Linear Programming-Based Algorithms for the Fixed-Hub Single Allocation Problem

Dongdong Ge,* Yinyu Ye,* Jiawei Zhang †

March 20, 2007

Abstract

This paper discusses the fixed-hub single allocation problem. In the model hubs are fixed and fully connected; and each terminal node is connected to a single hub which routes all its traffic. The goal is to minimize the cost of routing the traffic in the network. This paper presents linear programming-based algorithms that deliver both high quality solutions and a theoretical worst case bound. Computational results indicate that our algorithms solve large-sized problems efficiently. The algorithms are based on a new randomized rounding method, which might be of interest on its own.

Key words: hub location; network design; linear programming; worst case analysis

1. Introduction

Hub-and-spoke networks have been widely used in transportation, logistics, and telecommunication systems. In such networks, traffic is routed from numerous nodes of origin to specific destinations through hub facilities. The use of hub facilities allows for the replacement of direct connections between all nodes with fewer, indirect connections. One main benefit is the economies of scale as a result of the consolidation of flows on relatively few arcs connecting the nodes. In the United

*This research is supported by the Boeing Company. Email: {dongdong, yinyu-ye}@stanford.edu
†Stern School of Business, IOMS-Operations Management, New York University, New York, NY, 10012. Email: jzhang@stern.nyu.edu
States, hub-and-spoke routing is practically universal. Airlines adopted it after the industry was deregulated in 1978. Many logistics service providers such as UPS and Federal Express also have distribution systems using hub-and-spoke structure.

Given its widespread use, it is of practical importance to design efficient hub-and-spoke networks. In the literature, such problems are often referred to as hub location problems, in which two major questions need to be addressed: where hubs should be located and how the traffic/flow (be it passengers in transportation, packages in logistics, and communication packets) should be routed.

One important hub location problem is called the $p$-hub median problem. In this problem, the objective is to locate $p$ hubs in a network and allocate non-hub nodes to hub nodes such that the sum of the costs of transporting flows between all origin-destination pairs in the network is minimized. In 1987, O’Kelly [18] proposed a quadratic integer program for the $p$-hub median problem. Two primary heuristics, along with the applications to air transportation instances, were also reported by O’Kelly [18] to compute upper bounds of the objective function value. Later on, Klincewicz [14] developed an exchange heuristic and a clustering heuristic using a multi-criteria distance and flow-based allocation procedure. Campbell [5] proposed a greedy exchange heuristic for the $p$-hub median problem with multiple allocations and two heuristics for the single-allocation problem based on flow information. An efficient tabular search heuristic was suggested by Skorin-Kapov et al. [23].

The linearization of the quadratic model was also developed [4, 20, 19, 11, 12], and it can often generate integer solutions without forcing integrality for small-sized problems (up to 25 nodes). A common assumption of these papers is that each non-hub node is required to be assigned to exactly one hub. In this case, the problem is sometimes referred to as the $p$-hub median problem with single allocation.

Since the work of O’Kelly [18], the $p$-hub median problem and its variants have received substantial attention; see, for instance, [3, 17, 22, 10, 16]. An overview of research on the $p$-hub median problem and other hub location problems can be found in [6].

Very recently, Campbell et al. [7, 8] proposed the hub arc location problem where the question of interest is where hub arcs, each of which connects two hubs, should be located.

In both hub location problems and hub arc location problems, even when the locations of
hubs and/or hub arcs are specified in advance, optimally assigning the non-hub nodes to the hub nodes is still a challenging task. We refer to this problem as the fixed-hub allocation problem. This problem, although only a sub-problem to the hub location or hub arc location problems, is of particular importance. First, in many practical situations, the locations of hubs are pre-determined and remain unchanged in a long term. Second, the number of hubs can be relatively small, which makes it possible to enumerate all possible locations of the hubs. Further, solving the fixed-hub allocation problem efficiently would help us solve the hub location (or hub arc location) problem.

Therefore, we are concerned with the fixed-hub single allocation problem (FHSAP). For convenience, we may also use the notation $k$-FHSAP when the number of hubs is $k$. The FHSAP is known to be NP-hard and is a special case of the quadratic assignment problem. Sohn and Park [24] showed that, although the 2-FHSAP is a min-cut problem and thus is polynomial-time solvable, the 3-FHSAP is NP-hard already.

In several aforementioned heuristics for the $p$-hub median problem, instances of the FHSAP need to be solved when different subsets of hubs are fixed. And they are solved heuristically, given the complexity of the FHSAP. For example, the better one of two heuristics by O’Kelly [18] assigns a city to its nearest hub or the second nearest hub, which enumerates exponentially many allocation combinations. In Campbell’s paper [5], given an initial set of hub locations and flow information from the multiple allocations problem, the first heuristic assigns each city to the hub which routes its maximum flow, and the second heuristic assigns each city to a hub such that the total routing cost is minimized. Though the latter gives a tighter bound, it has to consider all possible single allocation combinations.

In this paper, we present a class of linear programming-based algorithms to tackle the FHSAP. Computational results show that our algorithms deliver high quality solutions that are very close to optimal. Further, our algorithms are capable of solving very large-scale problems in a reasonable amount of time. Equally important, we establish provable worst case bounds for our algorithms. We discuss our results in more details below.

The first step of our algorithms is to solve a linear programming (LP) relaxation of the FHSAP. A natural LP relaxation can be obtained from an LP formulation for the $p$-hub median problem suggested by Campbell [4]. This LP relaxation is extremely attractive. Skorin-Kapov et al. [23] improved this LP relaxation and reported the modified version was very tight and output
integral solutions automatically in 95% instances they tested. However, the size of the LP relaxation is relative large and restricts its applications to large-scale problems. Therefore, in order to solve large-scale problems, we also make use of the LP relaxation presented by Ernst and Krishnamoorthy [11, 12]. The size of this LP is significantly smaller than that in [23, 4]. We further modify this formulation by adding additional flow constraints, which delivers a better lower bound for the $FHSAP$. We consider all three LP relaxations. Although some relaxations are tighter than others, all these LPs often produce undesirable fractional solutions.

Therefore, the second step of our algorithms is to round fractional solutions to integral ones. The novelty of our algorithms is the introduction of a new type of randomized rounding method, which we call geometric rounding. Any optimal (fractional) solution of the LP relaxation falls in a simplex. By taking advantage of geometric properties of a simplex, we randomly round a fractional solution, which corresponds to a non-extreme point of the simplex, to an extreme point.

Our geometric rounding technique enables us to establish worst case bounds for our algorithms for certain LP relaxations. To the best our knowledge, no provable bound has been provided to any of the aforementioned heuristics. A polynomial-time $\rho$-approximation algorithm to a minimization problem is defined to be an algorithm that runs in polynomial time and outputs a solution with a cost at most $\rho(\geq 1)$ times the optimal cost. $\rho$ is called approximation ratio or performance guarantee. We show that our algorithms (based on two of the LP relaxations) have an approximation ratio of 2 for a special case of $k$-$FHSAP$ in which all hubs constitute an equilateral (i.e., distances between hubs are uniform). We also present a polynomial time algorithm for exactly solving a special case of $k$-$FHSAP$ in which all hubs are collinear (i.e., all hubs are on a single line). These two results imply an approximation ratio of 2 for solving general $k$-$FHSAP$ when $k = 3$ and lead to a data-dependent performance guarantee when $k \geq 4$ as well.

We consider the geometric rounding technique and its analysis our major contribution of this paper. We expect it will find more applications in designing efficient algorithms for solving other discrete optimization problems.

The results of the paper are organized as follows. In Section 2, we define the $FHSAP$ and present its linear programming relaxations. Section 3 presents our geometric rounding method and its analysis. In Section 4, we prove worst case bounds for our algorithms. Computational results are presented in section 5. In Section 6, we present an exact algorithm for a special case in which
hubs are collinear; we also discuss the implications of this algorithm in solving 3-FHSAP and the general FHSAP. Section 7 concludes the paper.

2. Problem Description and Formulation

This section defines the fixed-hub single allocation problem, reviews and modifies previously proposed mathematical programs. By the terminology of communication networks, the problem is to build a two-level network [9]; see figure 1. Hubs (airports, routers, concentrators, etc.) are transit nodes which route traffic. The network connecting hubs is called backbone network. Terminal nodes (cities, computers, etc.) are called access nodes, and they represent the origins and the destinations of the traffic. The model can be described as a backbone/tributary network design problem in which backbone networks are fully connected and tributary networks are star-shape.

In order to route the demands between two terminal nodes, the original node has to deliver all its demands to the hub it is assigned to. Then this hub sends them to the hub the destination node is assigned to (this step is skipped if both nodes are assigned to the same hub). Finally the destination node gets the demands from its hub. No direct routing between two terminal nodes is permitted. Two types of costs are counted: the cost of routing between terminal nodes and transit nodes and the cost of routing between transit nodes. There are often economies of scale for inter-hub traffic.
O’Kelly et al. [18] first formulated the uncapacitated single allocation p-hub median problem (USApHMP) as a quadratic integer program. We consider its adapted form for the FHSAP. Assume we are given a set of fixed hubs \( \mathcal{H} = \{1, 2, \ldots, k\} \) and a set of cities \( \mathcal{C} = \{1, 2, \ldots, n\} \). Directed demand \( d_{ij} \) to be routed from city \( i \) to city \( j \) is given. The distance from city \( i \) to hub \( s \) is \( c_{is} \), which is also called the per unit transportation cost. Similarly define \( c_{st} \) to be the distance from hub \( s \) to hub \( t \). Define \( \vec{x} = \{x_{i,s} : i \in \mathcal{C}, s \in \mathcal{H}\} \) to be the assignment variables. The quadratic formulation for the FHSAP is then

\[
\text{Problem FHSAP-QP}
\]

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j \in \mathcal{C}} d_{ij} \left( \sum_{s \in \mathcal{H}} c_{is} x_{i,s} + \sum_{t \in \mathcal{H}} c_{jt} x_{j,t} + \sum_{s,t \in \mathcal{H}} \alpha c_{st} x_{i,s} x_{j,t} \right) \\
\text{subject to} & \quad \sum_{s \in \mathcal{H}} x_{i,s} = 1, \quad \forall i \in \mathcal{C}, \\
& \quad x_{i,s} \in \{0, 1\}, \quad \forall i \in \mathcal{C}, s \in \mathcal{H}.
\end{align*}
\]

All coefficients \( d_{ij}, c_{is}, c_{jt}, c_{st} \geq 0 \), and \( c_{st} = c_{ts}, c_{ss} = 0, \forall i, j \in \mathcal{C}, \forall s, t \in \mathcal{H} \). \( \alpha \) is the discount factor and \( 0 \leq \alpha \leq 1 \). Without loss of generality, \( \alpha \) can be assumed to be one. Note that the transportation cost from cities to hubs, \( \sum_{i,j \in \mathcal{C}} d_{ij} \left( \sum_{s \in \mathcal{H}} c_{is} x_{i,s} + \sum_{t \in \mathcal{H}} c_{jt} x_{j,t} \right) \), is linear on \( \vec{x} \), we call it the linear cost of the objective function and denote it by \( L(\vec{x}) \). Similarly, call the other part of the objective function the inter-hub cost or quadratic cost, and denote it by \( Q(\vec{x}) \).

Campbell [4] linearized O’Kelly’s model by formulating an alternative MILP for USApHMP. Its adapted form for the FHSAP can be formulated as follows:

\[
\text{Problem FHSAP-MILP1}
\]

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j \in \mathcal{C}} \sum_{s,t \in \mathcal{H}} d_{ij} (c_{is} + c_{st} + c_{jt}) X_{ijst} \\
\text{subject to} & \quad \sum_{s,t \in \mathcal{H}} X_{ijst} = 1, \quad \forall i, j \in \mathcal{C}, \\
& \quad \sum_{t \in \mathcal{H}} X_{ijst} = x_{i,s}, \quad \forall i, j \in \mathcal{C}, s \in \mathcal{H}, \\
& \quad \sum_{s \in \mathcal{H}} X_{ijst} = x_{j,t}, \quad \forall i, j \in \mathcal{C}, t \in \mathcal{H}, \\
& \quad X_{ijst} \geq 0, \quad \forall i, j \in \mathcal{C}, s, t \in \mathcal{H}, \\
& \quad x_{i,s} \in \{0, 1\}, \quad \forall i \in \mathcal{C}, s \in \mathcal{H}.
\end{align*}
\]
Here $X_{ijst}$ is the portion of the flow from city $i$ to city $j$ via hub $s$ and $t$ sequentially. The formulation involves $O(n^2k^2)$ nonnegative variables and $O(n^2k)$ constraints. This formulation enables us to obtain an LP relaxation for the FHSAP by replacing the zero-one constraints with non-negative constraints. We will refer to this LP relaxation as FHSAP-LP1. As we have mentioned in the introduction, this LP relaxation is very tight and often produces integer solutions. However, the size of the LP relaxation is relative large, which restricts its applications to large-sized problems.

In order to reduce the time complexity, we consider a flow formulation for the FHSAP, which is adapted from a formulation for the USApHMP proposed by Ernst and Krishnamoorthy [11, 12]. In this formulation, we do not have to specify the route for a pair of cities $i$ and $j$, i.e., we do not need decision variable $X_{ijst}$. Instead, we define $\vec{Y} = \{Y_{st}^i : i \in \mathcal{C}, s, t \in \mathcal{H}, s \neq t\}$ where $Y_{st}^i$ is the total amount of the flow originated from city $i$ and routed from hub $s$ to a different hub $t$. Define $O_i = \sum_{j \in \mathcal{C}} d_{ij}$; $D_i = \sum_{j \in \mathcal{C}} d_{ji}$. Then the FHSAP can be bounded from below by Problem FHSAP-MILP2

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in \mathcal{C}} \sum_{s \in \mathcal{H}} c_{is}(O_i + D_i)x_{i,s} + \sum_{i \in \mathcal{C}} \sum_{s,t \in \mathcal{H}: s \neq t} c_{st}Y_{st}^i \\
\text{subject to} & \quad \sum_{s \in \mathcal{H}} x_{i,s} = 1, \quad \forall i \in \mathcal{C}, \quad (2) \\
& \quad \sum_{t \in \mathcal{H}: t \neq s} Y_{st}^i - \sum_{t \in \mathcal{H}: t \neq s} Y_{ts}^i = O_i x_{i,s} - \sum_{j \in \mathcal{H}} d_{ij}x_{j,s}, \quad \forall i \in \mathcal{C}, s \in \mathcal{H}, \quad (3) \\
& \quad x_{i,s} \in \{0, 1\}, \quad \forall i \in \mathcal{C}, s \in \mathcal{H}, \quad (4) \\
& \quad Y_{st}^i \geq 0, \quad \forall i \in \mathcal{C}, s, t \in \mathcal{H}, s \neq t. \quad (5)
\end{align*}
\]

Note that this modified formulation involves only $O(nk^2)$ nonnegative variables and $O(nk)$ linear constraints. In contrast to FHSAP-MILP1, the problem size is decreased by a factor $n$. We can then obtain an LP relaxation FHSAP-LP2 for the FHSAP from the formulation FHSAP-MILP2.

A given feasible assignment $\vec{x}$ to the FHSAP with the flow vector $\vec{Y}$ is always a feasible solution to FHSAP-MILP2. The value of objective function of FHSAP-MILP2 with this solution is equivalent to the transportation cost. Thus, FHSAP-MILP2 only provides a lower bound for the FHSAP, and there can be a strictly positive gap between the optimal value of FHSAP-MILP2 and that of the general FHSAP, as our simulation results indicate. However, it can be proved that FHSAP-MILP2 is an exact formulation of FHSAP when hubs in the network constitute an
equilateral.

It is possible to obtain a stronger LP relaxation than that of FHSAP-MILP2 by adding a set of valid constraints, which is particularly useful in deriving the worst case bound of our rounding algorithm.

**Lemma 1.** Let $\bar{x}$ and $\bar{Y}$ be defined as in Formulation FHSAP-MILP2. For any $i \in C$ and $s \in H$,

$$
\sum_{t \in H : t \neq s} Y^i_{st} + \sum_{t \in H : t \neq s} Y^i_{ts} = \sum_{j \in C} d_{ij} |x_{i,s} - x_{j,s}|.
$$

(6)

**Proof.** We verify equation 6 in two cases.

If $x_{i,s} = 0$, then

$$
\sum_{t \in H : t \neq s} Y^i_{st} + \sum_{t \in H : t \neq s} Y^i_{ts} = \sum_{t \in H : t \neq s} Y^i_{ts} = \sum_{j \in C} d_{ij} x_{j,s} = \sum_{j \in C} d_{ij} |x_{i,s} - x_{j,s}|.
$$

If $x_{i,s} = 1$, then

$$
\sum_{t \in H : t \neq s} Y^i_{st} + \sum_{t \in H : t \neq s} Y^i_{ts} = \sum_{t \in H : t \neq s} Y^i_{ts} = \sum_{j \in C, x_{j,s} = 0} d_{ij} (1 - x_{j,s}) = \sum_{j \in C} d_{ij} |x_{i,s} - x_{j,s}|.
$$

Therefore, equation 6 holds in both cases.

In view of Lemma 1, we obtain a strengthened LP relaxation for the FHSAP, which we call FHSAP-LP2'.

**Problem**

minimize $\sum_{i \in C} \sum_{s \in H} c_{is}(O_i + D_i)x_{i,s} + \sum_{i \in C} \sum_{s,t \in H : s \neq t} c_{st} Y^i_{st}$

subject to

$$
\sum_{s \in H} x_{i,s} = 1, \quad \forall i \in C,
$$

(8)

$$
\sum_{t \in H : t \neq s} Y^i_{st} - \sum_{t \in H : t \neq s} Y^i_{ts} = O_i x_{i,s} - \sum_{j \in H} d_{ij} x_{j,s}, \quad \forall i \in C, s \in H,
$$

(9)

$$
\sum_{t \in H : t \neq s} Y^i_{st} + \sum_{t \in H : t \neq s} Y^i_{ts} = \sum_{j \in H} d_{ij} |x_{i,s} - x_{j,s}| \quad \forall i \in C, s \in H
$$

$$
Y^i_{st}, x_{i,s} \geq 0, \quad i \in C, s, t \in H, s \neq t.
$$

(11)

Notice that if we sum up all the constraints generated from Lemma 1, we get a valid aggregate flow constraint:

$$
2 \sum_{i \in C} \sum_{s,t \in H : s \neq t} Y^i_{st} = \sum_{i,j \in C} \sum_{s \in H} d_{ij} |x_{i,s} - x_{j,s}|.
$$

(12)
Replace all constraints (10) in FHSAP-LP2’ by the single constraint (12), we get a new LP formulation. We refer to it as FHSAP-LP3. Considering the possible auxiliary variables to denote the absolute value of $x_{i,s} - x_{j,s}$ and the corresponding constraints, the numbers of variables and constraints in FHSAP-LP3 are both $O(n^2k + nk^2)$. Although it doesn’t reduce the size of the formulation of FHSAP-LP2’ significantly, computational results indicate that FHSAP-LP3 reduces the running time remarkably at the minor expense of the effectiveness of the algorithm. More importantly, FHSAP-LP3 is still sufficient for us to derive the worst case bound of our rounding algorithm.

In the next section, we discuss our rounding algorithm, that is, how to round fractional solutions of LP relaxations to integral ones.

3. Geometric Rounding

3.1 Rounding Procedure

Notice that a solution to the FHSAP can be completely defined by the assignment variable $\vec{x}$. After solving the LP relaxation FHSAP-LP1, FHSAP-LP2 or FHSAP-LP3, we only need to focus on rounding the fractional assignment variables to binary integers. Notice that, in all three relaxations presented above, for a terminal node $i$, any optimal solution $x_i = (x_{i,1}, \ldots, x_{i,k})$ on node $i$ must fall into a standard $k-1$ dimensional simplex:

$$\{w \in R^k | w \geq 0, \sum_{i=1}^{k} w_i = 1\}.$$  

We denote this simplex by $\Delta_k$.

Therefore, a fractional assignment vector on node $i$ corresponds to a non-vertex point in the simplex $\Delta_k$. Our goal is to round any fractional solution to a vertex point of $\Delta_k$, which is of the form:

$$(w \in R^k | w_i \in \{0, 1\}, \sum_{i=1}^{k} w_i = 1).$$

It is clear that $\Delta_k$ has exactly $k$ vertices. We denote the vertices of $\Delta_k$ by $v_1, v_2, \ldots, v_k$, where the $i$th coordinate of $v_i$ is 1.
Before presenting the rounding procedure, we will review some simple geometry concepts first. For a point \( x \in \Delta_k \), connect \( x \) with all vertices \( v_1, \ldots, v_k \) of \( \Delta_k \). Denote the polyhedron which exactly has vertices \( \{ x, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k \} \) by \( A_{x,i} \). Thus simplex \( \Delta_k \) can be partitioned into \( k \) polyhedrons \( A_{x,1}, \ldots, A_{x,k} \), and the interiors of any pair of these \( k \) polyhedrons do not intersect. Denote the volume of \( A_{x,i} \) by \( V_{x,i} \), and the volume of \( \Delta_k \) by \( V \).

We are now ready to present our randomized rounding algorithm. Notice that this rounding procedure is applicable to problems besides the FHSAP, as long as the feasible set of the problems is the set of vertices of a simplex.

**Geometric Rounding Algorithm (FHSAP-GRA):**

1. Solve an LP relaxation of the FHSAP: LP1, LP2 or LP3. And get an optimal solution \( \vec{x}^* \).
2. Generate a random vector \( u \), which follows a uniform distribution on \( \Delta_k \).
3. For each \( x^*_i = (x^*_i,1, \ldots, x^*_i,k) \), if \( u \) falls into \( A_{x^*_i,s} \), let \( \hat{x}_{i,s} = 1 \); other components \( \hat{x}_{i,t} = 0 \).

**Remark.** There are several direct methods to generate a uniform random vector \( u \) from the standard simplex \( \Delta_k \). One of them is to generate \( k \) independent unit-exponential random numbers \( a_1, \ldots, a_k \), i.e., \( a_i \sim \text{Exponential}(1) \). Then the vector \( u \), whose \( i \)th coordinate is defined as

\[
u_i = \frac{a_i}{\sum_{i=1}^{k} a_i},\]

Figure 2: By the geometric rounding method, \( \hat{x} = (1,0,0) \), \( \hat{y} = (0,0,1) \) as the graph indicates.
is uniformly distributed on $\Delta_k$.

Further, in the last step of our algorithm, we need to decide which polyhedron the generated point falls into. This can be easily done by observing the following fact.

**Lemma 2.** Given $w = (w_1, w_2, \ldots, w_k) \in \Delta_k$, vector $u$ in $\Delta_k$ is in the interior of polyhedron $A_{w,s}$ only if $s$ minimizes $\frac{w_l}{w_1}$, $1 \leq l \leq k$.

**Proof.** First, by symmetry we only need to discuss the case $s = 1$. If vector $u$ falls into polyhedron $A_{w,1}$, vector $u$ can be written as a convex combination of vertices of $A_{w,1}$. i.e., there exist nonnegative $\alpha_i$’s, such that $\sum_{i=1}^{k} \alpha_i = 1$ and $u = \alpha_1 w + \sum_{i=2}^{k} \alpha_i v_i$. It follows that

$$ u_1 = \alpha_1 w_1, \quad u_i = \alpha_1 w_i + \alpha_i, \forall i \geq 2. $$

Then, for each $i \geq 2$,

$$ \frac{u_i}{w_i} = \frac{\alpha_1 w_i + \alpha_i}{w_i} \geq \alpha_1 = \frac{u_1}{w_1}. $$

This completes the proof.

Thus, deciding which polyhedron the generated point falls into is an easy task if the index set $\arg \min_{1 \leq l \leq k} \{ \frac{w_l}{x_{i,l}} \}$ is a singleton. In case it is not, this can be done randomly, as it happens with probability zero if $u$ is generated uniformly at random.

### 3.2 Analysis of Geometric Rounding

In this subsection, we prove several properties of the geometric rounding procedure, which are useful in establishing the performance guarantee of our algorithms. We start with a simple fact regarding the volumes of a simplex $\Delta_k$ and the polyhedrons $A_{x,i}$.

**Lemma 3.** For any $w \in \Delta_k$ and any $i : 1 \leq i \leq k$, $V_{w,i}/V_k = w_i$. 


Proof. A linear transformation $T : \Delta_k \to A_{w,i}$ can be defined by an $n \times n$ matrix:

$$
\begin{pmatrix}
1 & 0 & \cdots & w_1 & 0 & \cdots & 0 \\
0 & 1 & & & & & \\
\vdots & & & & & & \\
0 & \cdots & w_i & & & & 1 \\
& & & & & & \\
0 & \cdots & w_n & & & & 1
\end{pmatrix}
$$

We denote the above matrix by $M$. According to the change of variables theorem \[13\], the volume ratio $V_{w,i}/V_k = |\det(M)| = w_i$.

Lemma 3 immediately implies a nice property regarding the expected value of the assignment variables.

**Theorem 4.** For any $i \in \mathcal{C}, l \in \mathcal{H}$, $E[\hat{x}_{i,l}] = x_{i,l}^*$.

**Proof.** For any $\hat{x}_{i,l}$, $E[\hat{x}_{i,l}] = \text{Prob}(\hat{x}_{i,l} = 1) = \text{Prob}(u \text{ falls into } A_{x_{i,l}^*})$.

On the other hand, since $u$ follows a uniform distribution on $\Delta_k$, the probability $u$ falls into $A_{x_{i,l}^*}$ is $V_{x_{i,l}^*}/V_k$. This fact, together with Lemma 3, implies that $E[\hat{x}_{i,l}] = x_{i,l}^*$. \(\square\)

The following theorem states that if two non-vertex points $x$ and $y$ are close in distance, then the rounded points $\hat{x}$ and $\hat{y}$ should not be too far from each other in expectation. One way to measure the distance of two points $x$ and $y$ is by the $l_1$ norm of $x - y$.

For any $x$ and $y$, define $d(x, y) := \sum s |x_s - y_s|$. Then we have

**Theorem 5.** For any $x, y \in \Delta_k$, randomly round $x$ and $y$ to vertices $\hat{x}$ and $\hat{y}$ in $\Delta_k$ by the procedure in FHSAP-GRA, then

$$E[d(\hat{x}, \hat{y})] \leq 2d(x, y).$$

**Proof.** Rather than proving the theorem directly, we prove an equivalent claim: For any $0 \leq m \leq k$, assume $x$ and $y$ have the same values on $m$ corresponding coordinates, then $E[d(\hat{x}, \hat{y})] \leq 2d(x, y)$. This claim can be proved by induction on $m$. \(12\)
If \( m = k \), then \( x = y \). It implies that \( d(x, y) = 0 \). On the other hand, with probability 1, \( x \) and \( y \) will be rounded to the same vertex. Therefore, \( E[d(\hat{x}, \hat{y})] = 0 \) as well. Thus, the desired claim holds in this case.

Assume the claim holds for \( m = k, k - 1, \cdots, m' + 1, m' \), where \( m' \geq 1 \). Now we consider the case that \( x \) and \( y \) have the same values on \( m = m' - 1 \) corresponding coordinates. Without loss of generality, assume \( \frac{x_1}{y_1} \geq \frac{x_2}{y_2} \geq \cdots \geq \frac{x_k}{y_k} \) (define \( \frac{x_k}{y_k} = +\infty \) if \( y_k = 0 \)). Because \( \sum x_i = \sum y_i = 1 \), \( x_i, y_i \geq 0 \), we must have \( \frac{x_i}{y_i} > \frac{x_k}{y_k} \) assuming \( x \neq y \).

We first consider the case in which both \( x_k \) and \( y_1 \) are nonzero. For any \( s : 0 < s < 1 \) let
\[
t = s + \frac{1 - s}{y_1}, \quad r = s + \frac{1 - s}{x_k}.\]
Further, we define two new points
\[
x(s) = (x_1 s, x_2 s, \cdots, x_{k-1} s, x_k r),
\]
\[
y(s) = (y_1 t, y_2 s, \cdots, y_{k-1} s, y_k s).
\]
Notice that \( s = 0 \) implies \( x_1 s < y_1 t \), \( s = 1 \) implies \( x_1 s > y_1 t \), and \( r, t \) increases as \( s \) decreases, so there exists \( 0 < s < 1 \), such that \( x_1 s = y_1 t \). Similarly, we can find \( 0 < s' < 1 \), such that \( x_k r' = y_k s' \). In the following proof, assume \( s \geq s' \) (the case \( s \leq s' \) can be handled similarly).

Then we know \( x_1 s = y_1 t; x_k r \leq x_k r' = y_k s' \leq y_k s \). This implies that \( x(s) \) and \( y(s) \) have the same values on \( m' \) corresponding coordinates.

Now we are ready to bound \( E[d(\hat{x}, \hat{y})] \). First, by the triangle inequality in \( l_1 \) metric,
\[
E[d(\hat{x}, \hat{y})] \leq E[d(\hat{x}, \hat{x}(s)) + d(\hat{x}(s), \hat{y}(s)) + d(\hat{y}(s), \hat{y})].
\]
From Lemma 6 below, we can show that \( E[d(\hat{x}, \hat{x}(s))] = d(x, x(s)) \), and \( E[d(\hat{y}, \hat{y}(s))] = d(y, y(s)) \). Further, by the assumption of the induction, and by the fact that \( x(s) \) and \( y(s) \) have the same values on \( m' \) corresponding coordinates, we know that
\[
E[d(\hat{x}(s), \hat{y}(s))] \leq 2d(x(s), y(s)).
\]
Therefore, in order to show \( E[d(\hat{x}, \hat{y})] \leq 2d(x, y) \), it is sufficient to prove the inequality:
\[
d(x, x(s)) + 2d(x(s), y(s)) + d(y, y(s)) \leq 2d(x, y),
\]
Figure 3: $v_k, x(s), x$ are collinear.

or

$$d(x, x(s)) + d(y, y(s)) \leq 2d(x, y) - 2d(x(s), y(s)).$$

By the definition of $d(x, y)$, the above inequality is equivalent to

$$2(r - 1)x_k + 2(t - 1)y_1 \leq 2((x_1 - y_1) + (y_k - x_k) - (x_1 s - y_1 t)),$$

which can be further reduced to the following trivial inequality

$$0 \leq (1 - s)(x_1 + y_k + \sum_{i=2}^{k-1} |x_i - y_i|).$$

This completes the proof for the case both $x_k$ and $y_1$ are non-zero.

If $x_k$ (or $y_1$ or both) is 0, replace the $x_k r$ (or $y_1 t$ or both) in the proof above with $1 - s$. The proof is similar.

Our proof of Theorem 5 has used a fact that is formalized in the following Lemma.

**Lemma 6.** Assume $x, x(s) \in \Delta_k$, and $x(s) = (sx_1, sx_2, \ldots, sx_{k-1}, sx_k + (1 - s)), 0 < s < 1$, then

$$E[d(\hat{x}, \hat{x}(s))] = d(x, x(s)).$$

**Proof.** First, we prove that $d(\hat{x}, \hat{x}(s))$ is non-zero if and only if $\hat{x}(s) = v_k$ but $\hat{x} \neq v_k$.  

14
Given random vector $u$ in $\Delta_k$, for each $1 \leq i \leq k - 1$, if $u \in A_{x(i)}$, then $u$ is a convex combination of $x(s), v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k$. Notice that $x(s) = sx + (1 - s)v_k$, so $u$ is also a convex combination of a convex combination of $x, v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_k$.

This implies that $u \in A_{x,i}$, and thus $A_{x(s),i} \subset A_{x,i}$. Thus, for $1 \leq i \leq k - 1$, if $\hat{x}(s) = v_i$, then with probability one, $\hat{x} = v_i = \hat{x}(s)$. Therefore, $d(\hat{x}, \hat{x}(s))$ is non-zero if and only if $\hat{x}(s) = v_k$ but $\hat{x} \neq v_k$.

Second, we claim that $A_{x,k} \subseteq A_{x(s),k}$. Then it follows that, the case that $\hat{x}(s) = v_k$ but $\hat{x} \neq v_k$ happens if and only if random vector $u$ falls into region $A_{x(s),k} - A_{x,k}$ (see Figure 3 for an illustration).

We prove this claim by verifying that $x$ can be written as a convex combination of $x(s)$ and $v_1, \ldots, v_{k-1}$.

The combination coefficients ($\alpha_1, \alpha_2, \ldots, \alpha_k$) are defined as follows: $\alpha_k = \frac{x_k}{sx_k + (1 - s)}$, and $\alpha_i = (1 - \alpha_k)sx_i$ for other $1 \leq \alpha_i \leq k - 1$.

We can see that $0 \leq \alpha_k \leq 1$ because $0 \leq x_k \leq 1$; and $0 \leq \alpha_i \leq 1$ for all $1 \leq i \leq k - 1$.

Furthermore, $\sum_{i=1}^{k} \alpha_i = (1 - \alpha_k)\sum_{i=1}^{k-1} x_i + \alpha_k = (1 - \alpha_k)(1 - x_k) + \alpha_k = 1$; and $x = \sum_{i=1}^{k-1} \alpha_i v_i + \alpha_k x(s)$.

So $x \in A_{x(s),k}$. Thus $A_{x,k} \subseteq A_{x(s),k}$.

Therefore

$$
Prob(d(\hat{x}, \hat{x}(s)) \neq 0) = \frac{V_{x(s),k} - V_{x,k}}{V_k} = \frac{x(s)_k - x_k}{x_k} = (1 - s)(1 - x_k).
$$

The third equality holds because of Lemma 3. Notice that, if $d(\hat{x}, \hat{x}(s)) \neq 0$, then $d(\hat{x}, \hat{x}(s)) = 2$. Thus,

$$
E[d(\hat{x}, \hat{x}(s))] = 2 * \frac{\theta}{\beta} = 2 \frac{1 - \alpha}{\alpha} = 2(1 - s)(1 - x_k).
$$
On the other hand, by definition,
\[ d(x, x(s)) = (1 - s) \sum_{i=1}^{k-1} x_i + (r - 1)x_k = 2(1 - s)(1 - x_k). \]

The lemma follows. \[\square\]

To end this subsection, we emphasize that the main results, i.e., Theorem 4 and 5 hold regardless of which LP relaxation we solve.

4. Worst-Case Analysis

We estimate the performance guarantee of FHSAP-GRA in this section. We will always assume the discount factor \(\alpha = 1\) for the convenience of the discussion because all theoretical analysis in this paper will hold for different \(\alpha\)’s.

Our goal is to bound the expected value of
\[ \sum_{i,j \in C} d_{ij} \left( \sum_{s \in H} c_{is} \hat{x}_{i,s} + \sum_{t \in H} c_{jt} \hat{x}_{j,t} + \sum_{s,t \in H} \alpha c_{st} \hat{x}_{i,s} \hat{x}_{j,t} \right). \]

Recall that
\[ L(\hat{x}) = \sum_{i,j \in C} d_{ij} \left( \sum_{s \in H} c_{is} \hat{x}_{i,s} + \sum_{t \in H} c_{jt} \hat{x}_{j,t} \right) \]
and
\[ Q(\hat{x}) = \sum_{i,j \in C} d_{ij} \sum_{s,t \in H} \alpha c_{st} \hat{x}_{i,s} \hat{x}_{j,t}. \]

We first focus on a special case in which the subgraph of hubs is an equilateral, i.e., distances between hubs are uniform. Without loss of generality, assume \(c_{st} = 2\) for any two different hubs \(s\) and \(t\). In this case, since \(\hat{x}\) is a feasible solution to the FHSAP,
\[ \sum_{s,t \in H} c_{st} \hat{x}_{i,s} \hat{x}_{j,t} = 2 \sum_{s,t \in H: s \neq t} \hat{x}_{i,s} \hat{x}_{j,t} = 2(1 - \sum_{s \in H} \hat{x}_{i,s} \hat{x}_{j,s}) = \sum_{s \in H} |\hat{x}_{i,s} - \hat{x}_{j,s}| = d(\hat{x}_i, \hat{x}_j). \]
Thus, in this special case,

\[ Q(\hat{x}) = \sum_{i,j \in C} d_{ij} d(\hat{x}_i, \hat{x}_j). \]

### 4.1 Bounds With Respect to \textit{FHSAP-LP1}

In this subsection, we assume that the LP relaxation \textit{FHSAP-LP1} is used in algorithm \textit{FHSAP-GRA}. We further assume that \((\hat{x}^*, \hat{X}^*)\) is an optimal solution to \textit{FHSAP-LP1}. Our main result of this subsection is summarized in the following Theorem.

**Theorem 7.** Assume that \(c_{st} = 2\) for all \(s \neq t\), then

\[
E[L(\hat{x})] + E[Q(\hat{x})] \leq \sum_{i,j \in C} \sum_{s,t \in H} d_{ij} (c_{is} + c_{jt}) X^*_{ijst} + 2 \sum_{i,j \in C} \sum_{s,t \in H} d_{ij} c_{st} X^*_{ijst}.
\]

**Proof.** From Theorem 4, we know that

\[
E[L(\hat{x})] = \sum_{i,j \in C} \sum_{s \in H} d_{ij} \left( \sum_{t \in H} c_{is} \hat{x}_{i,s} + \sum_{t \in H} c_{jt} \hat{x}_{j,t} \right) = \sum_{i,j \in C} \sum_{s \in H} \sum_{t \in H} c_{is} x^*_{i,s} + \sum_{t \in H} c_{jt} x^*_{j,t}.
\]

From the constraints of \textit{FHSAP-LP1}, \(x^*_{i,s} = \sum_{t \in H} X^*_{ijst}\) and \(x^*_{j,t} = \sum_{s \in H} X^*_{ijst}\).

Thus,

\[
E[L(\hat{x})] = \sum_{i,j \in C} \sum_{s \in H} \sum_{t \in H} d_{ij} \left( \sum_{s \in H} c_{is} x^*_{i,s} + \sum_{t \in H} c_{jt} x^*_{j,t} \right)
\]

\[
= \sum_{i,j \in C} \sum_{s \in H} \sum_{t \in H} d_{ij} \left( \sum_{s \in H} c_{is} \sum_{t \in H} X^*_{ijst} + \sum_{t \in H} c_{jt} \sum_{s \in H} X^*_{ijst} \right)
\]

\[
= \sum_{i,j \in C} \sum_{s,t \in H} d_{ij} (c_{is} + c_{jt}) X^*_{ijst}.
\]

Further,

\[
E[Q(\hat{x})] = \sum_{i,j \in C} d_{ij} E[d(\hat{x}_i, \hat{x}_j)]
\]

\[
\leq 2 \sum_{i,j \in C} d_{ij} (x^*_{i,s}, x^*_{j,s})
\]

\[
= 2 \sum_{i,j \in C} d_{ij} \sum_{s \in H} |x^*_{i,s} - x^*_{j,s}|,
\]

where the inequality holds because of Theorem 5. Further,

\[
x^*_{i,s} - x^*_{j,s} = \sum_{t \in H} X^*_{ijst} - \sum_{t \in H} X^*_{ijts} = \sum_{t \in H:t \neq s} (X^*_{ijst} - X^*_{ijts}).
\]
Thus,

\[ \sum_{s \in H} |x_{i,s}^* - x_{j,s}^*| \leq \sum_{s \in H} \sum_{t \in H: t \neq s} (X_{ijst}^* + X_{ijts}^*), \]

which implies that

\[ E[Q(\hat{x})] \leq 2 \sum_{i,j \in C} d_{ij} \sum_{s \in H} \sum_{t \in H: t \neq s} (X_{ijst}^* + X_{ijts}^*) = 4 \sum_{i,j \in C} \sum_{s \in H: s \neq t} d_{ij} X_{ijst}^*, \]

This completes the proof. \( \square \)

### 4.2 Bounds with Respect to FHSAP-LP3

In this subsection, we assume that the LP relaxation FHSAP-LP3 is used in algorithm FHSAP-GRA. We further assume that \((\bar{x}^*, \bar{Y}^*)\) is an optimal solution. Although FHSAP-LP3 has less variables and less constraints than FHSAP-LP1, we can still prove a bound that is similar to Theorem 7.

**Theorem 8.** Assume that \(c_{st} = 2\) for all \(s \neq t\), then

\[ E[L(\hat{x})] + E[Q(\hat{x})] \leq \sum_{i \in C} \sum_{s \in H: s \neq t} c_{is}(O_i + D_i)x_{i,s}^* + 2 \sum_{i \in C} \sum_{s,t \in H: s \neq t} c_{st}Y_{st}^{is}. \]

**Proof.** The proof is similar to that of Theorem 7. First,

\[ E[L(\hat{x})] = \sum_{i,j \in C} d_{ij}(\sum_{s \in H} c_{is}x_{i,s}^* + \sum_{t \in H} c_{jt}x_{j,t}^*) = \sum_{i \in C} \sum_{s \in H} c_{is}(O_i + D_i)x_{i,s}^*. \]

Second,

\[ E[Q(\hat{x})] = E[\sum_{i,j \in C} \sum_{s \in H} |\hat{x}_{i,s} - \hat{x}_{j,s}|] \leq \sum_{i,j \in C} \sum_{s \in H} d_{ij} (2|x_{i,s}^* - x_{j,s}^*|) = 2 \sum_{i \in C} \sum_{s,t \in H: s \neq t} c_{st}Y_{st}^{is}, \]

where the last equality follows from the aggregate flow constraint of FHSAP-LP3. This completes the proof. \( \square \)

**Remark 2.** Notice that the LP relaxation FHSAP-LP2’ has individual flow constraints from lemma 1, it is a stronger LP formulation. So the inequality above in Theorem 8 holds for GRA-LP2’ as well.
4.3 Performance Guarantee

Theorem 7 and 8 immediately imply that the algorithm \textit{FHSAP-GRA} has a performance guarantee of 2 when the subgraph of hubs is an equilateral. In fact, the approximation ratio on the inter-hub cost is at most 2, and the expected value of the city-to-hub cost is the same as that in the LP relaxation. Therefore, the performance guarantee of our algorithm can be improved depending on the ratio of the city-to-hub cost relative to the inter-hub cost.

Now we state our main theorem regarding the performance guarantee of algorithm \textit{FHSAP-GRA} for the general \textit{FHSAP}.

Define \( L = \max\{c_{st} : s,t \in H, s \neq t\} \), and \( l = \min\{c_{st} : s,t \in H, s \neq t\} \). Further, let \( r = \frac{L}{l} \), i.e., \( r \) is the ratio of the longest edge to the shortest edge among all inter-hub edges.

\textbf{Theorem 9.} The algorithm \textit{FHSAP-GRA} using the LP relaxation \textit{FHSAP-LP1} or \textit{FHSAP-LP3} has a performance guarantee of \( 2r \).

\textit{Proof.} Given an instance \( I \) of the \textit{FHSAP}, we build another instance denoted by \( I_L \), in which all of the inter-hub edges have the uniform length \( L \). Thus, the subgraph of hubs is an equilateral for instance \( I_L \). Let \( LP_I \) and \( LP_{IL} \) denote the optimal objective value of the LP relaxations of instance \( I \) and \( I_L \), respectively.

It is clear that
\[
LP_{IL} \geq LP_I \geq \frac{1}{r} LP_{IL}.
\]

Further, the expected cost of the solution generated by \textit{FHSAP-GRA} for instance \( I \) should be no more than the expected cost of the solution generated by \textit{FHSAP-GRA} for instance \( I_L \), which is at most
\[
2LP_{IL} \leq 2rLP_I,
\]
where the factor of 2 comes from Theorem 7 and 8.

We would like to point out that the ratio \( 2r \) is a worst case bound. The ratio is relative small when \( r \) is small. If a network is constructed in a way so that \( r \) is small, then even the worst case performance of our algorithm will not be too bad. In next section, we implemented our algorithm. The computational results suggest that it delivers solutions that are very close to the optimal ones.
Table 1: $n=50$, $k=5$.

<table>
<thead>
<tr>
<th>Discount Distribution</th>
<th>GRA-LP1</th>
<th></th>
<th>GRA-LP2</th>
<th></th>
<th>GRA-LP3</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>Gap1</td>
<td>CPU</td>
<td>Gap1</td>
<td>CPU</td>
<td>Gap2</td>
</tr>
<tr>
<td>$\alpha = 0.05$</td>
<td>U[0,20]</td>
<td>2.16</td>
<td>0.00%</td>
<td>0.03</td>
<td>0.00%</td>
<td>0.18</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>2.4</td>
<td>0.00%</td>
<td>0.02</td>
<td>0.00%</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td>2.88</td>
<td>0.00%</td>
<td>0.02</td>
<td>0.00%</td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td>2.14</td>
<td>0.00%</td>
<td>0.02</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
<tr>
<td>$\alpha = 0.25$</td>
<td>U[0,20]</td>
<td>2.59</td>
<td>0.00%</td>
<td>0.03</td>
<td>0.48%</td>
<td>7.28</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>2.79</td>
<td>0.00%</td>
<td>0.03</td>
<td>0.19%</td>
<td>4.18</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td>3.22</td>
<td>0.00%</td>
<td>0.04</td>
<td>0.31%</td>
<td>1.55</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td>3.36</td>
<td>0.00%</td>
<td>0.04</td>
<td>0.27%</td>
<td>0.27</td>
</tr>
<tr>
<td>$\alpha = 0.5$</td>
<td>U[0,20]</td>
<td>3.03</td>
<td>0.00%</td>
<td>0.03</td>
<td>0.77%</td>
<td>11.34</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>3.1</td>
<td>0.07%</td>
<td>0.03</td>
<td>0.09%</td>
<td>5.81</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td>3.71</td>
<td>0.00%</td>
<td>0.05</td>
<td>1.61%</td>
<td>1.61</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td>3.1</td>
<td>0.00%</td>
<td>0.04</td>
<td>9.25%</td>
<td>9.25</td>
</tr>
<tr>
<td>$\alpha = 1$</td>
<td>U[0,20]</td>
<td>3.3</td>
<td>0.00%</td>
<td>0.04</td>
<td>4.24%</td>
<td>12.19</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>3.08</td>
<td>0.00%</td>
<td>0.04</td>
<td>1.83%</td>
<td>1.83</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td>2.55</td>
<td>0.00%</td>
<td>0.04</td>
<td>4.47%</td>
<td>4.47</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td>2.04</td>
<td>0.00%</td>
<td>0.04</td>
<td>0.00%</td>
<td>0.00</td>
</tr>
</tbody>
</table>

5. Computational Results

Computational results for the implementation of $FHSAP$-$GRA$ are reported in this section. We applied $FHSAP$-$GRA$ to both randomly generated instances of three different sizes (Table 1, 2, 3, 4) and a benchmark problem data set (Table 5). All linear programs in the experiments were solved by CPLEX version 9.0 at a Stanford workstation (CPU: dual 3GHz/ memory: 8GB), and the rounding procedures were conducted on a notebook (CPU: Pentium 1.5GHz/memory: 1.0GB).

In all randomly generated examples, demands between cities are uniformly distributed on the interval $[0, 100]$ and all hub-to-city distances are uniformly distributed on the interval $[1, 11]$. We kept altering the distribution interval of inter-hub distances and the discount factor $\alpha$.

The benchmark problem set we used is called $AP$($Australia Post$) data set (Table 5) [11], which was collected from a real postal delivery network in Australia. It stores the coordinates and demands of 200 nodes (cities). Ernst and Krishnamoorthy solved $p$-hub location problems for $AP$ data set, and we tested our algorithms on hubs their solutions specified. Some of the hub-to-city cost coefficients are non-symmetric in the $AP$ data set, so we made adjustment to it accordingly.

In all the experiments, we run the rounding procedure 5000 times for those instances whose LP relaxations only have fractional optimal solutions. Considering that the running time of the algorithm is mainly spent in solving linear programs, CPU times reported in all tables are the
Table 2: n=100, k=10.

<table>
<thead>
<tr>
<th>Discount Distribution</th>
<th>GRA-LP1</th>
<th>GRA-LP2</th>
<th>GRA-LP3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>Gap1</td>
<td>CPU</td>
</tr>
<tr>
<td>α = 0.05</td>
<td>U[0,20]</td>
<td></td>
<td>768</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
<td>583</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
<td>567</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
<td>712</td>
</tr>
<tr>
<td>α = 0.25</td>
<td>U[0,20]</td>
<td></td>
<td>1115</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
<td>1329</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
<td>2276</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
<td>1318</td>
</tr>
<tr>
<td>α = 0.5</td>
<td>U[0,20]</td>
<td></td>
<td>1567</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
<td>9802</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
<td>9972</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
<td>10103</td>
</tr>
<tr>
<td>α = 1</td>
<td>U[0,20]</td>
<td></td>
<td>15249</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
<td>16851</td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
<td>15439</td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
<td>13780</td>
</tr>
</tbody>
</table>

Table 3: n=200, k=10.

<table>
<thead>
<tr>
<th>Discount Distribution</th>
<th>GRA-LP2</th>
<th>GRA-LP3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>Gap2</td>
</tr>
<tr>
<td>α = 0.05</td>
<td>U[0,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
</tr>
<tr>
<td>α = 0.25</td>
<td>U[0,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
</tr>
<tr>
<td>α = 0.5</td>
<td>U[0,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
</tr>
<tr>
<td>α = 1</td>
<td>U[0,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[14,20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>U[20,20]</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: n=1000, k=10.

<table>
<thead>
<tr>
<th>Discount Distribution</th>
<th>GRA-LP2</th>
<th>Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU</td>
<td>GRA</td>
</tr>
<tr>
<td>α = 1</td>
<td>U[0,20]</td>
<td>339820</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>538346</td>
</tr>
<tr>
<td></td>
<td>U[2,20]</td>
<td>274165</td>
</tr>
<tr>
<td>α = 0.5</td>
<td>U[0,20]</td>
<td>371262</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>260932</td>
</tr>
<tr>
<td></td>
<td>U[2,20]</td>
<td>270349</td>
</tr>
<tr>
<td>α = 0.25</td>
<td>U[0,20]</td>
<td>210725</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>213948</td>
</tr>
<tr>
<td>α = 0.05</td>
<td>U[2,20]</td>
<td>210725</td>
</tr>
<tr>
<td></td>
<td>U[4,20]</td>
<td>213948</td>
</tr>
</tbody>
</table>
Table 5: AP benchmark problems.

<table>
<thead>
<tr>
<th>n</th>
<th>k</th>
<th>Optimal</th>
<th>GRA-LP3</th>
<th>CPU Gap1</th>
<th>GRA-LP3</th>
<th>CPU Gap1</th>
<th>GRA-LP2</th>
<th>CPU Gap1</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>5</td>
<td>132367</td>
<td>132122</td>
<td>6.94</td>
<td>0.004%</td>
<td>132120</td>
<td>132372</td>
<td>0.02</td>
</tr>
<tr>
<td>50</td>
<td>4</td>
<td>143378</td>
<td>143200</td>
<td>4.04</td>
<td>0.000%</td>
<td>143139</td>
<td>143378</td>
<td>0.01</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>158570</td>
<td>158473</td>
<td>1.92</td>
<td>0.000%</td>
<td>158139</td>
<td>158570</td>
<td>0.01</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>134265</td>
<td>133938</td>
<td>2.17</td>
<td>0.000%</td>
<td>133908</td>
<td>134265</td>
<td>0.02</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>143969</td>
<td>143924</td>
<td>1.16</td>
<td>0.000%</td>
<td>143707</td>
<td>143969</td>
<td>0.01</td>
</tr>
<tr>
<td>40</td>
<td>3</td>
<td>158831</td>
<td>158831</td>
<td>0.60</td>
<td>0.000%</td>
<td>158642</td>
<td>158831</td>
<td>0.01</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>123574</td>
<td>123574</td>
<td>0.23</td>
<td>0.000%</td>
<td>123574</td>
<td>123574</td>
<td>0.01</td>
</tr>
<tr>
<td>25</td>
<td>4</td>
<td>139197</td>
<td>138727</td>
<td>0.17</td>
<td>0.000%</td>
<td>138727</td>
<td>139316</td>
<td>0.01</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>155256</td>
<td>155139</td>
<td>0.09</td>
<td>0.000%</td>
<td>154786</td>
<td>155256</td>
<td>0.01</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>123130</td>
<td>122333</td>
<td>0.11</td>
<td>0.000%</td>
<td>122329</td>
<td>123130</td>
<td>0.01</td>
</tr>
<tr>
<td>20</td>
<td>4</td>
<td>135625</td>
<td>134833</td>
<td>0.08</td>
<td>0.000%</td>
<td>134827</td>
<td>135625</td>
<td>0.01</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>151533</td>
<td>151515</td>
<td>0.05</td>
<td>0.000%</td>
<td>150724</td>
<td>151533</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>91105</td>
<td>89962</td>
<td>0.02</td>
<td>0.000%</td>
<td>89961</td>
<td>91105</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>112396</td>
<td>111605</td>
<td>0.01</td>
<td>0.000%</td>
<td>111321</td>
<td>112396</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>3</td>
<td>136008</td>
<td>135938</td>
<td>0.01</td>
<td>0.000%</td>
<td>135223</td>
<td>136008</td>
<td>0.01</td>
</tr>
</tbody>
</table>

running times for solving the LP relaxation of each instance.

Table 1 describes medium-sized examples, each of which has 50 cities and 5 hubs. Table 2 describes large-sized examples, each of which has 100 cities and 10 hubs. Table 1 and 2 present computational results for 32 instances by FHSAP-GRA with three different LP relaxations: LP1, LP2, and LP3. The running times and percentage gaps are given for each instance.

Denote algorithm FHSAP-GRA with the LP relaxation LPi by GRA-LPi (i = 1, 2, 3). For each algorithm GRA-LPi, denote the optimal objective value of the LP relaxation LPi by LPi, and denote the value of an integral solution by algorithm GRA-LPi by GRAi. Recall that LP1 is known to be a very tight lower bound, we define Gap1 = (GRAi - LPi) * 100% to measure the solution quality of each GRAi. Similarly we define Gap2 = (GRAi - LP3) * 100% considering that it is difficult to calculate LP1 when the problem size becomes large.

The computational results in table 1 and 2 show that FHSAP-GRA with different LP relaxations delivers solutions with variable qualities and time complexity. We have the following observations. They are compatible with the literature and analysis developed in this article.

The first, LP1 is a very tight lower bound. It automatically generates optimal integral assignments for 30 out of 32 instances. Furthermore, GRA-LP1 gives near optimal solutions for the remaining 2 instances. However, the running time increases rapidly to an intractable level when the problem size is increased to (100, 10) and the discount factor approaches 1. Therefore, GRA-LP1
is especially efficient for medium-sized problems.

The second, $GRA_2$ is of particular value in real applications for large-sized problems. We can observe that the much smaller running time comes at the expense of marginally larger gaps. It performs at most 1% worse than optimal assignments on 22 out of 32 instances comparing to $GRA-LP1$. Results also reveal that the effectiveness of the solutions by $GRA-LP2$ decreases as the discount factor gets larger.

The third, $GRA-LP3$ delivers high-quality solutions for large-sized problem in a reasonable amount of time. It generates solutions at most 1% worse than optimal ones on 28 out of 32 instances. And it outperforms $GRA-LP2$ on 15 out of 32 instances and is only inferior to $GRA-LP2$ on 1 instance. Moreover, $GRA-LP3$ always performs extremely well on instances where the graph of hubs has a (near) equilateral structure. We also observed that $LP_3$ is a tighter lower bound than $LP_2$. It improves $LP_2$ on 19 out of 32 instances.

For larger problems in table 3 we didn’t attempt to compute their tight lower bounds $LP_1$ due to the excessive running time. However, $GRA-LP3$ is still manageable at this size, which provides us a good lower bound $LP_3$ in most instances. There is one example in which both $GRA-LP2$ and $GRA-LP3$ have large values of $GAP2$. It is caused more possibly by the looseness of the lower bound $LP_2$ rather than by the algorithm itself.

For very large-sized problems in table 4, we only implemented $GRA-LP2$ because of its infeasible demands on memory. We reported the running times of these problems, the lower bounds from $FHSAP-LP1$ and the costs of rounded solutions. We also presented upper bounds derived from choosing the better one of two commonly used quick heuristics: the nearest neighborhood allocation heuristic and one-hub allocation heuristic. The former assigns every city to its nearest hub and the later assigns all cities to one single hub. We can observe that $GRA-LP2$ outperforms them on 6 out of 7 instances.

In table 5, we tested 15 $AP$ benchmark problems by $GRA-LP1$, $GRA-LP2$ and $GRA-LP3$ on fixed-hubs specified in their paper. Since solving $FHSAP-LP1$ already produced optimal integral assignments for all 15 problems in less than 120 seconds, we omitted it in the table. $GRA-LP3$ obtained optimal assignments on 14 out of 15 problems, and only 0.004% higher than the optimal cost on the remaining one, with much less time than $GRA-LP1$. $GRA-LP2$ is the fastest algorithm,
and generated optimal assignments on 13 of 15 problems. It performed 0.004% or 0.09% worse than the optimal cost on the remaining two problems.

6. Further Results

6.1 Polynomial-Time Solvable Case: Collinear Hubs

In this subsection we establish a polynomial-time solvable case in which the subgraph of hubs is degenerated to a line, i.e., all hubs are collinear. In this case, we can sort hubs and denote them by 1, 2, \ldots, k. See figure 4. For any two hubs s, t, s < t, we have \( c_{s,t} = \sum_{l=s}^{t-1} c_{l,l+1} \) by assumption.

The inter-hub cost in objective function can be written as:

\[
Q(\bar{x}) = \sum_{i,j \in C} d_{ij} \sum_{s=1}^{k-1} c_{s,s+1} \left[ \sum_{l=1}^{s} x_{i,l} \left( 1 - \sum_{t=s+1}^{k} x_{j,t} \right) + \sum_{l=1}^{s} x_{j,l} \left( 1 - \sum_{t=s+1}^{k} x_{i,t} \right) \right].
\]

\( FHSAP-QP \) can be converted to an equivalent mathematical program:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j \in C} d_{ij} \sum_{s=1}^{k-1} c_{s,s+1} \left[ \sum_{l=1}^{s} x_{i,l} \left( 1 - \sum_{t=1}^{s} x_{j,t} \right) + \sum_{l=1}^{s} x_{j,l} \left( 1 - \sum_{t=1}^{s} x_{i,t} \right) \right] + L(\bar{x}) \\
\text{subject to} & \quad \sum_{l \in H} x_{i,l} = 1, \quad \forall i \in C, \\
& \quad x_{i,l} \in \{0,1\}, \quad \forall i \in C, l \in H.
\end{align*}
\]

Now for each \( i \in C, s \in H, \) define \( X_{i,s} = \sum_{l=1}^{s} x_{i,l} \). The problem can be formulated equivalently...
as follows:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j \in \mathcal{C}} \sum_{s=1}^{k-1} d_{ij} c_{s,s+1} (X_{i,s}(1 - X_{j,s}) + X_{j,s}(1 - X_{i,s})) + L(\bar{x}) \\
\text{subject to} & \quad \sum_{l=1}^{k} x_{i,l} = 1, \quad \forall i \in \mathcal{C}, \\
& \quad X_{i,s} = \sum_{l=1}^{s} x_{i,l}, \quad \forall i \in \mathcal{C}, s \in \mathcal{H}, \\
& \quad x_{i,l}, X_{i,s} \in \{0,1\}, \quad \forall i \in \mathcal{C}, l, s \in \mathcal{H}.
\end{align*}
\]

Relax it to the following linear program:

\[
\begin{align*}
\text{minimize} & \quad \sum_{i,j \in \mathcal{C}} \sum_{s=1}^{k-1} d_{ij} c_{s,s+1} (X_{i,s} + X_{j,s} - 2X_{i,j,s}) + L(\bar{x}) \\
\text{subject to} & \quad \sum_{l=1}^{k} x_{i,l} = 1, \quad \forall i \in \mathcal{C}, \\
& \quad X_{i,s} = \sum_{l=1}^{s} x_{i,l}, \quad \forall i \in \mathcal{C}, s \in \mathcal{H}, \\
& \quad X_{i,j,s} \leq X_{i,s}, \quad \forall i, j \in \mathcal{C}, s \in \mathcal{H}, \\
& \quad X_{i,j,s} \leq X_{j,s}, \quad \forall i, j \in \mathcal{C}, s \in \mathcal{H}, \\
& \quad x_{i,l}, X_{i,s}, X_{i,j,s} \geq 0, \quad \forall i, j \in \mathcal{C}, l, s \in \mathcal{H}.
\end{align*}
\]

Now we present our randomized algorithm for the collinear-hubs case of the FHSAP, where the rounding procedure is inspired by Bertsimas et al. [1].

**Collinear-Hubs Algorithm for the FHSAP (FHSAP-CHA)**

1. Formulate the linear program relaxation as above; solve it to get an optimal solution \((\bar{x}^*, \bar{X}^*)\).
2. Uniformly generate a real number \(\rho\) on the interval \([0, 1]\), i.e., \(\rho \in U[0, 1]\).
3. For each \(i \in \mathcal{C}\), if \(X_{i,s}^* \leq \rho < X_{i,s+1}^*\), let \(\hat{x}_{i,s+1} = 1\), other \(\hat{x}_{i,l} = 0\), and determine \(\hat{X}_{i,l}\) accordingly.

This rounding generates a feasible solution to the original problem automatically. We will prove the expected value of this rounded solution is optimal.

**Theorem 10.** Algorithm FHSAP-CHA generates a randomized feasible assignment to the collinear-hubs case of the FHSAP. The expectation of this assignment is optimal.
Proof. We will analyze the expected performance of the linear cost and the inter-hub cost respectively.

First for each \( \hat{x}_{i,s} \), \( i \in C, s \in \mathcal{H} \),

\[
E(\hat{x}_{i,s}) = \text{Prob}(X_{i,s}^* - 1 < \rho < X_{i,s}^*) = X_{i,s}^* - X_{i,s}^* - 1 = \hat{x}_{i,s}^*.
\]

So the expected value of the linear cost is equal to the corresponding value in its LP relaxation, i.e., \( E[\hat{L}(\hat{x})] = L(\hat{x}^*) \).

Now consider the inter-hub cost. For each \( \hat{X}_{i,s} \), \( \hat{X}_{i,s} \hat{X}_{j,s} \), we have:

\[
E(\hat{X}_{i,s}) = \text{Prob}(\hat{X}_{i,s} = 1) = \text{Prob}(\rho < X_{i,s}^*) = X_{i,s}^*;
\]

\[
E(\hat{X}_{i,s} \hat{X}_{j,s}) = \text{Prob}(\hat{X}_{i,s} \hat{X}_{j,s} = 1) = \text{Prob}(\rho < \min(X_{i,s}^*, X_{j,s}^*)) = \min(X_{i,s}^*, X_{j,s}^*) = X_{i,j,s}^*.
\]

The last equality holds for any optimal feasible solution to the LP relaxation of the problem. Thus, for each pair of terminals \((i, j)\), the expected inter-hub cost is

\[
E \left[ \sum_{s=1}^{k-1} c_{s,s+1} (\hat{X}_{i,s} + \hat{X}_{j,s} - 2\hat{X}_{i,s}\hat{X}_{j,s}) \right] = \sum_{s=1}^{k-1} c_{s,s+1} (X_{i,s}^* + X_{j,s}^* - 2X_{i,j,s}^*).
\]

Therefore, the expected value of this assignment is equivalent to its LP relaxation value, which is a lower bound of the optimal cost.

\[ \square \]

6.2 The 3-hub Median Problem

Consider the \( p \)-hub median problem when \( p = 3 \). Notice that there are at most \( \binom{k}{3} \) possible combinations given \( k \) possible locations. Therefore, we can approach the 3-hub median problem in polynomial time by enumerating all allocation combinations and solving each \( FHSAP \) individually. So, consider the \( FHSAP \) first. Assume that in the optimal solution, three hubs \( \{a, b, c\} \) are open. Without loss of generality, assume \( 1 = c_{ab} \leq c_{ac} \leq c_{bc} \), and \( c_{ab} + c_{ac} \geq c_{bc} \). Now we need to solve a \( 3-FHSAP \) problem for the fixed hubs \( \{a, b, c\} \).

From theorem 9, we know that the \( 3-FHSAP \) problem can be approximated by a factor of \( 2r = 2c_{ac} \).
On the other hand, if we delete the edge \((b, c)\), we can assume all three hubs are in a line. Of course, the optimal cost will increase, by a factor of at most \(1 + \frac{1}{c_{bc}}\). In this way, we can easily obtain a \(1 + \frac{1}{c_{bc}}\)-approximation algorithm for the 3-FHSAP problem.

Therefore, by choosing the better of these two algorithms, we show that the 3-FHSAP problem can be approximated by a factor of

\[
\min\{1 + \frac{1}{c_{bc}}, 2c_{bc}\} = 2.
\]

**Corollary 11.** *There is a 2-approximation algorithm for the general 3-hub median problem with single allocation.*

### 7. Conclusion and the Future Work

In this paper we study the fixed-hub single allocation problem. We identify a special case which is polynomial time solvable. For the general problem, we propose an LP-based algorithm, which exhibits excellent performance in our computational study. Further, our algorithm enjoys a worst-case performance guarantee. To the best our knowledge, this is the first worst-case analysis for a heuristic proposed for the fixed-hub single allocation problem. Our results rely on a new randomized rounding technique, which might be of interest on its own.

There are still many interesting problems worth exploring in the future. It is very possible that there still exist other topologies of hubs which can be approached by constant approximation algorithms and to which a good embedding ratio from a metric graph can be found. The version with setup cost and capacity constraints is also useful in practice. Moreover, the rounding technique for the equilateral and colinear cases may have applications in other quadratic assignment problems.

**Acknowledgments**

We thank Huan Yang for his help in implementation and data collection.
References


