5. ALGORITHMS FOR COMPUTING REMOTENESS FUNCTION IN $\ell_1$-GRAPHS

5.1 Introduction

Algorithms for more general classes of graphs are included in this chapter. In Section 5.2 algorithms for computing remoteness function in partial Hamming graphs are given. A faster algorithm for computing median sets in quasi-median graphs is discussed in Section 5.3. Then in Section 5.4, the algorithm given in Section 5.2 is extended to $\ell_1$-graphs. In Section 5.5, algorithms for computing remoteness function in graphs which admit a scale embedding into Hamming graphs are presented. The algorithms in Sections 5.2, 5.4 and 5.5 are useful to compute antimedian and median sets as well. Recall the definitions of the above mentioned graph classes from Section 1.2.2.

5.2 Algorithm for Computing Remoteness Function in Partial Hamming Graphs

In this section we give an algorithm for computing remoteness function in partial Hamming graphs of complexity $O(n\text{idim}(G))$. It can also be used to compute the antimedian and median sets for arbitrary profiles on a partial Hamming graph.

Since partial Hamming graphs are isometric subgraphs of Hamming graphs, the distance between two vertices in a partial Hamming graph is also same as the Hamming distance between the corresponding vertices. Let $H$ be a Hamming graph with
vertex set \( \{u = u^{(1)} \ldots u^{(d)}\} \), such that \( 0 \leq u^{(i)} \leq m_i - 1 \), where \( m_i \geq 2 \) for each \( i \). Let \( \pi = (x_1, \ldots, x_k) \) be a profile on \( H \). Note that \( k \) is the cardinality of \( \pi \). For \( j = 1, \ldots, d \) let \( n_i^{(j)}(\pi), i = 0, \ldots, p \) be the number of vertices from \( \pi \) with the \( j^{th} \) coordinate equal to \( i \).

For a vertex \( x = (a^1, \ldots, a^d) \) of \( G \) we define the vector \( \vec{d}(x, \pi) \) as follows:

\[
\vec{d}_j(x, \pi) = k - n_i^{(j)}(\pi), \quad \text{for} \quad j = 1, \ldots, d.
\]

Using the definition of the Hamming distance, it is easy to observe, that for any vertex \( x \in V(G) \) we have

\[
D(x, \pi) = \sum_{j=1}^{d} \vec{d}_j(x, \pi).
\]

We derive the following algorithm for computing \( D(x, \pi) \) for any vertex of a partial Hamming graph \( G \).

**Algorithm 5.1:**

**Input:** A partial Hamming graph \( G \) isometrically embedded into a Hamming graph \( H \) and a profile \( \pi \).

**Output:** \( D(x, \pi) \) for all vertices \( x \in V(G) \), \( M(\pi, G) \) and \( AM(\pi, G) \).

**Step 1:** determine \( \vec{d}(\pi) = (n_i^{(1)}(\pi), \ldots, n_i^{(d)}(\pi)) \), for different values appeared in each coordinate position of the vertices in profile.

**Step 2:** For every \( x = (x_1, \ldots, x_d) \in V(G) \) compute

\[
\vec{d}_j(x, \pi) = k - \vec{a}_j(\pi), \quad \text{for} \quad j = 1, \ldots, d
\]

\[
D(x, \pi) = \sum_{j=1}^{d} \vec{d}_j(x, \pi).
\]

**Step 3:** Determine \( M(\pi, G) \) and \( AM(\pi, G) \) as the vertices with the smallest (largest) \( D(x, \pi) \).

**Theorem 5.2:** Let \( x \) be a vertex of a partial Hamming graph \( G \), and let \( \pi \) be a profile on \( G \). Then Algorithm 5.1 correctly computes \( D(x, \pi) \), the antimedian set and the median set of \( \pi \), and can be implemented in \( O(n \text{idim}(G)) \) time.
Proof. Observations preceding the algorithm imply the correctness of the algorithm. Since every vertex of $\pi$ has idim($G$) coordinates, Step 1 can be performed in $O(n \text{idim}(G))$ time as follows, set 0 as the initial frequency of different symbols in each coordinate position, as we scan through the $i^{th}$ coordinate of each vertex in the profile, increment the frequency corresponding to $i^{th}$ position for the observed symbol. Hence Step 1 can be performed in $O(n \text{idim}(G))$ time. For Step 2 we need $O(n \text{idim}(G))$ operations since for every vertex of $G$ we check each of its coordinates, this check is done in a constant time (upon the value of $x^{(j)}$) and choose the corresponding frequency to determine $k - a^j_i(\pi)$, then add this number to the sum in (4.1)). Along with Step 2 we can perform Step 3 in such a way that vertices $x$ with current largest and smallest value of $D(x, \pi)$ are kept during the entire algorithm. This yields a constant number of operations while processing each vertex. Hence the overall time complexity is $O(n \text{idim}(G))$.

Note that vertex set of hypercubes can be obtained as a special case of Hamming graphs with $m_i = 2$ for all $i$. And in this special case for each coordinate position $j$ we need to compute only $n^j_0(\pi)$ and $n^j_1(\pi)$, clearly $k - n^j_0(\pi) = n^j_1(\pi)$ and $k - n^j_1(\pi) = n^j_0(\pi)$. Hence the equations in Section 4.2 can be derived as a special case of the equations in this Section.

5.3 Algorithm for Computing Median Sets in Quasi-Median Graphs

Quasi-median graphs can be characterized as weak retracts of Hamming graphs [74]. An efficient algorithm for computing median set in quasi-median graph is presented in this section.

Now we generalize the concept of Majority and Minority rule into Hamming graphs. Let $H_d$ be a Hamming graph with $d$ components and let $\pi = (x_1, \ldots, x_k)$
be a profile. For $j = 1, \ldots, d$ and $i = 1, \ldots, p$ let $n_i^j(\pi)$ be the number of vertices from
$\pi$ with $j^{th}$ position equal to $i$ more formally,

$$n_i^j(\pi) = |\{x = (a_1, \ldots, a_d) \in \pi : a_j = i\}|$$

Define $Majority(\pi)$ as the set of vertices $u = (u_1, \ldots, u_d)$ where $u_j \in \{a_i\}$ such that
$n_{a_i}^j(\pi) = \text{maximum}(n_i^j(\pi))$. Similarly $Minority(\pi)$ for Hamming graphs are defined
as follows. $Minority(\pi)$ is the set of vertices $u = (u_1, \ldots, u_d)$ where $u_j \in \{a_i\}$
such that $n_{a_i}^j(\pi) = \text{minimum}(n_i^j(\pi))$. Note that in the case of Hamming graphs
$|Majority(\pi)|$ or $|Minority(\pi)|$ need not equal to 1. (i.e., the case is different from
that of hypercubes).

**Proposition 5.3:** Let $\pi = (x_1, \ldots, x_k)$ be a profile on a Hamming graph $H_d$ then
$M(\pi, H_d) = Majority(\pi)$ and $AM(\pi, H_d) = Minority(\pi)$

**Proof.** From Equation 5.1 we have

$$D(u, \pi) = \sum_{j=1}^{d} k - n_{a_i}^j(\pi).$$

It is clear that those vertices in $Majority(\pi)$ will have minimum on each component
of summation. Hence we have $M(\pi) = Majority(\pi)$. Similarly we get $AM(\pi) =
Minority(\pi)$, since vertices in $Minority(\pi)$ have maximum on each component of
summation.

Recall that every (weak) retract is a weak contraction, in particular, it is known
that quasi-median graphs are precisely retracts of Hamming graphs [160], hence due
to Theorem 4.3 we deduce:

**Corollary 5.4:** Let $G$ be a quasi-median graph, isometrically embedded into a
Hamming graph $H$. Then for any profile $\pi$ on $G$, $M(\pi, G) = M(\pi, H) \cap V(G)$. 
Combining Corollary 5.4 with Proposition 5.3 we get the following algorithm for computing median sets in quasi-median graphs.

**Algorithm 5.5:**

Input: A quasi-median graph $G$ isometrically embedded into a Hamming graph $H$ and a profile $\pi$ on $G$.

Output: $M(\pi, G)$.

Step 1: Compute $n_i^j(\pi) = |\{x = (a_1, \ldots, a_d) \in \pi : a_j = i\}|$.

Find $M(\pi, H)$ using the Majority rule as follows:

$\text{Majority}(\pi)$ equal to the set of vertices $u = (u_1, \ldots, u_d)$ where $u_j \in \{a_i\}$ such that $n_{a_i}^j(\pi) = \text{maximum}(n_i^j(\pi))$.

Step 2: Compute $M(\pi, G) = M(\pi, H) \cap V(G)$.

**Theorem 5.6:** Algorithm 5.5 correctly computes the median set $M(\pi, G)$ in a quasi-median graph $G$ and can be implemented in $O(n \text{idim}(G))$ time.

**Proof.** Correctness of the algorithm follows from Corollary 5.4 and Proposition 5.3.

Since every vertex of $\pi$ has $\text{idim}(G)$ coordinates, Step 1 can be performed in $O(n \text{idim}(G))$ time. Along with Step 1 we can perform Step 2 as follows. As soon as we determine the Majority in the $j$th coordinate, we mark all the vertices of $G$ that have the $j$th coordinate different from the Majority as non-median. Note that if the frequency of all symbols in the $j$th coordinate are equal, we do nothing. At the end we are left with the median set. Altogether we need $O(n \text{idim}(G))$ operations.

### 5.4 Algorithm for Computing Remoteness Function in $\ell_1$-Graphs

In this section we design an efficient algorithm for computing the remoteness function of arbitrary profile on graphs which are scale embeddable into $Q_d$. The algorithm also computes antimedian (median) set.
Let $\pi = (x_1, \ldots, x_k)$ be a profile on $Q_d$. For $i = 1, \ldots, k$ let $n_0^{(i)}$ and $n_1^{(i)}$ be the number of vertices from $\pi$ with the $i^{th}$ coordinate equal 0 and 1, respectively. More formally,

$$n_0^{(i)}(\pi) = |\{x \in \pi \mid x^{(i)} = 0\}|$$

and

$$n_1^{(i)}(\pi) = |\{x \in \pi \mid x^{(i)} = 1\}|.$$ 

Define Majority ($\pi$) as the set of vertices $u = u^{(1)} \ldots u^{(d)}$ of $Q_d$, where

$$u^{(i)} \begin{cases} 
= 0; & n_0^{(i)}(\pi) > n_1^{(i)}(\pi), \\
= 1; & n_0^{(i)}(\pi) < n_1^{(i)}(\pi), \\
\in \{0, 1\}; & n_0^{(i)}(\pi) = n_1^{(i)}(\pi).
\end{cases}$$

We may store the values $n_0^{(j)}(\pi)$ and $n_1^{(j)}(\pi)$ in two vectors, that is, for any profile $\pi$,

let

$$\vec{0}(\pi) = (n_0^{(1)}(\pi), \ldots, n_0^{(k)}(\pi)),$$

and

$$\vec{1}(\pi) = (n_1^{(1)}(\pi), \ldots, n_1^{(k)}(\pi)).$$

For a vertex $x$ of $G$ we define the vector $\vec{d}(x, \pi)$ as follows:

$$\vec{d}_j(x, \pi) = \begin{cases} 
\vec{0}_j(\pi); & x^{(j)} = 1, \\
\vec{1}_j(\pi); & x^{(j)} = 0,
\end{cases}$$

for $j = 1, \ldots, k$. It is easy to see, using the definition of the Hamming distance, that for any vertex $x \in V(G)$ we have
\[ D(x, \pi) = \sum_{j=1}^{k} \overrightarrow{d}_j(x, \pi) / \lambda. \] (5.2)

Based on these observations, an algorithm is obtained for computing remoteness function in any graph \( G \) which is scale \( \lambda \) embeddable into a hypercube \( Q_d \). The algorithm can also be used for computing the antimedian and median sets in such graphs.

**Algorithm 5.7:**

**Input:** A graph \( G \) scale embedded into a hypercube \( Q_d \), with scale factor \( \lambda \).

A profile \( \pi \).

**Output:** \( D(x, \pi) \) for all vertices \( x \in V(G) \), \( M(\pi, G) \) and \( AM(\pi, G) \).

**Step 1:** Using the Majority rule, determine \( \overrightarrow{0}(\pi) \) and \( \overrightarrow{1}(\pi) \).

**Step 2:** For every \( x \in V(G) \) compute \( D(x, \pi) \) as follows:

\[
\overrightarrow{d}_j(x, \pi) = \begin{cases} 
\overrightarrow{0}_j(\pi); & x^{(j)} = 1, \\
\overrightarrow{1}_j(\pi); & x^{(j)} = 0,
\end{cases}
\]

\[ D(x, \pi) = \sum_{j=1}^{k} \overrightarrow{d}_j(x, \pi) / \lambda. \]

**Step 3:** Determine \( M(\pi, G) \) and \( AM(\pi, G) \) as the vertices with the smallest (largest) \( D(x, \pi) \).

Observations preceding the algorithm imply the correctness of the algorithm.

In Step 1 of the above algorithm, vectors \( \overrightarrow{0}(\pi) \) and \( \overrightarrow{1}(\pi) \) are explicitly determined and saved. Hence Step 1 can be performed in \( O(nd) \) time, where \( d \) is the dimension of the smallest hypercube in which \( G \) is \( \lambda \)-embeddable. For Step 2 we need \( O(nd) \) operations since for every vertex of \( G \) we check each of its coordinates.
and this check is done in a constant time (based on the value of $x^{(j)}$ select either \(0_j\) or \(1_j\), then add this number to the sum in equation (4.1)). Along with Step 2 we can perform Step 3 in such a way that vertices with current largest and smallest value of $D(x, \pi)$ are kept during the entire algorithm. This yields a constant number of operations while processing each vertex. Hence the overall time complexity is $O(nd)$.

5.5 Algorithms for Computing Remoteness Function with Respect to $\lambda$-embedding into Hamming Graphs

In this section we give an algorithm for computing remoteness function of arbitrary profiles on graphs which are scale embeddable into Hamming graphs. Irrespective of the coordinatization of the partial Hamming graph and the choice of the profile, the number of operations are fixed in this algorithm.

Note that when the graph $G$ is embedded into a Hamming graph $H$ with scale factor $\lambda$, the distance between any two vertices in $G$ is same as the Hamming distance between the corresponding vertices divided by $\lambda$. The Hamming distance can be computed as follows. First compute frequency of different values $i$, $n_i^{(j)}(\pi), i = 0, \ldots, p$, appeared in each coordinate position $j$ of the vertices in profile $\pi$ and store these values in different vectors, that is, for any profile $\pi$ on a partial Hamming graph $G$ let \(\overrightarrow{d}(\pi) = (n_{a_1}^{(1)}(\pi), \ldots, n_{a_k}^{(k)}(\pi))\), where $k$ is the dimension of the host Hamming graph.

For a vertex $x$ of $G$ we define the vector \(\overrightarrow{d}(x, \pi)\) as follows: \(\overrightarrow{d}_j(x, \pi) = r - a_j^{(j)}(\pi)\), for $j = 1, \ldots, k$ (where $a = x^{(j)}$ and $r = |\pi|$). It is easy to see, using the definition of the Hamming distance, that for any vertex $x \in V(G)$ we have

\[
D(x, \pi) = \sum_{j=1}^{k} \overrightarrow{d}_j(x, \pi)/\lambda. \tag{5.3}
\]
We derive the following algorithm for computing \( D(x, \pi) \) for graphs which are scale embeddable into Hamming graphs.

**Algorithm 5.8:**

Input: A graph \( G \), which is \( \lambda \)-embeddable into a Hamming graph \( H \) and a profile \( \pi \).

Output: \( D(x, \pi) \) for all vertices \( x \in V(G) \), \( M(\pi, G) \) and \( AM(\pi, G) \).

Step 1: Using the frequency rule, determine \( \overrightarrow{d}(\pi) = (n_a^{(1)}(\pi), \ldots, n_a^{(k)}(\pi)) \), for different values appeared in each coordinate position of the vertices in profile.

Step 2: For every \( x \in V(G) \) compute \( D(x, \pi) \)

\[
D(x, \pi) = \sum_{j=1}^{k} \frac{\overrightarrow{d}_j(x, \pi)}{\lambda}.
\]

Step 3: Determine \( M(\pi, G) \) and \( AM(\pi, G) \) as the vertices with the smallest (largest) \( D(x, \pi) \).

**Theorem 5.9:** Let \( x \) be a vertex of a partial Hamming graph \( G \), and let \( \pi \) be a profile on \( G \). Then Algorithm 5.8 correctly computes \( D(x, \pi) \), the median set and the antimedian set of \( \pi \), and can be implemented in \( O(nd) \) time.

**Proof.** Observations preceding the algorithm imply the correctness of the algorithm. Since every vertex of \( \pi \) has \( d \) coordinates, Step 1 can be performed in \( O(nd) \) time as follows. Set 0 as the initial frequency of different symbols in each coordinate position, as we scan through the \( j^{th} \) coordinate of each vertex in the profile, increment the frequency corresponding to \( j^{th} \) position for the observed symbol. Hence Step 1 can be performed in \( O(nd) \) time. For Step 2 we need \( O(nd) \) operations since for every vertex of \( G \) we check each of its coordinates, and this check is done in a constant time (based on the value of \( x(j) \) select the corresponding frequency to determine \( r - \overrightarrow{a}_j^{(\pi)} \), then add this number to the sum in (4.1)). Along with Step 2 we can
perform Step 3 in such a way that vertices with current largest and smallest value of $D(x, \pi)$ are kept during the entire algorithm. This yields a constant number of operations while processing each vertex. Hence the overall time complexity is $O(nd)$.

5.6 Concluding Remarks

In this chapter we have focussed on the algorithms for computing remoteness function in $\ell_1$-graphs and graphs which admit a scale embedding into Hamming graphs. We obtained a fast algorithm for computing median sets in quasi-median graphs using their embedding via weak contraction into Hamming graphs.