A Heuristic Method of Optimal Generalized Indexware Encoding for Pictorial Databases

C. C. Yuen

Space Research Branch
Space Systems Division

D. E. Chang and E. L. Smith

Department of Information Engineering
University of Ill.
Chicago, Illinois

January 15, 1980
A HEURISTIC METHOD OF OPTIMAL GENERALIZED HYPERCUBE ENCODING FOR PICTORIAL DATABASES

C.C. Yang, S.K. Chang, K.K. Singh

Naval Research Laboratory
Washington, DC 20375

NRL Problem 79-0721-0-0

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)
UNCLASSIFIED

Approved for public release; distribution unlimited.

Similarity retrieval from a large pictorial database can be much more efficient by encoding the original database into certain convenient formats. Generalized hypercube (GH) encoding is one such technique. To optimize GH encoding, a heuristic approach has been formulated. Two optimization problems have been considered here: First, given the handle length m, find the optimal GH encoding. Second, given the threshold density, find the optimal GH encoding such that each generated hypercube has a density no less than a threshold density.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>GENERALIZED HYPERCUBE ENCODING</td>
<td>1</td>
</tr>
<tr>
<td>PROBLEM 1</td>
<td>2</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>2</td>
</tr>
<tr>
<td>Step 1</td>
<td>2</td>
</tr>
<tr>
<td>Step 2</td>
<td>3</td>
</tr>
<tr>
<td>Step 3</td>
<td>3</td>
</tr>
<tr>
<td>EXPERIMENTAL RESULTS</td>
<td>3</td>
</tr>
<tr>
<td>PROBLEM 2</td>
<td>4</td>
</tr>
<tr>
<td>ALGORITHM</td>
<td>7</td>
</tr>
<tr>
<td>Step 1</td>
<td>7</td>
</tr>
<tr>
<td>Step 2</td>
<td>7</td>
</tr>
<tr>
<td>Step 3</td>
<td>7</td>
</tr>
<tr>
<td>Step 4</td>
<td>7</td>
</tr>
<tr>
<td>Step 5</td>
<td>7</td>
</tr>
<tr>
<td>Step 6</td>
<td>8</td>
</tr>
<tr>
<td>Step 7</td>
<td>8</td>
</tr>
<tr>
<td>EXPERIMENTAL RESULTS</td>
<td>8</td>
</tr>
<tr>
<td>DISCUSSION</td>
<td>8</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>11</td>
</tr>
<tr>
<td>APPENDIX A</td>
<td>13</td>
</tr>
<tr>
<td>APPENDIX B</td>
<td>23</td>
</tr>
</tbody>
</table>
A HEURISTIC METHOD OF OPTIMAL GENERALIZED HYPERCUBE ENCODING FOR PICTORIAL DATABASES

INTRODUCTION

In similarity retrieval from a pictorial database [1], pattern recognition [2, 4], and clustering analysis, it is often desired to find the set of database records (or set of patterns, n-dimensional feature vectors, etc.) that are most similar to a test record (or test pattern, test feature vector, etc.). In the case of large databases (or patterns, clusters), it is very useful to encode the original database into certain convenient format in order to facilitate similarity retrieval and updating.

In this report, we shall consider one such technique called "Generalized Hypercube" encoding [3] and describe a heuristic method of generating minimum number of GH encoded tuples.

GENERALIZED HYPERCUBE ENCODING

Given a set of points \( S = \{ p_1, p_2, \ldots, p_k \} \) in n-dimensional space, where \( p_i = (x_1, x_2, \ldots, x_n) \) denotes a point in \( S \), we can have a family of n-dimensional GH codes. For each \( m, 1 \leq m \leq n + 1 \), the GH\(_m\) codes are:

\[
(x_1, \ldots, x_{m-1}; a_m, \ldots, a_n; b_m, \ldots, b_n),
\]

where \((x_1, x_2, \ldots, x_{m-1}, z_{m}, \ldots, z_n)\) is in \( S \) for some coordinates, \( z_{m}, \ldots, z_n \); and

\[
a_j = \min \{ y_j : \text{for some coordinates, } z_{m}, z_{k}, (x_1, \ldots, x_{m-1}, z_{m}, \ldots, y_j, \ldots, z_n) \text{ in } S \},
\]

\[
b_j = \max \{ y_j : \text{for some coordinates, } z_{m}, z_{k}, (x_1, \ldots, x_{m-1}, z_{m}, \ldots, y_j, \ldots, z_n) \text{ in } S \}.
\]

When \( m = 1 \), we are simply using the smallest n-dimensional hypercube containing \( S \) as the GH code. When \( m = n+1 \), the original point set \( S \) is used as the GH code. Other in-between values of \( m \) give GH codes of various levels of details.

For example, if point set \( S = \{ (1,1,3),(1,1,5),(1,2,1),(1,2,4),(2,3,6) \} \), GH\(_1\) is \( \{ (1,1,2,3,6) \} \). GH\(_2\) is \( \{ (1,1,1,2,5),(2,3,6,3,6) \} \). GH\(_3\) is \( \{ (1,1,3,5),(1,2,1,4),(2,3,6,6) \} \). GH\(_4\) is \( S \) itself. It should be noted that other GH codes can be generated, if we permute the coordinates. The previous example we have \( \{ (3,1,1,1,1),(5,1,1,1,1),(1,1,2,1,2), (6,2,3,2,3) \} \).

Therefore, it is important to select carefully the coordinates on which the GH encoding technique can be applied, i.e. to generate a minimum number of GH encoded tuples the problem reduces to select a special "handle" set. It is easy to see that once a handle vector has been selected, the GH encoding is unique.

In this report we shall describe the following heuristic methods for problems:

- Select a handle set \( \{ i_1, i_2, \ldots, i_{m-1} \} \) from \( \{ 1, 2, \ldots, n \} \) so that the least number of \( GH \) encoded tuples are generated.

- Suppose we define the density of a hypercube \( H \) as the number of data points in \( H \), divided by the total number of lattice integers. For example, the hypercube \((1,1,2,3,6)\) previously described has a density of \( 5/2\cdot3\cdot6 = 0.1388 \). Intuitively, we would like to generate hypercubes having high densities. Therefore Problem 2 can be stated as follows:

  Given a threshold density \( t \), \( 0 \leq t \leq 1.0 \), find optimal \( GH \) encoding such that each hypercube has density no less than \( t \).

**PROBLEM 1**

Given \( m \), the method of choosing \( m-1 \) handles which will generate optimal \( GH_m \) codes for a point set \( S \) is based on the following ideas:

- For each handle (column) \( i \), \( i = 1, 2, \ldots, n \) consider vector \( V_i = (x_{1i}, x_{2i}, \ldots, x_{ki}) \) where \( k \) is the number of points in the set \( S \). Among the elements \( x_{1i}, x_{2i}, \ldots, x_{ki} \) those having the same value are grouped together and their associated points (in \( S \)) will consist of disjoint groups, say, \( G_{1i}, G_{2i}, \ldots, G_{ji} \). Define the count (or cardinality) of \( G_{ji} = C_j(i) \). We next compute a measure of the priority of the \( i \)th column \( p_i \).

\[
p_i = \sum_{j=1}^{\ell} (C_j(i)-1)^2.
\]

Intuitively, we ignore those groups having only one element and put heavier weight to those groups having more than one element. Obviously, there are many other ways to define a priority measurement for each handle.

- Based on the measure \( p_i \), \( i = 1, 2, \ldots, n \) choose the best \( m-1 \) handles, i.e., \( i \)'s which have larger values of \( p_i \).

Following is the algorithm to implement the above ideas:

**ALGORITHM**

**Step 1.**

To compute \( p_i \), consider vector \( V_i = (x_{1i}, x_{2i}, \ldots, x_{ki}) \).

a. Make a stack of all \( k \) elements of \( V_i \).
b. Set a test element, TELMNT = top element of stack and set COUNT = 1.

c. Replace the top element of the stack by the last element and, set k = k-1; if k < 1, go to substep e.

d. Restack the k elements to check that the top element = TELMNT.
   • If it does, COUNT = COUNT + 1; then go to substep c.
   • If it does not, then
     \[
     \begin{align*}
     \text{If COUNT} > 1, \text{then } P_i &= P_i + \text{COUNT} \times \text{COUNT} \text{ and COUNT} = 1. \text{TELMNT} = \\
     \text{top element of the stack; go to substep c.}
     \end{align*}
     \]

e. Include the last group's population count in P_i.
   If COUNT > 1, then \( P_i = P_i + \text{COUNT} \times \text{COUNT} \).

Step 2.

Using STEP 1, compute all P_i's for \( i = 1, 2, \ldots, n \). Store handle numbers in an array, such as \([c(i) = i, i = 1, \ldots, n]\).

Step 3.

Sort array P in descending order, and while moving the elements of P during sort, move elements of c also in the same fashion.

For generating GH\(_m\) codes, choose handles in the first (m - 1) positions of array c, which correspond to the best (m - 1) values of P_i, \( i = 1, 2, \ldots, n \).

Once the set of handles has been selected, the GH encoding is unique.

For example, let point set \( S = \{(3,1,1), (5,1,1), (1,1,2), (4,1,2), (6,2,3)\} \). Using the previous algorithm, the measure of priority, \( P_i \) for \( i = 1, 2, 3 \) are: \( P_1 = 0, P_2 = 16, P_3 = 8 \). Based on the best values of \( P_i \), we choose handle set \((x_2, x_3, x_1)\) and with respect to this set of handles \((GH_1)\) is \(\{(1,1,2,2,3,6)\}\). \(GH_2\) is \(\{(1,1,1,2,5), (2,3,6,6)\}\). \(GH_3\) is \(\{(1,1,3,5), (1,2,1,4), (2,3,6,6)\}\). \(GH_4\) is \(\{(1,1,3), (1,1,5), (1,2,1), (1,2,4), (2,3,6)\}\).

EXPERIMENTAL RESULTS

The previous algorithm was implemented on an IBM/370 in FORTRAN; a listing of the program is given in Appendix A. The data for the program were random numbers with normal distribution and standard deviation, a number for each column vector \( V_i \) was also a random number with uniform distribution.
The selected set of handles by the previous method was used to generate \( GH_m \) encoded tuples, and their number was counted for a different number of point set and dimensions of space. To find the optimal \( GH_m \) encoding, great effort was taken, and the \( GH_m \) codes with minimum number of \( GH_m \) tuples was taken as the optimal \( GH_m \) encoding. (See Tables 1, 2, and 3.)

Few other heuristic approaches were tested to select the proper set of handles to generate optimal GH encoding, but the previous method has shown the best results and will be used in the next optimization problem.

**PROBLEM 2**

From the experimental results we know that the smaller the value of \( m \) (length of handles) the smaller the number of \( GH_m \) encoded tuples for a point set \( S \), but at the same time such hypercubes tend to be sparse. Therefore, to optimize GH codes for a given point set \( S \), the number of \( GH_m \) encoded tuples was counted for different dimensions and point sets. The results are shown in the following tables.

<table>
<thead>
<tr>
<th>No. of Points (M)</th>
<th>( GH_2 )</th>
<th>( GH_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( H )</td>
<td>( O )</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>40</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>50</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>60</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>70</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>80</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>90</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

**Table 1 — The number of optimal \( GH_m \) encoded tuples**

*Dimension \( N = 3 \)*  
*H = heuristic*  
*O = optimum*
Table 2 — The number of optimal GHₘ encoded tuples

<table>
<thead>
<tr>
<th>No. of Points (M)</th>
<th>GH₂</th>
<th></th>
<th>GH₃</th>
<th></th>
<th>GH₄</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H</td>
<td>O</td>
<td>Error (%)</td>
<td>H</td>
<td>O</td>
<td>Error (%)</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>8</td>
<td>0.0</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>16</td>
<td>16</td>
<td>0.0</td>
<td>20</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
<td>29</td>
<td>28</td>
<td>3.57</td>
</tr>
<tr>
<td>40</td>
<td>21</td>
<td>21</td>
<td>0.0</td>
<td>40</td>
<td>39</td>
<td>2.56</td>
</tr>
<tr>
<td>50</td>
<td>16</td>
<td>16</td>
<td>0.0</td>
<td>44</td>
<td>44</td>
<td>0.0</td>
</tr>
<tr>
<td>60</td>
<td>25</td>
<td>25</td>
<td>0.0</td>
<td>59</td>
<td>59</td>
<td>0.0</td>
</tr>
<tr>
<td>70</td>
<td>28</td>
<td>28</td>
<td>0.0</td>
<td>68</td>
<td>68</td>
<td>0.0</td>
</tr>
<tr>
<td>80</td>
<td>21</td>
<td>21</td>
<td>0.0</td>
<td>74</td>
<td>71</td>
<td>4.23</td>
</tr>
<tr>
<td>90</td>
<td>18</td>
<td>18</td>
<td>0.0</td>
<td>75</td>
<td>75</td>
<td>0.0</td>
</tr>
<tr>
<td>100</td>
<td>22</td>
<td>22</td>
<td>0.0</td>
<td>93</td>
<td>93</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Dimension N = 4
H = heuristic
O = optimum
Table 3 - The number of optimal $GH_m$ encoded tuples

<table>
<thead>
<tr>
<th>No. of Points (M)</th>
<th>$GH_2$</th>
<th>$GH_3$</th>
<th>$GH_4$</th>
<th>$GH_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>H</td>
<td>O</td>
<td>Error (%)</td>
<td>H</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>8</td>
<td>0.0</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>13</td>
<td>13</td>
<td>0.0</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
<td>29</td>
</tr>
<tr>
<td>40</td>
<td>16</td>
<td>16</td>
<td>0.0</td>
<td>38</td>
</tr>
<tr>
<td>50</td>
<td>21</td>
<td>21</td>
<td>0.0</td>
<td>49</td>
</tr>
<tr>
<td>60</td>
<td>18</td>
<td>18</td>
<td>0.0</td>
<td>55</td>
</tr>
<tr>
<td>70</td>
<td>18</td>
<td>18</td>
<td>0.0</td>
<td>64</td>
</tr>
<tr>
<td>80</td>
<td>21</td>
<td>21</td>
<td>0.0</td>
<td>75</td>
</tr>
<tr>
<td>90</td>
<td>26</td>
<td>26</td>
<td>0.0</td>
<td>84</td>
</tr>
<tr>
<td>100</td>
<td>22</td>
<td>22</td>
<td>0.0</td>
<td>96</td>
</tr>
</tbody>
</table>

Dimension N = 5
H = heuristic
O = optimum

threshold density $t$, find a set of handles with minimum length $m$, so that the resultant hypercubes have densities no less than the threshold density, $t$.

Two special cases can be solved immediately. If $t = 0$, then we can select $m = 1$, so that a single n-dimensional hypercube is generated, and $GH_1$ becomes the optimum $GH$ encoding. If $t = 1.0$, then all the hypercubes must have unit density, and we can select $m = n + 1$ so that the original point set $S$ becomes the optimum $GH$ encoding. For other values of $t$, $m$ should fall between 1 and $n + 1$. 
ALGORITHM

The algorithm optimize GH encoding for a given threshold density.

Step 1.

Start with $m = 1$;

generate $G_{1}$ codes, which is one hypercube. If the density of this hypercube is $\geq$ threshold density, then this is the optimum $G_{m}$ encoding, and go to Step 7, otherwise, set $m = m + 1$.

Step 2.

By using the previous heuristic method discussed in Problem 1, choose a set of handles which will generate the optimal $G_{m}$ encoding for a given $m$.

Set a flag $= 0$; and

NUM $= 0$, where NUM will be the number of $G_{m}$ encoded tuples with respect to the chosen handle vector of length $m$.

Step 3.

a. Generate the next hypercube with respect to the above chosen handle vector and compute its density.

b. If density of this hypercube $\geq$ threshold density, and if the number of points in this hypercube $\geq 1$, then

{ include this hypercube in optimal $G_{m}$ encoding.

NUM $= NUM + 1$; and

delete the points included in this hypercube from the original point set. }

c. If the density of this hypercube $< \text{threshold density}$, then set the flat $= 1$

Step 4.

Repeat Step 3 until all the points in point set have been considered once.

Step 5.

If the number of points in remaining point set $= 0$ then go to Step 7.

If (number of points in point set $= 1$ or

7
NUM <= 0 and flag = 0), then go to Step 6.

If (NUM <= 0 and flag = 1), then m = m + 1.

Go to Step 2.

Step 6.

Generate GH_m encoded tuples with respect to the last handle vector of length m.
Each GH_m encoded tuple is included in the optimal GH encoding.

Step 7.

Stop.

As an illustration, if point set S = {(3,1,1),(5,1,2),(1,1,2),(4,1,2),(6,2,3)} and threshold density = 50% (0.5), then with the previous algorithm, optimal encoding is attended as {(1;1,2,2,5), (2,3,6;3,6)} and the density of each hypercube is >= 50%.

EXPERIMENTAL RESULTS

The previous algorithm also was implemented on an IBM/370 in FORTRAN. A listing of the program is given in Appendix B. The test data were again random numbers.

The results are in Tables 4, 5, and 6.

DISCUSSION

In this section, some interesting applications of the generalized GH encoding technique are discussed.

Reference 1 suggests that a relational file R can be characterized by a logical expression E, so that every n-tuple (x_1,x_2,...,x_n) in R satisfies this logical expression.

The generalized GH encoding technique can be used to find a suitable logical expression characterizing a relational file.

For example, if S is as given in the Generalized Hypercube Encoding section of this report and m = 1, then GH_1 = { (1,1,1,2,3,6) } and the corresponding logical expression is (1 <= x_1) & (1 <= x_2) & (1 <= x_3) & (x_1 <= 2) & (x_2 <= 3) & (x_3 <= 6);

when m = 2, then GH_2 = { (1;1,1,2,2,5),(2,3,6;3,6) } and the corresponding logical expression is (x_1 = 1) & (1 <= x_2) & (1 <= x_3) & (x_3 <= 5)) V ((x_1 = 2) & (x_2 = 3) & (x_3 = 6)), thus, each GH encoding corresponds to a logical characteristic expression for the relational file.
### Table 4 — The number of optimal GH encoded tuples

<table>
<thead>
<tr>
<th>(M)</th>
<th>t=10</th>
<th>t=20</th>
<th>t=30</th>
<th>t=40</th>
<th>t=50</th>
<th>t=60</th>
<th>t=70</th>
<th>t=80</th>
<th>t=90</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>11</td>
<td>14</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>30</td>
<td>9</td>
<td>23</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>26</td>
<td>28</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>40</td>
<td>18</td>
<td>36</td>
<td>36</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>38</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td>50</td>
<td>23</td>
<td>36</td>
<td>39</td>
<td>40</td>
<td>42</td>
<td>48</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>60</td>
<td>12</td>
<td>43</td>
<td>48</td>
<td>53</td>
<td>55</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td>58</td>
</tr>
<tr>
<td>70</td>
<td>21</td>
<td>40</td>
<td>46</td>
<td>46</td>
<td>49</td>
<td>52</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>57</td>
<td>57</td>
<td>60</td>
<td>60</td>
<td>60</td>
</tr>
<tr>
<td>90</td>
<td>1</td>
<td>45</td>
<td>50</td>
<td>54</td>
<td>57</td>
<td>77</td>
<td>80</td>
<td>84</td>
<td>84</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>40</td>
<td>52</td>
<td>58</td>
<td>60</td>
<td>81</td>
<td>82</td>
<td>91</td>
<td>91</td>
</tr>
</tbody>
</table>

*M = number of points in point set

*t = percent threshold density

Dimension N = 3
Table 5 — The number of optimal GH encoded tuples

<table>
<thead>
<tr>
<th>(M)</th>
<th>t=10</th>
<th>t=20</th>
<th>t=30</th>
<th>t=40</th>
<th>t=50</th>
<th>t=60</th>
<th>t=70</th>
<th>t=80</th>
<th>t=90</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7</td>
<td>7</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>17</td>
<td>17</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>28</td>
<td>28</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>40</td>
<td>32</td>
<td>36</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
<td>39</td>
</tr>
<tr>
<td>50</td>
<td>38</td>
<td>45</td>
<td>46</td>
<td>49</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>60</td>
<td>46</td>
<td>52</td>
<td>56</td>
<td>57</td>
<td>57</td>
<td>57</td>
<td>59</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>70</td>
<td>42</td>
<td>57</td>
<td>60</td>
<td>68</td>
<td>68</td>
<td>68</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>80</td>
<td>45</td>
<td>72</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>90</td>
<td>58</td>
<td>72</td>
<td>78</td>
<td>79</td>
<td>82</td>
<td>87</td>
<td>88</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>100</td>
<td>66</td>
<td>78</td>
<td>81</td>
<td>90</td>
<td>91</td>
<td>92</td>
<td>92</td>
<td>93</td>
<td>93</td>
</tr>
</tbody>
</table>

M = number of points in point set

* t = percent threshold density

Dimension N = 4
In addition to applications in database characterization, the previously described technique also can be applied in clustering analysis where clusters are described by hypercubes with handles. Thus optimization technique will generate optimal encoding for the clusters.

A third application can be the description of n-dimensional pictures, which will have a practical significance in computer graphics and image processing applications.

REFERENCES


<table>
<thead>
<tr>
<th>(M)</th>
<th>t=10</th>
<th>t=20</th>
<th>t=30</th>
<th>t=40</th>
<th>t=50</th>
<th>t=60</th>
<th>t=70</th>
<th>t=80</th>
<th>t=90</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>47</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>60</td>
<td>57</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td>58</td>
<td>59</td>
<td>59</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>70</td>
<td>68</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>70</td>
<td>70</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>80</td>
<td>76</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>77</td>
<td>79</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>90</td>
<td>79</td>
<td>86</td>
<td>87</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
<tr>
<td>100</td>
<td>95</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

M = number of points in point

**Table 6 — The number of optimal GH encoded tuples**
APPENDIX A

1. //JUNK JDB
2. //JMP 1  = (8,0) L=2, R=256
3. //EXEC FORTGCLG
4. //PRINT SYTV DD *
5. CCCCCCCCCC
6. C
7. C MAIN PROGRAM TO GENERATE OPTIMAL GHM ENCODING FOR A POINTSET BY
8. C A HEURISTIC METHOD & TO CHECK THE RESULTS. NUMBER OF THESE GHM
9. C ENCODED TUPLES IS COMPARED WITH THE MINIMUM (REAL OPTIMUM) NO.
10. C OF GHM ENCODED TUPLES FOUND IN EXHAUSTIVE APPROACH.
11. C
12. C CALLS:
13. C (1) GENPTS: TO GENERATE TEST DATA I.E. A POINT SET,
14. C (2) HEURIS: TO GENERATE OPTIMAL GHM ENCODED TUPLES BY A
15. C HEURISTIC METHOD
16. C (3) EXHAUS: TO GENERATE OPTIMAL GHM ENCODING BY EXHAUSTIVE
17. C APPROACH
18. C
19. CCCCCCCCCC
20. INTEGER PTSET(110,6), P16, NCODE(5,3)
21. IFL=1
22. N = 50
23. CALL GENPTS(PTSET,N,N)
24. IF(IFL = EO. 1) GO TO 440
25. PRINT 1
26. FORMAT(*11,10X, 'INITIAL POINT SET IS: ')
27. DO 100 J = 1, M
28. PRINT 2 + (PTSET(I,J) = 1)
29. CONTINUE
30. 100 CONTINUE
31. 2 FORMAT(*11,10X,'HEURISTIC APPROACH')
32. PRINT 3
33. 3 FORMAT(100(1X))
34. PRINT 4
35. 4 FORMAT(*,25X,'HEURISTIC APPROACH')
36. PRINT 5
37. 5 FORMAT(25X,'--------- --------')
38. PRINT 6
39. 6 FORMAT(100(1X))
40. CALL HEURIS(PTSET,N,N, NCODE,IFL)
41. IF(IFL = EO. 1) GO TO 440
42. PRINT 7
43. 7 FORMAT(100(1X))
44. PRINT 8
45. 8 FORMAT(*,21X,'EXHAUSTIVE APPROACH')
46. PRINT 9
47. 9 FORMAT(21X,'--------- --------')
48. PRINT 10
49. 10 FORMAT(100(1X))
50. CALL EXHAUS(PTSET,N,N, NCODE,IFL)
51. N1 = N-1
52. PRINT 20
53. 20 FORMAT(*,10X,'NO. OF POINTS IS = ', 14)
54. PRINT 22
55. 22 FORMAT(10X,'NO. OF DIMENSINS = ', 13)
56. PRINT 15
57. 15 FORMAT(*,20X,'HEURISTIC',10X,'OPTIMUM',10X,'WORST',8X,'ERR')
58. JJ 200 = 1, N1
59. ERR = (NCODE(1,1) - NCODE(1,2) + 100)*NCODE(1,2)
60. 200 FORMAT(16x,1,12(1x,NCODE(1,1)),J= 1,3)ERR

13
YANG, CHANG AND SINGH

16 FORMAT('0F10.6,2X,2GF12.3(10X,14.1,F16.3)')
36 IF(PL.EQ.1) PRINT 66
66 FORMAT('0F10.6')
500 CONTINUE
560 CONTINUE
STOP
END

CCCECCCC
70. C RANDOM: GENERATES A RANDOM NUMBER X, 0 < X <= 1.0 WITH
71. C UNIFORM DISTRIBUTION.
72. C
73. CCECCCC
74.  SUBROUTINE RANDOM(X)
75.  REAL X
76.  INTEGER A, MULT, BASE
77.  A = MOD(MULT*A, BASE)
78.  X = FLOAT(A)/FLOAT(BASE)
79. RETURN
80. END
81. CCCECCCC
82. C GEN2PTS: TO GENERATE TEST DATA SET IN ARRAY PTSET.
83. C WHICH CONTAINS M NUMBER OF POINTS IN N DIMENSIONAL SPACE.
84. C CALLS:
85. (1) F4034: A LIBRARY SUBROUTINE TO GENERATE RANDOM NUMBERS OF
86. NORMAL DISTRIBUTION WITH A GIVEN STANDARD
87. DEVIATION.
88. (2) RANDOM: TO GET ANOTHER RANDOM NUMBER TO BE USED AS FIX
89. STANDARD DEVIATION FOR ONE COLUMN.
90. CCCECCCC
91.  SUBROUTINE GENPTS(PTSET, M,N)
92.  INTEGER PTSET(I,100)
93. DO 100 J = 1, N
94. CALL RANDOM(U)
95. STD = 9.0*U*3
96. DO 100 I = 1, M
97. CALL F4034(STD,X)
98. PTSET(I,J) = INT(X + .5)
99. 100 CONTINUE
100 RETURN
END

CCCECCCC
105. C PROCES: COMPUTES THE MEASURE OF PRIORITY P(I)
106. C FOR EACH COLUMN I WHERE I = 1,....,N FOR A GIVEN POINT SET
107. C IN ARRAY PTSET CONTAINING M POINTS IN N DIMENSIONAL SPACE.
108. C CALLS:
109. (1) H40DER: TO COMPUTE P(I) FOR ONE PARTICULAR VALUE OF I.
110. C
111. CCECCCC
112.  SUBROUTINE PROCES(PTSET, P, M,N)
113.  INTEGER PTSET(I,100), TEMPR(I, P)(6)
114. DO 100 I = 1, N
115. DO 200 J = 1, M
116. TEMPR(J) = PTSET(J, I)
117. P = TEMPR(J, 1)
118. CALL H40DER(I, TEMPR, P)
119. 100 CONTINUE
120 RETURN
121. END
122. CCCCCCCCC
123. C MODER(1,TEMPP,M,P): COMPUTES P(I) FOR A COLUMN VECTOR TEMPR
124. C AND USES HEAP STRUCTURE.
125. C CALLS:
126. C (1) REHEAP: TO MAKE HEAP OF ELEMENTS OF ARRAY TEMPR.
127. C
128. CCCCCCCCC
129. CCCCQUIRE HFODER(I,TEMPP,M,P)
130. INTEGER TEMPP(110),P(6)
131. INTEGER TEMP, ELEMENT, PSUM, SUM
132. M = M
133. L = M/2 + 1
134. 10 CALL REHEAP(L,M,TEMPP)
135. IF(L.GT. 11 GO TO 10
136. ELEMENT = 0
137. PSUM = 0
138. SUM = 0
139. TEMP = TEMPP(L)
140. TEMPP(1) = TEMPP(M)
141. IF(ELEMENT .GT. TEMP) GO TO 30
142. PSUM = PSUM + SUM
143. ELEMENT = TEMP
144. SUM = 1
145. GO TO 20
146. 20 SUM = SUM + 1
147. 30 M = M - 1
148. IF(M.GT. 0) GO TO 50
149. 50 CALL PHEAP(L,M,TEMPP)
150. GO TO 20
151. END
152. CCCCCCCCC
153. C REHEAP(L,M,TEMPP): MAKES HEAP OF ELEMENTS OF TEMPR STARTING
154. C FROM TEMPP(L) TO TEMPP(M).
155. C
156. CCCCCCCCC
157. SUBROUTINE REHEAP(L,M,TEMPP)
158. INTEGER TEMPP(110)
159. INTEGER FLAG, X
160. IT = L
161. JJ = 2*IT
162. X = TEMPP(JJ)
163. FLAG = 1
164. IF(JJ.GT. M1) RETURN
165. IF(JJ.GE. M1) GO TO 10
166. IF(X.LE. TEMPP(JJ)) GO TO 20
167. IF(JJ.GE. TEMPP(JJ)) GO TO 20
168. TEMPP(IT) = TEMPP(JJ)
169. JJ = JJ
170. IF(JJ.GE. M1) RETURN
171. IF(JJ.GE. M1) GO TO 10
172. IF(X.GE. TEMPP(JJ)) GO TO 20
173. IF(X.GE. TEMPP(JJ)) GO TO 20
174. IF(JJ.LT. TEMPP(JJ)) GO TO 20
175. IT = JJ
176. JJ = 2*IT
177. TEMPP(JJ) = X
178. GO TO 20
179. FLAG = 0
180. IF(JJ.GE. M1) AND. FLAG .EQ. 1) GO TO 40

15
SUBROUTINE SORT(P, O, N)
    INTEGER D(6), P(O)
    INTEGER FLAG, TOP, TEMP, BOTTOM
    FLAG = 1
    BOTTOM = N
    10 IF(FLAG .NE. 0 .AND. TOP .LT. BOTTOM) GO TO 20
    RETURN
    20 FLAG = 0
    IF TOP = BOTTOM - 1
    GO TO 100
    201 IF (P(1) .GE. P(1+1)) GO TO 100
    202 X = P(1)
    P(1) = P(1+1)
    203 P(1+1) = X
    204 TEMP = P(1)
    205 P(1) = P(1+1)
    206 P(1+1) = TEMP
    207 IF (FLAG .NE. 0) GO TO 100
    208 FLAG = 1
    209 CONTINUE
    BOTTOM = BOTTOM - 1
    GO TO 10
    RETURN
SUBROUTINE SORTPTSET(P, O, M, N)
    INTEGER PTSET(110*61, D(6))
    M = M
    L = L + 1
    CALL REHIP(M, M, N, O, PTSET)
    CALL EXCHAN(M, M, O, PTSET)
    IF (M .GT. L) GO TO 20
    RETURN
    END
SUBROUTINE REHIP(M, M, N, O, PTSET)
    M = M
    CALL EXCHAN(M, M, O, PTSET)
    IF (M .GT. L) GO TO 20
    END
C

SUBROUTINE REMIP(L,M,N,N,O,PTSET)
  INTEGER PTSET(I10,6),D16
  INTEGER FLAG, X
  J = 2#1
  X = M1
  DO 5 LL=1,N
  5 PTSET(X,LL) = PTSET(I,LL)
  FLAG = 1
  IF(J .GT. M1) RETURN
  DO 10 I = 1,L
  IF(J .GE. M1) GO TO 10
  CALL COMPARE(J,J+1,N,PTSET,ICOMP)
  IF (ICOMP .LT. 0) J = J+1
  CALL COMPARE(J,J+1,N,PTSET,ICOMP)
  IF (ICOMP .GE. 0) GO TO 20
  DJ 100 LL = J/N
  GO 100 PTSET(J,LL) = PTSET(I,LL)
  I = J
  J = 2#1
  GO 200 LL = I,N
  GO TO 30
  20 FLAG = 0
  GO 30
  IF(J .LE. M1 .AND. FLAG .EQ.1) GO TO 40
  RETURN
  END
  C

SUBROUTINE COMPARE(J,N,O,PTSET,ICOMP)
  INTEGER PTSET(I10,6),D16
  IF(J = 1) RETURN
  DO 100 I(L) = D111
  IF (PTSET(I,I+1).LT. PTSET(J,1)) ICOMP = -1
  100 I(L) = D111
  IF (PTSET(I,1).GE. PTSET(J,1)) ICOMP = 1
  RETURN
  END

SUBROUTINE EXCHANGE(J,N,PTSET)
  INTEGER PTSET(I10,6)

C

END
YANG, CHANG AND SINGH

```fortran
DO 100 II = 1,N
   ITEMP = PTSET(II)
   PTSET(II) = ITEMP
100 CONTINUE
RETURN
END

SUBROUTINE ENCODE(MN,N,O,PTSET,NUM,IFL)
   INTEGER PTSET(I),O(I),CODE(I)
   IF(IFL.EQ.1) GO TO 44
   PRINT 1
   PRINT 5,NM
   FORMAT(1X,GM*,12?I),* CODES ARE:
   IF(NM.EQ.1) GO TO 10
   IF(NM.EQ.0) GO TO 20
   NUM = 0
   30 J = M-1
   W = W + 1
   1 J = 1
   DO 100 II = 2,M
      IF(II.GT.W) GO TO 40
      FF = 0
      JD 700 JJ = J,NI
      JJ = O(JJ)
      IF(PTSET(II,JJ) .EQ. PTSET(I,JJ)) GO TO 200
      FF = 1
   40 CONTINUE
   50 J = J + 1
      GO TO 40
   100 CONTINUE
   30 IF(FLAG.EQ.1) GO TO 40
   40 DO 300 JJ = 1,NI
      JD = N - NI
      JJ = O(JJ)
      CODE(JJ) = PTSET(I,JJ)
   300 CONTINUE
   L = N
   NN = N - NI
   40 DO 400 JJ = NN,N
      CALL MINMAX(PTSET,M,JJ)
      L = L + 1
      CODE(JJ) = MIN
   400 CODE(NN) = MAX
   492 = L
   N3 = L + NN
```

NRL REPORT 8382

161. IF(FL,8Q,1) GO TO 85
162. IF(FL,8Q,1) GO TO 85
163. PRINT 11,(CODE(1),I=1,N1)
164. 11 FORMAT(1X,10X,1'H1313')
165. PRINT 13,(CODE(1),I=M,N2)
166. 13 FORMAT(1X,10X,1'H1313')
167. N22 = N2+1
168. PRINT 15,(CODE(1),I=22,N3)
169. 15 FORMAT(1X,33X,1'H1313')
170. PRINT 16
171. 16 FORMAT(1X,44X,1'H1313')
172. 85 I = I + 1
173. NUM = NUM + 1
174. 100 CONTINUE
175. RETURN
176. 10 DO 101 I = 1,N
177. 10 J = O(I)
178. CALL H1MAX(PTSET,MM,JJ,1,MN,MX)
179. CODE(I) = MIN
180. CONTINUE
181. 101 I = I + 1
182. IF(FL,8Q,1) RETURN
183. 8Q PRINT 90,(CODE(1),I=NN,N2)
184. 90 FORMAT(1X,24X,1'H1313')
185. PRINT 93
186. 93 FORMAT(1X,38X,1'H1313')
187. RETURN
188. 20 DO 200 I = 1,M
189. 21 DO 220 J = 1,N
190. 220 JJ = O(J)
191. IF(FL,8Q,1) GO TO 222
192. 22 FORMAT(10X,10X,1'H1313')
193. PRINT 22
194. 23 FORMAT(1X,24X,1'H1313')
195. 230 CONTINUE
196. 230 RETURN
197. END
198. C CCCCCCCCC
199. C MINMAX(PTSET,M,ICOL,1,J,MN,MX) TO FIND THE MINIMUM &
200. C MAXIMUM ELEMENTS OF COLUMN ICOL,STARTING FROM 1 TO J IN ARRAY
201. C PTSET.
202. C PTSET.
203. C CCCCCCCCC
204. C MINMAX(PTSET,M,ICOL,1,J,MN,MX)
205. INTEGER PTSET(110,64)
206. MIN = PTSET(1,ICOL)
207. MAX = PTSET(1,ICOL)
208. DO 100 I = 1,J
209. IF (PTSET(I,ICOL) LT MIN) MIN = PTSET(I,ICOL)
210. IF (PTSET(I,ICOL) GT MAX) MAX = PTSET(I,ICOL)
211. 100 CONTINUE
212. RETURN
213. END

19
YANG, CHANG AND SINGH

HEURIS(PTSET,M,N,NCODE,IFL):- TO STORE THE NUMBER OF GHI ENCODED TUPES FOR M=1,2,...,N+1 IN ARRAY NCODE, THE SET OF HANDLES

HEURIS(PTSET,M,N,NCODE,IFL):-

CALL PROCES(PTSET,M,N,I)

CALL SORT(PTSET,M,N,I)

CALL ENCODV(PTSET,M,NCODE,I)

INTEGER PTSET(110,6),0(6),P(16),NCODE(5,3)

1 FORMAT(//,'PRINT OUT OF ARRAY P IS:')

2 FORMAT(10X,3110)

2 DO 200 I = 1,N

10 200 I FORMAT(10X,413)

CALL ENCODV(PTSET,M,NCODE,I)

IF(I.EQ.1) GO TO 44

DO 300 J = 1,M

300 IF(J.EQ.I) GO TO 400

IF(J .LE. I) OR. I .GE. N) GO TO 400

NCODE(I+1,1) = NUM

NCODE(1+1,2) = NUM

NCODE(I+1,3) = NUM

400 CONTINUE

RETURN

END

EXHAUS(PTSET,M,N,NCODE,IFL):- TO STORE THE MINIMUM NUMBER OF GHI ENCODED TUPLES FOR M=1,2,...,N+1 IN ARRAY NCODE, ALL THE POSSIBLE SET OF HANDLES ARE USED TO GENERATE GHI ENCODED TUPES & THE CODES.

CALLS:

CALL (1) LOAD: TO FIND ALL POSSIBLE SETS OF HANDLES;

CALL (2) SORT: TO SORT THE POINT SET W.R.T. THE CURRENT SET OF HANDLES;

CALL (3) ENCODE: TO GENERATE GHI ENCODED TUPLES.
481. 20  RETURN
482. 10  DO 222 1 = 1,N
483. 222  JJ = 0(I)
484. 20  CALL MINMAX(PTSET,ORDER,JJ,1,NPTS,MIN,MAX)
485.  CODE(1) = MIN
486.  CODE(I+N) = MAX
487. 222  CONTINUE
488.  I1 = DENSITY(CODE,N,1,NPTS)
489.  IF(DEN*GE*THRESH) GOTO 444
490.  NUM = 0
491.  RETURN
492. 444  IF (FLG EQ. 1) GO TO 666
493.  PRINT 21,(CODE(I)*1=I,N)
494.  FORMAT(0'*IOX98(6I3)
495.  NI = N+1
496.  N2 = N*N
497.  PRINT 22,(CODE(I),I=M,J)
498.  FORMAT(*+I30X+I613)
499.  PRINT 23
500.  FORMAT(*+,50X+)
501.  666  NUM = 1
502.  RETURN
503.  END
504.  CCCC
505.  C MINMAX(PTSET,ORDER, JJ,1,J,MIN,MAX): FIND THE MINIMUM & MAXIMUM
506.  C OF ELEMENTS OF COLUMN JJ, STORED IN PTSET, THEIR POINTERS ARE
507.  C STORED IN ARRAY ORDER STARTING FROM I TO J.
508.  CCCC
509.  10  SUBROUTINE MINMAX(PTSET,ORDER, JJ,1,J,MIN,MAX)
510.   INTEGER PTSET(110,6),ORDER(110)
511.   11 = ORDER(I)
512.   MIN = PTSET(II, JJ)
513.   MAX = PTSET(II, JJ)
514.   JO = 100 K = 1,J
515.   11 = ORDER(K)
516.   IF(PTSET(I,JJ) .LT. MIN) MIN = PTSET(I,JJ)
517.   IF(PTSET(I,JJ) .GT. MAX) MAX = PTSET(I,JJ)
518.   CONTINUE
519.   RETURN
520.  END
521.  CCCC
523.  C GENERATED BY GHM ENCODED TUPLE STORED IN CODE WHICH INCLUDES
524.  C POINTS POINTED TO BY POINTERS IN ORDER STARTING FROM I TO J.
525.  CCCC
526.  50  FUNCTION DENSITY(CODE,MM,N,1,J)
527.   INTEGER CODE(12)
528.   NUMPTS = J-1 +1
529.   VOL = 1
530.   M1 = MM +1
531.   JO = 100 K = M1,N
532.   VOL = VOL*(CODE(K+N-MM)-CODE(K) +1)
533.   CONTINUE
534.   DENSITY = NUMPTS / VOL
535.   RETURN
536.  END
537.  CCCC
538.  CCCC
539.  21
L = L + ITERM
I = I + 1
LI = L - I
IF (I .GT. LI) GO TO 20

DO 500 IZ = LI + 1, LI
500 CONTINUE

TABLE(IZ,COL) = TABLE(I1,COL)

IF (I .GT. N11) GO TO 400

CALL FILL(TABLE,COL,N11,L)

CONTINUE

IF (NUM .GE. 2) GO TO 10

RETURN

END

CCCCCCCCC
559.
560. C FILL(TABLE,COL,N11,L1)- TO STORE SOME ELEMENTS IN LTH ROW OF
561. C ARRAY TABLE.
562. C

SUBROUTINE FILL(TABLE,COL,N11,L1)

INTEGER TABLE(720,6)
INTEGER COL, COL1

IX = TABLE(I1,COL)

ON 100 I = COL1,N

100 TABLE(I1-I) = TABLE(I1,I)

RETURN

END
APPENDIX B

```fortran
/*JHMPLA4 T=(2,0),L=4
*/ EXEC FORTRAN
/*FORTRAN,SYIN DD * 
****CCCCCCCCCCC
  C
  C MAIN PROGRAM TO OPTIMIZE GH ENCODING FOR A POINTSET,
  C SUCH THAT EACH HYPERCUBE HAS DENSITY NO LESS THAN A GIVEN
  C THRESHOLD DENSITY.
  C
  C PTSET: ARRAY CONTAINING POINTSET.
  C THRESH: THRESHOLD DENSITY.
  C M: NO. OF POINTS IN POINT SET.
  C N: DIMENSIONAL SPACE.
  C FLG: FLAG USED TO TURN OFF & ON PRINTING OF INTERMEDIATE
  C RESULTS.
  C
  C CALLS :
  C (1) GENPTS: TO GENERATE M POINTS IN N DIMENSIONAL SPACE &
  C STORE THEM IN ARRAY PTSET.
  C (2) OPTMUM: TO GENERATE OPTIMAL GH ENCODED TUPLES, AND
  C RETURNS THEIR NUMBER IN NCODE. IF FLG IS ON, PRINTS
  C THOSE GH ENCODED TUPLES ALSO. THRESH IS THE THRESHOLD
  C DENSITY SUCH THAT EACH HYPERCUBE HAS DENSITY NO LESS
  C THAN THRESH.

  C 24. CCCCCCCCC
  C INTEGER PTSET(110,6),TABLE(11)
  C
  N = 3
  FLG = 1
  1 D(J) = 10000
  CALL GENPTS(PTSET,M,N)
  CALL OPTMUM(PTSET,M,N,THRESH,NCODE,FLG)
  CALL PRINT(PTSET,M,N,THRESH,NCODE)

  C
  C 24. CCCCCCCCC
  C INTEGER PTSET(110,6),TABLE(11)
  C
  N = 3
  FLG = 1
  1 D(J) = 10000
  CALL GENPTS(PTSET,M,N)
  CALL OPTMUM(PTSET,M,N,THRESH,NCODE,FLG)
  CALL PRINT(PTSET,M,N,THRESH,NCODE)
```

23
YANG, CHANG AND SINGH

STLi6P

CCCCC CCCCCCCLC

PANLJO4(X): (NERATE A RANDM)CM 14UMUF-R

WITH1 C UNIFORM IISTR~cfUTIUN.

CCCCC CCCC CTCCCC CCCC CCCC

ULE3uUT INERANLUM(X) PE.A.L X 114TEGFP A/19727/.

MULT/2b211/.

EASE/J:7GE/ A = MO0(MUL7*A~BASE)

X = FLOAT(A)/FLOAT(BASE)

RETURN END

CCCCCCCCCC

CALLS:

(1) RANDUM: TO GET A RANDCM NUMBER.

CCCCCCCCC

SUBROUTINE RANDUM(X)

REAL X

INTEGER A/19727/,* MULT/25211/* EASE/32768/ A = MOD(MULT+BASE)

X = FLOAT(A)/FLOAT(BASE)

RETURN END

CCCCCCCCCC

GENPTS(PTSET, M,N):- TO GENERATE TABLE DATA SET IN ARRAY

PTSET WHICH CONTAINS M NUMBERS IN N DIMENSIONAL

SPACE.

CALLS:

(1) RANDUM: TO GET A RANDCM NUMBER.

CCCCCCCCCC

SUBROUTINE GENPTS(PTSET, M,N)

INTEGER PTSET(100,6)

DO 100 J = 1, N

DO 100 I = 1, M

CALL RANDUM(U)

PTSET(I,J) = INT(9 * U) + 1

100 CONTINUE RETURN END

CCCCCCCCCC

OPTMUM(PTSET, M,N,THRESH, NCODE, FLG):- TO GENERATE OPTIMAL GHENn1NG FOR A GIVEN THRESHOL

DENSITY, THRESH; AND RETURNS THE NUMBER OF SUCH TUPLES

IN NCODE.

CALLS:

(1) HC'REC: TO COMPUTE THE MEASURE OF PRIORITY P(I) FOR

EACH COLUMN I:

(2) SORT: TO GET THE PROPER SET OF HANDLES CORRESPONDING

TO BEST VALUES OF THEIR PRIORITIES, P(I).

(3) SORTPT: TO SORT POINTSET IN PTSET AND SET OF

HANDLES.

(4) LNCU3: TO GENERATE GHM EXCLUDED TUPLES WITH DENSITY

NO LESS THAN THRESHOLD DENSITY.

CCCCCCCCC

SUBROUTINE OPTMUM(PTSET, M,N,THRESH, NCODE, FLG)

INTEGER PTSET(110,6)+P(6),ID(6),GRDER(11101)

INTEGER FLG

NCODE = 0

DC 100 I = 1, M

100 IF (ID(I) = 1) THEN

MPTS = M

IDM = 1

110 IF (NCODE = 0) THEN

GO TO 100

ELSE

IDM = IDM + 1

END IF
NRL REPORT 8382

500  D(I) = 1
121. IF (MM .GT. 1) GOTO 80
122. CALL ENCODE(MM,N,PTSET,ORDER,D,NPTS,IFL,NUM,THRESH,FLG)
123. IF (NUM .EQ. 0) GOTO 70
124. NCODE = NUM
125. RETURN
126. 70 MM = MM + 1
127. 80 CALL PROCE5(PTSET,ORDER,D,NPTS,N)
128. IF (FLG .EQ. 1) GO TO 30
129. CALL SORT( P,D,N)
130. IF (NUM .LE. 0) GOTO 70
131. NCOD = NUM
132. RETURN
133. 70 MM = MM + 1
134. 80 CALL ENCODE(MM,N,PTSET,ORDER,D,NPTS,IFL,NUM,THRESH,FLG)
135. NCOD = NCOD + NUM
136. IF (NCOD .EQ. 0) GO TO 200
137. CONTINUE
138. 200 NPTS = NP
139. IF (ORDER(1) .EQ. 0) GO TO 200
140. ORDER(NP) = ORDER(1)
141. CONTINUE
142. 200 NPTS = NP
143. IF (NPTS .EQ. 1) GOTO 20
144. IF (NPTS .LE. 0) RETURN
145. NCOD = NCOD + NUM
146. IF (NPTS .GT. 0) GOTO 20
147. RETURN
148. IF (NUM .GT. 1) AND. (IFL .EQ. 1) MM = MM + 1
149. IF (NUM .LT. 1) AND. (IFL .EQ. 0) GO TO 20
150. GO TO 110
151. 20 IF (FLG .EQ. 1) GOTO 50
152. PRINT 5, MM
153. 5 FORMAT('/I0X,' HANDLE LENGTH = ', IS)
154. PRINT 6, ORDER(1), I = 1, MM
155. 6 FORMAT('/I0X,' HANDED ARE ':,615)
156. PRINT 7, FORMAT('/I0X,' CORRESPONDING GM CODES ARE:'),
157. 7 FORMAT('/I0X,' CORRESPONDING GM CODES ARE:'),
158. 50 NCOD = NCOD + NPTS
159. IF (FLG .EQ. 1) RETURN
160. 333 J1 = 1
161. 333 J1 = ORDER(1)
162. GO 333 J1 = N
163. JJ = 0(J)
164. 333 P(JJ) = PTSET(I1,J)
165. PRINT 8, (P(K), K = 1, MM)
166. 8 FORMAT('/I0X,'*,6515*,')
167. IF (NUM .GT. 0) GOTO 300
168. NJ = MM + 1
169. PRINT 9, (P(K), K = 1, N)
170. 9 FORMAT('/I0X,'*,6515*,')
171. PRINT 10, (P(K), K = NJ, H)
172. 10 FORMAT('/I0X,'*,6515*,')
173. PRINT 11
174. 11 FORMAT('/I0X,'*,6515*,')
175. 300 CONTINUE
176. RETURN
177. CONTINUE
178. 300 CONTINUE
179. C PROCES5(PTSET,ORDER,D,NPTS,N):= COMPUTES THE MEASURE OF PRIORITY
FOR EACH COLUMN I WHERE I=1,...,N FOR A GIVEN POINT SET IN ARRAY PTSET CONTAINING NPTS NO. OF POINTS IN N DIMENSIONAL SPACE.

CALLS:
(i) HPORDER TO COMPUTE P(I) FOR ONE PARTICULAR COLUMN I.

SUBROUTINE HPORDER(PTSET,ORDER,P,NPTS,N)

DO 100 I = 1,N
  DO 200 J = 1,NPTS
    JJ = ORDER(J)
  200 TEMP(J) = PTSET(JJ,1)
  100 CONTINUE
  RETURN
END

SUBROUTINE HPOORDER(I,TEMPR,NPTS,P)

INTEGER PTSET(100,6),ORDER(100),TEMPR(100),P(6)

M = N/2 + 1
L = M-1
CALL REHEAP(L,M,TEMPR)

10 L = L-1
  CALL REHEAP(L,M,TEMPR)
  IF (L .GE. M) GOTO 40
  IF (TEMPR(I) .GT. TEMP) GOTO 10
  R ELEMENT = 0
  PSUM = 0
  SUM = 0
  TEMP = TEMPR(I)
  TEMPR(I) = TEMPR(M)
  IF (TEMPR(I) .GT. TEMPR(M)) GOTO 30
  SUM = SUM + 1
  IF (SUM .GE. M) GOTO 40
  IF (M .GT. 0) GOTO 50
  IF (SUM .GE. M) GOTO 40
  SUM = SUM + 1
  M = M - 1
  IF (M .GT. 0) GOTO 50
  IF (P(I) .GE. SUM) GOTO 20
  IF (P(I) .GE. SUM) GOTO 20
  P(I) = P(I)
  RETURN
20 SUM = SUM + 1
30 M = M + 1
40 L = L + 1
50 M = M - 1
60 CALL REHEAP(L,M,TEMPR)
70 IF (P(I) .GE. SUM) GOTO 20
80 IF (P(I) .GE. SUM) GOTO 20
90 P(I) = P(I)
100 RETURN
END

SUBROUTINE REHEAP(L,M,TEMPR)

M = M - 1
RETURN
END

SUBROUTINE REHEAP(L,M,TEMPR)

M = M - 1
RETURN
END
241. \( i = l \)
242. \( j = 2+i \)
243. \( x = \text{TEMPR}(i) \)
244. \( \text{FLAG} = 1 \)
245. IF \((j < x \text{ AND } x \geq \text{MI})\) RETURN
246. IF \((\text{TEMPR}(j) < \text{TEMPR}(j+1))\) \(j = j+1\)
247. IF \((x \leq \text{TEMPR}(j))\) GOTO 20
248. \(\text{TEMPR}(i) = \text{TEMPR}(j)\)
249. \(i = j\)
250. \(j = 2+i\)
251. \(\text{TEMPR}(i) = x\)
252. GOTO 30
253. RETURN
254. END

```
C
SORT(P.O.N) :- SORTS P ARRAY CONTAINING N ELEMENTS AND DURING THIS SORT ALSO MOVES THE ELEMENTS OF ARRAY P IN THE SAME FASHION SO THAT IN ORDER TO CHOOSE A HANDLE SET OF M LENGTH, WE CAN CHOOSE FIRST M-1 HANDLES STORED IN FIRST M-1 POSITION OF P.

SUBROUTINE SORT(P,O,N)
   INTEGER O(6),P(6)
   INTEGER FLAG, TOP, TEMP, BOTTOM
   FLAG = 1
   BOTTOM = N
   TOP = 1
   IF(FLAG .NE. 0 .AND. TOP .LT. BOTTOM) GOTO 20
   RETURN
   FLAG = 0
   IF(FLAG .NE. 0 .AND. TOP .LT. BOTTOM) GOTO 20
   RETURN
   IF(FLAG .NE. 0 .AND. TOP .LT. BOTTOM) GOTO 20
   RETURN
   END
```

```
C
HEAPSORT IS USED HERE, THEREFORE:
CALLS:
(1) REHEP: TO MAKE HEAP.
(2) COMPAR: TO COMPARE ORDER OF TWO POINTS.
```
YANG, CHANG AND SINGH

301. C (3) EXCHANGE TO EXCHANGE POSITION OF TWO POINTS IN PTSET.

302. C

303. CCCCCC

304. SUBROUTINE EXCHANG(PTSET, ORDER, M, N, M2)

305. INTEGER PTSET(I10,6), ORDER(I10)

306. M1 = M

307. L = M1/2 + 1

308. CALL REHIP(L, M1, M2, N, PTSET, ORDER)

309. IF (L > GT, 1) GO TO 10

310. CALL EXCHANG(ORDER(1), ORDER(M1), N, PTSET)

311. M1 = M1 - 1

312. CALL REHIP(1, M1, M2, N, PTSET, ORDER)

313. IF (M1 > GT, 1) GO TO 20

314. RETURN

315. END

316.

317. CCCCCC

318. C

319. C REHIP(L, M1, M2, N, PTSET, ORDER): - MAKES A HEAP STRUCTURE

320. C OF THOSE POINTS IN PTSET WHOSE POINTERS ARE IN ARRAY ORDER.

321. C STARTING FROM ORDER(L) TO ORDER(M1).

322. C M IS THE NUMBER OF POINTS IN ORIGINAL PCINTSET

323. C

324. CCCCCC

325. SUBROUTINE REHIP(L, M1, M2, N, PTSET, ORDER)

326. INTEGER PTSET(I10,6), ORDER(I10)

327. INTEGER FLAG *X

328. JJ = 2 * I

329. J = ORDER(I)

330. DO 5 LL = 1, N

331. FLAG = 1

332. JJ = ORDER(J)

333. DO 100 LL = 1, N

334. PTSET(I*LL) = PTSET(J*LL)

335. IF (JJ > GT, M1) RETURN

336. 100 PTSET(I*LL) = PTSET(J*LL)

337. CALL COMPAR(JJ, ORDER(JJ), ORDER(ORDER(JJ)+1), N, PTSET, ICOMP)

338. IF (ICOMP > LE, 0) JJ = JJ+1

339. CALL COMPAR(JJ, ORDER(JJ), ORDER(ORDER(JJ)), N, PTSET, ICOMP)

340. IF (ICOMP > GE, 0) GO TO 20

341. JJ = ORDER(J)

342. 5 JJ = ORDER(I)

343. DO 100 LL = 1, N

344. PTSET(I*LL) = PTSET(J*LL)

345. IF (JJ > EQ, M1) AND. FLAG = 1) GO TO 40

346. RETURN

347. END

348.

349. CCCCCC

350. C

351. C COMPAR(I, J, N, PTSET, ICOMP): - COMPARES TWO POINTS I & J W.R.T. A

352. C HANDLE VECTOR IN D, AND RETURNS IN ICOMP.

353. C

354. C

355. C

356. C

357. C

358. C

359. C

360. C

28
CCCRCCCCC

SUBROUTINE COMPARE(I,N,O,PSET,ICOMP)
INTEGER PSET(110,6),O(6)
II = 1
100 II = II+1
IF (PSET(I,II) .GE. PSET(J,II)) GO TO 10
IF (I .LT. N) GO TO 5
ICOMP = 0
RETURN
5 II = II+1
GO TO 100
10 IF (PSET(I,II) .LT. PSET(J,II)) ICOMP = -1
IF (PSET(I,II) .GT. PSET(J,II)) ICOMP = 1
RETURN
END

CCCCCCC

SUBROUTINE EXCHANGE(I,J,N,PSET)
INTEGER PSET(110,6)
DO 100 II = 1,N
ITEMP = PSET(I,II)
PSET(I,II) = PSET(J,II)
PSET(J,II) = ITEMP
100 CONTINUE
RETURN
END

CCCCCCC

SUBROUTINE ENCODE(M,N,PSET,ORDER,O,NPTS,IFL,NUM,THRESH,FLG)
INTEGER PSET(110,6)
IF (FLG .EQ. 1) GO TO 70
PRINT 1
1 FORMAT(//)
PRINT S
70 FORMAT(*GH*V*CODES ARE:*)
PRINT 6,10(1),1=1,MM)
FORMAT(4*10X,'HANDLES ARE: '+6I5)
IF(MM.EQ.0) GO TO 20
IF (MM.EQ.0) GO TO 10
NUM = 0
IFL = 0
M1 = NPTS +1
I = 1
J = 1
DO 100 K = 2,M1
IF (K.GT. NPTS) GO TO 40
FLAG = 0
II = ORDER(K)
II = ORDER(K+1)
UU = 200 JJ = 1,MM
JJ = O(JJ)
IF(PTSET(I,JJ) .EQ. PTSET(I2,JJ)) GO TO 200
FLAG = 1
GO TO 330
CONTINUE
200 CONTINUE
IF (FLAG .EQ. 1) GO TO 40
J = J + 1
GO TO 100
40 II = ORDER(I)
DO 300 JJ = 1,MM
JJ = O(JJ)
CODE(JJ) = PTSET(I,JJ)
CONTINUE
300 CONTINUE
NI = MM+1
N2 = L = NN
DO 400 JJ = 1,N
JJ = O(JJ)
CALL MINMAX(PTSET,ORDER,II,JJ,MIN,MAX)
L = L + CODE(JJ)
MAX = CODE(JJ)
DEN = DENSITY(CODE,MM,N2;I,J)
IF(DEN .GE.TRESH) GO TO 111
IFL = I
GO TO 555
400 CONTINUE
111 IF (J .EQ. J) GO TO 555
N2 = L
N3 = L + NN
DO 333 K = 1,J
333 ORDER(K) = 0
IF (IFL .EQ. 1) GO TO 72
PRINT 32,(CODE(I),I=1,MM)
FORMAT('D\10X,*1,E13)
PRINT 33,(CODE[I],I=1,N2)
FORMAT('D\10X,*1,E13)
N22 = N2+1
PRINT 34,(CODE[I],I=N22,N3)
FORMAT('D\10X,*1,E13)
PRINT 35
FORMAT('D\10X,*1,E13)
72 NUM = NUM + 1
78 I = K
79 J = K
80 CONTINUE
SUBROUTINE EXHAUST(PTSET,M,N,NCODE,IFL)
INTEGER PTSET(100,6),O(6),TABLE(720,6),NCODE(5,3)
CALL &PROMBTABLE,N
ITEM = 1
DO 100 I = 1,N
ITEM = ITEM + 1
DO 200 J = 1,N
DO 300 J = 1,N
IF(I1FL .EQ. 1) GO TO 88
PRINT I
FORMAT(//,1X,'HANDLES CHOSEN ARE:');
PRINT 2,(O(J,J=1,N)
FORMAT(10X,4.I1)
CALL SORTPT(PTSET,O,M,N)
IF(I1FL .EQ. 1) GO TO 89
PRINT 3
FORMAT(//,1X,'FOLLOWING ARE GH CODES W.R.T. ABOVE:');
N1 = N + 1
DO 500 J = 1,N
CALL ENCODE(J,N,N,PTSET,N1,IFL)
IF(J .LT. I) OR. J .GE. N1) GO TO 500
IF(NUM .LT. NCODE(J-1,3)) NCODE(J-1,3) = NUM
IF(NUM .LT. NCODE(J-1,3)) NCODE(J,3) = NUM
500 CONTINUE
IF(I1FL .EQ. 1) GO TO 200
PRINT 4
FORMAT(1001)',')
513 CONTINUE
RETURN
END

SUBROUTINE PRCOMBTABLE,N
INTEGER TABLE(720,6)
INTEGER COL
NUM = N
DO 100 I = 1,N
K = 1
COL = 1
L = 1
NJ = NUM - 1
ITEM = 1
DO 200 K = 1,N
ITEM = ITEM + 1
DO 400 I = 1,N
1) = L
400 CONTINUE
31