Improve Near Neighborhood for Allocation in DDB with Fuzzy Approach

Alireza Rezaee
Department of system and Mechatronics Engineering, Faculty of New Sciences and Technologies, University of Tehran, Tehran, Iran
Email: arrezaee {at} ut.ac.ir

Sara Mosaferinejad Moghaddam
Islamic Azad University
Iran

***Abstract*— Migrating data fragment in distributed database systems is an important point in designing DDBMS. In this way, we are going to improve effectiveness of current NNA using a Fuzzy inference engine. Our results indicate that, fuzzy based NNA leads to have at least 5% in some of systems performance metrics. This algorithm, by providing data clustering, is very suitable for DDBS in the networks, with heavy loads, and frequent requests for data fragments coming from different sites.**

Keywords- Distributed Database Systems, Distributed Data, Fuzzy Inference Engine.

I. INTRODUCTION

Developments in database and networking technologies in the past two decades led to advances in distributed database systems. A DDS 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 [13]. Designing the fragmentation and allocation of the underlying data is the primary concern of a DDS. A major cost in executing queries in a distributed database system is the data transfer cost incurred in transferring relations (fragments) accessed by a query from different sites to the site where the query is initiated. The objective of a data allocation algorithm is to determine the assignment of fragments at different sites so as to minimize the total data transfer cost incurred in executing a set of queries. This is equivalent to minimizing the average query execution time, which is of primary importance in a wide class of distributed conventional or multimedia database systems. File allocation problem is studied extensively in the literature, started by Chu [14] and continued for non-replicated and replicated models [15, 16]. Some studies considered dynamic file allocation [17, 18].

Various approaches have already been described the data allocation technique in distributed systems [3], [6], [7], [8]. Some methods are limited in their theoretical and implementation parts [10], 11]. Other strategies are ignoring the optimization of the transaction response time. The other approaches present exponential time of complexity and test their performance on specific types of network connectivity [4].

Data allocation problem was introduced when Eswaran [19] first proposed the data fragmentation. Studies on vertical fragmentation [20, 21]; horizontal fragmentation [22] and mixed fragmentation [23] were conducted. The allocation of the fragments is also studied extensively.

In these studies, data allocation has been proposed prior to the design of a database depending on some static data access patterns and/or static query patterns. In a static environment where the access probabilities of nodes to the fragments never change, a static allocation of fragments provides the best solution. However, in a dynamic environment where these probabilities change over time, the static allocation solution would degrade the database performance. Initial studies on dynamic data allocation give a framework for data redistribution and demonstrate how to perform the redistribution process in a minimum possible time. In [5] a dynamic data allocation algorithm for non-replicated database systems is proposed named *optimal* algorithm, but no modeling is done to analyze the algorithm. In [7] the *threshold* algorithm is proposed for dynamic data allocation algorithm which reallocates data with respect to changing data access patterns. It focused on load balancing issues.

The threshold algorithm, which proposed in [7], uses a crisp method to migrate data fragment from one site to another. In this algorithm, if the access count of one site to a fragment exceeds the threshold value, fragment will migrate from owner to site, which frequently access to fragment. Optimal algorithm also uses a similar method. In this algorithm comparison between accesses count of owner and requesting site leads to migrate fragment. Near Neighborhood Allocation (NNA) is a variation of optimal algorithm. It is different with optimal algorithm in choosing destination of migrating fragment. In NNA, destination of moved fragment is the neighbor of the source, which is in the path from the source to the site with highest access pattern. This algorithm avoid moving data fragment too frequently.

All of above algorithms using crisp method to moving data along network paths. Estimating time and place (destination) of a data fragment depends on various parameters such as access pattern, bandwidth of network links and etc.

In this paper, we are going to present and analyze a new app

In this paper we are going to present and analyze a new approach for dynamic data allocation algorithm named Fuzzy Neighborhood Allocation. This algorithm is based on NNA [2], but with different strategy for selecting nodes for data movements and different strategy in migrating fragments.

The rest of this paper is organized as following: In section 2 we will describe about our method, section 3 is consists of our simulation environment and its results and section 4 is our conclusion.

II. METHODOLGY

The FNA algorithm is basically a variation of the NNA [3]. In NNA algorithm, all fragments are initially distributed over the nodes according to any static method but afterwards, any node j, runs the optimal algorithm described as follows for every fragment i, that it stores.

(1) For each (locally) stored fragment, initialize the access counter rows to zero. ($S_{ij} = 0$

were i ε fragment indexes and j ε nodes)

(2) Process an access request for the stored fragment

(3) Increase the corresponding access counter of the accessing node for the stored fragment. (If node (x) accesses fragment i, set $S_{ix} = S_{ix} + 1$)

(4) If the accessing node is the current owner, go to step 2. (i.e Local access, otherwise it is a remote access)

(5) If the counter of a remote node is greater than the counter of the current owner node, transfer the ownership of the fragment together with the access counter array to the neighbor of the source, which is in the path from the source to the node with highest access pattern. (i.e fragment migrates) (If node x accesses fragment i and $S_{ix} > S_{ij}$, send fragment i to node (x))

(6) Repeat from step 2.

The problem of optimal algorithm is that if the changing frequency of access pattern for each fragment is high, it will spend a lot of time for transferring fragments to different nodes. So, the response time and delay will be increased.

In NNA, the problem of optimal algorithm have been addressed:

In NNA algorithm, the requirement for moving a fragment is obtained as in optimal algorithm. But, the destination of the moving data is different. In this method, it considers the network topology and routing for specifying destination. In other words the destination of the moved fragment is the neighbor of the source, which is in the path from the source to the node with highest access pattern. It uses link state (Dijkstra) routing algorithm for its simplicity of implementation. Any routing algorithm can be used equally.

By using this approach it avoid moving data too frequently because since the fragment will be placed finally in a node which has the average access cost for nodes that use it. So, the delay of movement will be reduced and the response time will also be improved.

As mentioned above, detecting oscillation is important in DDBMS. Rapid changing of access pattern for a single fragment may cause problems in DDB system. Fragment migration between two sites leads high delay in accessing fragment. Migrating fragment is inaccessible during fragment migration because fragment is locked when it moves from one site to another. We are going to solve this problem by detectin oscillation state and avoiding from moving fragment. In this way, we consider access pattern of site and recognize oscillation state through differentiation of access pattern. After degrading of access pattern with using mean factor, we use a fuzzy and operator between smoothed access pattern and fuzzy compliment of differentiation of access pattern. The result show revised access pattern, which can be used in deciding of fragment migration. We trace access counts in 20 time sluts.

Each time slut comprises of 50-clock cycle. Access pattern of site j for fragment i during time slut of calculated as below:

$$P_{ijt} = \frac{S_{ijt}}{\sum_{K \in Sites} S_{ikt}}$$

Smoothed value of this pattern calculated as below:

$$Smoothed(P_{ijt}) = \frac{P_{ijt} + P_{ij(t+1)}}{2}$$

Comparison of these values for a sample data is showed in figure1. Differentiation of accessed patterns are calculated as:

$$\frac{dP_{ijt}}{dt} \approx D_{ijt} = \frac{P_{ij(t+1)} - P_{ijt}}{\Delta t}$$

Absolute value of differentiation fuzzified. Figure 2 shows these conversion states.

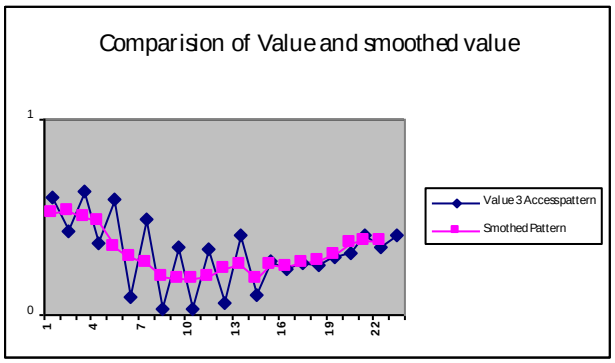

Figure 1 Value of access pattern and its smoothed value

Finally, revised accessed pattern is calculated through fuzzy and operation as below:

$$R_{ijt} = Smoothed(P_{ijt}) \wedge \sim D_{ijt}$$

Figure 3 shows fuzzy compliment of differentiation and Figure 4 shows revised access pattern.

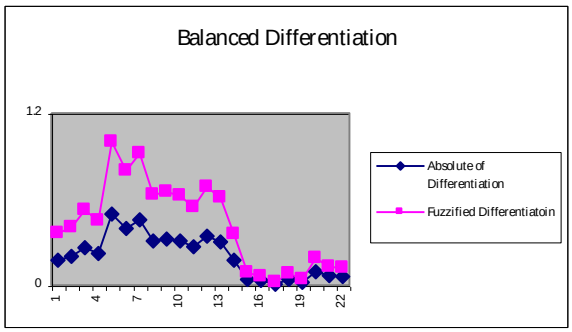

Figure 2 Balanced differentiation

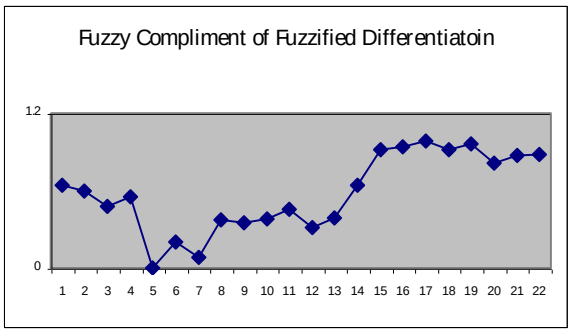

Figure 3 Compliment of fuzzified differentiation

Revised access pattern is defuzzified and used to compare with revised access pattern of owner. Decision for fragment migration is made according to these values.

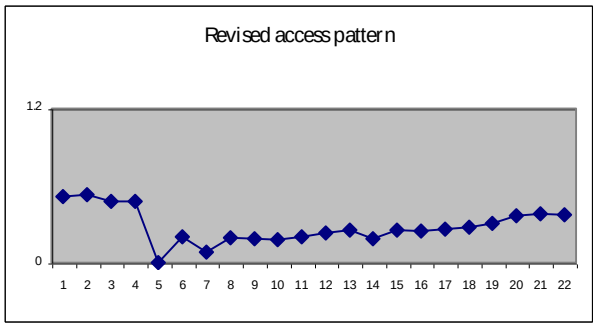

Figure 4 Revised access pattern

To find data fragment destination, we fuzzify distance vector of each site and then use its fuzzy compliment of this values. Suppose that fuzzified distance vector of site C is shown in Figure 5. So we calculate fuzzy compliment of this vector as shown in Figure 6. This vector would be multiplied by fuzzified access request vector.

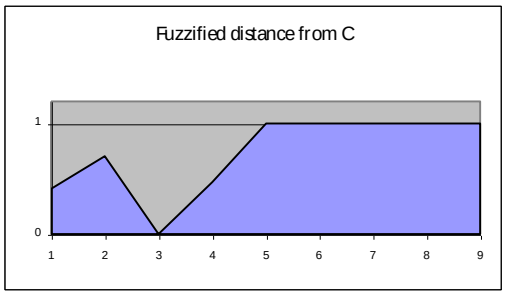

Figure 5 Fuzzified distance from C

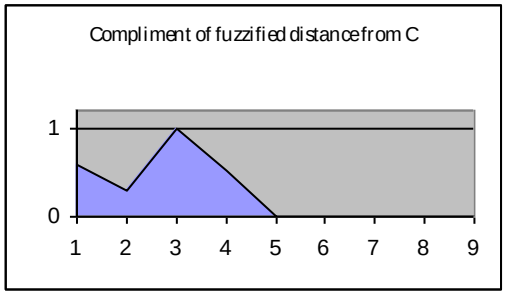

Figure 6 Compliment of fuzzified distance from C

Now we use link state algorithm. We are going to choose a proper link to move data fragment. In this way, we must compare links of owner. Each link is evaluated by the sites, which the link is in the path of reaching them using link state algorithm. We evaluated sites by compliment of fuzzified distance vector multiplied by fuzzified access request vector. Using a fuzzy OR operation, we conclude these parameters for a single link. Suppose N is the set of nodes which are connected to the owner via l in Dijkstra SPF algorithm. Then we will have:

$$W_l = \bigcup_{j \in N} V_j$$

Then after defuzzification phase, we can compare links. Figure 7 shows decision-making system schema.

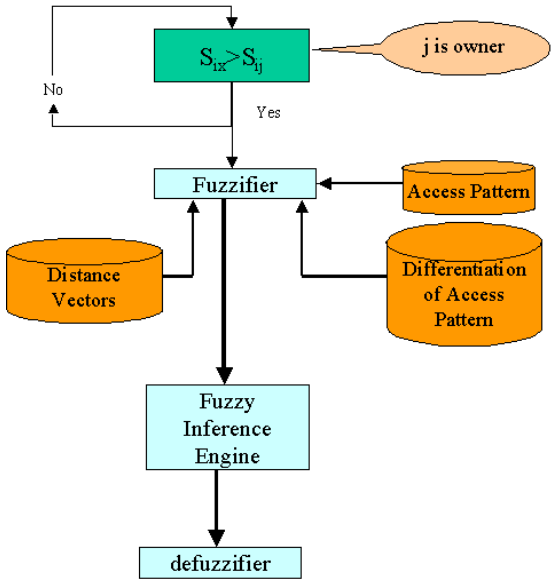

Figure 7 decision-making system schema

III. EXPERIMENTAL ENVIRONMENT AND SIMULATION RESULTS

We developed software to simulate FNA algorithm and compare it with NNA. This simulator is configurable for testing different network topologies and different data requests and/or allocation conditions. In our simulator we mark each packet's send and receive time. Using time stamp, we could compare algorithms in different factors. Detailed information regarding to the implementation of this software is available in [24] and [25].

In this experiment, we used the topology shown in figure 1. Our routing is based on link cost using Dijkstra algorithm (Shortest Path First). In our experiment links cost can change over simulation. So our system has 9 nodes as shown below. We used IKCo guarantee data as our benchmark. In this section we considered dealers as an independent site and each car referring is a requesting transaction.

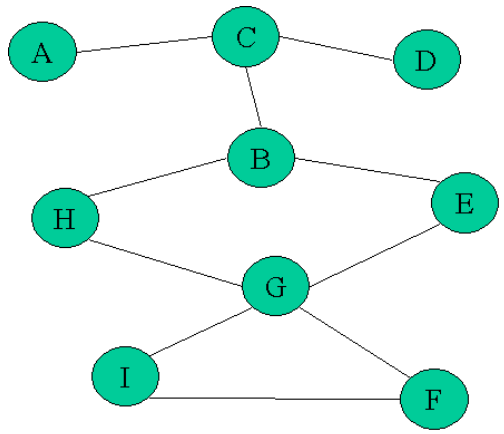

Figure 8 The topology of the experiment

In our experiments, we consider two factors: average delay for receiving the response (response time) for a fragment request and average time spent for moving data from one node to another (fragment data migration time). We will investigate the effect of different fragment size on these factors.

A. Fragment Size

Figures 10 and 11 show the effect of fragment size on these factors.

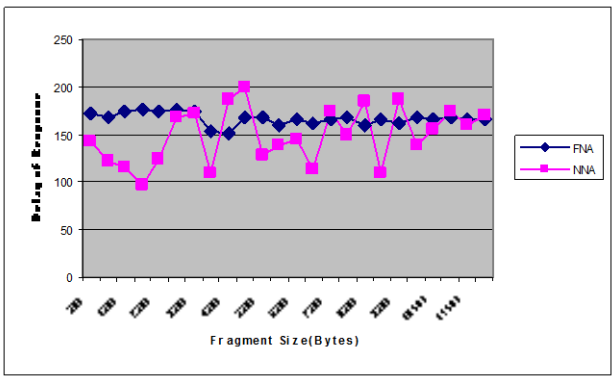

Figure 9 Effect of fragment size on the delay of responding to a fragment request in optimal approach and in NNA

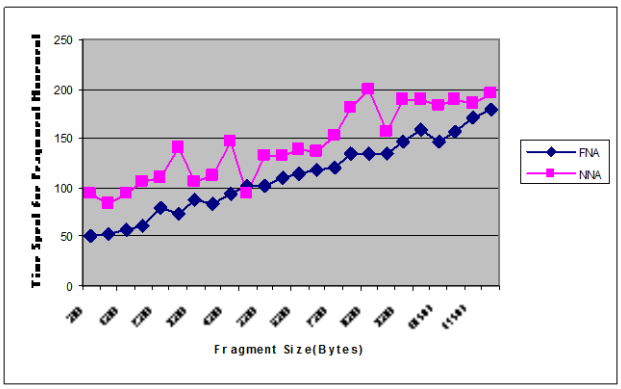

Figure 10 Effect of fragment size on the average time spent for moving fragments in optimal and NNA approach

According to Figure 3, for small fragments the average time spent for moving data in FNA algorithm is larger than NNA and for larger fragments this is reversed. The reason is that for small fragments the cost of moving data to destination node is low and so, the movement cost does not exceed the access cost. In the case of large fragments the movement of fragments takes more time and also increase the network traffic. So, less movement will produce some advantages that overcome the access cost. Avoidance of oscillation condition in FNA leads to have less traffic and saving in network resources such as bandwidths. In FNA destination of a data fragment is chosen according to access pattern of over all system. So, we direct our fragments more effective and this will be valuable in larger fragments.

B. Query Production Rate

Figures 12 and 13 show the effect of query production rate on our factors. In this way, we neglect some of transactions in our benchmark to evaluate our algorithm in different load condition. As the rate of query is increased, the delay of response and also the average time for fragment movement is decreased. Because the high rate of query causes each fragment find its proper owner node sooner and stays on it. So, the delay of response for a fragment will be decreased.

As shown, except the situation where the rate of query production is very low, FNA algorithm performs better than optimal algorithm. Deliberation in choosing destination of data fragment results avoid idle fragment movements. Less traffic is achieved by preventing oscillation condition.

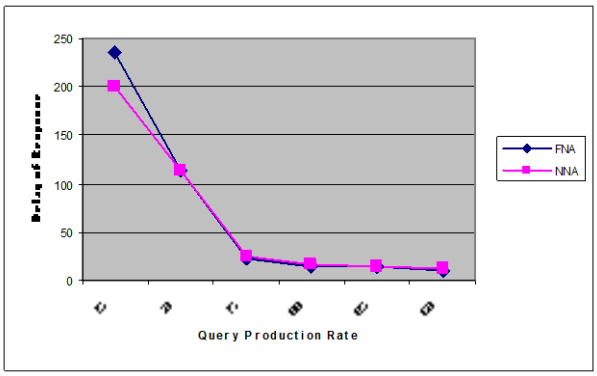

Figure 11 Effect of query production rate on the delay of responding to a fragment request in optimal and NNA approach

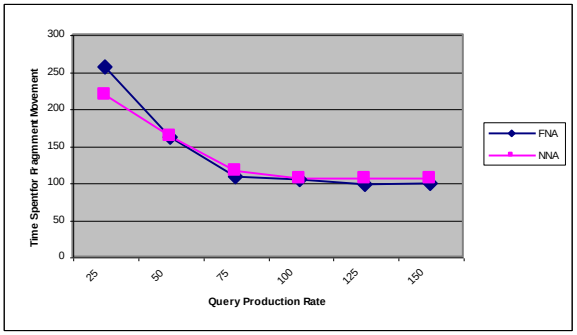

Figure 12 Effect of query production rate on the average time spent for moving fragments in optimal and NNA approach

C. Number of Active Nodes

Figure 14 shows that as the number of active nodes (the nodes generating queries) in the network increases, the difference between FNA and NNA algorithm appears better. In our experiment we change the number of active nodes form 2 to 8. To change the number of active nodes, we neglect some of transactions according to the node, which made the transaction. According to the results we conclude that in larger networks FNA algorithm respond to a request with much lower delay than NNA algorithm. Figure 15 shows that as the number of active nodes in the network increases, the average time spent for moving fragments in FNA algorithm is less than NNA algorithm. But this conclusion needs to be experimented on more complex network topologies.

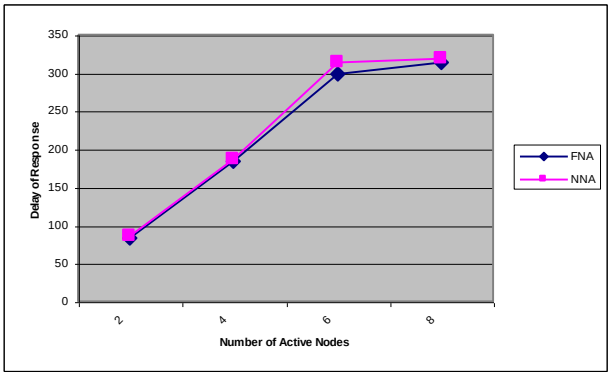

Figure 13 Effect of number of nodes in the network on the delay of responding to a fragment request in optimal and NNA approach]

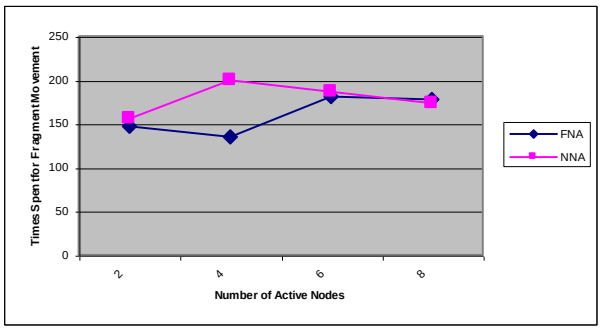

Figure 14 Effect of number of active nodes on the average time spent for moving fragments in optimal and NNA approach.

In our proposed algorithm, oscillations detected and recognized via a simple fuzzy inference engine. Recognizing oscillation condition leads to avoid oscillating data fragments between sites. So, idle data movement is decreased. Deliberation in fragment movement is another aspects of our proposed algorithm.

IV. CONCLUSION

In this study we represented a new dynamic data allocation algorithm in DDB named Fuzzy Neighborhood Allocation (FNA) algorithm. This algorithm is based on NNA algorithm but the difference in its approach is for selecting destination node of moving fragments and recognizing oscillation conditions. In our experiments we considered two factors, average delay of responding to a fragment request, and average time spent for moving fragments in the networks. We examined the effect of different parameters on these factors. Findings of our experiments indicated that, the FNA algorithm performs better for larger fragment size and query production rate, but for small fragment size and query production rate, the NNA and optimal algorithm performs better. For larger networks, by using FNA algorithm we can decrease the delay of response to a fragment regarding to optimal algorithm.

Here we just studied these algorithms on non-replicated distributed database systems. Further studies are needed to test FNA, NNA and optimal algorithms in replicated distributed database systems.

1. REFERENCES

[1] Berkan, R. C., Trubatch, S. L., Fuzzy Systems Design Principles, IEEE Press, New York, 1997

[2] Basseda, R., Tasharofi, S., Rahgozar, M., Near Neighborhood Allocation (NNA): A Novel Dynamic Data Allocation Algorithm in DDB, In proceedings of 11th Computer Society of Iran Computer Conference (CSICC2006), Tehran, 2006

[3] John, L. C., A Generic Algorithm for Fragment Allocation in Distributed Database Systems, *ACM*, 1994.

[4] Ahmad, I., Karlapalem, K., Kwok, Y. K., and So, S. K. Evolutionary Algorithms for Allocating Data in Distributed

Database Systems, *International Journal of Distributed and Parallel Databases*, 11: 5-32, The Netherlands, 2002.

[5] Brunstroml, A., Leutenegger, S. T. and Simhal, R., Experimental Evaluation of Dynamic Data Allocation Strategies in a Distributed Database with changing Workloads, *ACM Transactions on Database Systems*, 1995.

[6] Chin, A. G., Incremental Data Allocation and ReAllocation in Distributed Database Systems, *Journal of Database Management;* Jan-Mar 2001; 12, 1; ABI/INFORM Global pg. 35.

[7] Ulus, T., and Uysal, M., Heuristic Approach to Dynamic Data Allocation in Distributed Database Systems, *Pakistan Journal of Information and Technology 2 (3)*, 2003, ISSN 1682-6027, 231-239.

[8] Voulgaris, S., Steen, M. V., Baggio, A., and Ballintjn, G., Transparent Data Relocation in Highly Available Distributed Systems. *Studia Informatica Universalis*. 2002.

[9] Navathe, S. B., Ceri, S., Wiederhold, G. and Dou, J., Vertical Partitioning Algorithms for Database Design, *ACM Transaction on Database Systems*, 1984, 680-710.

[10] Apers, P. M. G. , "Data allocation in distributed database systems," *ACM Transactions on Database Systems*, vol. 13, no. 3, 1988, 263–304.

[11] Huang, Y. F. and Chen, J. H., Fragment Allocation in Distributed Database Design, *Journal of Information Science and Engineering 17*, 2001, 491-506.

[12] Hababeh, I. O., A Method for Fragment Allocation Design in the Distributed Database Systems, *The Sixth Annual U.A.E. University Research Conference*, 2005.

[13] Özsu, T., and Valduriez, P., Principles of Distributed Database Systems. Prentice-Hall Book Co., Englewood Cliff, USA, 1991.

[14] Chu, W. W., Optimal File Allocation in a Multiple Computer System, IEEE Transactions on Computers, C-18, 1969, 885-889.

[15] Morgan, H.L., and Levin, K. D., Optimal Program and Data Locations in Computer Networks, Communications of ACM, 20, 1977, 315-321.

[16] Azoulay-Schwartz, R., and Kraus, S., Negotiation on Data Allocation in Multi-Agent Environments, Autonomous Agents and Multi-Agent Systems, 5, 2002, 123-172.

[17] Wah, B. W., Data Management in Distributed Systems and Distributed Data Bases, Ph.D. Dissertation, University of California, Berkeley, CA, USA, 1979.

[18] Smith, A. J., Long-term File Migration: Development and Evaluation of Algorithms, Communications of ACM, 24, 1981, 512-532.

[19] Eswaran, K. P., Placement of Records in a File and File Allocation in a Computer Network, in Proceedings of IFIP Congress on Information Processing, Stockholm, Sweden, 1974, 304-307.

[20] Navathe, S. B., Ceri, S., Wiederhold, G., and Dou, J., Vertical Partitioning Algorithms for Database Design, ACM Transaction on Database Systems, 9, 1984, 680-710.

[21] Ceri, S., Pernici, B., and Wiederhold, G., Optimization Problems and Solution Methods in the Design of Data Distribution, Information Systems, 14, 1989, 261-272.

[22] Ceri, S., Navathe, S. B., and Wiederhold, G., Distribution Design of Logical Database Schemas, IEEE Transactions on Software Engineering, 9 , 1983, 487-503.

[23] Zhang, Y., and Orlowska, M. E., On Fragmentation Approaches for Distributed Database Design, Information Sci., 1, 1994, 117-132.

[24] Basseda, R. and Tasharofi, S., Design and Implementation of an Environment for Simulation and Evaluation of Data Allocation Models in Distributed Database Systems, Technical Report No. DBRG.RB-ST.A50701, 2005

[25] Basseda, R. and Tasharofi, S., Data Allocation in Distributed Database Systems, Technical Report No. DBRG.RB-ST.A50715, 2005