Using Genetic Algorithm for Optimal Allocation of Data Fragments in Distributed Database Systems

Ogheneovo, E.E.
Department of Computer Science
University of Port Harcourt
Port Harcourt, Nigeria.
edward_ogheneovo@yahoo.com

Okoye, C.
Department of Computer Science
University of Port Harcourt
Port Harcourt, Nigeria.
noblelink123@gmail.com

ABSTRACT

Communication and other related costs has been a major concern when discussing distributed database system since data often reside in different sites that are dispersed over long distances. These costs can be minimized by partitioning tables into smaller fragments. This paper proposed the genetic algorithm technique for allocating fragments in a distributed database. The choice of the genetic algorithm is important because the technique can handle data over large databases or sites that are geographically spread across large distances. Also, genetic algorithm provide a platform for natural selection and evolution since it is a robust and adaptive method for solving search and optimization problems in a large irregular search space. Thus genetic algorithm is used to determine the most efficient allocation procedure. Finally, we show through experimental analysis, that the procedure adopted produced solution that is better and optimal when compared with the results obtained by other researchers in terms of the run time and hence costs of communication and other associated costs.

Keywords: Database, genetic algorithm, fragment allocation, optimal solution & search algorithm

Aims Research Journal Reference Format:
Ogheneovo, E.E. & Okoye, C. (2017): Using Genetic Algorithm for Optimal Allocation of Data Fragments in Distributed Database Systems. Advances in Multidisciplinary & Scientific Research Journal. Vol. 3. No.1, Pp 63-72

1. INTRODUCTION

Data allocation is an important factor when considering the design of distributed databases and database systems in general [1] [2] [3]. However, finding optimal solutions as a means of resolving data allocation issue in distributed databases is of major concern and a serious challenge to database developers. This is mainly due to the fact that several sites or databases that are geographically spread at relatively long distances holds fragments of the data to be allocated. If a distributed database is not properly developed, the implication is that a lot of time and resources will be expended on allocating such data and this could increase costs of information distribution and retrieval drastically. Thus in designing a distributed database, a lot of factors must be put into consideration if such a database must be optimal in terms of costs, efficiency, and data contained in them must be easily accessible to users on demand. Data or fragment allocation is used to determine the best location for each data fragment. Fragment allocation is a problem with high complexity as a result; it is a Non-deterministic Polynomial (NP)-complete problem. Hence many suggestions have been proposed in the past to reduce its complexity [4] [5] [6] [7].

A distributed database system (DDBS) [8] [9] [10] is a collection of sites connected by a communication network, in which each site is a database system in its own right, but the sites have agreed to work together, so that a user at any site can access data anywhere in the network exactly as if the data were all stored at the user's own site [11] [12]. The essence of developing a DDBS is to meet the information requirements of organizations whose businesses involve distributed operations in their day to day transactions. This was accelerated by the advancement in telecommunication systems, hence such business organizations have branches or facilities where their computer systems are connected together with some communication links. Data or fragment allocation technique is used to determine the best location for each data fragment. Fragment allocation is a problem with high complexity as a result; it is a Non-deterministic Polynomial (NP)-complete problem. Hence many suggestions have been proposed in the past to reduce its complexity [13] [14].

Fragmentation is important because it improves the performance of the DDBMS; this is done by an increase in the efficiency of DDBMS and making sure that data are stored only where it is needed. Storing data where it is needed is achieved through a process called fragment allocation [15] [16]. Fragmentation is a design technique to divide a single relation or class of a database into two or more partitions such that the combination of the partitions provides the original database without loss of information [17]. Three types of fragmentation include: horizontal, vertical and hybrid or mixed fragmentation.

Fragmentation helps to minimize the quantity of unrelated data accessed in the DDBMS by the application running on it, which in turn reduces the frequency of data accesses. In fragment allocation, fragments are placed at the sites of DDBS. This is done to minimize the data transfer costs which include: cost of querying, cost of storing and cost of updating while application is being processed; which are the major problems of fragment allocation. Hence, each fragment is allocated to a location within a distributed environment such that the system functions more effectively and efficiently [18] [19].

In this paper, Genetic Algorithm (GA) [20] [21] is used for optimal allocation of data fragments in distributed databases. Genetic Algorithms are search procedures modeled on natural selection and evolution that is very robust for solving and optimization problems. They are mostly used for solving problems that has large irregular search space and for problems that are very complex and the best solution is obtained after navigating through a large search space. The best solution is gotten after searching through a large search space of complex problems [22] [23] [24] [25].

GA uses the same pattern which human genetic process operates in arriving at solution to problems. It also genetic algorithm provide a platform for natural selection and evolution since it is a robust and adaptive method for solving search and optimization problems in a large irregular search space [26] [27]. Thus genetic algorithm is used to determine the most efficient allocation procedure. This work is aimed at seen how fragments can be allocated in distributed databases more easily and efficiently. In this paper, we proposed the genetic algorithm technique for allocating fragments in a distributed database. The choice of the genetic algorithm became imperative because the algorithm can handle data over large database or sites that are geographically spread across large distances.

2. RESEARCH PROBLEM

Communication cost has been a major concern when discussing distributed database system since data often reside in different sites that are geographically dispersed. This cost can be minimized by partitioning tables into smaller fragments. This can be done horizontally, vertically, or mixed, a condition often referred to as mixed fragmentation. The fragments are then allocated or re-allocated to network sites where they can be easily accessed. This is done to ensure that most data resides in local databases for easy access as opposed to remote accesses which could cause delay and incur high costs. An important challenge in fragment allocation is how to allocate the fragment to achieve the minimal communication cost. Many previous researches have proposed network sites clustering as a solution to minimizing communication cost. In these works, sites which have similar communication costs are placed together so as to increase the performance of the distributed database system. Fragments are allocated to clusters and subsequently allocated to network sites.

One major setback in these research works is that the clustering technique is mostly conducted on a small scale distributed database. The implication is that as the volume of distributed database increases nowadays there is need for an efficient technique which can be used to handle fragment allocation. This paper proposed the genetic algorithm technique for allocating fragments in a distributed database. The choice of the genetic algorithm is important because the technique can handle data covers a large distances or database that are geographically spread across large distances. Also, genetic algorithm provide a platform for natural selection and evolution since it is a robust and adaptive method for solving search and optimization problems in a large irregular search space.

3. METHODOLOGY

As seen in figure 1, the genetic algorithm an initial population to determine the run time allocation of fragments for sites and generate allocation solution. This is done using the genetic operators: selection, crossover, and mutation which are discussed fully in figure 2. Figure 2 shows the detail or lower level design for figure 1. Detail in the sense that the various activities and steps involved in the genetic process are fully represented and discussed.

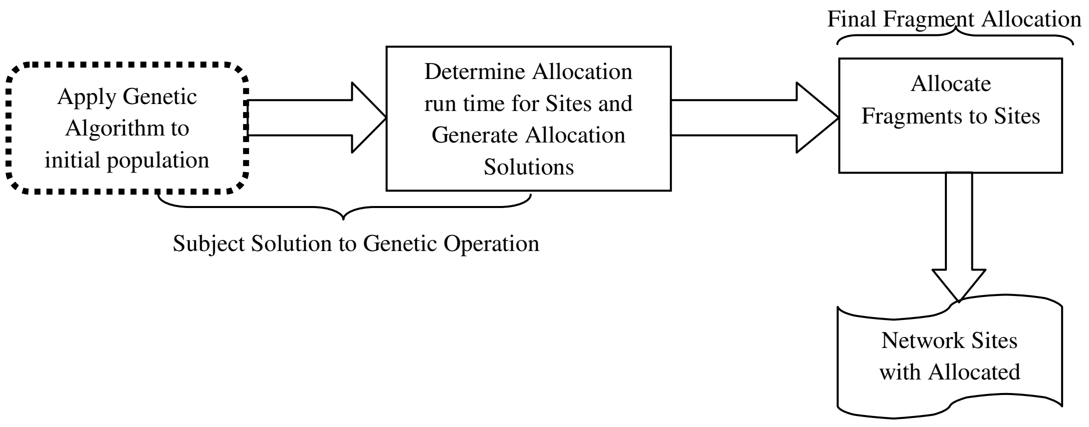
Final Fragment Allocation
Apply Genetic Algorithm to initial population
Determine Allocation run time for Sites and Generate Allocation Solutions
Allocate Fragments to Sites
Subject Solution to Genetic Operation
Network Sites with Allocated

Fig. 1: The genetic algorithm for determining cost function for sites and allocation solution

In figure 2, the initial solution of the entire population is evaluated and calculating their fitness. The selection is done using the selection operator for individual with lower fitness function to guarantee lower cost. The pairs of individuals randomly selected are the crossover using the crossover operation. After this, the mutation operator is applied to the selected population by computing individual's fitness level and sometimes altering the a random bit in a string for the worst fitness individuals in order to obtain better individuals with good genetic traits.

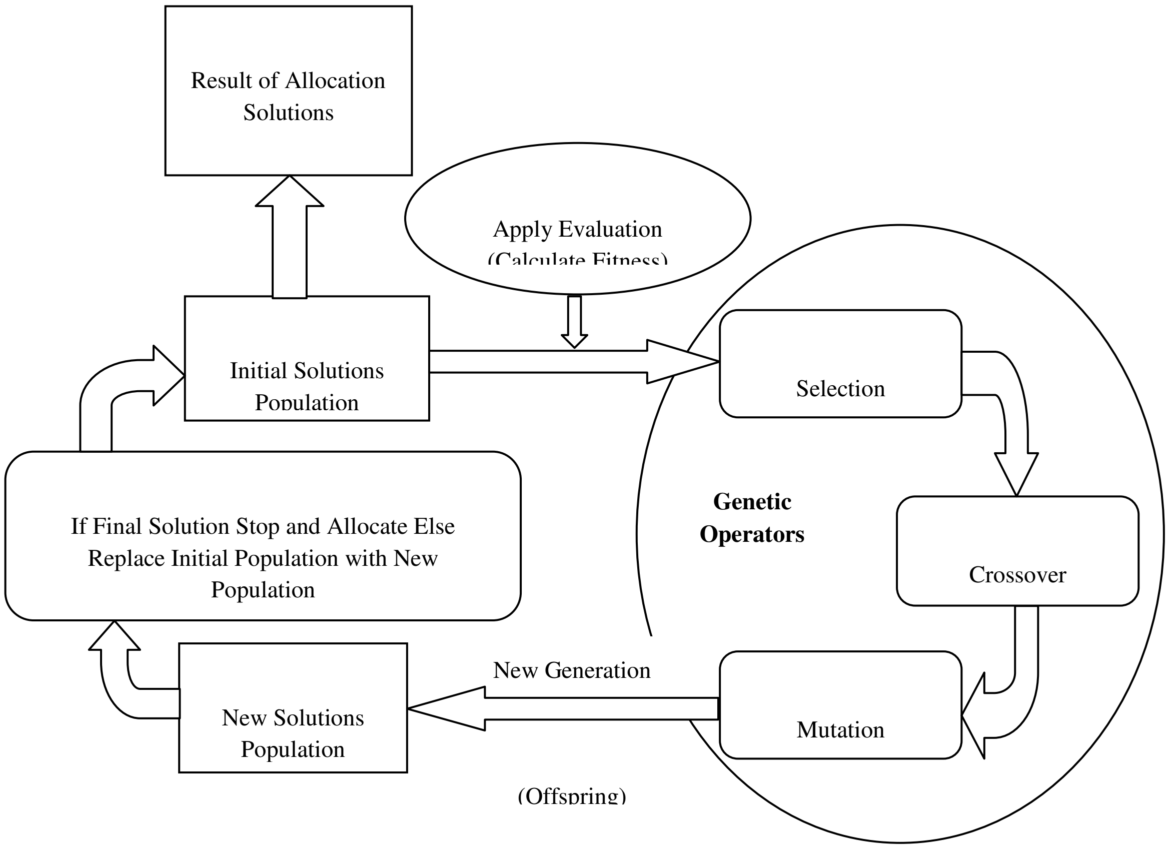
Result of Allocation Solutions
Apply Evaluation (Calculate Fitness)
Initial Solutions Population
Selection
Genetic Operators
Crossover
If Final Solution Stop and Allocate Else Replace Initial Population with New Population
New Generation
Mutation
New Solutions Population
(Offspring)

Fig. 2: Detail design of allocating fragments using the generic algorithm.

Based on this, new solution of offspring is produced from the old generation. The process is repeated until the final population is selected. Figure 2 shows the detailed diagram of the genetic algorithm procedure and steps. The final result is the allocated solutions produced.

3.1 The Genetic Algorithm Operators

To obtain the best and optimal solution, GA uses some operators. These are: selection, crossover and mutation. These operators are applied to an initial population of solutions to generate a new and better population of solutions. The initial population comprises of many chromosomes or strings each of which represents a solution to the problem. The fitness function is computed for each individual solution after which GA operators are applied to the population to generate a new one. The input is subjected to these operators iteratively in order to obtain the optimum solution. Each iteration is called a generation. Each generation obtained after iteration is better and closer to the optimum solution. The iteration is continued until the best solution is obtained.

i. **The Selection Operator**

The selection operator selects the individuals (chromosomes or strings) which have a lower fitness function value. Individuals with a lower fitness represent the ones with lower communication cost, thus they have higher probability of being selected. After selection crossover is performed on them.

ii. **The Crossover Operator**

Crossover is very similar to what is obtainable in reproduction. Pairs of individuals are randomly selected for the crossover operation is performed on them. After the crossover, new pair of individuals with better fitness level is obtained when compared with the parents. Crossover is performed on these individuals by combining different parts of the selected string to form a new pair of string. The resultant off springs obtained is accepted only if it has a better fitness than the parent strings is then added to the new population which then replaces the individuals that have been performing poorly in the previous population

iii. **The Mutation Operator**

After crossover, the mutation operator is applied to the population. This is done by computing individual's fitness level and occasionally altering a random bit in a string for the worst fitness individuals in order to obtain better individuals with good genetic traits. This process is repeated until an optimal (or near optimal) solutions are obtained. In each generation, the generated individuals will always replace the worst fitness individual in the population. After completing the iteration by genetic algorithm on the population of solutions, an optimal or near optimal solution is obtained. This solution is then used determine the final fragment allocation to the individual network sites.

3.2 The Fragment allocation algorithm

GA is applied on each site with the fragments divided into F elements (parts) where each element represent a fragment in the database

Algorithm 1: Fragment allocation algorithm

```
 1. Input: Data to be fragmented
 2. Output: Fragmented data allocated to sites
 3. begin
 4.   for all members of population
 5.       sum += fitness of some individuals
 6.   end //end for
 7.     for all members of population
 8.        probability = sum of probabilities + (fitness / sum)
 9.       sum of probabilities += probability
10.     end //end for
11.    while sum <  population // cost function is high
12.    do this twice
13.       number = Random between 0 and 1
14.       repeat
15.           for all members of population
16.               if number > probability but less than next probability then select
17.               end if
18.           end //end for
19.             create fragments
20.       until new allocation is complete //optimum solution with lower cost function
21.    end //end loop
22.   end.
```

Fig. 3: Fragment allocation algorithm

Algorithm 1 is a genetic algorithm used to fragment data located in different sites for easy access. First, the algorithm determines the total number of individuals in the population. It then performs crossover to produce offspring based on the genetic mutation of the individuals selected. This process continues until the best individuals with good fitness and best genetic traits are produced.

4. RESULTS AND DISCUSSION

To obtain a solution of representation by GA where F fragments is allocated to S sites, fourteen (14) sites were considered. Each element has an integer value which represents the site where the fragment is proposed to be allocated.

Table 4.2: Experimental Result

No of Sites	Rizik et al. (2014) Time(sec)	Abdalla (2012) Time(sec)	Singn et al. (2014) Time(sec)	Proposed GPA Time(sec)
1	1.126	1.201	1.32	0.83
2	1.354	1.412	1.702	1.162
3	1.503	2.04	2.8	1.05
4	2.087	2.542	3.01	2.25
5	3.501	4.212	5.5	3.18
6	5.194	7.505	6.82	4.32
7	7.964	12.324	11.2	5.67
8	9.385	14.152	14	7.58
9	11.011	17.511	16	8.76
10	11.521	18.31	16.7	9.45
11	11.882	18.61	18.3	10.25
12	11.951	19.31	19.01	8.76
13	12.169	19.72	19.54	9.39
14	12.815	20.501	19.75	8.62

After computing, the computation time of the results obtained were compared with the results of previous related works. From the result we can conclude that GA effectively gives optimal result and this result even improves as the number of sites increases. A graph of these comparisons is plotted as shown in figure 3.

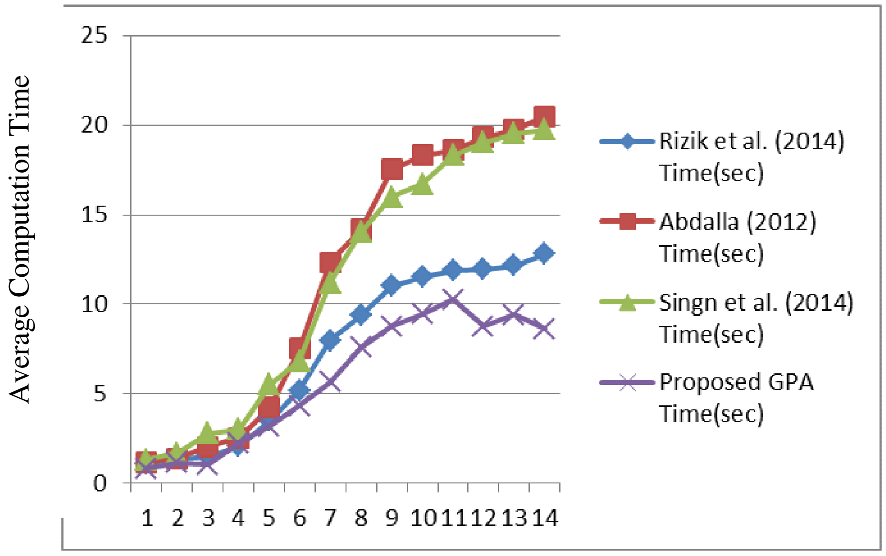

Fig. 3: Graph Showing Result Comparison.

The Figure illustrate the effectiveness of the proposed fragment allocation model in terms of the average computation time over the three other fragment allocation techniques as seen in the graph. Thus figure 3 shows that our algorithm has a better performance in terms of time complexity and that the result has least average computation time needed to allocate fragments to network sites where they are needed. Moreover, it can be inferred that the average computation time for fragment allocation increases as the number of sites increases and vice versa. Hence, our algorithm produces a better and more optimal solution.

5. CONCLUSION

Data allocation is an important factor when considering the design of distributed databases and database systems in general. However, finding optimal solutions as a means of resolving data allocation issue in distributed databases is of major concern and a serious challenge to database developers. This is mainly due to the fact that several sites or databases that are geographically spread are holding fragments of the data to be allocated. If a distributed database is not properly developed, the implication is that a lot of time and resources will be expended on allocating such data and this could increase costs of information distribution and retrieval drastically. Thus in designing a distributed database, a lot of factors must be put into consideration if such a database must be optimal in terms of costs, efficiency, and data contained in them must be easily accessible to users on demand. In this paper, genetic algorithm is used to find the best allocation of fragments to sites in a distributed database system. The choice of the genetic algorithm is based on fact that the technique can handle data over large database or sites that are geographically spread across large distances. Also, genetic algorithm provide a platform for natural selection and evolution since it is a robust and adaptive method for solving search and optimization problems in a large irregular search space. Thus genetic algorithm is used to determine the most efficient allocation procedure. Finally, we show through experimental analysis, that the procedure adopted produced solution that are closer to optimal solution when compared with the results obtained by other researchers.

REFERENCES

1. Tiwari, P. and Chande, S. V. (2013). Optimization of Distributed Database Queries Using Hybrids of Ant Colony optimization Algorithm. Int'l Journal of Advanced Research in computer Science and Software Engineering, Vol. 3, Issue 6, pp. 609-614.
2. Reid, D. J. (1997). Minimizing the Response Time of Executing a Join between Fragmented Relations in a Distributed Database System. Mathematical Computer Modelling, Vol. 25, No. 1, pp. 59-75.
3. Tosun, U. (2014). Distributed Database Design Using Evolutionary Algorithms. Journal of Communications and Networks, Vol. 16, Issue 4, pp. 430-435. doi: 10.1109/JCN.2014.000073.
4. Ahmad, I., Karlapalem, K., Kwok, Y.-K., and So, S.-K. (2002). Evolutionary Algorithms for Allocating Data in Distributed Database Systems. Distributed and Parallel Databases, Vol. 11, Issue 1, pp. 5-32. doi: 10.1023/A:1013324605452.
5. Zynali, M. and Khanli, L. M. (2010). Fuzzy Based Approach for Load Balanced Distributed Database on Senor Network. Int'l Journal of Future Generation Communication and Networking, Vol. 3, No. 2, pp. 41-52.
6. Abdalla, H. I. (2011). Improving Data Management in a Distributed Environment. Journal of Digital Information Management, Vol. 9, No. 3, pp. 122-125.
7. Jagannatha, S., Geetha, D. E., Kumar, T. V. S., and Kanth, K. R. (2013). Load Balancing in Distributed Database System Using Resource Allocation Approach. Int'l Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 7, pp. 2529-2535.
8. Kaur, P. (2015). Query Optimization. Int'l Journal of Engineering Science and Computing, pp. 1547-1550. doi: 10.4010/2015.389.
9. Gope, D. C. (2012). Dynamic Data Allocation Methods in Distributed Database System, American Academic & Scholarly Research Journal Vol. 4, No. 6, pp. 1-8.
10. Sarhan, A. (2009). A New Allocation Technique for Methods and Attributes in Distributed Object-Oriented Databases Using Genetic Algorithms. The Int'l Arab Journal of Information Technology, Vol. 6, No. 1, pp. 17-26.
11. Raipurkar, A. and Bamnote, G. R. (2013). Query Optimization in Distributed Database System. Int'l Journal of Computer Science and Application, Vol. 6, No. 2, pp. 319-322.
12. Karlapalem, K., Ahmad, I., So, S.-K., and Kwok, Y.-K. (1997). Empirical Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems. In Proceedings of IEEE Computer Society's Int'l Computer Software and Applications Conference, **Vol., No.,** pp. 296-301.
13. Singh, A., Khalon, K. S. and Virk, R. S. (2014). Nonreplicated Static Data Allocation in Distributed Databases Using Biogeography-Based Optimization, Chinese Journal of Engineering, Vol. 2014, pp. 1-9.
14. Sleit, A., AlMobaideen, W., Al-Areqi, S. and Yahya, A. (2007). American Journal of Applied Sciences, Vol. 4, No. 8, pp. 613-618.
15. Singh, A. and Khalon, K. S. (2009). Non-Replicated Dynamic Data Allocation in Distributed Database Systems. Int'l Journal of Computer Science and Network Security, Vol. 9, No. 9, pp. 176-180.
16. Motzkin, D. and Yurk, P. D. (1990). A Data Distribution Model for Distributed Relational Databases with Dissimilar Computers and Network Costs, Mathematics and Computer Modelling, Vol. 14, pp. 172-177.
17. Amalarethinam, D. I. G. and Balakrihnan, C. (2012). A Study on Performance Evaluation of Peer-to-Peer Distributed Databases. IOSR Journal of Engineering, Vol. 2, No. 5, pp. 1168-1176.
18. Salunuke, D. and Potdar, G. P. (2014). A Survey Paper on Database Partitioning. Int'l Journal of Advanced Research in Computer Science & Technology, Vol. 2, Issue 3, pp. 210-212.
19. Amalarethinam, D. I. G. and Balakrihnan, C. (2013). oDASuANCO – Ant colony Optimization Based Data Allocation Strategy in Peer-to-Peer Distributed Databases. Int'l Journal of Enhanced Research in Science, Technology & Engineering, Vol. 2, Issue 3, pp. 1-8.

20. Huang, Y.-F. and Chan, J.-H. (2001). Fragment Allocation in Distributed Database Design. Journal of information Science and Engineering, Vol. 17, pp. 491-506.
21. Rho, S. and March, S. T. (1994). A Nested Genetic Algorithm for Distributed Design. In Proceedings of the 27th Annual Hawaii Int'l Conference in System Sciences, pp. 33-42.
22. Kumar, R. and Gupta, N. (2012). Non-Redundant Dynamic Data Allocation in Distributed Database Systems. Special Issue of Int'l Journal of Computer Applications on Issues and Challenges in Networking, Intelligence, and Computing Technologies –ICNICT 2012, pp. 6-10.
23. Padia, S., Khulge, S., Gupta, A. and Khadilikar, P. (2015). Query Optimization Strategies in Distributed Databases. Int'l Journal of Computer Science and Information Technologies, Vol. 6, No. 5, pp. 4228-4243.
24. Amalarethinam, D. I. G. and Balakrihnan, C. (2015). An Optimized Strategy for Data Allocation in Peer-To-Peer Distributed Databases. Int'l Journal of Fuzzy Mathematical Archive, Vol. 6, No. 2, pp. 187-195.
25. Hababeh, I. O., Bowring, N. and Ramachandran, M. (2003). An Integrated Strategy for Data Fragmentation and Allocation in a Distributed database Design. In Proceedings of Int'l Conference on Information Technology and Natural Sciences,
26 Apers, P. M. G. (1988). Data Allocation in Distributed Database Systems. Journal of ACM Transactions on Database Systems (TODS), Vol. 13, Issue 3, pp. 263-304. doi: 10.1145/44498.45063.
27. Corcoran, A. L. M. and Hale J. (1994). A Genetic Algorithm for Fragment Allocation in Distributed Database System. In Proceedings of the SAC'94 ACM Symposium on Applied computing, pp. 247-250, March 6-8, 1994, Phoenix, Arizona, USA. doi: 10.1145/326619.326738.