Computers & Operations Research 36 (2009) 3143 – 3151

Contents lists available at ScienceDirect

Computers & Operations Research

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

A 2-phase algorithm for solving the single allocation *p*-hub center problem

T. Meyer[a,*], A.T. Ernst[b], M. Krishnamoorthy[b]

[a]Department of Mathematics, University of Kaiserslautern, Gottlieb-Daimler-Strasse, 67663 Kaiserslautern, Germany
[b]CSIRO Mathematical and Information Sciences, Private Bag 33, Clayton South MDC, Clayton VIC 3169, Australia

ARTICLE INFO

Available online 7 August 2008

Keywords:
Hub location
Branch and bound
Ant colony optimization

ABSTRACT

The single allocation *p*-hub center problem is an NP-hard location–allocation problem which consists of locating hub facilities in a network and allocating non-hub nodes to hub nodes such that the maximum distance/cost between origin–destination pairs is minimized. In this paper we present an exact 2-phase algorithm where in the first phase we compute a set of potential optimal hub combinations using a shortest path based branch and bound. This is followed by an allocation phase using a reduced sized formulation which returns the optimal solution. In order to get a good upper bound for the branch and bound we developed a heuristic for the single allocation *p*-hub center problem based on an ant colony optimization approach. Numerical results on benchmark instances show that the new solution approach is superior over traditional MIP-solver like CPLEX. As a result we are able to provide new optimal solutions for larger problems than those reported previously in literature. We are able to solve problems consisting of up to 400 nodes in reasonable time. To the best of our knowledge these are the largest problems solved in the literature to date.

1. Introduction

Hub location problems arise when it is desirable to transport commodities (e.g., goods or passengers) between origin–destination (o–d) pairs. In general, a direct transportation of these commodities cannot be realized due to the fact that establishing such a network is extremely costly. As an alternative, one uses a special logistic network with a so-called hub-and-spoke structure where the hubs act as collection, consolidation, transfer and distribution points. The advantage of using hubs is flow consolidation, which in turn allows one to achieve economies of scale on the flow between hubs. Hence, transferring flow between hubs is cheaper than moving commodities directly between non-hub nodes (spokes). Spoke nodes can be connected to one or more hubs, depending on whether the design constraints allow single or multiple allocation. It is usually assumed that the hubs are fully interconnected, while spoke nodes are not connected to each other. Therefore all commodities have to be routed via at least one hub (Fig. 1).

In general, hub location problems deal with two different tasks: *hub selection* where some nodes are selected to be hubs and *spoke allocation* where an assignment of spoke nodes to hubs is made.

The most common hub location problems are those with center and median objective. In the *hub median problem* the objective is to minimize the total transportation cost. It is applicable, for instance, in airline and telecommunication systems. This model of hub network design can sometimes lead to unsatisfactory results when worst-case o–d distances are excessively large. In order to avoid this drawback, *hub center* problems may be a better suited model. Here the main objective is to minimize the maximum distance or cost between o–d pairs. This objective is particularly important for the delivery of perishable or time sensitive items.

The hub location problem to be discussed in this paper is known as the *uncapacitated single allocation p-hub center problem* (USApHCP). In this problem there is no capacity restriction on the hubs or on the flow between arcs. We want to choose a fixed number *p* of the nodes to be hubs and allocate the spoke nodes to exactly one of the chosen hubs in such a way that the maximum path between any o–d pair is minimized. The USApHCP is known to be NP-hard and even its single allocation subproblem (HCSAP) with respect to a given number of hubs is already NP-hard (see e.g. [1]).

In 1994, Campbell [2] presented a quadratic formulation for the USApHCP and a linearization of it which had the drawback of many additional variables. In 2000, Kara and Tansel [3] provided a linearization of Campbell's quadratic formulation, which clearly

* Corresponding author. Tel.: +49 631 205 4824.
E-mail addresses: tmeyer@mathematik.uni-kl.de (T. Meyer), andreas.ernst@csiro.au (A.T. Ernst), mohan.krishnamoorthy@csiro.au (M. Krishnamoorthy).

0305-0548/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2008.07.011

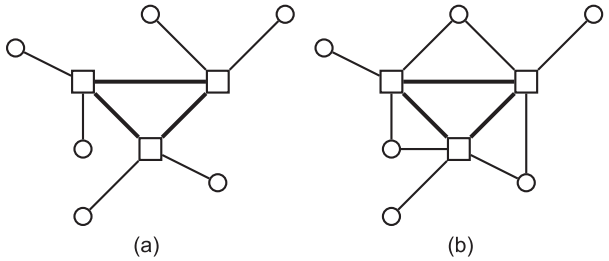

Fig. 1. Hub-and-spoke networks with different allocation schemes: (a) single allocation; (b) multiple allocation.

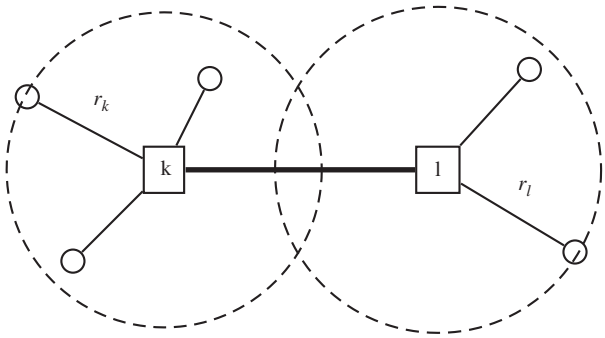

Fig. 2. Example of hubs and spokes with hub radii r_k and r_l and a path with length $r_k + r_l + \alpha c_{kl}$.

outperformed that of Campbell. The most promising formulation of the USApHCP was provided in 2002 by Ernst et al. [1]. Their formulation is based on the concept of a *hub radius*. Integer programming approaches based on these formulations solve the USApHCP exactly. Several heuristic methods were proposed in Ernst et al. [1,4]. The heuristics were analyzed with respect to their worst case behavior for the hub selection as well as for the allocation part of the USApHCP. Hamacher and Meyer [5] proposed an approach based on a combination of algorithms for solving hub covering problems with a binary search procedure. Recently, Gavriliouk [6] applied an aggregation technique to solve the USApHCP. For extensive surveys of hub location problems we refer to [7,8].

The paper is organized as follows: In the next section we introduce basic definitions, review the radius formulation and discuss a preprocessing procedure. Then in Sections 3 and 4 we describe methods for obtaining upper and lower bounds. The 2-phase algorithm for solving the USApHCP is presented in Section 5. In Section 6, we discuss the computational performance of the algorithm and the allocation subproblem—the HCSAP, which is used in the second phase. Section 7 concludes the paper with a short summary of our results.

2. Basics and preprocessing

Let $G = (N, E)$ be a complete undirected graph with node set $N = \{1, \dots, n\}$. Each pair of nodes is connected by an arc (i,j) with cost c_{ij}. We assume that the graph is symmetric, i.e., $c_{ij} = c_{ji}$, and satisfies the triangle inequality $c_{ij} \leqslant c_{ik} + c_{kj}$ for all $i,j,k = \{1, \dots, n\}$ (this can be done without loss of generality, since otherwise c_{ij} can be replaced by the shortest path distances between nodes i and j). The economy of scale is modeled by a discount factor $\alpha \in [0,1]$ on each of the hub-to-hub links. Due to the triangle inequality any o–d path between nodes i and j which are allocated to hubs k and l, respectively, has the length $d_{ij} = \chi c_{ik} + \alpha c_{kl} + \delta c_{jl}$ where χ and δ representing collection and distribution cost factors.

As already mentioned, the hub center problem is important for a hub network involving for instance perishable or time sensitive items. Hence, costs refer to time and the economy of scale is achieved due to higher speed on hub-to-hub links. Therefore, flow volume as well as collection and distribution cost factors are of less interest for hub center problems and are usually neglected in the literature.

Next, we briefly review the radius formulation. To this end, let $x_{ik} \in \{0,1\}$ be 1 if node i is allocated to hub k and 0 otherwise. Note that this definition implies that $x_{kk} = 1$ if and only if node k is a hub node. Moreover let r_k be a continuous variable representing the *radius* of hub k, i.e., the maximum cost between hub k and a spoke node allocated to k (Fig. 2).

The mixed integer formulation can then be stated as:
USApHCP-rad:

$$\begin{aligned}
\min\quad & z \\
\text{s.t.}\quad & z \geqslant r_k + r_l + \alpha c_{kl} \quad \forall k,l \in N && (1)\\
& r_k \geqslant c_{ik} x_{ik} \quad \forall i,k \in N && (2)\\
& \sum_{k \in N} x_{kk} = p && (3)\\
& \sum_{k \in N} x_{ik} = 1 \quad \forall i \in N && (4)\\
& x_{ik} \leqslant x_{kk} \quad \forall i,k \in N && (5)\\
& x_{ik} \in \{0,1\}, \quad r_k \geqslant 0 \quad \forall i \in N,\ k \in N && (6)
\end{aligned}$$

Constraint (1) defines the objective function z as the maximum cost of a path between any two nodes by using the respective hub radii. Constraints (5) and (2) make sure that node i can only be allocated to k, if k is a hub and if the cost c_{ik} between i and k is at most the radius r_k of k, respectively. With constraint (3) it is guaranteed that exactly p hubs are selected, while constraint (4) ensures that every node is assigned to exactly one hub.

Assume we are given an upper bound (UB) on the path length between any o–d pair. We can then apply a preprocessing step to eliminate allocations which cannot occur in a single allocation solution, since they would lead to a violation of the UB. Hence, only valid allocations need to be considered:

$$\mathscr{V} = \{(i,k) | 2c_{ik} \leqslant UB \text{ and } c_{ik} + \alpha \max_j c_{kj} \leqslant UB\}$$

In other words, i can be assigned to k if it can serve itself within the UB (first inequality) and if it is possible to get to every j in time (second inequality). As we will see in Section 4 the set $\mathscr{V}$ is very essential for the lower bound computation and can be further reduced during the algorithm.

In the first phase of our algorithm we use a shortest path based branch and bound, which was originally developed by Ernst and Krishnamoorthy [9] to solve the *uncapacitated single allocation p-hub median problem* (USApHMP). We will adapt this algorithm to obtain potential optimal hub locations for the USApHCP. To this end, let us introduce the definitions of *clusters* and *scenarios*.

A cluster C_i is defined as a subset of N whereas $\{C_1, \dots, C_r\}$ form a partition of N. Each cluster C_i has a hub-content h_i, where $\sum_{i=1}^{r} h_i = p$. Moreover let $H^+ = \bigcup_{i:h_i>0} C_i$ and $H^0 = \bigcup_{i:h_i=0} C_i$ be collections of nodes in clusters with positive and zero hub-content, respectively. Finally we define a scenario, $S = \{\{(C_i, h_i) : i = 1, \dots, r\}, \mathscr{V}\}$ to be a collection of clusters and their hub-contents, and the set of current valid allocations. Let $L(S)$ denote the lower bound for scenario S.

3. Obtaining upper bounds

In order to reduce the size of the branch and bound tree, it is desirable to find a good feasible solution to start with. There are a number of widely used meta-heuristic approaches that could be used as a guide to developing such a heuristic for this problem. Here we consider a number of options. The simplest is to use a constructive heuristic. In [1] a constructive heuristic is proposed that performs well for multiple allocation problems but rather poorly for single allocation problems. Furthermore, generally meta-heuristics that look at a large number of different solutions tend to perform significantly better than constructive heuristics that produce a single solution. We discarded evolutionary algorithm approaches as there is no clear way to encode hub location solutions in such a way as to allow efficient crossover of chromosomes while also ensuring that significant features of the parent are preserved in the offspring.

Another option is to use a local search method. For single allocation p-hub *median* problems a simulated annealing heuristic has been shown to work well in [10]. However, adapting this method to the USApHCP did not yield good results. For example for the 40.4 AP data set problem it produced a gap of over 2% in about the same time that the heuristic presented here takes for 100 node problems. One of the reasons for this is the structure of the solution space. This can be seen illustrated by looking at an example with 100 nodes and five hubs for which the optimal solution and a near optimal solution (the second best ever found by our heuristics) are shown in Fig. 3. Clearly though these solutions have very similar objective values they are structurally *very* different. In order to move between these two solutions a local search method would need to change four of the five hub locations and many allocations. This illustrates why it is difficult for a local neighborhood search method to move from good solutions to better solutions for this problem. Hence, we believe that other local search methods including tabu search or variable neighborhood search would also struggle to find good solutions.

Another class of meta-heuristic algorithms uses repeated solution construction to search for good solutions, including methods like GRASP, problem space search and ant colony optimization (ACO). We decided to use ACO as the bases for our heuristic though similar results could perhaps be obtained with one of the other methods in this group. The basic approach of ACO is based on an analogy with the foraging behavior of ants (see [11] for more details). In this algorithm an "ant" represents an individual solution. Solutions are constructed randomly with probabilities based on "pheromones" (tables of weights for selecting elements of the solution). These pheromones "evaporate" (that is the weights decrease) as the algorithm proceeds in order to discourage the same solution being tried again. However, ants deposit pheromones (increase the weights) for good solutions. Typically if τ_{iv} represents the pheromone level for variable i and value v, then the probability of setting $x_i = v$ is calculated as

$$\frac{\tau_{iv}\eta_{iv}^{\beta}}{\sum_w \tau_{iw}\eta_{iw}^{\beta}}$$

where η_{iv}^{β} are constant heuristic weights (possibly $\eta_{iw}^{\beta} = 1$).

The basic outline of the algorithm is given in Algorithm 1 which shows that the algorithm simply chooses hubs and then determines what nodes are to be allocated to the hub with emphasis on trying to find the furthest node to be allocated to each hub. Note that we essentially use two types of pheromones, τ_k to control how likely node k is to be chosen to be a hub and τ_{ik} which controls how likely node i is to be the furthest node allocated to k if k is a hub.

Algorithm 1. ACO heuristic for USApHCP

```
Initialize pheromone values τ_k ← 1 and τ_ik ← 1 ∀i,k ∈ N
Best bound B = ∞
for n_iter iterations do
  for n_ant iterations do
    Set ℋ ← ∅, 𝒩 ← N {Construct a new random solution}
    for p iterations do
      Randomly choose a hub k ∈ 𝒩 with probability determined by pheromones τ_k
      Set ℋ ← ℋ + k, 𝒩 ← 𝒩\k
      Randomly choose a node i ∈ 𝒩 to allocate to k with probability determined by pheromones τ_ik, 𝒩 ← 𝒩\i
      Allocate all nodes j ∈ 𝒩 such that d_jk < d_ik to k. 𝒩 ← 𝒩\{j ∈ 𝒩 : d_jk < d_ik}
    end for
    while 𝒩 ≠ ∅ do
      Pick a i ∈ 𝒩 with uniform probability
      if i can be allocated to any k ∈ ℋ without increasing the objective then
        Allocate i to k, 𝒩 ← 𝒩\i
      else
        Randomly choose a hub k ∈ ℋ with probability determined by pheromones τ_ik and allocate i to k; 𝒩 ← 𝒩\i
      end if
    end while
    Evaluate the current solution
    Update the best bound B and store the best solution found so far
  end for
  Update τ_k and τ_ik by rewarding the best solution found so far
end for
```

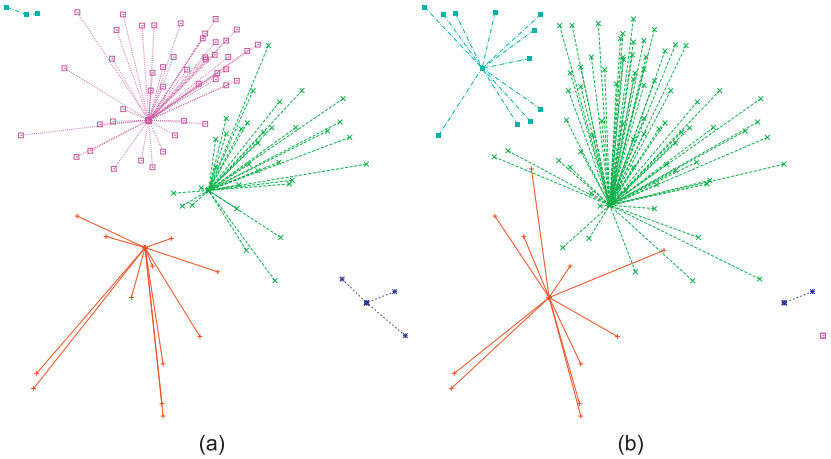

Fig. 3. Two solutions for the AP problem with 100 nodes and five hubs: (a) optimal; (b) gap 0.68%.

As Algorithm 1 shows we are repeatedly choosing hub nodes and then non-hub nodes to allocate to the hubs. Below we describe how the pheromone values control this choice and how the pheromones are updated as the algorithm precedes.

Choosing hub nodes: Each time a new hub node is to be chosen in Step 7, we normally randomly pick one of the nodes $k \in \mathcal{N}$ with probability

$$p_k = \frac{\tau_k}{\sum_{j\in\mathcal{N}}\tau_j}$$

However, with probability p_{greedy} we simply make a greedy choice, that is we select $\text{argmax}_{k\in\mathcal{N}}\tau_k$ as new hub node. Every time a hub node is chosen "local evaporation" is carried out. That is we update

$$\tau_k \leftarrow (1-\rho)*\tau_k + \rho * \frac{1}{n}$$

This encourages subsequent solution constructions to avoid using the same hub again in a bid to increase diversity. It also ensures that none of the pheromone values approach zero as the algorithm progresses. The factor ρ is referred to as the evaporation rate.

Choosing nodes to allocate to a hub: For choosing a "furthest" node to allocate to a hub in Step 9, a similar approach is used but also including a set of heuristic biases η_{ik}. Also we restrict the possible choices of nodes i to those that do not lead to an increase in solution cost above the current best bound. That is we define for hub k

$$b_k = \min\left\{\tfrac{1}{2}B, \min_{\ell\in\mathcal{H}}\min_{j \text{ allocated to } \ell}\{B - \alpha d_{k\ell} - d_{\ell j}\}\right\}$$
$$\mathcal{N}_k = \{k\} \cup \{i \in \mathcal{N} : d_{ik} \leqslant b_k\}$$

The restricted set of choices $\mathcal{N}_k$ of nodes to allocate to hub k discourages solutions that would be worse than the best already found. The node i to be allocated to a given hub k is then chosen with probabilities

$$p_{ik} = \frac{\tau_{ik}\eta_{ik}^{\beta}}{\sum_{j\in\mathcal{N}_k}\tau_{jk}\eta_{jk}^{\beta}}$$

As before we use probability p_{greedy} to determine if just a greedy choice should be made to select a node $\text{argmax}_{i\in\mathcal{N}_k}\tau_{ik}\eta_{ik}^{\beta}$. The parameter β limits the effect of the η_{ik}. The heuristic parameters η_{ik} are calculated according to

$$\eta_{ik} = \left(1 + \left|d_{ik} - \frac{1}{p+1}\max_{j\in N} d_{ij}\right|\right)^{-1}$$

This provides a bias based on the expectation that nodes that are far away will not be allocated to hub k while those that are too close will be chosen anyway if a node further away is selected for allocation to k. Clearly the expected distance of a non-hub node to a hub node will decrease as more hubs are available. Once a non-hub node i has been chosen local pheromone evaporation is carried out:

$$\tau_{ik} \leftarrow (1-\rho)\tau_{ik} + \rho\frac{1}{n} \tag{7}$$

Choosing a hub for an unallocated node: In Step 17 of Algorithm 1 a hub needs to be chosen for a node that has not been allocated yet. We use the same pheromones τ_{ik} but a set of dynamically calculated heuristic parameters $\bar{\eta}_{ik}$ that depend on the currently chosen set of hubs $\mathcal{H}$:

$$\bar{\eta}_{ik} = \max_{\ell\in\mathcal{H}}\{d_{i\ell}\} - d_{ik} + 1$$

Table 1
Results for the ACO heuristic on AP data sets

$n \cdot p$	Opt. found (%)	Avg. gap (%)	Avg. CPU	Avg. iter.
25.2	100.00	0.00	0.26	8
25.3	100.00	0.00	0.32	9
25.4	100.00	0.00	0.36	4
25.5	100.00	0.00	0.40	1
40.2	100.00	0.00	0.50	9
40.3	100.00	0.00	0.63	28
40.4	100.00	0.00	0.72	25
40.5	100.00	0.00	0.82	17
50.2	100.00	0.00	0.68	21
50.3	98.00	0.01	0.87	142
50.4	100.00	0.00	1.03	33
50.5	100.00	0.00	1.15	20
100.2	96.50	0.02	2.15	107
100.3	83.20	0.35	2.57	327
100.4	77.80	0.12	2.96	344
100.5	1.70	0.24	3.35	228
200.2	99.60	0.01	8.20	94
200.3	100.00	0.00	9.84	65
200.4	97.00	0.04	11.34	408
200.5	70.90	0.25	12.78	291

The probability for choosing a hub $k \in \mathcal{H}$ to allocate the given node i is then

$$\bar{p}_{ik} = \frac{\tau_{ik}\bar{\eta}_{ik}^{\bar{\beta}}}{\sum_{\ell\in\mathcal{H}}\tau_{i\ell}\bar{\eta}_{i\ell}^{\bar{\beta}}}$$

Here we use a value of $\bar{\beta} = 0.5$. As before with probability p_{greedy} a greedy choice is made and irrespective of how k is chosen, local evaporation is carried out according to Eq. (7).

Rewarding the best solution: To compensate for the local evaporation of pheromones carried out during the construction of solutions, we periodically increase (reward) the pheromones corresponding to the best solution. In particular if $\mathcal{H}^*$ is the set of hubs of the best solution found so far and i_k^* is the furthest node from k that is allocated to each hub $k \in \mathcal{H}^*$ then we set

$$\tau_k \leftarrow (1-\rho)\tau_k + Q\frac{1}{B} \quad \forall k \in \mathcal{H}^*$$
$$\tau_{i_k^* k} \leftarrow (1-\rho)\tau_{i_k^* k} + Q\frac{1}{B} \quad \forall k \in \mathcal{H}^*$$

Here B is the best solution objective found so far as before and Q a scaling parameter.

3.1. Computational results

Clearly the heuristic described in this section requires a significant number of parameters to be set, though as discussed below the exact value used is not crucial for the performance of the algorithm. After some limited experimentation we have chosen the following parameters for use in generating the results in this paper: n_{iter}=1000, $n_{\text{ants}} = 10 + n$, $\rho = 0.005$, $p_{\text{greedy}} = 0.1$ and $Q = 0.05\max_{i,j\in N}\{d_{ij}\}$. The algorithm was coded in C++, compiled with the Intel C++ compiler using −O3 optimization flag and run on a cluster of Linux nodes with dual Intel Xeon 3.2 GHz processors. The algorithm as implemented does not require large amounts of memory (about 11Mb for 200 node problems).

In Table 1 we provide results from running the ACO heuristic on some AP data sets. For each instance the heuristic was run 1000 times. We report the percentage of runs that found the optimal solution and the average gap between the heuristic solution found and

Table 2
Results for the ACO heuristic on the 100.5 AP data set with changes to various parameters

Param.	Opt. found (%)	Avg. gap (%)	Avg. CPU	Avg. iter.
Original	1.60	0.25	3.33	234
10ρ	0.30	0.28	3.43	348
$\rho/10$	2.00	0.23	3.47	258
$2p_{\text{greedy}}$	2.80	0.23	3.27	207
$p_{\text{greedy}}/2$	1.50	0.24	3.37	245
$n_{\text{ants}} = 10 + n/2$	1.00	0.31	1.83	326
$n_{\text{ants}} = 10 + 2n$	2.20	0.23	6.39	158
$n_{\text{iter}}/2$	0.50	0.28	1.71	209
$2n_{\text{iter}}$	4.30	0.22	6.62	266
$4Q$	3.20	0.25	3.20	155
$Q/5$	0.30	0.28	3.47	386

the optimum. We also provide the average CPU seconds and the average number of iterations until the best solution was found. This shows that for problems with less than 100 nodes the heuristic finds the optimal solution and generally would need much less than the 1000 iterations allowed for. However, for larger problems it occasionally fails to find the optimal solution and in fact rarely finds the optimal solution in the case of 100 nodes with five hubs. The run times are generally very consistent with little variation between the average reported and the times required by individual runs.

While this may appear to be a large number of parameters that have to be tuned to obtain these results, the algorithm is in fact fairly insensitive to the value chosen for many of these parameters. Of course the run time increases proportionally to the number of iterations or the number of ants (solutions) generated per iteration. Similarly having very small evaporation ρ and reward Q would mean the algorithm takes a very long time to converge while large Q could lead to premature convergence. However, in practice the exact values chosen for these parameters are not very important. This is demonstrated in Table 2 where we show how the results vary with changes to the ACO parameters for 1000 runs of the 100 node 5 hub data set. The columns are as before, except that the first column describes how one of the parameters was changed from the defaults described at the beginning of this section (e.g. $2p_{\text{greedy}}$ indicates that the greedy probability was doubled to 0.2). Clearly some of the changes are more by chance then due to the parameter change. For example the first row in principle should be the same as the 100.5 row of Table 1. However, generally the results are as expected. The biggest impact is by the parameters controlling the number of solutions generated (n_{ants} and n_{iter}) which directly affect run time and also have a clear impact on the solution quality. The impact of some of the other parameters is much less pronounced. It is unsurprising that increasing the reward leads to quicker convergence (smaller number of iterations until the best solution found), though this appears to have very little effect on the average quality of solutions found. Even the fairly large changes to the evaporation rate or the greedy probability tried appear to have only fairly little effect on the quality of solutions produced. Overall the results support the conclusion that the types of results achieved are due to the structure of the heuristic algorithm more so than due to the particular parameter values chosen.

4. Obtaining lower bounds

The lower bound calculation in our branch and bound tree is restricted to scenarios which are generated by the branching process described in Section 5.1.2. Let us define the set $\mathscr{A}_i$ as the set of hubs to which node i can be allocated. In other words, $\mathscr{A}_i = \{k \in H^+ | (i,k) \in \mathscr{V}\}$.

Based on the preprocessing step there are two possibilities for scenario S:

1. If there exists $i \in H^0$ with $|\mathscr{A}_i| = 0$, node i cannot be allocated to any of the hubs in H^+. Hence, there would never be a feasible single allocation solution, i.e., $L(S) = \infty$.
2. Otherwise, we construct a three layered graph $\overline{G}$ and compute a lower bound.

Graph $\overline{G}$ was first introduced in [9]. However, due to our preprocessing step we slightly change the construction of $\overline{G}$. Let N_1, N_2 and N_3 be the node sets for the three layers. N_1 and N_2 contain copies of all nodes in N (i.e., $H^0 \cup H^+$) while N_3 contains only nodes in H^+. Now we add arcs with cost

- $\overline{c}_{ik} = c_{ik}$ for all nodes $i \in N_2 \backslash H^+$ and $k \in N_3$ with $(i,k) \in \mathscr{V}$ and $k \in \mathscr{A}_i$.
- $\overline{c}_{kj} = c_{kj}$ for all nodes $j \in N_1 \backslash H^+$ and $k \in N_3$ with $(j,k) \in \mathscr{V}$ and $k \in \mathscr{A}_j$.
- $\overline{c}_{ik} = \overline{c}_{kj} = 0$ for all nodes $i \in N_2 \backslash H^0$, $j \in N_1 \backslash H^0$ and $k \in N_3$ with $i, j = k$.
- $\overline{c}_{kl} = \alpha c_{kl}$ between nodes k and l if they are either in different clusters or in the same cluster C_i with $h_i > 1$.
- $\overline{c}_{kl} = c_{kl}$ if k and l are in the same cluster C_i with $h_i = 1$.

Now the cost for the maximum path between any nodes $i, j \in N$ is bounded below by the shortest path cost between any $i \in N_2$ and $j \in N_1$ for any choice of hub locations that agrees with the given hub-content of the partition. In Fig. 4 we see an example of how $\overline{G}$ is constructed.

For a given scenario we treat all nodes in regions that have positive hub-contents as hubs and calculate a lower bound using an all pairs shortest path algorithm in $\overline{G}$. Although, due to the preprocessing step, these lower bounds are already quite tight, we can improve them further by taking into account the single allocation constraint and by updating the set of valid allocations. To this end let $\overline{d}_{kj}$ be the shortest path distance from hub k to a non-hub node j, i.e., $\overline{d}_{kj} = \min_{l \in H^+}\{\overline{c}_{kl} + \overline{c}_{lj}\}$ for all $k \in H^+$ and $j \in H^0$. Then the lower bound can be computed as follows:

$$L(S) = \max_{i \in H^0} \min_{k \in H^+} \max_{j \in N} (\overline{c}_{ik} + \overline{d}_{kj})$$

In other words, for all non-hub nodes the lower bound is calculated by finding the maximum path of collecting all flow to a single hub and distribute it via the cheapest route to its destination. The calculation above assumes single source with multiple distribution points. Note that, since in the p-hub center problem $\chi = \delta = 1$, the scheme with multiple sources and single distribution points would produce the same lower bound.

The reduction of $\mathscr{V}$ for scenario S can be combined with the lower bound calculation and is described in Algorithm 2.

Algorithm 2

Let $\overline{d}_{kj} = \min_{l \in H^+}\{\overline{c}_{kl} + \overline{c}_{lj}\}$ for all $k \in H^+$ and $j \in H^0$.
for $i \in H^0$, $k \in H^+$ and $j \in N$ **do**
  if $\overline{c}_{ik} + \overline{d}_{kj} > UB$ **then**
    $\mathscr{V} \leftarrow \mathscr{V} \backslash (i,k)$
    $\overline{c}_{ik} = \infty$
  end if
end for

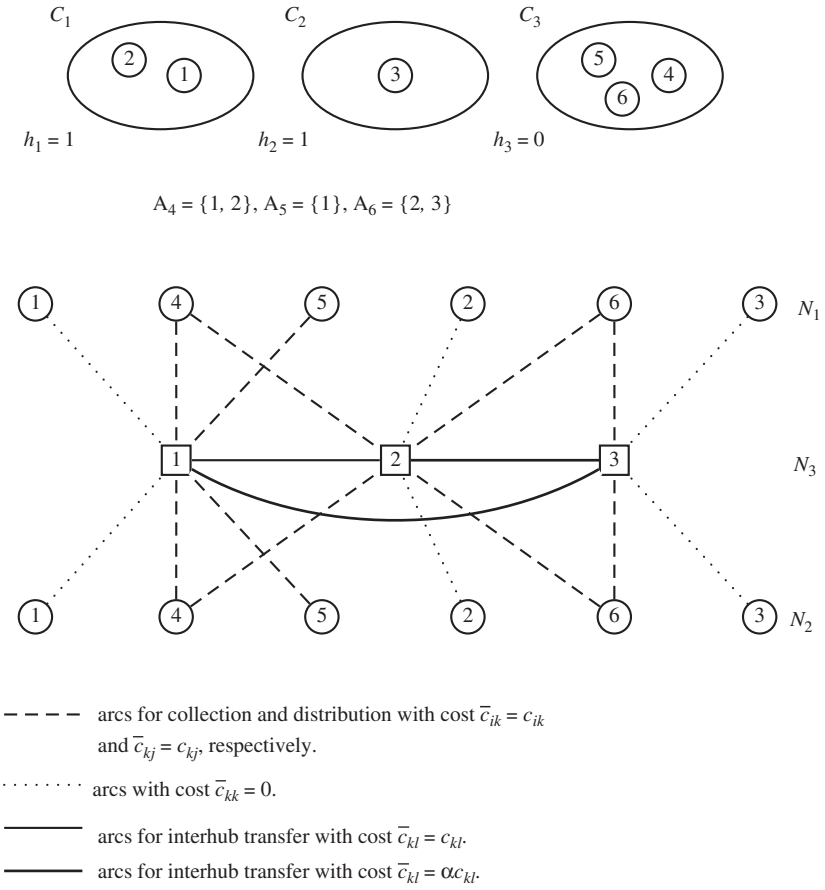

Fig. 4. Illustration of the construction of $\bar{C}$.

Let S^p and S^c be a parent and a child scenario and let H^p and H^c the corresponding hub sets. Since $H^c \subset H^p$, any invalid allocation of S^p would also be invalid for S^c. Hence, reducing $\mathscr{V}$ leads to a more efficient lower bound calculation in the child scenario.

5. The 2-phase algorithm

Our 2-phase algorithm tackles the location of hubs and the single allocation problem separately. The first phase is carried out by a branch and bound returning a set of potential optimal hub locations. We refer to this set as the dominating hub set (DHS). This is followed by an allocation phase using a reduced sized formulation which returns an optimal solution. The DHS approach is motivated by two main reasons. Firstly, branching on allocation within the branch and bound would lead to a massive increase of the search tree and therefore it is likely to run into memory problems. Even if the cardinality of the DHS is large, we can split it and solve the allocation problem in parallel. In fact, in an upcoming paper, we also applied the 2-phase algorithm to the USApHMP and compared it to the original branch and bound. It turned out that the new approach was able to solve much larger problems where the branch and bound ran out of memory—reason being the allocation process. Secondly, if a user (e.g. airline industry) has multiple objectives the DHS provides multiple options without resolving the problem.

As already mentioned we adapt the shortest path based branch and bound for the USApHMP to compute the DHS for the USApHCP. In the next subsection, we shall give an outline of the branch and bound algorithm. See [9] for more details.

5.1. Phase 1: branch and bound

Our search tree does not start with a single root node but with a given number r of initial cluster. As reported in [10] this strategy has significantly improved computational efficiency for the USApHMP. Each node of the branch and bound forest is fully specified by a scenario S. We solve the subproblem associated with scenario S.

There are three possibilities:

1. If $L(S) < UB$ and the solution to the subproblem is feasible for the original problem with respect to the number of hubs, i.e., $|H^+| = p$, we add this hub set to DHS and backtrack.
2. If $L(S) \geqslant UB$, then backtrack.
3. Otherwise, if $L(S) < UB$ and $|H^+| > p$, we perform further branching. This process continues until no further backtrack or no further branching is possible and all nodes in the forest are explored.

5.1.1. The root level

Let the diameter of a cluster be the maximum distance between any two nodes in the cluster. The r clusters are generated as follows:

1. Let each node be a cluster itself.
2. Combine any two clusters such that the diameter of the new combined cluster is no larger than that obtained by joining any other two current clusters.
3. If the number of clusters is r, STOP; else go to 2.

Preliminary computational tests have shown that we get the best overall performance with $r = n/p$ and $n/10$.

5.1.2. Branching strategy

Our branch and bound tree is searched in a depth first manner. A tie between scenarios is broken by choosing the one that has the maximum number of nodes per hub. Once a node in the branch and bound forest is identified for performing branching, new child nodes can be generated by firstly generating two new subregions and then forming new scenarios. Two subregions of a parent node are created by first choosing two nodes s and t such that the distance between s and t is the diameter of the parent region. Then two regions are formed by assigning each node in the parent region to the closer node among s and t. New scenarios are encompassed by the scenario of the parent node and cover all of the possibilities entailed by the parent scenario—including the (updated) set $\mathscr{V}$. More precisely, a region with a positive hub-content is selected and split into two subregions. We then construct all possible scenarios from these two subregions and hub-content combinations.

5.2. Phase 2: single allocation problem

In real world applications, such as airline and postal systems, the hub locations are usually fixed over a period of time because of managerial and economic reasons. However, changing the allocations (e.g. flights) is more practical and less expensive. Hence, good methods for solving the spoke allocation are required. In 2002, the HCSAP was first formulated by Ernst et al. [4] as a linear integer program based on the radius formulation. We refer to this formulation as **HCSAP-r**. In the same paper they proved that the HCSAP is NP-hard and proposed four heuristics to tackle the allocation subproblem. Campbell et al. [12] also presented complexity results and IP formulations. However, one formulation is similar to that in [4]. Moreover they established some polynomially solvable cases, in particular, when $\alpha = 0$, when there are only two hubs, and when the graph is a tree or a path.

Let $\mathscr{H}$ be the set of hub facilities with $|\mathscr{H}| = p$. Define $h(i) \in \mathscr{H}$ to be the hub to which node i is assigned and let $\mathscr{S}$ and $\mathscr{M}$ be the set of single allocations and multiple allocations, respectively:

$$\mathscr{S} := \{(i,k) : |\{k|h(i) = k\}| = 1\}$$

$$\mathscr{M} := \{(i,k) : h(i) = k\}$$

It is easy to see that for any given hub set $\mathscr{H}$ it holds that $\mathscr{S} \subseteq \mathscr{M}$. **HCSAP-r**, which is the best known formulation so far, can be improved by taking into account the lower bound solution. Once we have solved an all pairs shortest path problem we can fix all spoke node assignments which are already single allocated. The above observations result in the following model:

HCSAP-sp:

$$\begin{aligned}
\min\quad & z \\
\text{s.t.}\quad & z \geqslant r_k + r_l + \alpha c_{kl} \quad \forall k,l \in \mathscr{H} && (8)\\
& r_k \geqslant c_{ik} x_{ik} \quad \forall i : (i,k) \in \mathscr{M} && (9)\\
& \sum_{k \in \mathscr{H}} x_{ik} = 1 \quad \forall i : (i,k) \in \mathscr{M} \setminus \mathscr{S} && (10)\\
& x_{ik} = 1 \quad \forall (i,k) \in \mathscr{S} && (11)\\
& x_{ik} \in \{0,1\}, \quad r_k \geqslant 0 \quad \forall (i,k) \in \mathscr{M},\ k \in \mathscr{H} && (12)
\end{aligned}$$

Note that if $\alpha = 1$ every non-hub node is allocated to its nearest hub and thus **HCSAP-sp** and **HCSAP-r** would coincide. However, in most applications economies of scale can be achieved using hub links and therefore α would be less than 1.

Having discussed the two phases of the algorithm in more detail, we shall give a more compact description. To this end let $\mathscr{H}^*$ be the hub set obtained by the ACO heuristic.

Algorithm 3. 2-Phase algorithm

```
1:  Set DHS = ∅
2:  Phase 1:
3:  Compute an upper bound using the ACO heuristic
4:  DHS = DHS ∪ {H*}
5:  Compute the set of valid allocations
6:  Apply shortest path based B&B to get the DHS
7:  if |DHS| = 1 then
8:      STOP: upper bound is optimal
9:  else
10:     Phase 2:
11:     for H ∈ DHS do
12:         solve HCSAP-sp
13:     end for
14: end if
```

Note that we have already calculated the lower bound solution for each $\mathscr{H} \in DHS$ in the first phase of our algorithm, thus in Step 12 of Algorithm 3 we can drop the preprocessing needed for **HCSAP-sp**.

6. Computational results

The proposed 2-phase algorithm, coded in C/C++, is compared to the radius formulation for the USApHCP and has been compiled with GNU gcc/g++ v 4.2.2 using options –O6 and –funroll-loops. All numerical tests were performed on a dual Intel Xeon 3.2 GHz machine equipped with 4 GB RAM, running a 32-bit Linux, Kernel 2.6.5. We used the built-in branch and bound routine of Ilog CPLEX 11.0 to solve the integer programs.

6.1. Test problems

The AP (Australian Post) data set and a newly created data set are used to prove the effectiveness of our new solution approach. The AP data set was introduced by Ernst and Krishnamoorthy [10] and is considered to be a benchmark by most researchers in the hub location area. It is derived from the real-world application of a postal delivery network and consists of 200 nodes, each representing a postal district. The discount factor is a constant and given by α=0.75.

To demonstrate that the 2-phase algorithm can also solve larger problems we created a new data set consisting of 400 nodes. For this data set the x and y coordinates were randomly generated from $U[0, 100\,000]$. We created different instances by choosing subsets with $n \in \{100, 200, 300, 400\}$, $p \in \{2, 3, 4, 5\}$ and $\alpha = 0.75$. In the following we will refer to this data set as URAND.

Although the HCSAP is NP-hard it can be solved much easier than the USApHCP. Consequently, to test the performance of our reduced size formulation, we generated three large data sets consisting of 300, 400, 500, 600, and 1000 nodes. For each problem size, the number of hubs p is set to 3, 5, 7, 10, and 20. We will refer to these data sets as RAND1, RAND2 and RAND3, respectively. As for the URAND the x and y coordinates were taken randomly from $U[0, 100\,000]$ and we set $\alpha = 0.75$.

6.2. Results for the 2-phase algorithm

Tables 3 and 4 present results for the 2-phase algorithm based on the AP and URAND data sets. The meanings of the column headings are as follows: The first column specifies the test problem $n \cdot p$ where n is the number of nodes in the graph and p the required number of hubs. The optimal objective is given in column 2. The next two columns correspond to the percentage upper gap and the CPU_h time of the ACO heuristic. The gap between the optimal solution and that one obtained by the ACO heuristic is defined as $(z_{\text{ACO}} - z^*)/z^* \times 100$, being z^* the optimal objective function value and z_{ACO} the

Table 3
Results for the 2-phase algorithm (AP data set)

$n \cdot p$	Opt. cost	Heuristic		2-phase algorithm					CPLEX
		%	CPU_h	nodes	$\|DHS\|$	CPU_{bb}	CPU_a	**CPU**	
25.2	53 207.46	0.00	0.39	7	1	0.00	0.00	0.39	1.36
25.3	46 608.31	0.00	0.47	4	2	0.01	0.00	0.48	2.37
25.4	45 552.50	0.00	0.50	6	1	0.00	0.00	0.50	2.34
25.5	45 552.50	0.00	0.56	14	1	0.00	0.00	0.56	0.33
40.2	61 682.50	0.00	0.68	13	1	0.00	0.00	0.68	26.82
40.3	58 192.76	0.00	0.85	22	1	0.00	0.00	0.85	12.86
40.4	52 265.27	0.00	0.96	5	1	0.00	0.00	0.96	12.95
40.5	49 741.20	0.00	1.05	9	1	0.00	0.00	1.05	6.12
50.2	65 523.37	0.00	0.93	19	1	0.01	0.00	0.94	82.92
50.3	60 132.14	0.00	1.15	66	1	0.01	0.00	1.16	90.29
50.4	52 905.77	0.00	1.37	6	1	0.00	0.00	1.37	28.09
50.5	50 707.87	0.00	1.49	38	1	0.01	0.00	1.50	16.68
100.2	65 914.82	0.00	2.72	34	1	0.07	0.00	2.79	322.66
100.3	60 658.88	0.00	3.23	157	1	0.14	0.00	3.37	1006.39
100.4	56 124.74	0.00	3.66	50	1	0.10	0.00	3.76	3927.14
100.5	54 243.50	0.22	3.98	408	2	0.49	0.02	4.49	1891.49
200.2	68 231.97	0.00	9.43	47	1	0.70	0.00	10.13	***
200.3	64 237.35	0.00	11.32	495	1	1.47	0.00	12.79	***
200.4	59 999.08	0.00	12.82	163	5	1.30	0.08	14.20	***
200.5	58 561.76	0.00	14.79	1740	1	10.48	0.00	25.27	***

Table 4
Results for the 2-phase algorithm (URAND data set)

$n \cdot p$	Opt. cost	Heuristic		2-phase algorithm										CPLEX
				$r = \frac{n}{p}$					$r = \frac{n}{10}$					
		%	CPU_h	nodes	$\|DHS\|$	CPU_{bb}	CPU_a	**CPU**	nodes	$\|DHS\|$	CPU_{bb}	CPU_a	**CPU**	
100.2	127 987.22	0.00	3.35	58	1	0.07	0.00	3.42	63	1	0.08	0.00	3.43	2578.00
100.3	119 919.87	0.00	4.08	358	1	0.21	0.00	4.29	221	1	0.25	0.00	4.33	***
100.4	111 452.93	0.00	4.81	576	1	0.37	0.00	5.18	414	1	0.58	0.00	5.39	***
100.5	106 174.56	0.00	5.54	349	1	0.42	0.00	5.97	183	1	0.30	0.00	5.84	***
200.2	132 146.71	0.00	12.14	176	1	0.70	0.00	12.84	124	1	0.76	0.00	12.90	***
200.3	121 782.28	0.00	15.31	1274	1	2.60	0.00	17.91	273	1	1.66	0.00	16.97	***
200.4	116 242.74	1.09	17.57	22 443	40	43.37	0.37	60.94	3279	40	14.81	0.39	32.77	***
200.5	108 511.29	0.00	20.46	2563	1	13.06	0.00	33.52	849	1	6.00	0.00	26.46	***
300.2	132 146.71	0.00	27.73	319	1	3.53	0.00	31.26	334	1	4.68	0.00	32.41	***
300.3	122 270.07	0.38	34.91	14 376	522	103.30	10.00	148.21	3705	2	34.80	0.01	69.72	***
300.4	116 573.58	0.41	41.20	63 535	15	299.89	0.18	341.27	19 730	15	206.09	0.19	247.48	***
300.5	109 522.22	0.00	48.19	16 338	1	144.60	0.00	192.79	3184	1	40.86	0.00	89.05	***
400.2	131 721.57	0.32	54.75	368	3	9.67	0.03	64.45	666	5	15.90	0.05	70.70	***
400.3	122 270.07	1.54	68.17	14 376	522	108.21	10.71	187.09	5843	522	82.72	10.78	161.67	***
400.4	115 843.77	2.06	88.36				Out of memory		52 384	1218	868.61	25.88	982.85	***
400.5	109 522.22	0.00	92.30				Out of memory		6967	1	162.23	0.00	254.53	***

objective function value obtained by the heuristic, respectively. For the 2-phase algorithm we report:

nodes: number of branch and bound nodes;
CPU_{bb}: CPU time required for the branch and bound;
CPU_a: CPU time required for the allocation;
CPU: $CPU_h + CPU_{bb} + CPU_a$.

The last column gives the CPU time for solving the radius formulation with CPLEX. "***" indicates that the problem could not be solved by CPLEX within 6000 CPU seconds. Note that for all experiments the CPU time was measured in seconds.

Tables 3 and 4 provide strong evidence that the proposed solution approach can solve large scale instances in a reasonable amount of time, where CPLEX fails to find an optimal solution. For both data sets the 2-phase algorithm always required significantly less CPU time (except for AP25.5) than the MIP-based approach. For the AP data the optimal solution was found very fast—even for the largest instances. In 17 out of 20 AP problems the UB was proven to be optimal in the first phase, i.e., $|DHS| = 1$. As can be seen in Table 1, the second phase was only needed three times with at most five hub set combinations for which the allocation problem had to be solved. To determine the DHS (or to prove optimality of the UB) fewer than 500 branch and bound nodes were needed except for AP200.5. Note that the number of tree search nodes corresponds to the number of shortest path calculations. The observations indicate that our lower bounding technique provides extremely tight bounds. For the AP data set we have chosen $r = n/p$ for the number of initial clusters to start with. Although the overall performances for $r = n/p$ and $n/10$ are comparable with respect to the number of branch and bound nodes and CPU time. However, for the larger URAND data set the difference became more significant, especially for the 300 and

Table 5
Results for the HCSAP (RAND1, RAND2, RAND3)

p	3		5		7		10		20	
n	sp	r	sp	r	sp	r	sp	r	sp	r
300	0.01	0.29	0.09	3.81	0.13	14.45	0.86	21.50	2.12	106.79
300	0.02	0.17	0.03	1.09	0.28	2.55	1.07	12.73	1.11	34.46
300	0.01	0.41	0.04	0.84	0.09	8.78	1.31	12.77	0.66	20.67
400	0.03	0.31	0.08	0.86	0.18	2.28	0.56	10.53	5.26	29.12
400	0.04	0.35	0.05	1.20	0.37	1.31	2.95	45.11	5.94	24.05
400	0.05	0.76	0.08	1.04	0.19	1.55	0.78	46.01	3.04	38.70
500	0.03	0.55	0.07	1.38	0.23	1.67	0.14	4.71	5.20	164.24
500	0.04	0.69	0.15	1.17	0.59	8.20	0.34	11.41	5.34	411.37
500	0.03	0.41	0.1	1.38	0.18	1.80	0.42	7.75	12.7	130.09
600	0.04	0.80	0.07	1.24	0.21	8.02	3.44	245.94	16.64	228.34
600	0.04	0.84	0.11	13.08	0.96	9.27	1.19	55.87	10.03	717.04
600	0.03	0.53	0.11	2.22	1.87	7.18	6.57	63.64	14.75	774.10
1000	0.08	2.26	0.29	5.37	0.58	66.80	2.60	16.44	119.60	1942.12
1000	0.09	2.27	0.08	39.2	1.28	33.02	1.17	16.16	16.06	953.95
1000	0.06	1.85	0.34	5.27	9.15	92.7	1.45	62.28	27.12	994.51

Table 6
Average CPU time

Data set	**HCSAP-sp**	**HCSAP-r**
RAND1	168.62	2889.99
RAND2	59.4	2406.78
RAND3	91.26	2287.45
Total avg.	106.42	2528.07

400 nodes instances. Here we observed that the results were much better with $r=n/10$. In order to confirm that this is, indeed, the better choice for the initial number of clusters, we provide computational results for URAND for both values of r, namely, $r=n/10$ and n/p.

From Table 4 we can see the same behavior of the 2-phase algorithm for the URAND data set as for the AP data set—the proposed method performs consistently better than CPLEX and the cardinality of the DHS is very small compared to the problem size. Even the largest instances could be solved in a reasonable amount of computation time.

6.3. Results for the HCSAP

In the following we provide results of our reduced size formulation, which was discussed in Section 5.2. Table 5 gives the CPU time required for solving **HCSAP-sp** and **HCSAP-r** with CPLEX. For each problem size n the three corresponding rows give the results for data sets RAND1, RAND2, and RAND3. The number of hubs p is given in each column block. As already mentioned the hubs needed as input for the HCSAP are provided from the ACO heuristic presented in Section 3. As can be seen from the table formulation **HCSAP-sp** clearly outperforms the original radius formulation for the HCSAP. The results of Table 5 are summarized in Table 6 where we provide the total CPU time for each data set as well as the average CPU time over all three data sets. The average CPU time used by **HCSAP-r** is more than 20 times the CPU time of **HCSAP-sp**.

7. Conclusions

In this paper, we proposed a 2-phase algorithm for solving the uncapacitated single allocation p-hub center problem. In the first phase we compute a set of potential optimal hub combinations using a shortest path based branch and bound. Then in the second phase the allocation part is solved with a reduced sized formulation. Moreover we developed a heuristic algorithm based on ant colony optimization. This heuristic was used to compute the upper bound needed for the branch and bound. Our computational experiments indicate that our solution approach is significantly faster than standard MIP solvers like CPLEX. The effectiveness of the 2-phase algorithm can be traced back to the fact that the new heuristic and the lower bounding technique are very efficient. As a result, we are able to provide exact solutions to problems larger than those attempted previously in the literature.

Acknowledgments

We wish to thank two anonymous referees for improving this paper through their helpful comments and criticism.

References

[1] Ernst AT, Hamacher HW, Jiang H, Krishnamoorthy M, Woeginger G. Uncapacitated single and multiple allocation p-hub center problems. Computers and Operations Research 2008, accepted.
[2] Campbell JF. Integer programming formulations of discrete hub location problems. European Journal of Operational Research 1994;72:387–405.
[3] Kara BY, Tansel BC. The single-assignment p-hub center problem. European Journal of the Operational Research 2000;125:648–55.
[4] Ernst AT, Hamacher HW, Jiang H, Krishnamoorthy M, Woeginger G. Heuristic algorithms for the uncapacitated single allocation p-hub center problems. Unpublished Report; 2002.
[5] Hamacher HW, Meyer T. Hub cover and hub center problems. Technical Report 98, FB Mathematik TU Kaiserslautern; 2006.
[6] Gavriliouk O. Aggregation in hub location problems. Computers and Operations Research 2008, submitted for publication.
[7] Alumur S, Kara BY. Network hub location problems: the state of the art. European Journal of Operational Research 2008;190(1):1–21.
[8] Campbell JF, Ernst AT, Krishnamoorthy M. Hub location problems. In: Hamacher H, Drezner Z, editors. Location theory: applications and theory. Berlin: Springer; 2002. p. 373–406.
[9] Ernst AT, Krishnamoorthy M. An exact solution approach based on shortest-paths for p-hub median problems. INFORMS Journal on Computing 1998;10(2):149–62.
[10] Ernst AT, Krishnamoorthy M. Efficient algorithms for the uncapacitated single allocation p-hub median problem. Location Science 1996;4(3):139–54.
[11] Dorigo M, Stützle T. Ant colony optimization. Cambridge, MA: MIT Press; 2004.
[12] Campbell AM, Lowe TJ, Zhang L. The p-hub center allocation problem. European Journal of Operational Research 2007;176:819–35.