A Genetic-Firefly Hybrid Algorithm to Find the Best Data Location in a Data Cube

M. Faridi Masouleh
Information Technology Management Department
Science and Research Branch
Islamic Azad University
Tehran, Iran
m.faridi@srbiau.ac.ir

M. A. Afshar Kazemi
Information Technology Management Department
Science and Research Branch
Islamic Azad University
Tehran, Iran
m.afsharkazemi@yahoo.com

M. Alborzi
Information Technology Management Department
Science and Research Branch
Islamic Azad University
Tehran, Iran
Mahmood_alborzi@yahoo.com

A. Toloie Eshlaghy
Information Technology Management Department
Science and Research Branch
Islamic Azad University
Tehran, Iran
toloie@gmail.com

Abstract—**Decision-based programs include large-scale complex database queries. If the response time is short, query optimization is critical. Users usually observe data as a multi-dimensional data cube. Each data cube cell displays data as an aggregation in which the number of cells depends on the number of other cells in the cube. At any given time, a powerful query optimization method can visualize part of the cells instead of calculating results from raw data. Business systems use different approaches and positioning of data in the data cube. In the present study, the data is trained by a neural network and a genetic-firefly hybrid algorithm is proposed for finding the best position for the data in the cube.**

Keywords-database; data cube; genetic algorithm; firefly algorithm; neural network

I. INTRODUCTION

Business Intelligence (BI) is a vast category of methods, applications and technologies of gathering data, accessing information, and analyzing a large amount of data for getting knowledge in the organization to make effective business decisions and made Information Technology (IT) measurements powerful. Typical BI technologies include business rule modeling, ETL (Extraction, Transformation and Load) tools, data profiling, data warehousing and online analytical processing, and data mining. The focus of BI is to fully utilize massive data to help organizations gain competitive advantages. Decision support systems are increasingly used in business. They provide access to data that has been locked in a database and convert it into useful information. Many organizations and companies are developing integrated decision support systems known as data warehouses, where users can carry out analysis. As operational databases update the state of their information, data warehouses typically store the history of that information. As a result, data warehouses tend to grow over time.

Users of decision support systems are generally reluctant to identify behaviors instead of searching for an independent record. Queries in a decision support system build aggregation. The size of the data warehouse and complexity of the queries can produce long queries that require much time for completion. This causes delays that are not acceptable in most decision support system environments because it reduces the efficiency of the system.

Data cube analysis is a powerful tool for analyzing multi-dimensional data. Cube analysis allows users to simply explore data by calculating the aggregate size of all possible groups as defined by their new dimensions. A number of techniques have been used to design effective methods to calculate the cube. Technically, a cube is a multi-dimensional redundancy plan relationship that calculates all SQL-grouped operators and aggregates their results in n-dimensional space to answer OLAP queries. This aggregation is calculated in derivative summary tables or multi-dimensional arrays and mathematical tools are required to estimate the aggregation sizes. Because an aggregation is generally quite large, indexing, which adds redundancy, is deemed necessary to speed up the inquiry [42, 43, 44].

II. OLAP

Online analytical processing (OLAP) is rooted in systems such as IRI Express, Comshare, and Essbage. Unlike statistical databases that hold census data and economic information, OLAP collects everyday commercial transactions data, such as that for sales or health. The main objective of OLAP is to enable analysts to build a mental image of the underlying data by exploring different perspectives at various stages of generalization and interaction [39]. As a component in decision support systems, OLAP interacts with other components such as data warehouses and data mining to assist business analysts in the decision-making. Data warehouses collect data from multiple data sources, such as transactional databases, across an organization. The data is cleaned and converted to a sustainable format before being collected in the data warehouse. Subsets of the data in the warehouse are extracted to meet the specific needs of the organization. Unlike transactional databases, where data is continuously updated, the data collected in a warehouse updates itself; this is only possible by data sources in cyclic form.

OLAP and data mining both allow analysts to discover knowledge about the data in a warehouse. Data mining algorithms automatically generate such knowledge in a pre-defined state, such as by classification or collaborative laws [2, 3]. OLAP does not produce knowledge directly, but relies on human analysis to interpret the results of the query. OLAP is more flexible than data mining because analysts can access different patterns and trends rather than a specific set of knowledge. OLAP and data mining can be combined to enable analysts to obtain data mining results from various sectors of the data and in the generalization stages [9, 39]. In a typical session of OLAP, analysts count aggregated queries about the underlying data. OLAP can obtain data in a few seconds, even if the query has a very long record. The results allow analysts to collect large amounts of data to determine general patterns and trends. Based on their observations about an exception to a pattern, analysts can collect smaller amounts of data in greater detail to fill in parts. This process can be repeated in different parts of the data using data segmentation until the mental imagery is satisfied. OLAP system needs are determined in ways such as the FASMI test and Codd law. Some requirements are unique to OLAP. To enable OLAP for an interactive process, the system must first effectively respond to queries. OLAP often relies on large-scale pre-computing, specialized indexing, and storage to improve performance.

In the next step, analysts are allowed to explore the data from different perspectives and stages of generalization to organize and generalize the data into multiple dimensions and hierarchies. The data cube model described below is a popular abstract model for this purpose [9]. The data to be analyzed by OLAP is collected in the data warehouse using the communication model and are organized using stellar models measuring dimension and other features. Each dimension has a table that displays contributions to dimension hierarchically. The table of dimensions could have redundancies which can be eliminated by dividing each dimension into other tables. The result is called a snowflake model.

ROLAP and MOLAP are popular architectures of OLAP. ROLAP provides a front tool that translates the multi-dimensional queries in the corresponding SQL queries for processing within a relationship. OLAP is lightweight and is scalable to large datasets, but its effectiveness is limited because intra-relational optimization methods are not designed for multi-dimensional queries. OLAP does not rely on the relational model, but focuses on a multi-dimensional view. MOLAP can achieve better performance by visualization and improvement of a multi-dimensional view, although it requires significant storage for visualization and is not always scalable [2, 4, 23].

III. DATA CUBE

A data cube is an SQL operator that supports OLAP functions such as histograms and subsidiary operations. Histograms are aggregations at the calculation level. Even if such tasks are possible using standard SQL queries, they become too complicated. The number of sets needed for the number of dimensions is exponential. A complex query could produce results during the explorations of the base table which result in weak performance. Because a subtotal of all queries is common for OLPA queries, a new operator can be arbitrarily determined to collect subtotals as a data cube. Users of the system want the data cube to offer a summary of the data in detail from different aspects.

A data cube comprises reciprocal table generalization that occurs in multiple approaches. At first, a data cube is n-dimensional. It is divided into eight parts, each of which is called a cube. The first cube is a three-dimensional cube known as the nucleus. The Next Tuesday cube contains data in which all values are 2D surface values. The next three cubes contain all values and 2D levels. The next three cubes are one-dimensional. The last cube has a single value and a dimensionless point [1, 11, 16]. The second generalization approach is the aggregation function. This aggregation will be discussed with SUM. In general, any aggregation function can be used with a custom value to build data cubes. This function can be divided into distributive, algebraic, and comprehensive categories. I is a set of values and $P(I)=\{p1, p2,..., pn\}$ is part of I. The aggregation function of F( ) is distributive if the function of G is $F(I)=G(\{F(p_i):1\le i\le n\}$. Under these circumstances, distributive SUM, SOUNT, MIN, and MAX are easy to examine. After generalization of function G( ) to one, the vector of m is restored and algebraic aggregation shows characteristics similar to the distributive sample of $F(I)=G(\{F(p_i):1\le i\le n\}$. The third approach is dimensional hierarchy. For a data cube, each dimension has two stages of weak hierarchy. Each dimension can have many features. Dimension features can create a weak hierarchical network. Features on the base table have lower bounds on the network and all queries have higher bounds. The network structure plays an important role in most data cube approaches [25, 35].

Core cube visualization is necessary because this cannot be achieved by calculating other items. If any cube with all values has a value comparable to the core cube, it is not necessary to visualize the core cube because responding to a query using the cube incurs the same cost as the core cube. Many greedy algorithms have been proposed to position data in data cube [16, 25]. Even if all data cubes must be calculated, the network structure can be used to improve the efficiency of calculations. For example, if calculation is based on ordering records, then core cubes can be ordered in three ways. Every choice leads to simplification of the calculation to one cube with all values, because that cube can be calculated without additional ordering, although calculation of the two other cubes will require a core cube to be re-ordered. Based on the estimated cost, the algorithms available as the optimal choice for ordering each cube and the choices leading to construction of calculation pipelines again reduce the size of the cube ordered [11, 12, 25]. Data warehouse users work in a graphical environment and data is displayed as a data cube in multi-dimensional form. Dimensions are usually 2D or 3D, but higher dimensions can also be generated to increase information exploration. The values in each cell of a data cube contain a measuring value [4, 23, 30, 35].

IV. Related work

Effective calculations and multi-dimensional aggregation in a data cube have been studied by many researchers. A data cube has been classified as an aggregation of generalizations by a relational operator based on subsidiary data cubes with distributive, algebraic and comprehensive categories [10]. A greedy algorithm has been developed to provide a partial cube for data cube calculations [35, 36]. A calculation method has been developed based on piece processing to effectively organize large multi-dimensional arrays [32]. Several guidelines have been suggested for effective calculation of multidimensional aggregations for ROLAP servers [31]. A model has been developed based on multi-path array aggregation piece processing for data cube calculations in MOLAP [40]. A method has been also been developed to calculate scattered data cubes [18]. After iceberg queries were developed [25, 26], the BUC method, a scalable method with the ability to calculate iceberg cubes head downward, was introduced [16, 17]. The H cube method was also developed for calculating iceberg cubes with complex measurements using H-tree structure [12]. The star cube method was introduced for calculation of an iceberg cube with a dynamic star tree structure [7]. The MM cube is an effective method of calculating the iceberg cube using factorized network space [41]. A shell-piece method based on data cubes is very effective for OLPA multi-dimensional data processing [23].

Apart from these studies on data cube models, the smart dwarf method has been suggested for the calculation of a declined data cube and is known as the condensed model [38]. Calculation of compression data cubes (dwarf cubes) uses a smart cube structure that summarizes the semantics of a data cube [21]. The same method was also developed in a QC tree structure [22]. An aggregation-based approach known as a closed cube or cube C has been developed [8] to close calculation in a closed cube using a new algebraic method of measurement. Studies have been conducted on calculations using compressed data cubes by estimation, including the quasi-cubic [6] and the wavelet cube [13] methods. Compressed cubes for estimation in continuous dimensions [14] have been applied using the linear logarithm models [5] to a compressed data cube.

V. Neural network and evolutionary algorithms

Everything is done in an environment directly or indirectly and is a collection of decisions. It is important at the beginning of a decision to have sufficient knowledge of the issue investigated to make appropriate decisions. Evolutionary algorithms have been used to identify environments. In the present paper, two algorithms are used to comprehend the issues to facilitate knowledge [37].

A. Neural Networks

The perceptron network is a well-known neural network and its multilayer state is a widely-used neural network. Perceptron networks probably had the greatest effect on early neural networks. The perceptron learning rule is stronger than those such as the Hebbian rule. With reasonable assumptions, this method of learning by repetition converges to correct weights. Convergence means that network learning leads to estimation of weights that allow it to produce correct output values for each education input pattern and similar patterns. Moreover, the performances of neural networks are related to the choice of the neurons count, architecture of networks and learning algorithms [28, 44].

The perceptron learning rule is supervised learning in which a stimulus, response, input, optimal output, pattern, and pattern class are available. Learning error occurs; hence, in practice, at each step the learning error must be used to set network parameters such that the learning error becomes less when the same input is applied again. The perceptron learning rule is generated for one-layer neural networks consisting of neurons with a conversion function with a two-value limit threshold [20, 32, 37]. Normally, primary perceptrons have three layers: sensory neurons, connecting units, and the response unit forming an approximate retina model. Although these networks are only taught weights between the second and third layers, the activation function for each connecting unit is a binary step function with an optional, but fixed, threshold value. The signal sent from the connecting units to the output unit is a binary signal and $y = f(y_{in})$ is the perceptron output with the activation function:

$$f(y_{in}) = \begin{cases} 1 & if \quad y_{in} > \theta \\ 0 & if \quad -\theta \le y_{in} \le \theta \\ -1 & if \quad y_{in} < -\theta \end{cases}$$

This function determines the network output in addition to the usual outputs of +1 (belonging to the category or group) and -1 (not belonging to the category or group) and includes the area between θ and –θ in which a decision will not be made [34, 37]. The unit weights connected to the response or output units are set using the perceptron learning rule. In the perceptron training rule, the network computes the output unit response for each input learning vector and then determines whether or not an error has occurred for the model. Here, an error occurs when the output calculated by the network does not equal the target value. Learning this rule is similar to learning the Hebbian rule, except that the weights only alter when the network response for the input contains an error.

The perceptron network does not identify the difference between errors when the calculated network output is 0 (area without decision) and the target value is -1 or when the calculated output value is +1 and the target value is -1. The learning rule changes the direction of network weights such that the response becomes equal to the target value. Only the weights on the joints of connecting units that send non-zero signals to the output unit will be changed, because these signals have caused the error. In the perceptron training rule, if an error occurs in the training input pattern, the weights vary as:

$$w_i(new) = w_i(old) + ax_i t$$

where t represents the target value as +1 or -1 and a is the learning rate and determines the speed of the weights. In this network, if no error occurs, the weights will not change and training will continue until no further error occurs.

An optimization algorithm which supplies the best training dataset for appropriate Artificial Neural Networks is proposed in this study like [24,44] that updates weights and bias and the trained net is added as input dataset for genetic algorithm. The perceptron learning rule convergence theorem states that if there are weights that allow the network to produce the correct answer for all training patterns, then the perceptron learning method uses these values when setting weights. This means that the perceptron network will be able to solve the problem or learn the desired categorization. In addition, the network will use these weights several times with a limited number of training repetitions [23, 44].

B. Genetic Algorithm

In genetic algorithms, every solution is displayed as a binary string and the measurement of the related fitness function. A genetic algorithm solves a problem by first randomly generating a population of chromosomes. Second, the fitness of each chromosome in the population is assessed and, third, new chromosomes are created by random mating and mutation of some bits. Fourth, less fit members are excluded from the current population and, fifth, new chromosomes are evaluated and included in the population. Finally, the third to fifth steps are repeated until population convergence occurs, which is also the termination condition. The fitness of the total population increases as the number of people in the population increases; however, the number of fit people, or more fit people, is equal at t and t-1. The population is considered for evolution rather than an individual or components. A set of chromosomes makes up the population. The effect of genetic operators on the population is to form a population with the same number of chromosomes. The fitness function provides an indicator for independent functioning in the work area. Fenestration and linear normalization are terms within the fitness function [20, 27, 29, 33, 37].

Reproduction takes place in two stages when parental chromosomes are duplicated in bits of a child's chromosomes and functions alter the chromosomes of each child. The major operators are intersection, mutation, and inverse mutation. Intersection is the strongest supporter of genetic algorithms and is usually associated with high probability. Mutation is less important than intersection and less likely to occur. As the parameters change and simulation advances, mutation become more important and population change decreases [14]. Intersection is the driving force behind a genetic algorithm which uses reproduction. Initially, a genetic algorithm uses a random number generator to select a random point in each parent at an intersection where a genetic algorithm occurs [14, 15, 27]. Mutation causes searches in untouched spaces and it can be deduced that the most important task of mutation is to avoid convergence to the local optimum. When a member of the new population arises, each gene will mutate if possible. Inversion is used in the genetic algorithm because it is biologically plausible. It can be said that in inversion, one chromosome can be selected at any time and then two points are randomly selected and their locations reversed. There are different strategies for selecting people from a population for reproduction. The selection operator selects a number of chromosomes from a population for reproduction and the more fit chromosomes are more likely to be selected [15, 33, 37].

C. Firefly Algorithm

The firefly algorithm is inspired by observation of actual fireflies. Fireflies are social insects that live in colonies and their behavior tends toward survival of the colony rather than survival of a small fraction of them. One of the most important and interesting behaviors of fireflies is finding food, especially how to find the shortest path between a food source and their nest. The firefly algorithm is a swarm intelligence method; the difference between this algorithm and fireflies in nature is memory. The most important features of this algorithm are its high convergence speed, flexibility, high error tolerance, and its insensitivity to initial values [15]. The similarity of the data placement in a data cube with a firefly algorithm and the parameters used to select the appropriate position for data and suggest the best location prompted the use of this algorithm in the present study [19].

Two very important factors in the firefly algorithm are variation in light intensity and formulation of attractiveness. For simplicity, it is always assumed that the attractiveness of a firefly is determined by its brightness, which is associated with target function coding. For maximum optimization problems, brightness I is selected for fireflies in position $I(x) \infty f(x)$. β attractiveness is a common point that can be seen by fireflies and there is no difference in distance r_{ij} for fireflies i and j. Note that the light intensity decreases as the distance from its own resources increases. In the simplest case, the light intensity of I(r) differs according to inverse square law and is defined as [4, 19]:

$$I(r) = \frac{I_s}{r^2}$$

where I_s is the intensity at the source. For a given media, a constant light absorption coefficient is considered with variable γ where the brightness intensity I differs at distance r. This relation is defined as:

$$I = I_0 e^{-\gamma r}$$

where I_0 is the main brightness intensity. To avoid the singularity at r = 0 in relation I_s / r^2 , the effect of inverse square law and absorption is Gaussian and defined as:

$$I(r) = I_0 e^{-\gamma r^2}$$

Firefly attractiveness develops as intensity that can be observed by fireflies and β attractiveness of fireflies can be defined as:

$$\beta = \beta_0 e^{-\gamma r^2}$$

where β_0 attractiveness in r = 0. $1/(1+r^2)$ is faster than the power function for calculation and is generally defined as:

$$\beta = \frac{\beta_0}{1+\gamma r^2}$$

To implement the project, attractiveness function β(r), is assumed to be a uniform reduction function such as:

$$\beta(r) = \beta_0 e^{-\gamma r^m} \quad , \quad (m \geq 1)$$

The distance between two fireflies (or equipment) i and j in x_i and x_j is similar to a Cartesian distance:

$$r_{ij} = \left| \left| x_i - x_j \right| \right| = \sqrt{\sum_{k=1}^{d} (x_{i,k} - x_{j,k})^2}$$

where $x_{i,k}$ is the kth location coordinate component of x_i and the ith firefly. In the 2D state, it can be expressed as:

$$r_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

The motion of firefly i is attractive for firefly j as expressed by:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2} (x_i - x_j) + \alpha \in_i$$

The second part of relation (11) represents the attractiveness of the firefly where α is a random parameter and $\in_i$ is the vector of random numbers drawn from the Gaussian or uniform distribution [15].

To use this algorithm, first an n-member population of fireflies in different locations is randomly generated. All fireflies initially have the same amount of light and with each iteration a light update phase and a location update phase are carried out. The location of the fireflies is the randomly-placed location and the purpose of the light is for best placement of the data. The amount of light for each firefly in each iteration is determined by its location fitness value. The fitness value is the value added to the light in each iteration. Firefly light for each update can be expressed as:

$$\varphi_i(t) = (1-p)\varphi i(t-1) + \gamma\, j(x_i(t))$$

where $j(x_i(t)$ is the new value of firefly light when used again, $(1-p)\varphi_i(t-1)$ is the fitness of firefly i in iteration t of the algorithm and p and φ are constants with which to model the gradual decline and its effect on light.

Relation 13 is used to detect the position of other pieces of data in the neighborhood as:

$$p_{ij}(t) = \frac{\varphi_j(t) - \varphi_i(t)}{\sum_{k \in N_i(t)} \varphi_k(t) - \varphi_i(t)}$$

where $N_i(t)$ is the set of fireflies in the neighborhood of firefly i at time t. The distance between fireflies i and j at time t uses the Euclidean distance $d_{ij}(t)$ Discrete-time movement of fireflies with probability p is expressed as:

$$x_i(t-1) = x_i(t) + s\left(\frac{x_j(t) - x_i(t)}{\left|\left| x_j(t) - x_i(t) \right|\right|} \right)$$

where operator ||…|| shows the Euclidean norm r, s is its step-size movement. The term $x_t(t)$ is an m-dimensional vector of firefly location over time t. A neighbor is assigned to each firefly i in which radial range r_d^i is naturally dynamic. Updating of the neighbors is achieved as:

$$r_d^i(t-1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|\}\}$$

where β is a constant parameter and n_t controls the number of neighbors.

VI. Proposed method

In this project, a data set containing information from a printing company was used. This data warehouse contained information such as user name, company ID, company name, company address, company phone number, company-enabled state, and company ads. User name, company ID, and company name are used as location dimension features. Company address, company phone number, company-enabled state, and company ads are used as time dimension features.

Other pieces of data contain information about orders by these companies, including ID, user name, user ID advertised, company ID, order name, color of ordered item, requested size (X and Y), material used for ordered item, order date, enabled or disabled state, confirmed or unconfirmed order state, send order to user, and change of order by user. ID, user name, user ID advertised, and company ID are used as location dimension features. Order name, color of ordered item, requested size (X and Y), material used for ordered item, order date, enabled or disabled state, confirmed or unconfirmed order state, send order to user, and change of order by user are used as time dimension features.

As like [28] we use the genetic algorithms techniques and firefly algorithm to optimize finding best location for data in data cube. The data was to consider being genetic algorithm inputs and the initial population totaled 1000. The initial population of the firefly algorithm was set at 700. The values were inserted experimentally and the population distribution in the environment was very effective because the area was very large. Determination of the initial population was based on trial and error. Reduction of data size was done using the genetic algorithm and placement improvement was done using the firefly algorithm. The proposed algorithm was written as a pseudo-code. Algorithm 1 gives the output (S) set by testing the MLP as input for optimized placement and algorithm 2 receives this input and makes a hybrid genetic-firefly algorithm to find the appropriate place for data in the data cube.

Algorithm 1. MLP for optimizing data place.

```
Input: Dataset
Output: (S)
 1. Define weights and bias for MLP;
 2. Learn dataset with MLP;
 3. If error_occurs in learning of MLP
 4. Update weights and bias
 5. End
 6. Test dataset with MLP;
 7. Use output (S) from testing of MLP as input for optimized placement
```

Algorithm 2. Genetic algorithm and firefly algorithm as a hybrid algorithm for data cube.

```
Input: Output (S) from MLP (algorithm 1)
Output: Place data in data_cube
 1. Use the output (S) from testing of MLP as input for optimized placement;
 2. Initialize the parameters for both genetic and firefly algorithms;
 3. Generate main_population_size from the dataset for the genetic algorithm;
 4. Use the fitness function for the genetic algorithm;
 5. While generating for genetic from 1 to n do:
 6. Use genetic fitness function;
 7. Create new_population;
 8. Evaluate each individual;
 9. While new_population < main_population_size do:
10. Select parents from new_population using roulette wheel method;
11. Use genetic crossover;
12. Use genetic mutation;
13. Place offspring into new_population;
14. Endwhile
15. main_population_size ← new_population;
16. Lighting from new_generation for new population for firefly generated;
17. Define new firefly (component or population) for firefly algorithm;
18. Define distance beyond firefly (component or population);
19. Lighting for attractiveness of firefly;
20. Move firefly (component or population) for attractiveness with light;
21. Place data in data_cube
22. Endwhile
```

VII. SIMULATION RESULTS

Applying the optimization algorithm to the Genetic-firefly hybrid algorithm, the initial population of genetic algorithm totaled 1000 and firefly algorithm totaled 700. User name, company ID, and company name are used as location dimension features. Company address, company phone number, company-enabled state, and company ads are used as time dimension features. As it shown two dimensions are defined for this simulation; one "location" and the other "time". Simulation was done in a MATLAB environment using the observational method. At first step weights and bias are defined for multi-layer perceptron and learning with MLP occurred. If errors occurred in learning, weights and bias are updated. At the end in this phase, test dataset are tested. The data was first inserted into the data cube environment sporadically as shown in Figure 1. Now this phase output is as input for optimized algorithm. Parameters for both genetic and firefly algorithms initialize. The mail population size from dataset generate for genetic algorithm. Fitness function is used in this level and new population has been created and evaluated. In this part with using "Roulette Wheel" method, parents from new population are selected. Crossover and mutation in genetic algorithm occurs and offspring is placed into new population. The best data position was as shown in Figure 2. Firefly algorithm applies in new population and generated lighting. Distance beyond firefly population is defined. Component for attractiveness with light are moved and placed in data cube. Figure 3 shows the reduction of dimensions over time determined visually for the operation.

VIII. CONCLUSION

In this study, placement of data in the scalable space of a data cube was done using a hybrid genetic-firefly algorithm. The data was first inserted into the data cube environment sporadically then the best data position was shown and the reduction of dimensions made over time determined visually for the operation. Dataset clustering was carried out with the help of a perceptron multi-layer neural network and then the hybrid genetic-firefly algorithm was used for placement to find the best position for the data in the data cube. Decision-based programs include large-scale complex database queries. If the response time is short, query optimization is critical. Users usually observe data as a multi-dimensional data cube. Each data cube cell displays data as an aggregation in which the number of cells depends on the number of other cells in the cube. At any given time, a powerful query optimization method can visualize part of the cells instead of calculating results from raw data. Business systems use different approaches and positioning of data in the data cube. In the present study, the data is trained by a neural network and a genetic-firefly hybrid algorithm is proposed for finding the best position for the data in the cube.

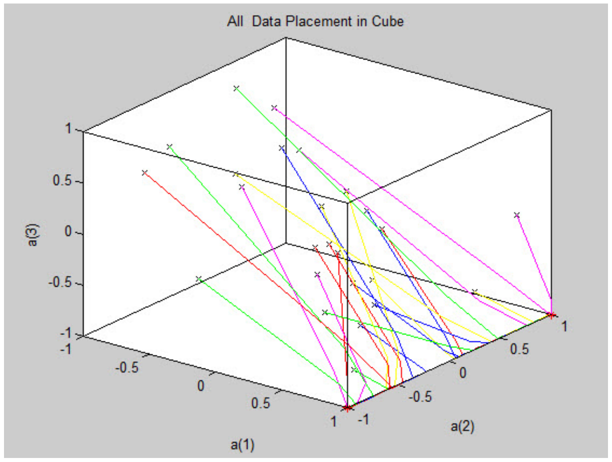

Fig. 1. Data in data cube.

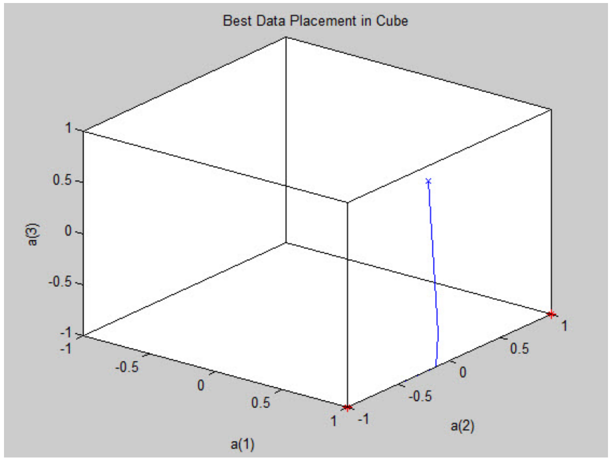

Fig. 2. Best data position in data cube.

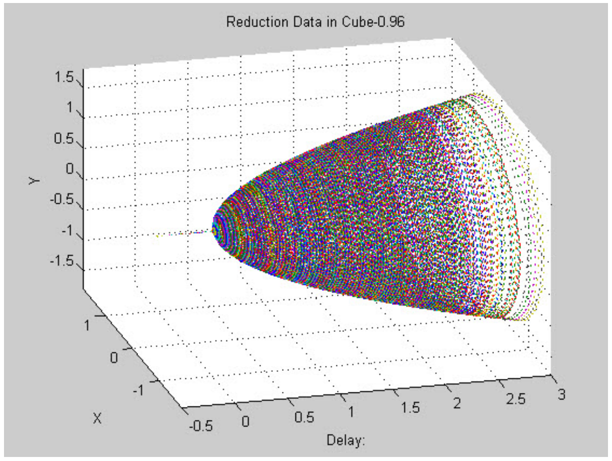

Fig. 3. Visualization of data dimensions for reducing dimensions of data cube.

REFERENCES

[1] A. Nandi, C. Yu, P. Bohannon, R. Ramakrishnan, "Data cube materialization and mining over mapreduce", IEEE Transactions on Knowledge and Data Engineering, Vol. 24, No. 10, pp. 1747-1759, 2012

[2] H. Cheng, Y. Lu, C. Sheu, "An ontology-based business intelligence application in a financial knowledge management system", Expert Systems with Applications,Vol. 36, No. 2, pp. 3614-3622, 2009

[3] L. Ogiela, "Data management in cognitive financial systems", International Journal of Information Management, Vol. 33, No. 2, pp. 263-270, 2013

[4] H. K. Chow, K. L. Choy, W. B. Lee, F. T. Chan, "Design of a knowledge-based logistics strategy system", Expert Systems with Applications, Vol. 29, No. 2 ,pp. 272-290, 2005

[5] D. Barbara, X. Wu, "Using loglinear models to compress datacubes", In Web-Age Information Management, Springer Berlin Heidelberg, pp. 311-323, 2000

[6] D. Barbará, M. Sullivan, "Quasi-cubes: exploiting approximations in multidimensional databases", ACM SIGMOD Record, Vol. 26, No. 3, pp. 12-17, 1997

[7] D. Xin, J. Han, X. Li, B. W. Wah, "Star-cubing: Computing iceberg cubes by top-down and bottom-up integration", 29th International Conference on Very Large Data Bases, Vol. 29, pp. 476-487, 2003

[8] D. Xin, Z. Shao, J. Han, H. Liu, "C-cubing: Efficient computation of closed cubes by aggregation-based checking in data engineering", 22nd International Conference on Data Engineering, (ICDE), April 3-7, 2006

[9] E. F. Codd, S. B. Codd, C. T. Salley, Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate, Codd & Associates, 1993

[10] S. Li, B. Lin, "Accessing information sharing and information quality in supply chain management", Decision Support Systems, Vol. 42 No. 3, pp. 1641-1656, 2006

[11] M. Kamber, J. Han, J. Chiang, "Metarule-guided mining of multi-dimensional association rules using data cubes", 3rd International Conference on Knowledge Discovery and Data Mining, Vol. 97, pp. 207-211, 1997

[12] J. Han, J. Pei, G. Dong, K. Wang, "Efficient computation of iceberg cubes with complex measures", In ACM SIGMOD Record, Vol. 30, No. 2, pp. 1-12, ACM, 2001

[13] J. S. Vitter, M. Wang, B. Iyer, "Data cube approximation and histograms via wavelets", 7th ACM International Conference on Information and Knowledge Management, pp. 96-104, 1998

[14] K. C. Lee, S. Lee, I. W. Kang, "KMPI: measuring knowledge management performance", Information and Management, Vol. 42, No. 3, pp. 469-482, 2005

[15] J. Brownlee, Clever Algorithms. Nature-Inspired Programming Recipes, Jason Brownlee, 2011

[16] K. Beyer, R. Ramakrishnan, "Bottom-up computation of sparse and iceberg cube", In ACM SIGMOD Record ,Vol. 28, No. 2, pp. 359-370, 1999

[17] N. Wiwatwattana, H. V. Jagadish, L. V. Lakshmanan, D. Srivastava, "X^3: A cube operator for xml olap", IEEE 23rd International Conference on Data Engineering, pp. 916-925, 2007

[18] K. A. Ross, D. Srivastava, "Fast computation of sparse datacubes", 23rd International Conference on Very Large Data Bases, Vol. 97, pp. 25-29, 1997

[19] A. Kavousi-Fard, H. Samet, F. Marzbani, "A new hybrid modified firefly algorithm and support vector regression model for accurate short term load forecasting", Expert Systems with Applications, Vol. 41, No. 13, 6047-6056, 2014

[20] K. J. Kim, I. Han, "Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index", Expert systems with Applications, Vol. 19, No. 2, pp. 125-132, 2000

[21] L. V. Lakshmanan, J. Pei, J. Han, "Quotient cube: How to summarize the semantics of a data cube", 28th International Conference on Very Large Data Bases, pp. 778-789, 2002

[22] L. V. Lakshmanan, J. Pei, J. Han, "QC-Trees: An efficient summary structure for semantic OLAP", 2003 ACM SIGMOD International Conference on Management of Data, pp. 64-75, 2003

[23] C. K. M. Lee, W. Ho, G. T. Ho, H. C. Lau, "Design and development of logistics workflow systems for demand management with RFID", Expert Systems with Applications, Vol. 38, No. 5, pp. 5428-5437, 2011

[24] D. Pelusi, "Designing neural networks to improve timing performances of intelligent controllers", Journal of Discrete Mathematical Sciences and Cryptography, Vol. 16, No. 2-3, pp. 187-193, 2013

[25] Y. Yuan, X. Lin, Q. Liu, W. Wang, J. X. Yu, Q. Zhang, "Efficient computation of the skyline cube", 31st International Conference on Very Large Data Bases, pp. 241-252, 2005

[26] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, J. D. Ullman, "Computing Iceberg Queries Efficientl", Internaational Conference on Very Large Databases, New York, Stanford InfoLab, 1998.

[27] S. H. Min, J. Lee, I. Han, "Hybrid genetic algorithms and support vector machines for bankruptcy prediction", Expert Systems with Applications, Vol. 31, No. 3, pp. 652-660, 2006

[28] D. Pelusi, "PID and intelligent controllers for optimal timing performances of industrial actuators", International Journal of Simulation: Systems, Science and Technology, Vol. 13, No. 2, pp. 65-71, 2012

[29] D. Pelusi, "Optimization of a fuzzy logic controller using genetic algorithms", IEEE International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), Vol. 2, pp. 143-146, 2011

[30] R. T. Ng, A. Wagner, Y. Yin, "Iceberg-cube computation with PC clusters", In ACM SIGMOD Record, Vol. 30, No. 2, pp. 25-36, 2001

[31] S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. Naughton, R. Ramakrishnan, S. Sarawagi, "On the computation of multidimensional aggregates", 22th International Conference on Very Large Data Bases, Vol. 96, pp. 506-521, 1996

[32] S. Sarawagi, M. Stonebraker, "Efficient organization of large multidimensional arrays", 10th International Conference on Data Engineering, pp. 328-336, 1994

[33] K. S. Shin, Y. J. Lee, "A genetic algorithm application in bankruptcy prediction modeling", Expert Systems with Applications, Vol. 23, No. 3, pp. 321-328, 2002

[34] S. Haykin, Neural Networks, a comprehensive foundation, 2004

[35] V. Poosala, V. Ganti, "Fast approximate answers to aggregate queries on a data cube", IEEE 11th International Conference on Scientific and Statistical Database Management, pp. 24-33, 1999

[36] V. Harinarayan, A. Rajaraman, J. D. Ullman, "Implementing data cubes efficiently", In ACM SIGMOD Record, Vol. 25, No. 2, pp. 205-216, 1996

[37] A. Vellido, P. J. Lisboa, P. J., J. Vaughan, "Neural networks in business: a survey of applications (1992–1998)", Expert Systems with Applications, Vol. 17, No. 1, pp. 51-70, 1999

[38] W. Wang, J. Feng, H. Lu, J. X. Yu, "Condensed cube: An effective approach to reducing data cube size", IEEE 18th International Conference on Data Engineering, pp. 155-165, 2002

[39] X. Li, J. Han, H. Gonzalez, "High-dimensional OLAP: a minimal cubing approach", 30th International Conference on Very large Data Bases, Vol. 30, pp. 528-539, 2004

[40] Y. Zhao, P. M. Deshpande, J. F. Naughton, "An array-based algorithm for simultaneous multidimensional aggregates", In ACM SIGMOD Record, Vol. 26, No. 2, pp. 159-170, 1997

[41] Z. Shao, J. Han, D. Xin, "MM-Cubing: Computing iceberg cubes by factorizing the lattice space in scientific and statistical database management", IEEE 16th International Conference on Scientific and Statistical Database Management, pp. 213-222, 2004

[42] C. W. Choo, "The knowing organization: How organizations use information to construct meaning, create knowledge and make decisions", International journal of information management, Vol. 16, No. 5, pp. 329-340, 1996.

[43] P. K. Dey, S. O. Ogunlana, "Selection and application of risk management tools and techniques for build-operate-transfer projects", Industrial Management and Data Systems, Vol. 104, No. 4, pp. 334-346, 2004

[44] J. A. Johannessen, B. Olsen, B., J. Olaisen, "Aspects of innovation theory based on knowledge-management", International Journal of Information Management, Vol. 19, No. 2, pp. 121-139, 1999

[45] D. Pelusi, R. Mascella, "Optimal control algorithms for second order systems", Journal of Computer Science, Vol. 9, No. 2, pp. 183-190, 2013