97	429884.63	706	427555.00	0.54	0.50	404.97	3089.81
10	423919.06	428088.81	430	426245.41	0.43	0.55	271.67	1841.43
11	422578.50	427053.44	389	425480.63	0.37	0.69	535.44	1836.24
12	421709.41	426491.84	518	424919.03	0.37	0.76	626.48	2392.34
13	421040.28	424736.25	651	424118.22	0.15	0.73	523.09	2679.90
14	420373.59	425468.66	634	423243.03	0.52	0.68	1031.04	3062.05
15	419718.66	426167.63	567	423352.03	0.66	0.87	871.44	2694.69
16	419120.53	425516.22	1340	423293.66	0.52	1.00	953.96	5211.16
17	418516.91	426550.63	502	423129.47	0.80	1.10	1023.43	2562.23
18	417937.28	427183.41	616	422572.44	1.08	1.11	1067.76	2927.14
19	417399.59	426672.22	420	422484.66	0.98	1.22	1174.85	2453.51
20	416911.44	429006.38	314	422127.66	1.60	1.25	1665.58	2622.97
21	416451.53	426754.38	303	421684.84	1.19	1.26	1812.45	2713.26
22	416009.47	426333.78	387	421964.81	1.02	1.43	1778.84	2924.26
23	415571.06	429055.78	455	422282.38	1.58	1.61	1424.70	2714.35
24	415168.06	428965.66	1179	421331.78	1.78	1.48	1566.31	5021.95
25	414752.00	430291.16	1143	421737.41	1.99	1.68	4596.93	7916.26

We observe that the heuristic reports solutions worse than the solutions of the two formulations for six of the instances. These are instances with at most 50 nodes and the average percentage difference is -0.608%. For two instances, the heuristic reports the optimal solution like both of the formulations. For the remaining 13 instances, the heuristic solution is better than the best solution found using the formulations, the average percentage difference is 1.38%. Besides, the heuristic outperforms the formulations for larger size instances. The average improvement over the best solution obtained using the formulations is 0.66%, the minimum improvement is -1.32% and the maximum improvement is 2.38%. The average and maximum cpu times are 496.25 and 2677.62 s, respectively.

The local search algorithm improves the best solution by 1.22% on the average. The maximum improvement is 5.45%. The average cpu time for the search is 100.39 s which is about 20% of the total cpu time. We observe that the percentage improvement decreases as n increases. As the search time increases with growing n, one can turn off the local search mode of the algorithm or put a time limit for large size instances if the solution time is a big concern.

In Table 3, we report the results of our second experiment where we solve the 81 city instances for different values of p using the heuristic algorithm. In column $gap\%$, we report the percentage gap of our heuristic solutions, i.e., $gap = (ub - lb)/ub * 100$. Here, we observe that the heuristic solutions have a gap of 0.8% on the average with the maximum gap being 1.68%. The gap increases as the value of p increases. This is not surprising since the number of integer variables that have nonzero value increases with p.

The average solution time is 2985.34 cpu seconds and the average search time is 858.27 cpu seconds which is about 29% of the total cpu time. The local search improves the best solution by 0.64% on the average and the improvement increases as p increases. This can also be observed in Table 2.

We can summarize the findings of our computational study as follows. The LP relaxation of formulation $SM1$ has a better lower bound compared to that of formulation $SM2$ but it takes very long to compute. So it is only practical for small size instances. Formulation $SM2$ can be used to solve medium-sized problems if the user has some tolerance on optimality. The heuristic proposed outperforms the two formulations for medium and large size instances. As it also computes a lower bound, it reports a measure of quality for its solution. The results were very good for the Turkish network for different values of p.

References

[1] Labbé M, Yaman H. Solving the hub location problem in a star–star network. Networks, forthcoming.
[2] Elmastas S. Personal communication.
[3] Mirzaian A. Lagrangian relaxation for the star–star concentrator location problem: approximation algorithm and bounds. Networks 1985; 15:1–20.
[4] Pirkul H. Efficient algorithms for the capacitated concentrator location problem. Computers and Operations Research 1987;14:197–208.
[5] Cornuéjols G, Nemhauser GL, Wolsey LA. The uncapacitated facility location problem. In: Mirchandani PB, Francis RL, editors. Discrete location theory. New York: Wiley; 1990. p. 119–71.
[6] Cornuéjols G, Sridharan R, Thizy JM. A comparison of heuristics and relaxations for the capacitated plant location problem. European Journal of Operational Research 1991;50:280–97.
[7] Krarup J, Pruzan PM. The simple plant location problem: survey and synthesis. European Journal of Operational Research 1983;12:36–81.
[8] Labbé M, Peeters D, Thisse JF. Location on networks. In: Ball MO, Magnanti TL, Monma CL, Nemhauser GL, editors. Network routing, handbooks in operations research and management sciences, vol. 8. Amsterdam: North-Holland; 1995. p. 551–624.
[9] Sridharan R. The capacitated plant location problem. European Journal of Operational Research 1995;87:203–13.
[10] Gavish B. Topological design of centralized computer networks: formulations and algorithms. Networks 1982;12:355–77.
[11] Chardaire P, Lutton JL, Sutter A. Upper and lower bounds for the two-level simple plant location problem. Annals of Operations Research 1999;86:117–40.
[12] Helme MP, Magnanti TL. Designing satellite communication networks by zero–one quadratic programming. Networks 1989;19:427–50.
[13] Campbell JF, Ernst AT, Krishnamoorthy M. Hub location problems. In: Drezner Z, Hamacher HW, editors. Facility location: applications and theory. Berlin: Springer; 2002. p. 373–407.
[14] O'Kelly ME. A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research 1987;32: 393–404.
[15] Campbell JF. Integer programming formulations of discrete hub location problems. European Journal of Operational Research 1994;72: 387–405.
[16] Skorin-Kapov D, Skorin-Kapov J, O'Kelly M. Tight linear programming relaxations of uncapacitated p-hub median problem. European Journal of Operational Research 1996;94:582–93.
[17] Ernst AT, Krishnamoorthy M. Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science 1996;4: 139–54.
[18] Ebery J. Solving large single allocation p-hub problems with two or three hubs. European Journal of Operational Research 2001;128:447–58.
[19] Labbé M, Yaman H, Gourdin E. A branch and cut algorithm for hub location problems with single assignment. Mathematical Programming 2005;102:371–405.
[20] Sohn J, Park S. A linear program for the two hub location problem. European Journal of Operational Research 1997;100:617–22.
[21] Ernst AT, Krishnamoorthy M. An exact solution approach based on shortest paths for p-hub median problems. INFORMS Journal on Computing 1998;10:149–62.
[22] Pirkul H, Schilling D. An efficient procedure for designing single allocation hub and spoke systems. Management Science 1998;44:S235–42.
[23] Hamacher HW, Labbé M, Nickel S, Sonneborn T. Adapting polyhedral properties from facility to hub location problems. Discrete Applied Mathematics 2004;145:104–16.
[24] Labbé M, Yaman H. Projecting the flow variables for hub location problems. Networks 2004;44:84–93.
[25] Yaman H. Polyhedral analysis for the uncapacitated hub location problem with modular arc capacities. SIAM Journal on Discrete Mathematics 2005;19:501–22.
[26] Yaman H, Carello G. Solving the hub location problem with modular link capacities. Computers and Operations Research 2005;32:3227–45.
[27] Picard JC, Ratliff HD. Minimum cuts and related problems. Networks 1975;5:357–70.