2010 Second International Conference on Network Applications, Protocols and Services

A Novel Evolutionary Algorithm for Solving Static Data Allocation Problem in Distributed Database Systems

Ali Safari Mamaghani
Young Researcher Club, Islamic Azad University
Bonab Branch
Bonab, Iran
Safari_m_61@yahoo.com

Mostafa Mahi
Computer Engineering Department
Islamic Azad University, Bonab Branch
Bonab, Iran
mostafamahi@gmail.com

Mohammad Reza Meybodi
Computer Engineering Department
Amir Kabir University of Technology
Tehran, Iran
mmeybodi@aut.ac.ir

Mohammad Hosseinzadeh Moghaddam
Islamic Azad University, Hashtrood Branch
Hashtrood, Iran
mh.moghaddam@yahoo.com

***Abstract*—Given a distributed database system and a set of queries from each site, the objective of a data allocation algorithm is to locate the data fragments at different sites so as to minimize the total data transfer cost incurred in executing the queries. The data allocation problem, however, is NP-complete, and thus requires fast heuristics and random approaches to generate efficient solutions. In this paper an approximate algorithm has been proposed. This algorithm is a hybrid evolutionary algorithm obtained from combining object migration learning automata and genetic algorithm. Experimental results show that proposed algorithm has significant superiority over the several well-known methods.**

Keywords-object migration learning automata;genetic algorithms;Distributed database system; Data fragment allocation; Evolutionary algorithm.

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.

The primary concern of a DDS is to design the fragmentation and allocation of the underlying data. Fragmentation unit can be a file where allocation issue becomes the file allocation problem. 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 [1]. The objective of a data allocation algorithm is to determine an 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 as well as multimedia database systems.

The data allocation problem, is NP-complete [2], and thus requires fast heuristics to generate efficient solutions. Furthermore, the optimal allocation of database objects highly depends on the query execution strategy employed by a distributed database system, and the given query execution strategy usually assumes an allocation of the fragments.

File allocation problem is studied extensively in the literature started by Chu [2] and continued for non-replicated and replicated models [3, 4].

Data allocation problem was introduced when Eswaran first proposed the data fragmentation [5]. Studies on vertical fragmentation and horizontal fragmentation and mixed were conducted. The allocation of the fragments is also studied extensively. Data allocation algorithms were studied in static and dynamic environments. In a static environment where the access probabilities of sites 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. Different dynamic data allocation algorithms in distributed database systems were explained in [6, 7, and 8].

Many reports on static environments have been published such as random neighborhood search algorithm [1]. The main idea in a neighborhood search algorithm is to generate an initial solution with moderate quality. Then, according to some pre-defined neighborhood, the algorithm probabilistically selects and tests whether a nearby solution in the search space is better or not. If the new solution is better, the algorithm adopts it and starts searching in the new neighborhood; otherwise, the algorithm selects another

978-0-7695-4177-8/10 $26.00 © 2010 IEEE
DOI 10.1109/NETAPPS.2010.10

solution point. Using evolutionary algorithms is an alternative method for solving such problems. Corcoran et al. were first pioneers who allocated data fragments in distributed data base systems by using genetic algorithms [9]. In this application, every gene in chromosome resembles a data fragment, so that the length of a chromosome represents the number of fragments. A similar algorithm suggested by Corcoran was used for file allocation in distributed systems as well [10].

Another genetic-based algorithm was used by Ishfaq Ahmad et al. [1]. In contrast to the approach used by Corcoran, a binary encoding approach was implemented in this work. Another posed evolutionary algorithm which tries to solve the above-mentioned problem is the simulated evolution algorithm. They differ mainly in style for which the first method is based on a crossover operator as a stochastic mechanism which in turn is proper for data exchange among solutions in order to find the most appropriate solution while the latter one applies a mutation operator as an initial search mechanism [11]. The mean field annealing technique combines the collective computation property of the famous Hopfield neural network with simulated annealing [12].

In this paper an approximate algorithm has been proposed. This algorithm is a hybrid evolutionary algorithm obtained from combining object migration learning automata and genetic algorithm. Experimental results imply the suggested algorithm has significant superiority over the several well-known methods. The rest of this paper is organized as follows: Section 2 elaborates the data fragment allocation problem in distributed database systems. Section 3 is an introduction to learning automata and genetic algorithms. In section 4, we describe new hybrid algorithm for solving the problem. Section 5 and 6 is dedicated to describe experimental results and paper conclusion respectively.

II. THE DATA FRAGMENT ALLOCATION PROBLEM

We now present a formal description of the problem. A distributed database is composed of a collection $S = \{c_1, c_2, c_3, ..., c_m\}$ of m sites, where each site i is characterized by its capacity c_i and a set $F = \{s_1, s_2, ..., s_n\}$ of n fragments, where each fragment j is characterized by its size s_j. Each fragment is required by at least one of the sites. The site requirements for each fragment are indicated by the requirements Matrix.

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ r_{m,1} & r_{m,2} & \cdots & r_{m,n} \end{bmatrix}$$

Where $r_{i,j}$ indicates the requirement by site i for fragment j. In general, this requirement is represented by a real value, that is, a weight. A variation of this is to use a Boolean value to indicate that fragment j is either required or not required by site i. Transmission cost is given by the transmission cost

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,m} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ r_{m,1} & r_{m,2} & \cdots & r_{m,m} \end{bmatrix}$$

Where $t_{i,j}$ indicates the cost for site i to access a fragment located on site j.

Given the above definitions, the distributed database allocation problem is one of finding the optimal placement of the fragments at the sites. That is, we wish to find the placement $P = \{p_1, p_2, ..., p_j, ..., p_n\}$ (where p_j =i indicates fragment j is located at site i) for the n fragments so that the capacity of any site is not exceeded. $\sum_{j=1}^{n} r_{i,j} \times s_j \le c_i \quad \forall i | 1 \le i \le m$ And the total transmission cost, $\sum_{i=1}^{m}\sum_{j=1}^{n} r_{i,j} \times t_{i,p_j}$ is minimized [9].

By restricting the use of the requirements matrix and having zero transmission cost, the distributed database allocation problem can be transformed to the bin packing problem, which is known to be NP-complete.

III. LEARNING AUTOMATA AND GENETIC ALGORITHMS

Learning automata are adaptive decision-making devices operating on unknown random environments. The learning automaton has a finite set of actions and each action has a certain probability (unknown for the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal action [13]. Figure 1 illustrates how a stochastic automaton works in feedback connection with a random environment.

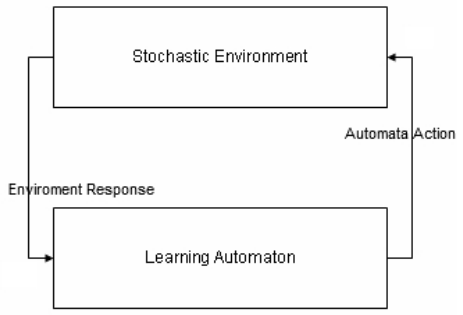

Figure 1. The interaction between learning automata and environment

The automaton chooses one of the offered actions according to a probability vector which at any time instant contains the probability of choosing each action. The chosen

action triggers the environment, which responds with an answer (reward or penalty), according to the reward probability of the chosen action. The automaton takes into account this answer and modifies the probability vector by means of a learning algorithm. A learning automaton is one that learns the action that has the maximum probability to be rewarded and that ultimately chooses this action more frequently than other actions.

Genetic algorithm (GA) based search methods are inspired by the mechanisms of natural genetic leading to the survival of the fittest individuals. Genetic algorithms manipulate a population of potential solutions to an optimization problem [16]. Specifically, they operate on encoded representations of the solutions, equivalent to the genetic material of individuals in nature, and not directly on the solutions themselves. In the simplest form, solutions in the population are encoded as binary strings. As in nature, the selection mechanism provides the necessary driving force for better solutions to survive. Each solution is associated with a fitness value that reflects how good it is, compared with other solutions in the population. The higher the fitness value of an individual, the higher the chance of survival in the subsequent generation. Recombination of genetic material in genetic algorithms is simulated through a crossover mechanism that exchanges portions between strings. Another operation, called mutation, causes sporadic and random alternation of the bits of strings. Mutation also has a direct analogy with nature and plays the role of regenerating lost genetic material.

IV. THE PROPOSED HYBRID ALGORITHM

Speed for reaching to solution in search process get high if genetic algorithms, learning automata, integration of concepts of gene, action and depth are combined .This approach is prevented from being trapped into the local minima. The suggested algorithm in this section is an attempt in this direction. Self-remedy, reproduction and penalty and reward (guidance) are some of the characteristics of hybrid algorithm. For more information, refer simply to the reference [14, 15].

Chromosome and gene: in spite of classic genetic algorithms, in the suggested algorithm, we don't use binary encoding for chromosomes. Chromosomes are shown by an object migration learning automata so that any gene of chromosomes is associated to one of the automata actions and is located in a specific depth of that action. We show object migration automata as $< V, \alpha, \varphi, \beta, F, G >$ .In this Automaton $\alpha = \{\alpha_1, \alpha_2, \alpha_3, ...\alpha_n\}$ is set of allowed actions of automaton. This automaton has n actions (the number of actions equals with the number of data fragments).

$V = \{V_1, V_2, ..., V_n\}$ is set of objects. These objects are sites number that allocated to data fragments. The objects have values 1, 2… m. The objects move on different states of automata and create new allocations.

$\phi = \{\phi_1, \phi_2, ..., \phi_{nN}\}$ is set of states and N is memory depth for automata . The set of automata states are divided to k subsets $\{\phi_1, \phi_2, ..., \phi_N\}$, $\{\phi_{N+1}, \phi_{N+2}, ..., \phi_{2N}\}$ … $\{\phi_{(k-1)N+1}, \phi_{(k-1)N+2}, ..., \phi_{kN}\}$, so objects are classified in terms of their states. If object u is situated in the set of states $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, ..., \phi_{jN}\}$, in this case, data fragment j will be on u^{th} site. In the set of states of action j, states $\phi_{(j-1)N+1}$ and ϕ_{jN} are referred as internal state and boundary states respectively. The objects lying in $\phi_{(j-1)N+1}$ and ϕ_{jN} are referred as more and less certainty objects respectively.

B = {0, 1} is the set of inputs of automata. In this set, 1 and 0 stand for failure and success respectively.

F: $\phi \times \beta \rightarrow \phi$ stands for state mapping function. This function produces the next state in terms of current state and the input of automata. This function determines the movement of objects in states of automata. The function F is different for diverse automata.

G: $\phi \rightarrow \alpha$ is output mapping function. This function decides what action to do in exchange for any automata state. If object u is in the set of states $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, ..., \phi_{jN}\}$, action j is selected (therefore, data fragment j will be on u^{th} site).

For example, if we have 6 data fragments and 4 sites in distributed database, we represent allocation p={2,3,4,1,2,1} as object migration learning automata is shown in figure 2 (where p_j =i indicates fragment j is located at site i).

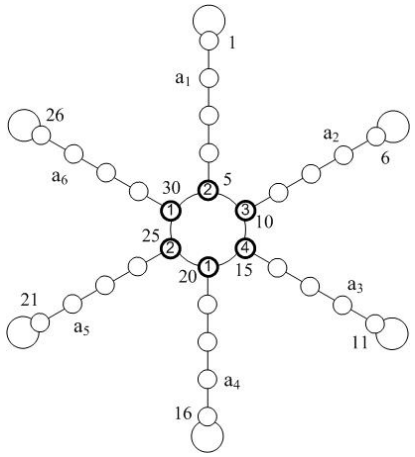

Figure 2. Allocation p={2,3,4,1,2,1} as object migration learning automata based on Tsetlin automata connections.

Operations: Since every chromosome is represented as a learning automaton in hybrid algorithm, crossover and mutation operators are not similar to classic genetic operators. Also this hybrid algorithm has penalty and reward operation.

Penalty and reward operator: In the chromosome, an object is chosen randomly, then, it takes penalty or reward. If data fragment location cost (corresponding action with object) goes beyond threshold (the threshold amount is calculated as average total transition cost of data fragments),

this object will get reward and moves toward the more internal states of this action. Otherwise, the object gets penalty. Having taken reward or penalty, the state of object in relevant set states change. If an object is located in boundary state of an action, its getting penalty leads to the change of action and creating a new solution. The condition of penalizing or rewarding an object is as follows:

The amount of cost for data fragment transition to the given site is calculated from the following relationship:

$$\cos t(datafragment) = \sum_{i=1}^{m} r_{i,datafragment} \times t_{i,p_{dataffragment}}$$

Then, the threshold amount is calculated as average total transition cost of data fragments. If the amount of data fragments transition cost is less then or equal to the amount of threshold, this object is given reward, otherwise, and it can be penalized. Figure 3 shows the pseudo code of reward operator.

```
Procedure Reward( LA, u )
    If (LA.State(u)-1) mod N <> 0 then
            Dec (LA.State(u));
    End If
End Reward
```

Figure 3. Pseudo code of reward operator.

And Figure 4 shows the pseudo code of penalty operator.

```
Procedure Penalize( LA, u )
    repeat
        For U = 1 to n do
            If (LA.State(U)) mod N <> 0 then
                Inc(LA.State(U));
            End If
        End for
    Until at least one object appears in the boundary state
    bestcost = ∞ ;
    for U = 1 to n do
        Create Allocation LA′ from LA by swapping u and U
        If cost_i( LA′) < bestcost then
            bestcost = cost_i( LA′);
            bestfragment = U;
        End If
    End for
    LA.State(bestfragment) = LA.Action(bestfragment)*N;
    LA.State(u) = LA.Action(u)*N;
    Swap(LA.State(u),LA.State(bestfragment));
End Penalize
```

Figure 4. Pseudo code of penalty operator

Selection operator: Roulette wheel is used for selecting learning automaton (chromosome) for mutation or crossover.

Crossover operator: In order to do this operation, we can use k-point crossover.

Mutation operator: When mutation occurred, a randomly selected gene was replaced with a randomly selected choice from the range of valid site numbers.

Now regarding pervious descriptions, we can show the hybrid algorithm applied for solving the problem. Pseudo code of this algorithm is shown in figure 5.

```
Procedure DDA_Hybrid (problem);
    Begin
        n=number of fragments in the system;
        m= number of sites in the system;
        sp = Size of Population;
        Create the initial population LA_1 … LA_sp;
        EvalFitness();
        Iteration=1;
    Repeat
    NewLA1 = NewLA2 = LA with min Value of fitness;
    for i = 2 to sp do
        Select LA1; Select LA2 ;
        if (Random ≤ CrossoverRate) then
            Crossover ( LA1, LA2 );
        if (Random ≤MutationRate) then
            Mutation ( LA1 ); Mutation ( LA2 );
        NewLAi+1 = LA1;
        NewLAi+2 = LA2 ;
        i=i+2;
    end for
    for i = 1 to sp do
    LAi = NewLAi;
    u = Random *n;  //1 ≤ u ≤ n
        If cost(u) ≤ threshold then Reward(LAi , u );
//threshold= Σ_{i=1}^{m} Σ_{j=1}^{n} r_{i,j} × t_{i,p_j} /n
//cost(data fragment)= Σ_{i=1}^{m} r_{i,data_fragment} × t_{i,p_{dataf_fragment}}
        Else penalize(LAi , u);
    end for
    EvalFitness();
    Inc(iteration);
    Until(iteration=maxiteration)
End DDA_Hybrid;
```

Figure 5. The hybrid algorithm for solving data fragment allocation Problem**.**

V. EXPERIMENTAL RESULTS EVALUATION

In this section the results of the implementation of the new algorithm and the comparison of that with some existing algorithms is shown. The algorithms applied to be compared with this new algorithm are as follows: the neighborhood random search algorithm (RS), Corcoran Genetic Algorithm , the Ishfaq Genetic Algorithm and object migration learning automata (OMA).

The comparison was made from solution quality view that generated by different algorithm. Before the new algorithm compared to existing algorithms, Test on the new hybrid algorithm with various learning automata connections such as Krinsky, Krylov and Tsetline was made. So, different cases in number of sites and data fragments were rendered. As an example, a test on a sample with 20 data fragments and sites numbering from 10 to 100 was accomplished. Binary values, i.e. 0 or 1, were assigned to the entries of requirements matrix according to the case. In addition, a value from 0 to 100 is assigned to the transfer cost and size of any given fragment as well. The results are shown in figure 6.

The results indicate that solutions are generated by new hybrid algorithm based on Tsetline connections have less Transmission cost than other connections. So we used the hybrid algorithm based on Tsetline connections as representative for hybrid algorithm in experiments.

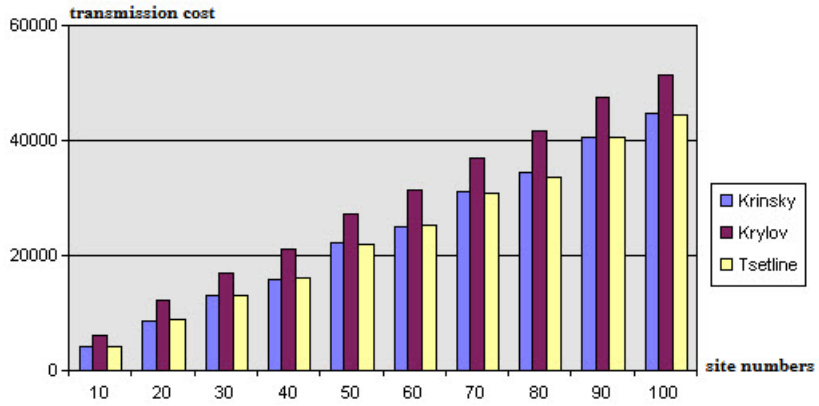

Figure 6. The impact of increasing site numbers over data fragment transfer cost among different learning automata connections.

In the next stage, we compare new algorithm with other existing algorithms such as the neighborhood random search algorithm (RS), Corcoran Genetic Algorithm and the Ishfaq Genetic Algorithm and object migration learning automata (OMA). The results are shown in figure 7.

The point is implied by this figure is that first by increasing the number of existing sites in the distributed system, the data fragment transfer cost during the query execution increases. Second the quality of yielded solutions by the hybrid algorithm (LAGA) in an analogy is higher than those of other algorithms because of the fact that transition cost that obtained by allocating data in this algorithm is lower than other algorithms.

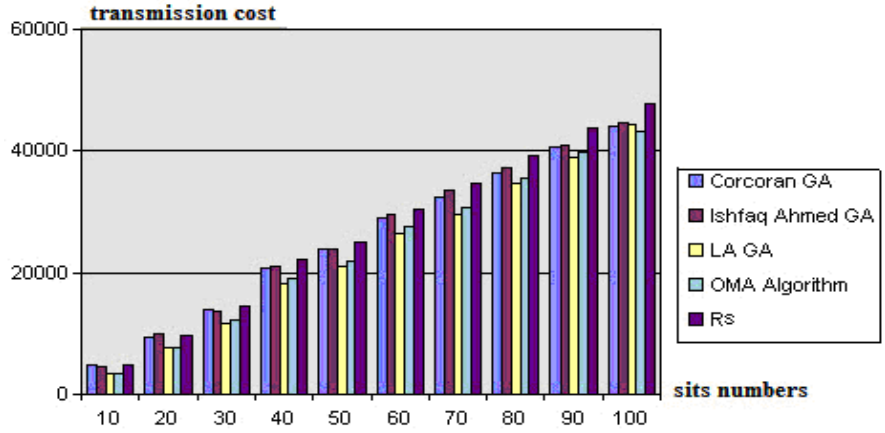

Figure 7. The impact of increasing site numbers over data fragment transfer cost among different algorithms.

In order to render the third test with a constant given number of sites (say 15 sites) we increase the number of fragments from 10 to 100. The outcome results are showed in figure 8. In this figure the comparison of algorithms is shown based on the views of data transfer cost. Figure 8 gives out two points for consideration, first by increasing the number of current data fragments the cost of data fragment transition during the query execution is increased as well. Second notification focuses on quality of generated solution which shows that new algorithm is an effective approach.

VI. CONCLUSIONS

In this paper, a hybrid evolutionary algorithm has been proposed for static data fragment allocation in distributed database systems. This algorithm uses two methods of genetic algorithm and learning automata synchronically for searching the states space of problem. It has been showed in this paper that by synchronic use of learning automata and genetic algorithms in searching process, the quality of generated solutions has been accelerated. The results of experiments show that hybrid algorithm has dominance over the methods of genetic and learning automata-based and neighborhood random search algorithms.

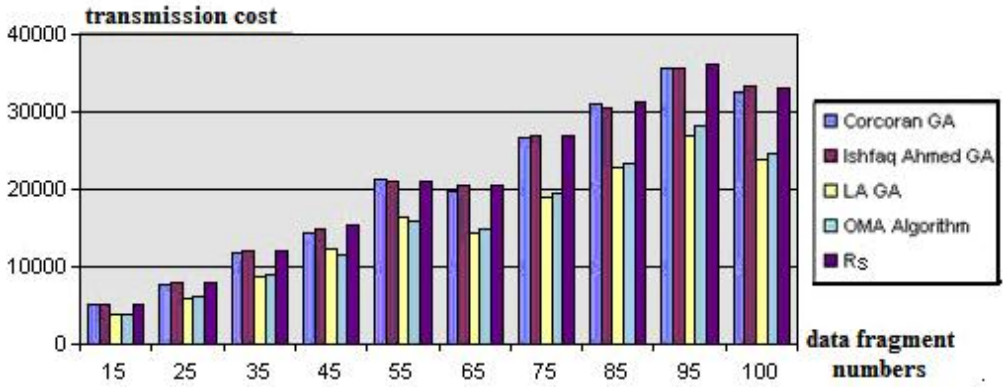

Figure 8. The impact of increasing the number of data fragments over the transfer cost among different algorithms.

REFERENCES

[1] I. Ahmed, K. KARLAPALEM, and Y. K. Kowok, "Evolutionary Algorithms for Allocating Data in Distributed Database Systems," International Journal of Distributed and Parallel Databases, vol. 11, no. 1, pp. 5-32, 2002.

[2] W.W. Chu, "Optimal file allocation in a multiple computer system," IEEE Transactions on Computers, vol. C-18, no. 10, pp. 885–889, 1969.

[3] H. L. Morgan and K. D. Levin, "Optimal program and data locations in computer networks," Communications of the ACM, vol. 20, no. 5, pp.315–322, 1977.

[4] R. A. Schwartz and S. Kraus, "Negotiation on data allocation in multi-agent environments," Autonomous Agents and Multi-Agent Systems, vol. 5, no. 2, pp. 123–172, 2002.

[5] K. P. Eswaran, "Placement of records in a file and file allocation in a computer network," Information Processing,, pp. 304–307, 1974.

[6] A. G. Chin, "Incremental Data Allocation and ReAllocation in Distributed Database Systems," Journal of Database Management, vol.12, no. 1, pp. 35–45, 2001.

[7] T. Ulus and M. Uysal, "Heuristic approach to dynamic data allocation in distributed database systems," Pakistan Journal of Information and Technology, vol. 2, pp. 1682–6027, 2003.

[8] Y. F. Huang and J. H. Chen, "Fragment allocation in distributed database design," Journal of Information Science and Engineering, vol. 17, no. 3, pp. 491–506, 2001.

[9] A. Corcoran and J. Hale, " A Genetic Algorithm For Fragment Allocation In a Distributed Database System," ACM, pp.247-250, 1994.

[10] A. Corcoran and D. L. Schoenefeld, "A Genetic Algorithm For File and Task Allocation In a Distributed System," IEEE, pp.247-250, 1994.

[11] Y.K. Kwok, K. Karlapalem, I. Ahmad, and N. M. Pun, "Design and evaluation of data allocation algorithms for distributed multimedia database systems," IEEE Journal on Selected Areas in Communications, vol 14, no. 7, pp 1332–1348, 1996.

[12] T. Bultan and C. Aykanat, "A new mapping heuristic based on mean field annealing," Journal of Parallel and Distributed Computing, vol. 16, pp. 292–305, 1992.

[13] K. Narendra and M. A. L. Thathachar, Learning Automata: An Introduction. Prentice Hall, 1989.

[14] B. J. Oommen, and D. C .Y. Ma, "Deterministic Learning Automata Solutions to the equipartitioning problem," IEEE Trans. on Computers, Vol. 37 , pp. 2-13, 1998.

[15] A. Safari Mamaghani and M.R. Meybodi, "Clustering of Software Systems using New Hybrid Algorithms," Proc. Int. Conf. on Computer and Information Technology (CIT 2009), Xiamen, China, pp. 20-25, 2009.

[16] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.