ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 6, No. 4, 2011, *pp*. 269-278

A Framework for Reducing Multidimensional Database to Two Dimensions

Adio Akinwale [1], Kolawole Adesina[2] and Olusegun Folorunso[3]

Department of Computer Science, University of Agriculture, Abeokuta, Nigeria

(*Received April 8, 2011, accepted May 17, 2011*)

Abstract. This work used a method of Matrix Decomposition Algorithm to obtain a new dataset of genetic epistasis as a surrogate for a multidimensional dataset which transformed multidimensional database to a 2-dimensional database. It employed decomposition algorithms based on Boyce Codd Normal Form for minimizing anomalies. The decomposition and reversible algorithms were used on relationship among object attributes and were implemented. The implemented program ran on sample genetic epistasis datasets of up to 10 dimensions and it was shown that multidimensional datasets can be reduced to two dimensions. It was established that the time taken to generate a sequence of tuples from multidimensional database to a 2-dimensional dataset was directly proportional to the number of genes considered. The result showed that the reduced 2-dimensional database did not require any in-built functions which take long processing time for generating query result as against querying of multidimensional dataset. The reduced 2-dimensional dataset was reversible to the original multidimensional dataset for lossless join operation which indicated that there was no loss of data values or tuple. The method was compared with existing reduction techniques and it was found that data access was very fast with decomposition algorithm than relational model.

Keywords: Matrix decomposition algorithm, multidimensional database, genetic epistatis, principal component analysis, project pursuit method, relational model

1. Introduction

Nowadays, emerging data are multidimensional in nature. Multidimensional database technology is being applied to distributed data and to new types of data that current technology often cannot adequately analyze. The global reason for the multidimensional database's rise is to facilitate flexible, high performance access and analysis of large volumes of complex and interrelated data [24]. Multidimensional database structure has evolved to match closely the way people visualize data [17]. Thus, people think of their businesses in multidimensional terms. For example, a sales manager will want to know the sales of a particular product over a period of time and may be in a particular region or location. This is a multidimensional view of sales data. The higher the dimensionality, the higher the volume of the data, and consequently, the more complex the data is to manage.

To lessen the scalability problem of processing large datasets, data reduction technique is proffered. Data reduction techniques are useful to reduce the scalability problem of processing large datasets. In order to efficiently archive and process growing multidimensional datasets, the work presents and compares reduction and reversibility technique algorithms that reduce multidimensional database to a two dimensional database and also reverse the two dimensional back to its original form without loss of data.

2. Literature Review

Multidimensional database is stored in such a way as to be represented to the user as a hypercube or multidimensional array, where each core value or fact occupies a cell indexed by a unique set of dimension values. Agrawal et. al. also asserted that multidimensional database is a key technology in the enabling of interactive analyses of large amounts of data for decision-making purposes [1]. It is used to process and analyze large and complex datasets. Multidimensional database originated from the multidimensional matrix

[1] *E-mail address*: aatakinwale@yahoo.com;
[2] *E-mail address*: kolawole_adesina@yahoo.co.uk;
[3] *E-mail address*: folorunsolusegun@yahoo.com

algebra, which has been used for (manual) data analysis since the late 19th century. An example of multidimensional software is On-Line Analytical Processing (OLAP) which supports multidimensional view of data in various ways. Sarawagi summarized OLAP as **F**ast **A**nalysis of **S**hared **M**ultidimensional **I**nformation [27]. Gupta et.al. cited that there is no theoretical limit to the number of dimensions although, current multidimensional database tools experience performance problems when the number of dimensions is more than 10. Multidimensional databases do not support the logical joining of multiple multidimensional arrays. The inability to join multidimensional databases with relational databases, limits query flexibility by eliminating the possibility of using characteristics tables to dynamically segment data [12], [23].

Colliat cited that there are two main approaches to building a multidimensional database. One approach maintains the data as a k-dimensional cube based on a non-relational specialized storage structure for storing k-dimensional data. Another approach uses a relational model wherein operations on the data cubes are translated to relational queries. This means it is posed in a possibly enhanced dialect of structural query language. This approach thus calls for dimensionality reduction [11].

The problem of dimension reduction is introduced as a way to overcome the curse of the dimensionality when dealing with vector data in high-dimensional spaces and as a modeling tool for such data. It is defined as the search for a low-dimensional manifold that embeds the high-dimensional data. It is basically a mapping from a D-dimensional space onto an L-dimensional space, for D > L. Lindsay devised a method that automates the task of projection pursuit. He characterized a given projection by a numerical index that indicated the amount of structure that is present. The index can be used as the basis for a heuristic search to locate the interesting projection [20].

Jensen et. al. considered analyzing multidimensional database with a two-dimensional spreadsheet. However, a table is considerable for only two-dimensional dataset. If there is need to go further to three dimensions, the obvious solution is to use separate worksheet for each extra dimension, with one worksheet for each dimension values and only to some extent [16]. This implies that there will be need to continuously create extra sheets for additional dimension. Analyses involving several values of the extra dimensions are cumbersome, and with many thousands of dimension values, the solution becomes infeasible or complex thus, attention is necessary to resolve the problem.

3. Theoretical Form of the System

A gene consists of 2 alleles – dominant and recessive denoted by **A** and **a** respectively. Each allele has a biological population frequency of $p = q = 0.5$ with genotype frequencies of p^2 for **AA**, 2pq for **Aa** and q^2 for **aa** following Hardy-Weinberg Equilibrium. Alleles **A** and **a** can combine in the following forms: **AA**, **Aa**, **aA**, **aa** [6]. Because the order of the alleles is unimportant, a genotype can have one of 3 values **AA**, **Aa** and **aa**. The penetrance function defines the probability (P) of disease (D) for genotypes (G) for one or more genetic variations, denoted by P[D/G]. Examples are P[D/AA] and P[D/AA,BB]. Thus for **n** genotypes interaction there will be $\mathbf{3^n}$ penetrance functions. The fixed dimensional length n-genotype Single Nucleotide Polymorphism (SNP) gives a total number of $\mathbf{3^n}$ sequences with the same number of penetrances, thus 3 SNP gives $3^2 = 9$ sequences.

Examples: 2 genotypes have 3^2 functions = 9 functions, i.e., 3 x 3 array.

3 genotypes have 3^3 functions = 27 functions, i.e., 3 x 3 x 3 array.

4 genotypes have 3^4 functions = 81 functions, i.e., 3 x 3 x 3 x 3 array.

26 genotypes have 3^{26} functions = 67,108,864 functions, i.e., 3 x 3 x 3 x … x 3 array.

n genotypes have 3^n functions = 3 x 3 x 3 x 3,…, x n array.

Equally, business sales outlet may have interesting data to manipulate and evaluate the efficient of the algorithm. These types of data will have varying dimensions. The achievement was to turn any of these multidimensional arrays into a 2-dimensional list, which has been detailed and described in figure 1a-d.

```
Procedure Generate_Penetrance_List(C: dimensions) is
Begin
Set P = 3^C (the total no of possible penetrance functions)
Call procedure BuildALPHABETS to set up array of alphabets A – Z
Call procedure BuildSNPString[C,3]
Declare cSNP[P] – a list to store penetrance functions
for p in 1 .. P       If p = 1 Then    for j in 1 .. C    Index(j) = 1
loop j      else
for m = C down to 1   If Index(m) <> 3 Then
Index(m) = Index(m) + 1 'increase index
Exit For  else
Index(m) = 1          'set index back to 1
End If     Next m
End If
IndexString = null
For l in 1 .. C                    'Set index for the SNPCombination
sIndex(p, l) = Index(l)
IndexString = IndexString + "," + StringValue(Index(l))
loop l
SNPCombinationIndex(p) = Mid(IndexString, 2, Length(IndexString) - 1)
SNPString = null
For i in 1 .. C     SNPString = SNPString & SNP(i, Index(i))
loop i
SNPCombination(p) = SNPString
Get the penetrance value for SNPString
Store the SNPString and its penetrance function
loop p'
    End Generate_Penetrance_ List
```

Figure 1a: The Structure of Decomposition Algorithm

```
Procedure BuildALPHABETS is
    Begin
    Declare ALPHA(26) as an array of strings
    for a in 1 .. 26
    ALPHA(a)= ALPHABET_a
    loop a
        End BuildALPHABETS
```

Figure 1b: The Decomposition Algorithm to call Build Alphabets Procedure

4. Methodology

4.1. Principal Component Analysis

Fordor (2002) stated that principal component analysis is the best in respect to it's mean square error which can be used to reduce the number of dimensional dataset. It is a statistical method that involves mean, variance, standard deviation, covariance, eigenvectors and eigenvalues. The eigenvectors with highest eigenvalues is the principal component. This technique is not possible to compare the same dataset with other techniques.

```
Procedure BuildSNPString(C: dimensions, 3: dimension cardinality) is
    Begin
    Declare SNP(C,3) as array of strings
    for i in 1 to C;  for j in 1 to 3
    CAP = ALPHA(i)
    Select Case j:
    Case 1:  SNP(i,j) = CAP + CAP
    Case 2:  SNP(i,j) = CAP + lower(CAP)
    Case 3:  SNP(i,j) = lower(CAP + CAP)
    End Select
    Loop j ;  loop i
    End BuildSNP
```

Figure 1c : The Decomposition Algorithm to call Build SNP String Procedure

Input: tables R and set of functional dependencies for R
1. compute keys of R based on functional dependencies
2. until no more Boyce Codd Normal Form violations
3. pick any R' with AA→BB that violate Boyce Codd Norma Form
4. decompose R' into R1(AA, BB) and R2(AA, CC) where CC is all attribute in R' except (AA U BB)
5. compute functional dependencies for R1 and R2
6. compute keys for R1 and R2 based on the functional dependencies

Figure 1d: Decomposition Algorithm based on Boyce Codd Normal Form

4.2. Projection Pursuit Method

Carreira-Perpinan (2001) stated that Projection Pursuit is an unsupervised technique that picks interesting low-dimensional linear orthogonal projections of a high-dimensional point cloud by optimizing a certain objective function called projection index. It is typically used to take profit of the human ability to discover patterns in low-dimensional of one dimensional to three dimensional projections. The idea of a projection pursuit is to locate the projection or projections from high to low dimensional space that reveal the most details about the structure of the dataset. Once an interesting set of projections has been found, existing structures (clusters, surfaces, etc) can be deduced and analyzed separately [5], [10].

4.3. Decomposition Algorithm

The decomposition algorithm adopts the principle of divide and conquered paradigm. Multidimensional database is decomposed into several sub-instances and independently solve each of the sub-instance as illustrated in figure 1a-d. The sub-instance solutions are combined to form the original instance. The algorithm in figure 3 and 4 are used to achieve each of these sub-instances which are eventually merged together taking into consideration the principle of lossless join decomposition. A decomposition R1 and R2 is a lossless-join decomposition if $R1 \cap R2 \rightarrow R1$ *or* $R1 \cap R2 \rightarrow R2$. It uses decomposition algorithm of Boyce Codd Normal Form for free data anomalies during database operations as described in figure 1d above. Multidimensional dataset are captured via inner loops of the algorithm in figure 3. The figure illustrates how each dimension is transformed to an additional field of the base table while keeping track of the location of each dimension in the original multidimensional data structure. The algorithm makes sure that each newly created field is transformed to its position in the multidimensional data structure. It also takes care of interconnectivities among dimensions which preserve each cell of the dataset.

```
Procedure Get_Input_DataStructure is
Begin
Get the no of gene counts (dimensions) C (assuming all dimensions have the same cardinality c, e.g.,
3 with a limit of 26 dimensions)
Declare array arr of C dimensions
for a in 1 .. c ;  for b in 1 .. c ;  for c in 1 .. c  …  for z in 1 .. c
get a value v to store into the multidimensional array location
arr(a,b,c,…,z) = v
loop z  …  loop c ;  loop b ;  loop a
    End Get_Input_DataStructure
```

Figure 3: Data Set Input into Multidimensional Database

```
Procedure Reverse_Penetrance_Matrix(P: length of new list, c: cardinality) is
Begin
Set C (dimensions) = P^1/C or ∛P
Declare array arr of C dimensions, with c cardinality
Open List cSNP[P]    (cSNP is a list that stored transformed values of length P)
Declare a Counter set to 0
for a in 1 .. c ;  for b in 1 .. c ;  for c in 1 .. c    …   for z in 1 .. c
set Counter := Counter + 1
Retrieve the value of cSNP at index Counter
arr(a,b,c,…,z) = cSNP[Counter]
loop z  …   loop c ;  loop b ; loop a
    End Reverse_Penetrance_Matrix
```

Figure 4: Reverse Algorithm

5. Implementation Approach

The implementation of decomposition algorithms in figure 1a-d, 3 and 4 were carried out on Visual Studio 6.0 (Visual Basic), Structured Query Language Server Database 2000, Microsoft Excel 2000 and Online Analytical Processing 2.0 running on Pentium M 1.6GHz and 512 MB RAM of Personal Computer System. SQL Server was used to store the transformed data into two dimensional dataset while Microsoft Excel was used to store the reversed transformed two dimensional dataset to its original structure. There are three major menus, namely, submission, setup and functionalities. The SNP data capture sub-menu under submission is used to capture multidimensional dataset to be transformed. Bearing in mind the scope on which the algorithms are designed, the system gets the number of genes count to simulate.

The system automatically generates the list of data inputs with the SNP combinations required for the multidimensional dataset as a result of the interaction among the genes. The user then fills in the penetrances for each of the combinations. After entering the data, the transformation sub-menu under functionalities menu is used to perform reduction operation on the dataset. The SNP visualization sub-menu is used to show the pictures of the transformed dataset and graphical output for performance analysis.

To achieve this, a user account with database owner right is created in the Micro-Soft Structural Query Language Server 2000 database that is used to store the data. This database is thus implemented over a Local Area Network system via this account. A user can connect to the database within the application.

6. Performance Analysis and Evaluation

The decomposition algorithm was demonstrated for GeneCount 2, 3, 4, 6, 7, 8, 9 and 10. Each SNP count was selected to generate its penetrance sequence. For example, a GeneCount of 2 produced $3^2 = 9$ penetrance items. The sequence pattern is a loop through the rows and columns of the structures, starting from the lowest row and column to the highest row and column, to give 1,1; 1,2; 1,3; 2,1; 2,2; 2,3; 3,1; 3,2 and 3,3. This is depicted in figure 6 in column 4. The same program code was run for other genecounts in the same manner. Figure 7 shows genecount 4 produced $3^4 = 81$ penetrance items which starts at index 1111,

1112, 1113,…, 4441, 4442, 4443, 4444 in column 4. Thus, we have the following convertibility: 3→2; 4→2; 5→2; 6→2; 7→2; 8 →2; …; n→2.

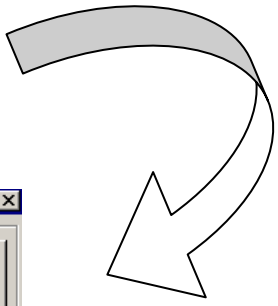
For fixed length dimensions, the relationship between dimension counts and sequence elements yielded gives a Geometric Progression (G.P.) sequence of the form ar^{n-1}.

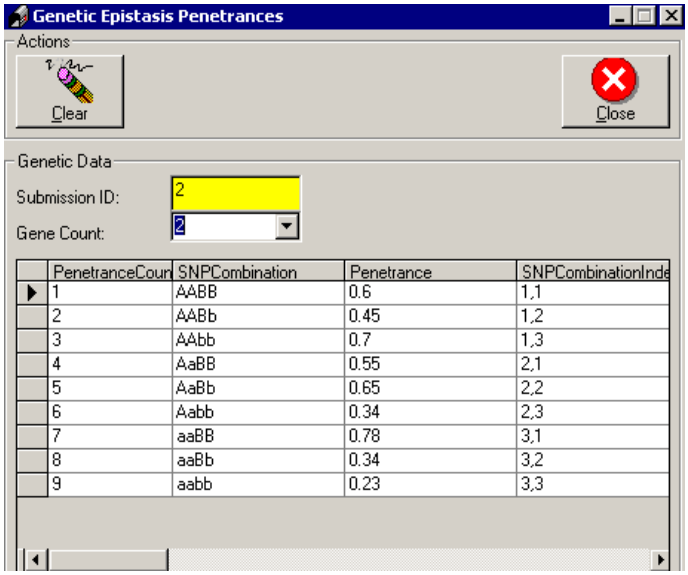
Based on the operations performed on a database management system, namely, searching, inserting, updating and deleting, it has been revealed that searching through a multidimensional database (a database of more than 2 dimensions) is slower than searching through a 2- dimensional database, especially when the cardinality of the dimension is high. This is due to the fact that searching through such a database will technically involve moving through several inner loops of data. On the other, searching through an indexed list of data makes data retrieval faster. The same analogy goes for INSERT, UPDATE and DELETE, since each of these operations especially, UPDATE and DELETE, needs to know the point (location) at which the operation is to be performed. This can be seen in figure 8 by illustrating the relationship between genotype counts and the corresponding time in seconds taken to form the linear array generate its sequences. As the number of genecount (dimension) increases, the length of the transformed list and the amount of time (seconds) taken to generate the list increases, meaning that, the higher the cardinality of dimension of a database, the higher the time required to perform operations on such a database.

On the other hand, it is of interest to have the transformed multidimensional dataset reversed to its original state. The reverse algorithm was run on a transformed dataset of figure 7 which was depicted in figure 9. This gave the following multidimensional structure presented in MS Excel Worksheet in figure 10. The reduced 2-dimensional dataset was reversible to the original multidimensional dataset using the principle of lossless join operation which indicated that there was no loss of data.

7. Comparison of Decomposition Algorithm with existing Reduction Techniques

	Table penetrance			Margin penetrance
	AA (.25)	*Aa* (.50)	*aa* (.25)	
BB (.25)	0	0	1	0.25
Bb (.50)	0	0.50	0	0.25
bb (.25)	1	0	0	0.25
Margin penetrance	0.25	0.25	0.25	





	PenetranceCoun	SNPCombination	Penetrance	SNPCombinationInde
▶	1	AABB	0.6	1,1
	2	AABb	0.45	1,2
	3	AAbb	0.7	1,3
	4	AaBB	0.55	2,1
	5	AaBb	0.65	2,2
	6	Aabb	0.34	2,3
	7	aaBB	0.78	3,1
	8	aaBb	0.34	3,2
	9	aabb	0.23	3,3

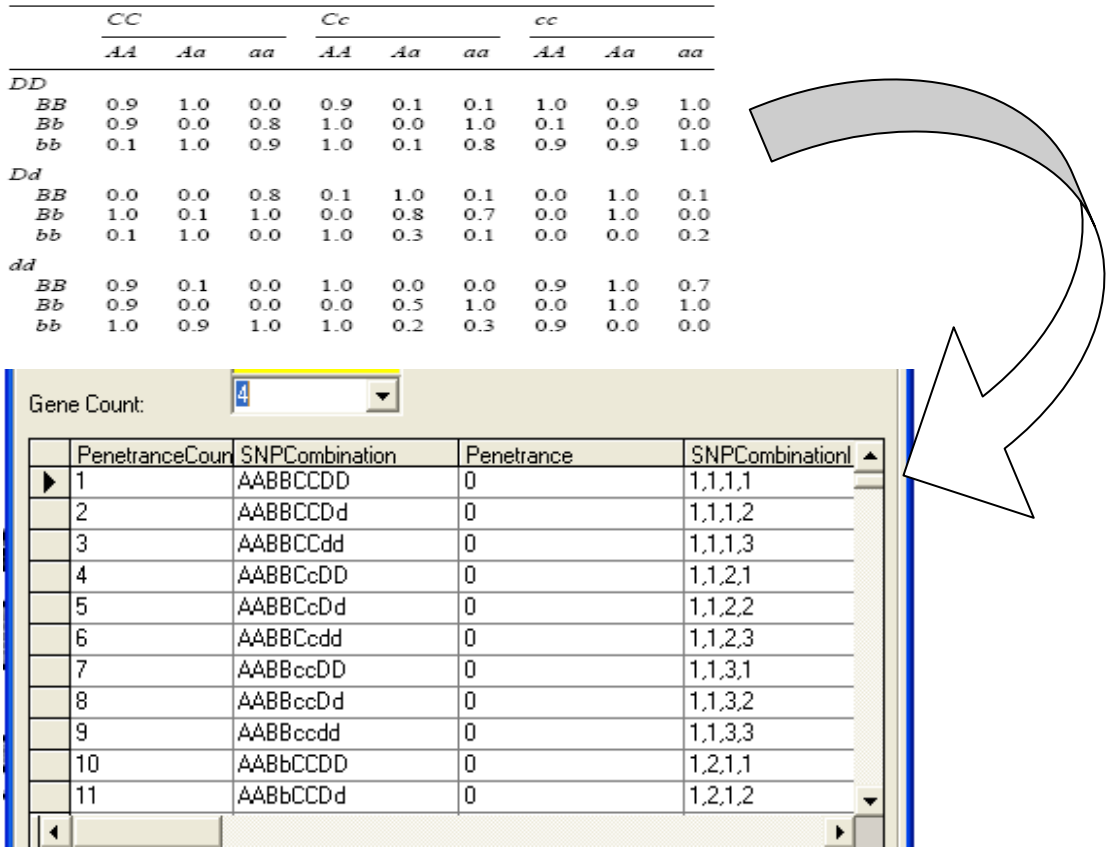
Figure 6: A list showing a typical linear array generated (▶) and the corresponding sequences of Gene count 2

The conventional database reduction techniques such as principal component analysis, projection pursuit among which relational model is the best in the area of comparing the same set of data. This decomposition

algorithm was compared with relational model by using the same data set. A multi-dimensional dataset of five different dimensions was chosen and served as input values to both relational model and decomposition algorithm. The composite graph shown in figure 11 depicts the comparison between the two algorithms using genes count and time taken to process. Pure data preservation was confirmed in the decomposition algorithm. No atomic data item got loss during the reduction exercise. Data processing and access were very fast, but with large storage unit. In the final analysis, the decomposition method performed better than relational model to access data in multidimensional database.

8. Conclusion and Future Work

A decomposition algorithm was used for reducing 10-multidimensional database to a two-dimensional database. The algorithm was simulated with series of multi-dimensional dataset to reduced two-dimensional dataset. The reduced two-dimensional dataset were reversed into multi-dimensional dataset and it was found that there was no loss of data in the database. The algorithm was experimented with the relational model by using the same dataset as input. It was found that data access was very fast with decomposition algorithm than relational model. The work is effectively limited to representation of 26 dimensions due to the coding system on alphabets A, B, C,…, Z (26 letters). It will be of a great interest if a coding technique can be achieved to represent dimensions greater than 26, while still using the same algorithm to manipulate multi-dimensional dataset.



	CC			Cc			cc		
	AA	Aa	aa	AA	Aa	aa	AA	Aa	aa
DD									
BB	0.9	1.0	0.0	0.9	0.1	0.1	1.0	0.9	1.0
Bb	0.9	0.0	0.8	1.0	0.0	1.0	0.1	0.0	0.0
bb	0.1	1.0	0.9	1.0	0.1	0.8	0.9	0.9	1.0
Dd									
BB	0.0	0.0	0.8	0.1	1.0	0.1	0.0	1.0	0.1
Bb	1.0	0.1	1.0	0.0	0.8	0.7	0.0	1.0	0.0
bb	0.1	1.0	0.0	1.0	0.3	0.1	0.0	0.0	0.2
dd									
BB	0.9	0.1	0.0	1.0	0.0	0.0	0.9	1.0	0.7
Bb	0.9	0.0	0.0	0.0	0.5	1.0	0.0	1.0	1.0
bb	1.0	0.9	1.0	1.0	0.2	0.3	0.9	0.0	0.0

Gene Count: 4

	PenetranceCoun	SNPCombination	Penetrance	SNPCombinationI
▶	1	AABBCCDD	0	1,1,1,1
	2	AABBCCDd	0	1,1,1,2
	3	AABBCCdd	0	1,1,1,3
	4	AABBCcDD	0	1,1,2,1
	5	AABBCcDd	0	1,1,2,2
	6	AABBCcdd	0	1,1,2,3
	7	AABBccDD	0	1,1,3,1
	8	AABBccDd	0	1,1,3,2
	9	AABBccdd	0	1,1,3,3
	10	AABbCCDD	0	1,2,1,1
	11	AABbCCDd	0	1,2,1,2

Figure 7: A list showing a typical linear array generated (▶) and the corresponding sequences of Gene count 4**.**

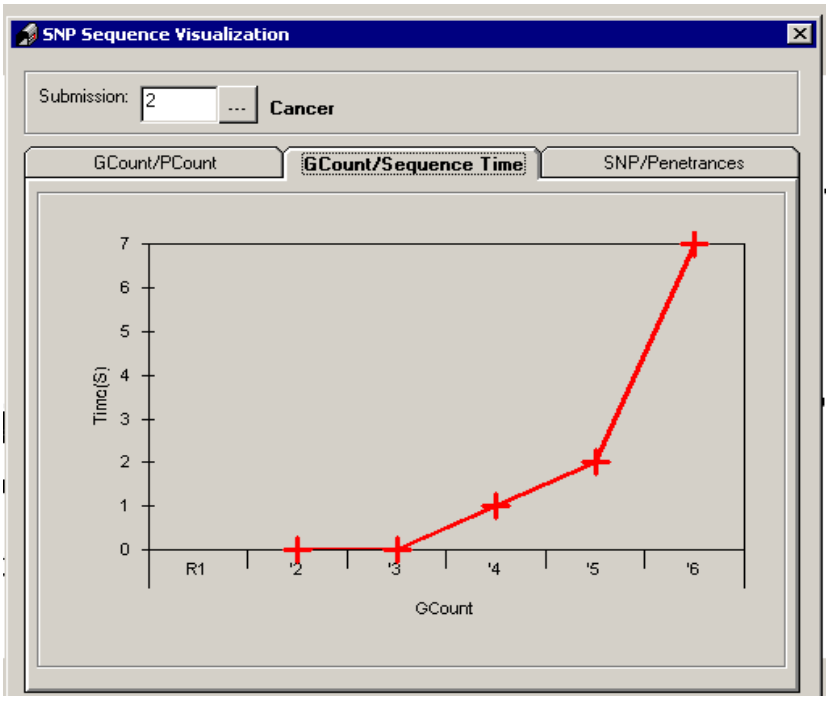
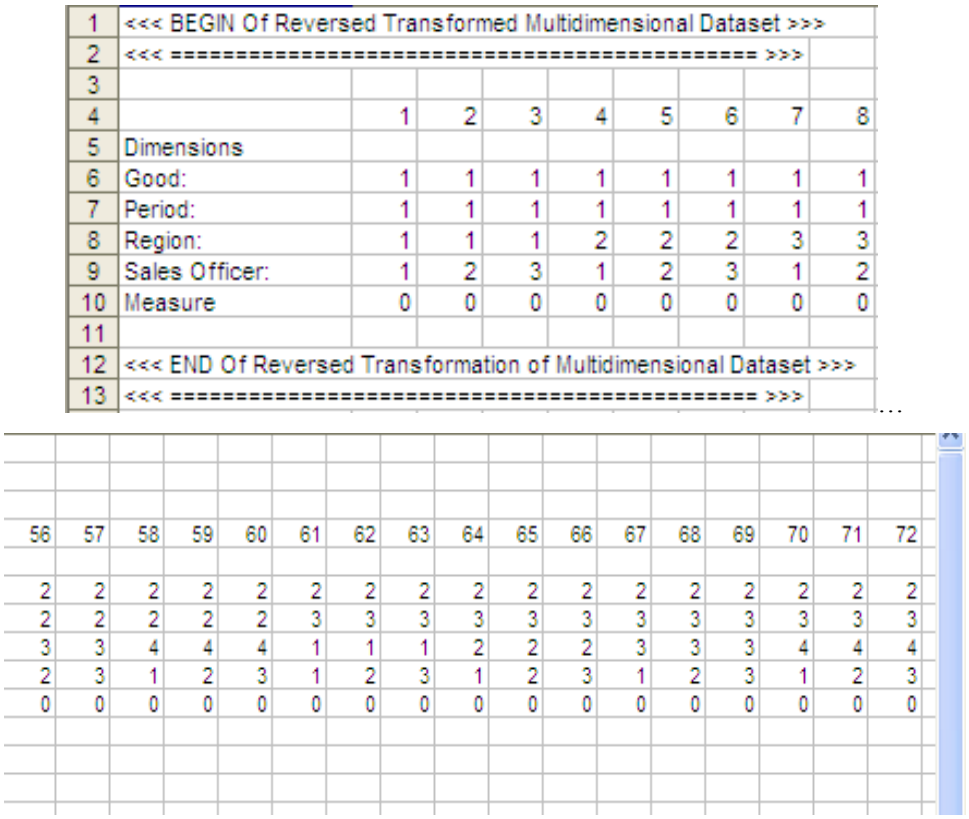


Figure 8: A graph showing the relationship between genotype counts and the corresponding time (secs) taken to form the linear array generate its sequences

SNo	IndexCombination	ElementCombinatio	Dimensi
1	1.1.1.1.	G1.P1.R1.S1	0
2	1.1.1.2.	G1.P1.R1.S2	0
3	1.1.1.3.	G1.P1.R1.S3	0
4	1.1.2.1.	G1.P1.R2.S1	0
5	1.1.2.2.	G1.P1.R2.S2	0
6	1.1.2.3.	G1.P1.R2.S3	0
7	1.1.3.1.	G1.P1.R3.S1	0
8	1.1.3.2.	G1.P1.R3.S2	0
9	1.1.3.3.	G1.P1.R3.S3	0
10	1.1.4.1.	G1.P1.R4.S1	0
11	1.1.4.2.	G1.P1.R4.S2	0
12	1.1.4.3.	G1.P1.R4.S3	0
13	1.2.1.1.	G1.P2.R1.S1	0
14	1.2.1.2.	G1.P2.R1.S2	0
15	1.2.1.3.	G1.P2.R1.S3	0
16	1.2.2.1.	G1.P2.R2.S1	0
17	1.2.2.2.	G1.P2.R2.S2	0
18	1.2.2.3.	G1.P2.R2.S3	0
19	1.2.3.1.	G1.P2.R3.S1	0
...			
40	2.1.2.1.	G2.P1.R2.S1	0
41	2.1.2.2.	G2.P1.R2.S2	0
42	2.1.2.3.	G2.P1.R2.S3	0
43	2.1.3.1.	G2.P1.R3.S1	0
44	2.1.3.2.	G2.P1.R3.S2	0
45	2.1.3.3.	G2.P1.R3.S3	0
46	2.1.4.1.	G2.P1.R4.S1	0
47	2.1.4.2.	G2.P1.R4.S2	0
48	2.1.4.3.	G2.P1.R4.S3	0
49	2.2.1.1.	G2.P2.R1.S1	0
50	2.2.1.2.	G2.P2.R1.S2	0
51	2.2.1.3.	G2.P2.R1.S3	0
52	2.2.2.1.	G2.P2.R2.S1	0
53	2.2.2.2.	G2.P2.R2.S2	0
54	2.2.2.3.	G2.P2.R2.S3	0
55	2.2.3.1.	G2.P2.R3.S1	0
56	2.2.3.2.	G2.P2.R3.S2	0
57	2.2.3.3.	G2.P2.R3.S3	0
58	2.2.4.1.	G2.P2.R4.S1	0
59	2.2.4.2.	G2.P2.R4.S2	0

Figure 9: Transformation of 4 Dimensional Non-Uniform Length Dataset



1	<<< BEGIN Of Reversed Transformed Multidimensional Dataset >>>								
2	<<< ========== >>>								
3									
4		1	2	3	4	5	6	7	8
5	Dimensions								
6	Good:	1	1	1	1	1	1	1	1
7	Period:	1	1	1	1	1	1	1	1
8	Region:	1	1	1	2	2	2	3	3
9	Sales Officer:	1	2	3	1	2	3	1	2
10	Measure	0	0	0	0	0	0	0	0
11									
12	<<< END Of Reversed Transformation of Multidimensional Dataset >>>								
13	<<< ========== >>>								

56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3
3	3	4	4	4	1	1	1	2	2	2	3	3	3	4	4	4
2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 10: Reverse Transformation of 4 Dimensional Non-Uniform Length Dataset

Time

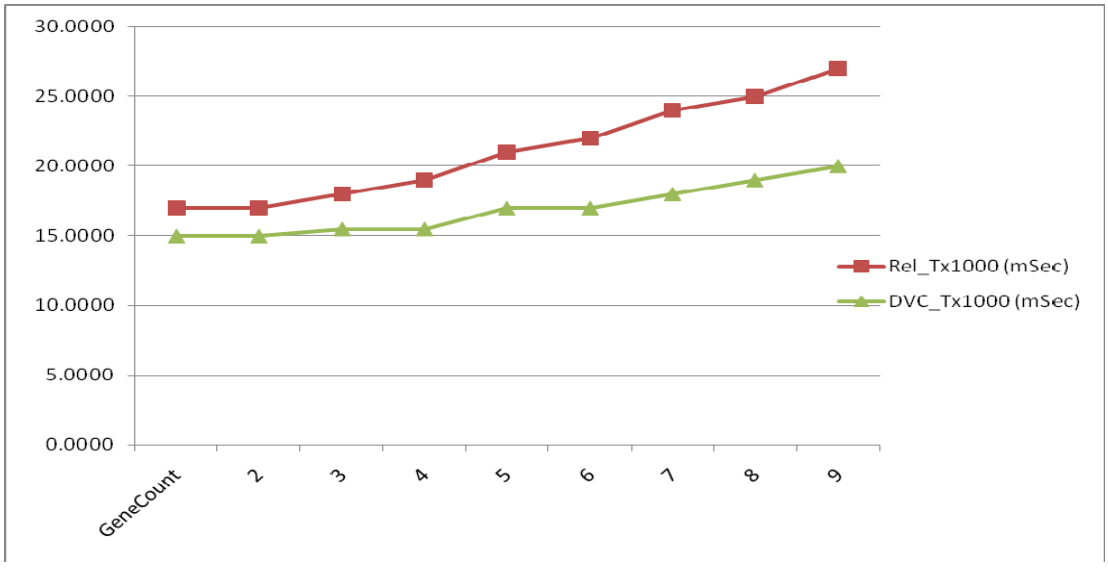


Figure 11: A Line Graph showing GeneCount and Time Taken to Process and Generate Data Sequence for Relational Model and Decomposition Algorithm

9. References:

[1] S. Agrawal. On the Computation of Multidimensional Aggregates. *Proceeding Very Large Data Base Conference*. Morgan Kaufmamn, San Francisco. 1996, pp. 506-512.

[2] D.E. Avion and Fitzgerald. *Information Systems Development Methodologies Techniques and Tools*, 3rd Edition. New York, MCGraw-Hill Education, 2003.

[3] K. Basso, A. A.Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera and A. Califano. Reverse Engineering of Regulatory Networks in Human Cells. Nat Genet. *Biomedical Central Journal*. Hong Kong. 2005, **37**(4): 382-390.

[4] B. Inmon. Different Kinds of Data Warehouse Performed Based. *XING AG*, 2009.

[5] M. A. Carreira-Perpinan. Continuous latent variable method for dimensionality reduction and sequential data construction, PhD thesis, Department of Computer Science, University of Sheffield, UK, http://faculty.ucmerced.edu/mcarreira-perpinan/papers/phd-thesis.html, 2001.

[6] C. Gibas and P. Jambeck. *On Developing Bioinformatics Computer Skills.* O'Reilly Media Inc., USA, 2001.

[7] D. O'Neil. *Modern Theories of Evolution: An Introduction to the Concepts and Theories that led to our Current Understanding of Evolution*. Behaviour Sciences Department, Palomar College, San Marcos, California, 2010.

[8] A. Essenberg and J. Melton. SQL Standardisation: The next steps. *ACM SIGMOD Record*. ACM, New York, USA. 2000, **29**(1): 63-67.

[9] I. K. Fodor. and C. Kamath C.. Dimension reduction techniques and the classification of Bent Double Galaxies. *Computational statistics and data analysis Journal*. 2002, **41**: 91-121.

[10] J. H.Friedman and J. W. Turkey. Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Trians Computing*. UK. 1974, **23**: 881-890.

[11] G. Colliat, *OLAP.* Relational, and Multidimensional Database Systems. *ACM SIGMOD Record.* ACM, New York, USA. 1996, **25**(3): 64-69.

[12] H. Gupta, V. Harinaraya, A. Rajaraman, and J. D. Ullman. Index Selection for OLAP. *In Proceeding 13th ICDR*. Manchester UK. 1997, pp. 208-209.

[13] H. Helen, and H. Peter. Using OLAP and Multidimensional Data. *ACM SIGMOD Record vol.* ACM, New York, USA. 1999, **28**(4): 64-69,

[14] A. Howard. *Elementary Linear Algebra*, 5th edition. New Jesey: John Wiley & Sons Inc., USA, 2005.

[15] H. J. Moore, L. W. Hahn, M. D. Ritchie, T. A. Thornton and B. C. White. Routine Discovery of Complex Genetic Models Using Genetic Algorithm. *Applied Soft Computing*. Elsevier. 2004, **1**(1).

[16] C. S. Jensen, A. Kligys, T. B. Pedersen and I. Timko. Multidimensional Data Modeling for Location Based Services. *The VLBD Journal, The International Journal on very large data bases*. Springer Berlin, 2004, 13(1): 1-21,

[17] Kenan Systems Corporation. *An Introduction to Multidimensional Database Technology*. Cambridge: Kenam Systems Corporation, UK., 1995

[18] R. A. King., J. J. Rotter and A. G. Motsulsky. *The Genetic Basis of Common Diseases.* Oxford: Oxford University Press, 1992.

[19] L. W. Hahn, M. D. Ritchie and J. H. Moore. Multifactor Dimensionality Reduction Software for Detecting Gene-Gene and Gene-Environment interactions. Oxford: Oxford University Press, 2002.

[20] I. S. Lindsay. *A Tutorial on Principal Components Analysis*. www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002.

[21] J. H. Moore and S. M. Williams. Traversing the Conceptual Divide between Biological and Statistical Epistasis. Systems Biology and a More Modern Synthesis. BioEssays. *Biomedical Central Journal*. Hong Kong. 2005, **27**(6): 637-646,

[22] R. L. Nagel. *Epistasis and the Genetics of Human Diseases*. C R Biol. *Biomedical Central Journal.* Hong Kong. 2005, **328**(7) : 606-615,

[23] N. M. Gatto, U. B. Campbell, A. G. Rundle and H. Ahsan. Further Development of the Case-Only Design for Assessing Gene-Environment Interaction: Evaluation of and Adjustment for Bias. *International Journal of Epidemiology*. Oxford: Oxford University Press, UK, 2004.

[24] T. B. Pedersen and C. S. Jensen. Multidimensional Database Technology. *IEEC Computer Society*. Los Alamitus. 2001, **34**: 40-46,

[25] T. B. Pedersen, C. S. Jensen and C. E. Dyreson. Extending Practical Pre-aggregation in OLAP. *Proceeding 25th Int's Conference on very large Databases*. Morgan Kaufmamn, San Mateo, Calfonia, 1999, pp. 667-674.

[26] R. Agrawal., A. Gupta and S. Sarawagi. Modeling Multidimensional Databases. *Proceeding of the 13th International Conference on Data Engineering*. Birmingham: IEEC Computer Society, UK. 1997, pp. 232 – 243.

[27] S. Sarawagi. User Adaptive Exploration of OLAP Data Codes. *Proceeding VLDB*. San Francisco: Morgan Kaufmamn. 2000, pp. 307-316.

[28] P. Yue and J. Monlt. Identification and Analysis of Deleterious Human SNP**.** *Journal of Biology.* **356**: 1253-1274. Elsevier.