Vertical fragmentation and allocation in distributed databases using k-mean algorithm

Ahmed Benmelouka[1]*, Benameur Ziani[1], and Youcef Ouinten[1]

[1]LIM Laboratory, Amar Telidji University of Laghouat, Bp 37G, Ghardaia Road, Laghouat 03000, Algeria
*Corresponding author Email: a.benmelouka@univ-djelfa.dz

***Abstract*— Distributed database systems are increasingly becoming the dominant tools for data management. However, in these systems, the sites are remote and exchange a huge amount of data, which leads to bottlenecks as well as large disk accesses in data transfers that may be irrelevant. Query processing optimization techniques are an important concern for system administrators to improve the performance of distributed database systems (DDBS). Vertical fragmentation is a suitable solution but its complication lies in the large number of alternatives to obtain an optimal solution. This paper provides a new approach better suited to the problem of vertical fragmentation by the K-means classification algorithm but with our new adequate distance. To validate our approach, we compared our solution first with a vertical fragmentation algorithm called VFAR and second with the same k-means algorithm with the hamming distance.**

Keywords- vertical fragmentation; distributed databases; K-means; distance.

I. INTRODUCTION

The complexity and diversity of computing systems, which connect geographically dispersed computing and storage resources across many enterprises and layered network architectures, manage massive, unbelievable, and expanding volumes of information, create exceptional challenges on distributed implementation.

Distributing data over several networked systems lowers costs while enhancing performance and availability [1]. The physical design of distributed databases requires not only the exact description of the data and data types but also the choice of suitable optimization techniques that can improve system performance by lowering processing times and transfer requirements for the evaluation of the requests.

Finding a design method that lowers processing costs while requiring the least amount of data access is a top priority. One of the most popular methods is vertical fragmentation, which involves dividing up a given table into a number of vertical fragments in order to lower query processing costs by reducing the amount of data.

The number of ways a set of elements can be split into non-empty subsets is almost equal to KK (B(k) bell number) [2]. This is motivated by the fact that the fragmentation problem has a huge search space for optimal solutions and is difficult enough to study on its own.

Since determining the perfect fragmentation scheme is a NP-complete problem [3]-[4], optimization methods are used.

The fundamental tenet of partitioning techniques is to maximize distances between recognized clusters and decreasing distances inside identified clusters.

Queries are no longer seen as unique items, but rather as groupings of similar elements.

In this paper, we apply the most well-known classification approach in the literature, the K-means algorithm, which is still the most widely used classification tool in scientific and industrial applications [3]-[4]-[5].

There are various distances used in this type of classification, the most well-known such as the Euclidean distance, the Hamming distance, and so on. Our contribution in this article is: first, we propose a new distance that is perfectly adapted to the binary representation of the workload; and second, we present a solution to vertical fragmentation via the automatic classification k-means based on the proposed distance.

The rest of the article is organized as follows: The previous researches on vertical fragmentation are described in Section II. Motivation is covered in Section III. The proposed strategy is presented in Section IV. The experiment's results are covered in Section V. In section VI, the conclusion is presented.

II. RELATED WORK

In different contexts, such as centralized relational databases, distributed databases, object-oriented databases, and in the context of centralized or distributed data warehouses, different work on vertical fragmentation is being done. While in a centralized context, fragmentation aims to lower the costs associated with processing requests, in a distributed context, it also aims to improve the distribution of data across remote sites in order to achieve parallel processing of requests.
As a result, we restrict ourselves to listing the primary works on vertical fragmentation in reverse chronological order.

Hoffer presents a method for vertical fragmentation in [6] and [7] that is based on the Bond Energy Algorithm (BEA) by taking use of the attribute affinity matrix, where the affinity is calculated between pairs of attributes and clusters with high affinities are formed.

After the introduction of BEA, a new algorithm was developed in [8] by adding a second step to the previous approach of recursively dividing the affinity matrix into two halves by minimizing the number of queries that access attributes common to both halves in an effort to reduce the subjectivity of the final interpretation. The proposed approach is called binary fragmentation because it generates two fragments.

By proposing a vertical fragmentation method using a graphical technique, the authors of [9] replicated the previous work. This method uses a network termed an affinity graph to describe the affinity matrix and then uses that graph to obtain cycles that specify the detected fragments.

Optimal Binary Partitioning Algorithm (OBP), a further method for vertical binary fragmentation, has been put out in [10]. It creates the many fragments of a tree structure.

The attributes are divided into two categories for each query: those that the query references and those it does not. The technique generates a tree whose leaves include the attribute groups. When merging neighboring leaves of the tree, the best binary partitioning is then determined by using a cost function.

[11] Propose an implementation strategy for a design approach that performs vertical fragmentation on the database in accordance with frequent elements and, more precisely, sets of elements closed by the *FPClose* algorithm.

[12] formalized the issue of fragmentation as a linear programming problem and divided views into smaller tables, which increased access costs that might be decreased by combining views into bigger tables.

[13] suggest a genetic algorithmic solution to the issue of vertical fragmentation that also incorporates the path selection strategy to enhance database performance.

To lower the cost of query execution in a relational data warehouse environment, the authors of [14] suggest a hybrid partitioning based on a genetic algorithm.

In [15], when a partition is formed, the attributes are distributed among different geographical locations. The schema determines the success rate of a partition. Moving an attribute that is loosely coupled to a different subset in a partition improves the success rate.

[16] adapts the *Apriori* algorithm to propose a method for resolving the issue of vertical fragmentation in three steps. The first step entails searching for frequent motives in a workload, and the second step generates candidates for fragmentation schemes where the largest motive for the first fragment is taken into account. The remaining patterns are then analyzed in decreasing order of their size, with the optimal scheme being suggested in the last phase using a cost model.

According to the update cost values of the clusters, the authors in [17], offer a strategy for grouping sites into clusters to which attributes would be allocated during the initial allocation phase. The second stage of the allocation process will be completed based on the attribute priority value. The candidate site with the highest value will be chosen to store the attribute.

An technique for vertical partitioning in distributed databases is presented by the authors in [18]. Valley Based Vertical Partitioning Algorithm is the name of the suggested algorithm (VBVPA). This approach uses the Clustered Affinity Matrix (CAM), which is created by combining the Attribute Usage Matrix (AUM) and the Frequency Matrix (FM).

The Authors offer an automated solution for vertical fragmentation in [19]. The suggested method uses the affinity matrix to identify sets with frequent attributes and keeps the top-k sorted sets based on their supports. A binary partition is defined by each set of attributes that is kept. The method then evaluates if the binary partitions created for each load request are optimal. The system optimizer makes an estimation of a partition's relevance for a certain query. The resultant bit groups may then be combined, and the cost of every conceivable merged partition can be calculated, in a merge step. The division combined with the optimum cost is finally selected.

The fuzzy logic idea is used to divide a single relationship's attributes into two or more partitions in [20], or the distributed database administrator determines the relative weights of each attribute's importance before fragmentation.

An approach that uses the Bond Energy Algorithm (BEA) for data fragmentation vertically and site allocation is described in [21]. The program concurrently constructs attribute clusters, determines the cost of each cluster's allocation at each site, and assigns each cluster to the right site using a superior affinity metric.

In order to efficiently find an optimum solution in a smaller possible search space, [22] provides vertical partitioning based on maximum frequent item sets.

The authors in [23] advise complete vertical allocation, fragmentation, and replication (FVFAREL). First, a specification analysis of the extended create, read, update, and delete (CRUD) matrix is performed. The scattered data collecting tables are then vertically sliced using Prim's Enhanced Minimum Spanning Tree (MST) approach, after copies are assigned to distant sites as part of the FVFAREL procedure.

In [24], the authors present an optimized scheme for vertical fragmentation in a distributed database system using Differential Evolution Algorithm (DE). This algorithm is a hybrid of differential evolution (scalable algorithm) and bond energy algorithm (classical vertical fragmentation algorithm).

III. MOTIVATION

Most clustering approaches employ Euclidean distance as a metric of similarity. However, in cases like ours, we are dealing with a request load represented in binary form, which necessitates an acceptable binary distance. Two queries are similar if they share a large number of attributes, but distant if they share fewer attributes; based on this concept, we proposed a new distance that focuses on the number of attributes shared by queries; for the workload, we chose a binary representation in which each attribute in the query is represented by the number 1; an inspiring illustration of this may be seen in the following:

Consider the points :

X : 1111000000000
Y : 1100000000000
Z : 1101000110000

We want to calculate the distance between *Y*, *Z*, and *X* using three metrics: Euclidean, Hamming, and our distance; the results are shown in *Table01*.

Point	Euclidean	Hamming	Our distance
X:1111000000000	0.00	0.00	0.00
Y:1100000000000	**1.41**	**2.00**	2.50
Z:1101000110000	1.73	3.00	**2.40**

Table 01: *Distances from point X.*

The given example illustrates that the Euclidean distance brings point *Y* closer to point *X*. The same is true for the Hamming distance; however, for our distance, it is point *Z* that is near, which is practically right because a request with three attributes in common is closer than a request with two attributes in common.

Given the particularity of the problem, ie the search for queries having the maximum number of attributes in common to be in the same group, for this our binary distance gives a better classification; We will give a brief description of our approach and the suggested distance in the sections that follow.

IV. OUR APPROACH: K-MEANS WITH NEW DISTANCE

1. Proposed distance.

A. *Definition*

A real function d defined on any set E is also a distance function if and only if the following four conditions are satisfied [25]:

- $\forall x, y \in E,\ d(x,y) \geq 0$
- $\forall x, y \in E,\ d(x,y) = 0 \Leftrightarrow x = y$
- $\forall x, y \in E,\ d(x,y) = d(y,x)$
- $\forall x, y, z \in E,\ d(x,z) \leq d(x,y) + d(y,z)$

We see that the preceding four requirements are satisfied by the suggested distance. Distance formula.

B. *Distance formula*

Be two points:

x et y such as: $x\,(x_1, x_2 \dots, x_n)$ $y\,(y_1, y_2 \dots, y_n)$

$$x_i = 0 \text{ or } 1 \text{ and } y_i = 0 \text{ or } 1$$

$$\boxed{d\,(x,y) = 1 - \frac{In\ common(1_{(x\ ,\ y)})}{Max(1_{(x\ ,\ y)}) + Max[\,different\,(1_{(x\ ,\ y)})]}}$$

- $In\ common(1_{(x\ ,\ y)}) = \sum_{i=1}^{n}(x_i * y_i)$
- $Max(1_{(x\ ,\ y)}) = Max(\sum_{i=1}^{n} x_i\,, \sum_{i=1}^{n} y_i)$

$$Max[\,different\,(1_{(x\ ,\ y)})] = Max\left(\sum_{i=1}^{n} x_i - \sum_{i=1}^{n}(x_i * y_i)\,, \sum_{i=1}^{n} y_i - \sum_{i=1}^{n}(x_i * y_i)\right)$$

$$\boxed{d\,(x,y) = 1 - \frac{\sum_{i=1}^{n}(x_i * y_i)}{Max(\sum_{i=1}^{n} x_i\,, \sum_{i=1}^{n} y_i) + Max\left(\sum_{i=1}^{n} x_i - \sum_{i=1}^{n}(x_i * y_i)\,, \sum_{i=1}^{n} y_i - \sum_{i=1}^{n}(x_i * y_i)\right)}}$$

2. SUGGESTED TECHNIQUE.

A. *Illustrative example.*

An example of query characteristics and their representations for the categorization process is shown in *Table 02* and *Table 03*, respectively.

Queries	Attributes	Frequency
q_1	a,b,e	6
q_2	b,e	3
q_3	a,c,d,f	2
q_4	b,d,e	3
q_5	b,d,e,f	4

Table 02: *Example of request characteristics.*

	a	b	c	d	e	f	Frequency.
q_1	1	1	0	0	1	0	6
q_2	0	1	0	0	1	0	3
q_3	1	0	1	1	0	1	2
q_4	0	1	0	1	1	0	3
q_5	0	1	0	1	1	1	4

Table 03: *Preparing Query Classification*

B. *The stages of our technique.*

Assume that for a three-class classification *(k=3)*, the produced groups are $G_1 = \{q_1; q_2\}$, $G_2 = \{q_4; q_5\}$ *and* $G_3 = \{q_3\}$. The optimum fragmentation pattern search is generated as follows:

- We begin with G_1 since the total of its request frequencies is the highest. The query q_1 is the most frequent in this category. As a result, *{a; b; e}* initializes the first fragment. In the case of query q2, the fragment stays unchanged since *{a; e}* is included within *{a; b; e}*.

- We now look at the group *G2 = {q4; q5}*. The most frequent query, *q5*, results in the production of the fragment *{d;f}* (the attributes *b* and *e* being already assigned to the first fragment). The q_4 query produces no results since all of its attributes are already in use.

- The final group $G_3 = \{q_3\}$ results in the formation of the fragment *{c}*.

  Thus, the recommended fragmentation scheme is *{a, b, e}; {d; f};{c}*.

V. EXPERIMENTAL STUDY

A. *Experimental Setup*

The scientific community can evaluate the effects of various technical decisions by using benchmarks [26]. We took use of the data warehouse from the TPC-H test bench that the TPC (Transaction Processing Council) updated and made accessible to the scientific community[27].
A *Lineitem* fact table and seven dimension tables *Part*, *Supplier*, *Partsupp*, *Customer*, *Orders*, *Nation*, and *Region* make up the warehouse under consideration. The characteristics of the warehouse tables are illustrated in *Table 04*.
There are *22* queries total in the load being examined. Each request is identified by the collection of attributes it mentions as well as how frequently it is used. *Table 05* contains a description of each attribute that the fragmentation process takes into account.

Table	Number of records
Lineitem	6 000 000
Part	200 000
Supplier	10 000
Partsupp	800 000
Customer	150 000
Orders	1 500 000
Nation	25
Region	5

Table 04: *Characteristics of TPC-H warehouse tables*

Code	Attribut	Type	Size (octets)
a	LineNumber	Integer	4
b	Quantity	Decimal	8
c	ExtendedPrice	Decimal	8
d	Discount	Decimal	8
e	Tax	Decimal	8
f	ReturnFlag	Fixed text	1
g	LineStatus	Fixed text	1
h	ShipDate	Date	7
i	CommiDate	Date	7
j	ReceipDate	Date	7
k	ShipInStruct	Fixed text	25
l	ShipMode	Fixed text	10
m	Comment	Variable text	44

Table 05: *Attribute characteristics.*

All experiments were carried out on an Intel(R) Core(TM) i3 2.40GHz processor with 4.00 GB of RAM running the Linux Ubuntu 21.04 operating system. In order to validate our contribution, manipulate the data, and test the performance of the generated configurations, we created a java application for k-means classification that integrated two distances: Hamming and our proposed distance.

B. Tests and Results.

In the first experiment, we determined the total cost of the query load for the two approaches, including the VFAR strategy (a vertical fragmentation, allocation, and replication scheme of a distributed database) suggested in [28]. for the various values of *k*: *k=2*, *k=3*, *k=4*, *k=5*, *k=6*, and *k=7*, and our method based on our newly suggested binary distance, described in the previous section, the results are shown in *Table 06*. Our technique provides a better execution cost essentially for all values of k, as can also be observed in *figure 01*. We can see that we have improved over *60%* over the *VFAR* technique if we use the best execution cost for our strategy, which is *k=4* with a cost of *5 432 504,88*.

	VFAR	OurApproach
K = 2	12 187 631,84	6 552 377,93
K = 3	14 310 922,85	7 193 979,49
K = 4	13 778 452,15	5 432 504,88
K = 5	13 262 094,73	7 049 692,38
K = 6	12 213 999,02	7 099 497,07
K = 7	11 743 051,76	7 099 497,07

Table 06: *Execution cost according to the value k.*

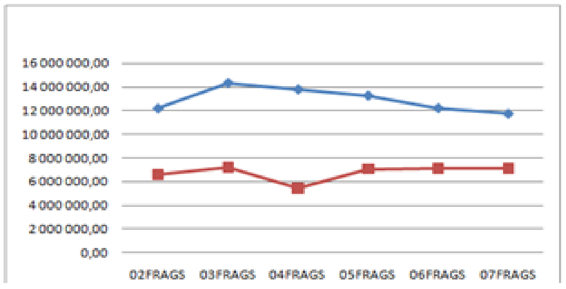

Figure 01: *Execution cost according to the value k*

For our second experiment we calculated the execution cost for the values *K=2*, *K=4* and *K=6* using the two distances for the classification, namely the binary Hamming distance and our proposed distance; the results show that with our distance we obtain better execution costs as shown in *Table 07* and *Figure 02*. Our approach therefore constitutes an improvement in the accuracy of the k-means classification by the new distance which adapts well to the binary representation of the query load.

	Hamming distance.	Our distance
K = 2	7 032 846,68	6 552 377,93
K = 4	6 706 918,95	5 432 504,88
K = 6	7 110 483,40	7 099 497,07

Table 07: *Comparison of execution costs according to distance.*

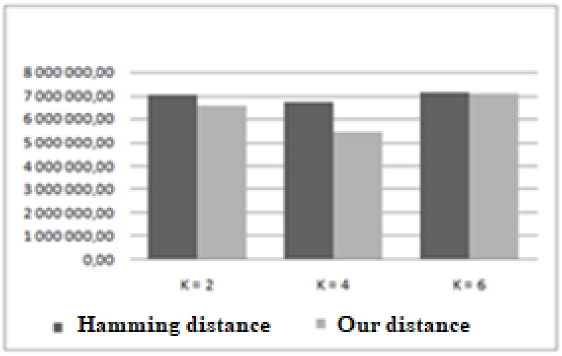

Figure 02: *Comparison of execution costs according to distance.*

In a third experiment, we varied the scal factor from 1, 10, 30,..., at 100,000. We then calculated the query load execution cost for each case with k=4 (best execution cost), and the results are shown in *Table 08*. As the selectivity factor is significant because it determines the number of blocks to be transferred, we did this to gauge its impact; it can be seen that the execution cost increases proportionally as the selectivity factor's value increases.

Scale factor (SF)	Cardinality of LINEITEM Table	Execution cost
1	6 001 215	5 432 504,88
10	59 986 052	40 735 639,72
30	179 998 372	119 215 375,83
100	600 037 902	393 892 104,62
300	1 799 989 091	1 178 576 982,48
1 000	5 999 989 709	3 925 086 175,67
3 000	18 000 048 306	11 772 292 462,80
10 000	59 999 994 267	39 237 345 015,66
30 000	179 999 978 268	117 709 014 240,93
100 000	599 999 969 200	392 359 887 217,33

Table 08: *Impact of the Scale factor on the execution costs.*

VI. Conclusions and Perspectives

Distributed databases are becoming increasingly important as the volume of stored data increases. Despite the popularity and development of distributed databases, performance remains a major issue. This is because database queries are usually complex, often requiring expensive computational operations such as joins and aggregations.. The cost of a query, on the other hand, can vary by several orders of magnitude in various physical implementations of the same logic diagram. Therefore, the physical architecture chosen for the implementation has a considerable influence on the performance of distributed databases.

In this paper, we proposed a solution for vertical fragmentation guided by a data mining technique based on k-means classification by integrating a binary distance that allowed us to have a better classification, proven by the tests carried out on the TPCH Benchmark and which gave us satisfactory results.

As perspectives, it is suggested that we extend our approach to other physical design techniques, such as horizontal or mixed fragmentation on the one hand, and consider vertical fragmentation of databases distributed by other data mining techniques on the other hand, which are very beneficial to reducing the complexity of exploring the solution search space and building an optimized solution.

VII. References

[1] S. Mehta, P. Agarwal, P. Shrivastava, and J. Barlawala, "Differential bond energy algorithm for optimal vertical fragmentation of distributed databases," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, pp. 1466-1471, 2018.

[2] M. Hammer and B. Niamir, "A heuristic approach to attribute partitioning," in *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, 1979, pp. 93-101.

[3] N. Rogouschi and Y. Bennani, *Classification à base de modèles de mélanges topologiques des données catégorielles et continues*, 2009.

[4] H. Elghazel and A. D.), *Classification et prévision des données hétérogènes: application aux trajectoires et séjours hospitaliers*, 2007.

[5] D. Weiss, "Descriptive clustering as a method for exploring text collections," *Unpublished doctoral dissertation, Poznan University of Technology, Poznan, Poland*, 2006.

[6] J. A. Hoffer and D. G. Severance, "The use of cluster analysis in physical data base design," in *Proceedings of the 1st International Conference on Very Large Data Bases*, 1975, pp. 69-86.

[7] W. T. McCormick Jr, P. J. Schweitzer, and T. W. White, "Problem decomposition and data reorganization by a clustering technique," *Operations research*, vol. 20, pp. 993-1009, 1972.

[8] S. Navathe, S. Ceri, G. Wiederhold, and J. Dou, "Vertical partitioning algorithms for database design," *ACM Transactions on Database Systems (TODS)*, vol. 9, pp. 680-710, 1984.

[9] S. B. Navathe and M. Ra, "Vertical partitioning for database design: a graphical algorithm," in *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, 1989, pp. 440-450.

[10] W. W. Chu and I. T. Ieong, "A transaction-based approach to vertical partitioning for relational database systems," *IEEE Transactions on Software Engineering*, vol. 19, pp. 804-812, 1993.

[11] A. S. Al-Shannaq and S. Almotairi, "Vertical Fragmentation for Database Using FPClose Algorithm," *Journal of Information Security and Cybercrimes Research*, vol. 2, pp. 110-115, 2019.

[12] M. Golfarelli, D. Maio, and S. Rizzi, "Vertical Fragmentation of Views in Relational Data Warehouses," in *SEBD*, 1999, pp. 19-33.

[13] S.-K. Song and N. Gorla, "A genetic algorithm for vertical fragmentation and access path selection," *The Computer Journal*, vol. 43, pp. 81-93, 2000.

[14] Z. Elhoussaine, D. Aboutajdine, and A. El Qadi, "Complete algorithm for fragmentation in data warehouse," *Age*, vol. 1, p. 1, 2008.

[15] E. S. Abuelyaman, "An optimized scheme for vertical partitioning of a distributed database," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, pp. 310-316, 2008.

[16] N. Gorla and P. W. Betty, "Vertical fragmentation in databases using data-mining technique," *International Journal of Data Warehousing and Mining (IJDWM)*, vol. 4, pp. 35-53, 2008.

[17] H. I. Abdalla, A. A. Amer, and H. Mathkour, "A Novel Vertical Fragmentation, Replication and Allocation Model in DDBSs," *J. Univers. Comput. Sci.*, vol. 20, pp. 1469-1487, 2014.

[18] S. Medhavi and A. Kumar, "Valley based vertical partitioning in distributed database," *e-Journal of Science & Technology*, vol. 10, 2015.

[19] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala, "Database tuning advisor for microsoft sql server 2005," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 930-932.

[20] G. H. Eladl, "A Modified Vertical Fragmentation Strategy for Distributed Database," 2017.

[21] H. Rahimi, F.-A. Parand, and D. Riahi, "Hierarchical simultaneous vertical fragmentation and allocation using modified Bond Energy Algorithm in distributed databases," *Applied computing and informatics*, vol. 14, pp. 127-133, 2018.

[22] B. Ziani, Y. Ouinten, and M. Bouakkaz, "MaxPart: An Efficient Search-Space Pruning Approach to Vertical Partitioning," *Computing and Informatics*, vol. 37, pp. 915-945, 2018.

[23] P. Sathishkumar and M. Gunasekaran, "An improved vertical fragmentation, allocation and replication for enhancing e‐learning in distributed database environment," *Computational Intelligence*, vol. 37, pp. 253-272, 2021.

[24] S. M. a. P. A. a. P. S. a. J. Barlawala, "Differential bond energy algorithm for optimal vertical fragmentation of distributed databases," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 1466-1471, 2022.

[25] FrancoiseGeandier, "Cours de topologie et analyse Hilbertienne ".

[26] J. Darmont, O. Boussaïd, and F. Bentayeb, "DWEB: A Data Warehouse Engineering Benchmark," in *International Conference on Data Warehousing and Knowledge Discovery*, 2005.

[27] TPC-H., "Transaction processing performance council," 2022.

[28] A. E. a. B. Abdel Raouf, Nagwa L. and Tolba, M. F., "An optimized scheme for vertical fragmentation, allocation and replication of a distributed database," in *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2015.