Journal of Advances in Computer Research
Quarterly ISSN: 2008-6148
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 4, No. 3, August 2013), Pages: 1-24
www.jacr.iausari.ac.ir

Designing a New Structure Based on Learning Automaton to Improve Evolutionary Algorithms (With Considering Some Case Study Problems)

Ali Safari Mamaghani[1] *, Kayvan Asghari[2], Mohammad Reza Meybodi[3]

(1)Computer Engineering Department, Islamic Azad University, Bonab Branch, Bonab ,Iran
(2) Islamic Azad University, Khameneh Branch, Khameneh ,Iran
(3)Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran

ali.Safari.m@gmail.com; k.asghari@yahoo.com; mmaybodi@aut.ac.ir

Received: 2013/05/08; Accepted: 2013/07/15

Abstract

Evolutionary algorithms are some of the most crucial random approaches to solve the problems, but sometimes generate low quality solutions. On the other hand, Learning automata are adaptive decision-making devices, operating on unknown random environments, So it seems that if evolutionary and learning automaton based algorithms are operated simultaneously, the quality of results will increase sharply and the algorithm is likely to converge on best results very quickly. This paper contributes an algorithm based on learning automaton to improve the evolutionary algorithm for solving a group of NP problems. It uses concepts of machine learning in search process, and increases the efficiency of evolutionary algorithm (especially genetic algorithm). In fact, the algorithm is prevented from being stuck in local optimal solutions by using learning automaton. Another positive point of the hybrid algorithm is its noticeable stability since standard division of results, which is obtained by different executions of algorithm, is low; that is, the results are practically the same. Therefore, as the proposed algorithm is used for a set of well-known NP problems and the results are very suitable it can be considered as a precise and reliable technique to solve the problems.

Keywords: *Learning Automaton, Genetic Algorithm, Hybrid Algorithm, NP Problems*

1. Introduction

The genetic algorithm, which is a search heuristic, is a subset of evolutionary algorithms. This method is mostly used to solve the optimization problems; it uses techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. Besides, the algorithm is a directed random technique that moves gradually toward optimal point; moreover, it solves the problem without additional information about that. Generally, evolutionary approaches have the drawback of getting stuck in local optimal solutions; thus, learning automaton is used to collaborate with them to tackle the problem. This hybrid approach is called GALA.

To evaluate the efficiency of the algorithm, a number of NP problems as case studies are chosen. The evaluation results clearly indicate the capability of the algorithm to solve the studied problems; indeed, they point out that the hybrid algorithm is a stable and precise approach since the generated solutions have less standard division. It means that the approach not only finds suitable solutions but also produces solutions which are approximately the same.

The efficiency of the hybrid algorithm is also theoretically justifiable, since the important function of GA is to evolve the candidate chromosomes; in other words, solutions gradually develop over the iterative process of GA. Meanwhile, if each chromosome has self-improvement ability (learning), the solutions obtained by GA will have noticeable developments.

The structure of the paper remainder is as follows: Section 2 elaborates definition of the studied problems. Section 3 is an introduction to learning automaton and GA. In section 4, the proposed GALA algorithm is described. Section 5 and 6 are dedicated to show experimental results and paper conclusion respectively.

2. Definition of Studied NP Problems

Definition of NP problems used as case studies is elaborated in this section.

2.1 The Join Ordering Problem in database systems

Query optimization is an action in which an efficient query execution plan (qep) is provided, and this is one of the basic stages in query processing. At this stage, database management system selects the best plan among execution plans in a way that query execution bears the low cost, especially the cost of input/output operations. Input of optimizer is the internal form of query that has been entered into database management system by user.

The general purpose of query optimizing is the selection of the most efficient execution plan for achieving the suitable data and responding to data queries. In the other words, in case, we show the all of allocated execution plans for responding to the query with S set, each member qep that belongs to S set has cost(qep) that this cost includes the time of processing and input/output. The purpose of each optimization algorithm is finding a member like qep_0, which belongs to S set, that [1]:

$$Cost(qep_0) = \min_{qep \in s} cost(qep) \qquad (1)$$

The execution plan for responding to query is the sequence of algebra relational operators applied on the database relations, and produces the necessary response to the query. Among the present relational operators, processing and optimizing the join operators, which are displayed by symbol ∞, are difficult operations [31].

Although all of the present execution plans for responding to a definite query have similar outputs, generated inter-relation's cardinalities aren't similar; thus, generated execution plans have different costs, so selecting of suitable ordering for join execution affects on total cost. The problem of query optimizing, which is called the problem of selecting suitable ordering for join operators' execution, is a NP-hard problem [2 and 3].

2.2 The total weighted tardiness scheduling problem

In the single machine, total weighted tardiness scheduling problem is an optimization problem. We consider N job that must be processed uninterruptedly over a machine which is continuously available, and this machine can process only one job in each moment. Each job like j is available for processing in the time zero and has a positive processing time p_i, a positive weight w_i and a due date time d_j. For an assumed sequence of jobs, tardiness of job named j is defined as $T_i = \max\ \{0, C_i - d_i\}$, which the completing time of j is

$$C_j = \sum_{i=1}^{j} p_i \qquad (2)$$

The purpose of the total weighted tardiness problem is to find an order of scheduling all jobs that minimizes the total weighted tardiness of all jobs with this definition:

$$\sum_{j=1}^{n} W_j T_j \qquad (3)$$

Consider the various permutations of the N jobs to find the optimal schedule. Even for a problem with the rather small size, complete enumeration of permutations is computationally impossible since it requires the evaluation of n! Sequences. For the arbitrary positive job weights, this problem is strongly NP-hard [4, 5 and 6].

2.3 The Hamiltonian Cycle Problem

In the graph theory, the Hamiltonian Cycle is a cycle that once touches every nodes of graph exactly, and finally returns to the start node. Also, a graph which has a Hamiltonian Cycle is called Hamiltonian Graph. The problem of finding Hamiltonian path and cycle in graph is NP-Complete [7]. The input of the problem is edge list, or in some cases adjacency matrix of graph. The aim of the problem is to find a permutation of graph nodes in which the number of visited edges is to be maximized.

2.4 The Traveling Salesman Problem

The TSP problem is generalization of the Hamiltonian Cycle. Suppose a complete graph G=(V, E)in which every edge $(u, v) \in E$has nonnegative cost C(u,v) . In the problem, graph vertices equal cities, and graph edges and the cost of them are equivalent to path between cities and length of path respectively.

The salesman should start with an origin city and visit all other cities once and return to the origin in a tour with minimum cost [8]. For example in figure 1, the tour$< u, w, v, x, >$, which is bold, has minimal cost. Therefore, the objective of the TSP problem is to find a permutation with minimum cost.

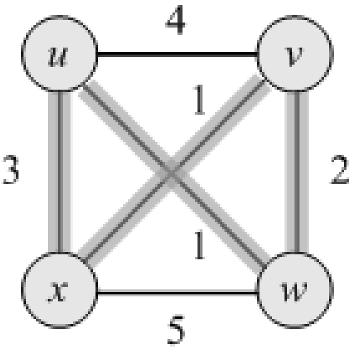


Figure 1.An Example of Complete Graph and It's Best Tour

2.5 The Graph Bandwidth Minimization Problem

Suppose G= (V, E) as a graph. A labeling L of G assigns the integers {1, 2… n} (n=the number of edges) to the vertices of G. Let L(v) is the label of vertex v, where each vertex has a different label. The bandwidth of a vertex v, $B_L(v)$, is the maximum of the differences between L(v) and the labels of its adjacent vertices. That is:

$$B_L(v) = \max\{|L(v) - L(u)|: (v,u) \epsilon E\} \quad (4)$$

The bandwidth of a graph G respecting the labeling L is then

$$B_L(G) = \max\{B_L(v): v \in V\} \quad (5)$$

The bandwidth B (G) of graph G is thus the minimum $B_L(G)$ value over all possible labeling L. In other words, the bandwidth minimization problem consists of finding a labeling L that minimizes $B_L(G)$. If we consider the incidence matrix of graph G, the problem can be formulated as finding a permutation of the rows and the columns of this matrix that keeps all the non-zero elements in a band that is as close as possible to the main diagonal. This is why this problem is known as the Matrix Bandwidth Minimization Problem (MBMP). It has been proved that MBMP Problem is NP-Complete [9] and also shown that it is NP-complete even if the input graph is a tree, maximum vertex degree of which is 3 [10].

2.6 The Software Clustering Problem

Software module clustering involves partitioning of connected software modules into clusters according to predetermined criterion which is called Basic MQ. The Basic MQ measurement [11] is the objective function of this problem. It measures inter-connectivity (i.e., connections between the components of two distinct clusters) and intra-connectivity (i.e., connections between the components of the same cluster) independently.

The MQ is defined as a measurement of the quality or fitness of a particular system modularization. This is designed to produce higher values as the intra-connectivity increases and the inter-connectivity decreases. To accomplish this goal, source code analysis tools are relied on to transform the source code of a system into a language-independent directed graph which is called the Module Dependency Graph (MDG). It's considered that find finding the optimal clustering is an NP hard problem.

The Basic MQ measures inter-connectivity and intra-connectivity independently.

Intra-Connectivity (cohesion): It measures the degree of connectivity between the components that are grouped in the same cluster. A high degree of intra-connectivity indicates good subsystem partitioning because the modules grouped within a common subsystem share many software-level components. A low degree of intra-connectivity indicates poor subsystem partitioning because the modules assigned to a particular subsystem share few software-level components. We define the intra-connectivity measurement A_i of cluster i consisting of N_i components and μ_i intra-edge dependencies as:

$$A_i = \frac{m_i}{N_i^2} \quad (6)$$

This measurement is a fraction of the maximum number of intra-edge dependencies that can exist for cluster i, which is N_i^2. The value of A_i is between 0 and 1. A_i is 0 when modules in a cluster do not share any software-level resources; on the contrary, A_i is 1 when every module in a cluster uses a software resource from all of the other modules in its cluster (i.e., the modules and dependencies within a subsystem form a complete graph).

Inter-Connectivity: It measures the degree of connectivity between two distinct clusters. A high degree of inter-connectivity indicates a poor subsystem partition. Having a large number of inter-dependencies complicates software maintenance because changes to a module may affect many other parts of the system due to the subsystem relationships. A low degree of inter-connectivity is desirable as it indicates that the clusters of the system are, to a large extent, independent. Therefore, changes applied to a module are likely to be localized to its subsystem, which reduces the likelihood of introducing faults into other parts of the system. We define the inter-connectivity E_{ij} between clusters i and j consisting of N_i and N_j components, respectively, and with ε_{ij} inter-edge dependencies as:

$$E_{ij} = \begin{cases} 0 & if\ i = j \\ \frac{e_{ij}}{2N_i N_j} & if\ i \neq j \end{cases} \tag{7}$$

The inter-connectivity measurement is a fraction of the maximum number of inter edge dependencies between clusters i and j ($2N_iN_j$). This measurement is bounded between the values of 0 and 1. E_{ij} is 0 when there are no module-level relations between subsystem i and subsystem j; E_{ij} is 1 when each module in subsystem i depends on all of the modules in subsystem j and vice-versa.
Now that inter-connectivity and intra-connectivity have been described formally, we define the Basic MQ measurement for a MDG partitioned into k clusters, where A_i is the interconnectivity of the i^{th} cluster and E_{ij} is the inter-connectivity between the i^{th} and j^{th} clusters as:

$$BasicMQ = \begin{cases} \frac{1}{k}\sum_{i=1}^{k} A_i - \frac{1}{\frac{k(k-1)}{2}} \sum_{i,j=1}^{k} E_{i,j} & if\ k > 1 \\ A & if\ \ k = 1 \end{cases} \tag{8}$$

The Basic MQ measurement demonstrates the trade-off between inter-connectivity and intra-connectivity by rewarding the creation of highly cohesive clusters while penalizing the creation of too many inter-edges. The Basic MQ measurement is bounded between -1 (no cohesion within the subsystems) and 1 (no coupling between the subsystems) [12].

2.7 The Data Fragment Allocation Problem in Distributed Systems

A distributed database is composed of a collection $S = \{C_1, C_2, \ldots, C_m\}$ of m sites, where each site i is characterized by its capacity c_i and a set $F = \{S_1, S_2, \ldots, S_n\}$ 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,

$$R = \begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,n} \\ r_{2,1} & r_{2,2} & \cdots & r_{2,n} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ r_{m,1} & r_{m,2} & \cdots & r_{m,n} \end{bmatrix} \tag{9}$$

, 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,

$$T = \begin{bmatrix} t_{1,1} & t_{1,2} & \cdots & t_{1,m} \\ t_{2,1} & t_{2,2} & \cdots & t_{2,m} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ r_{m,1} & r_{m,2} & \cdots & r_{m,m} \end{bmatrix} \tag{10}$$

, 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 to find the optimal placement of the fragments at the sites. In fact, it is to find the placement $P = \{p, p_2, \dots, p_n\}$(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

$$\sum_{j=1}^{n} r_{i,j} \times s_j \le c_i \qquad \forall i | 1 \le i \le m \tag{11}$$

,and the total transmission cost is minimized [13].

$$\sum_{i=1}^{m} \sum_{j=1}^{n} r_{i,j} \times t_{i,p_j} \tag{12}$$

The data allocation is a NP- complete problem [14 and 30].

3. The Learning Automata and Genetic Algorithms

Learning automata is an adaptive decision-making device 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, the iterative process of interacting on the environment can result in selection of the optimal action [15]. Figure 2 illustrates how a stochastic automaton works in feedback connection with a random environment.

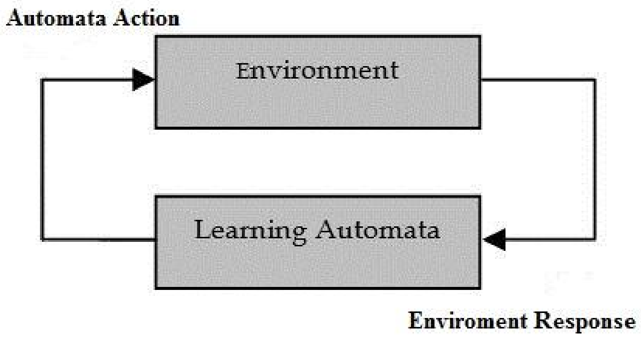


Figure2. The interaction between Learning Automaton and Environment [15]

The automaton chooses one of the offered actions according to a probability vector which at any time instantly 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. 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. 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.

4. The Proposed Approach Based On Genetic Algorithm and Object Migration Learning Automaton

If genetic algorithm and learning automaton as well as concepts such as gene, action and depth are integrated, the speed of finding appropriate solutions by GA can accelerate. In fact, using these concepts is an effort to prevent the algorithm from being trapped in local optimal solutions. Also, Self-improvement, reproduction and penalty and reward are other remarkable characteristics of the hybrid algorithm.

4.1 Representation

Unlike classic genetic algorithms, which use binary encoding, the chromosomes of new proposed approach are encoded as an object migration learning automaton, in which each gene is associated with automaton actions, and is located in a specific depth of the action [16, 17].

In general, an object migration automaton is shown as {V, α, , F, G}[17]. For example, to solve the TSP problem, elements of automaton are as follows [21]:

- The V= {V_1 ,$V_{2,\ \ldots,}$ V_n} set is a group of objects. Indeed, in the TSP problem the objects are nodes of input graph.
- Theα= { α_1 ,$\alpha_{2,\ \ldots,}$ α_k} set is the actions of automaton; that is, the automaton has k actions that the number of action equals the number of graph's node. Moreover, if node u of the graph is placed on action m, node u will be the m^th City in the tour.
- The = { $_1$, $_{2,\ldots,}$ $_{KN}$} is the set of states. In addition, N is depth of automaton. The set is divided into k subsets, { $_1$, $_{2,\ldots,}$ $_N$},{ $_{N+1}$, $_{N+2,\ldots,}$ $_{2N}$}, …, { $_{(k-1)N+1}$, $_{(k-1)\ N+2,\ldots,}$ $_{kN}$} .Each object of automaton is classified according to its state. In other words, If object u is placed in one state of{ $_{(i-1)N+1}$, $_{(i-1)\ N+2,\ldots,}$ $_{iN}$}, it will be the j^th city in the tour. Besides, $_{(i-1)N+1}$ and $_{iN}$ are internal and boundary states respectively. Also, The object located in an internal state is more important.
- The B= {0, 1} is set of automaton inputs; that is, 1 and 0 means failure and success respectively.
- Function F is transition function, which determines next sate according to current state and automaton input. In other words, the function defines how objects move in states of automaton.
- Function G is output mapping function, which chooses action for each state.

For Example, a complete graph is given in figure 3.

$$M = \begin{array}{c|cccccc} & a & b & c & d & e & f \\ \hline a & 0 & 5 & 8 & 7 & 2 & 10 \\ b & 5 & 0 & 3 & 8 & 7 & 5 \\ c & 8 & 3 & 0 & 5 & 7 & 1 \\ d & 7 & 8 & 5 & 0 & 4 & 3 \\ e & 2 & 7 & 7 & 4 & 0 & 6 \\ f & 10 & 5 & 1 & 3 & 6 & 0 \end{array}$$

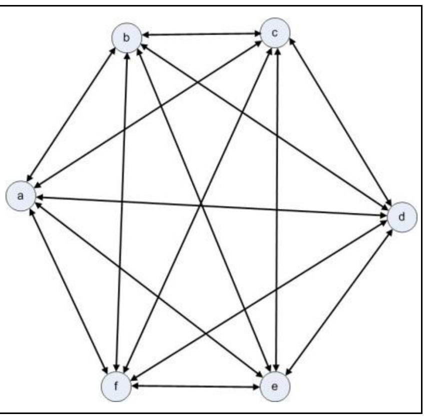


Figure 3.An instance of 6-node complete graph.

Suppose (c, b, f, d, a, e) as a permutation of nodes of mentioned-above graph in figure 3. This tour can be shown as a learning automaton based on Tsetline connections [17] in figure 4.

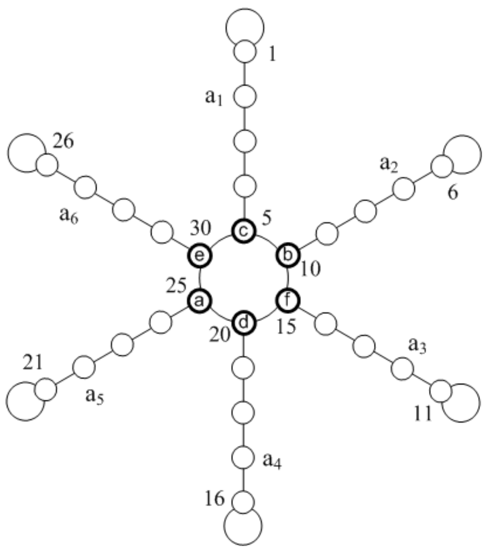


Figure 4.An Object Migration Learning Automaton Based on Tsetline Connections

4.2 Operations

As mentioned before, chromosomes in hybrid algorithm are represented as learning automaton, so the crossover and mutation operators are not similar to classic genetic operators; in addition to them, the algorithm has more operators which are called penalty and reward.

4.2.1 The Selection Operator

The Roulette Wheel is used as selection operator although other selection operators are eligible to be used.

4.2.2 The Crossover Operator

At first, two genes r1 and r2 are chosen randomly. The set of genes between r1 and r2 is referred to as crossover set. Then, in the first chromosome all genes in crossover set are swapped for their corresponding gens of the second chromosome and vice versa. The pseudo code of the operator is shown in figure 5.

```
Procedure Crossover ($LA_1$, $LA_2$)
Begin
Generate two random numbers $r_1$ and $r_2$ between 1 to n.
$r_1$ = Random *n; $r_2$ = Random *n;
$r_1$ = Min(r1, r2); $r_2$ = Max($r_1$, $r_2$);
for i = r1 to r2 do
j = Action of LA1 where  $LA_1$ .Object($LA_1$.Action(j))=  $LA_2$.Object($LA_2$.Action(i)));
Swap(LA1.State(LA1.Action(i), $LA_1$.State($LA_1$.Action(j));
j = Action of $LA_2$ where  $LA_2$ .Object($LA_2$.Action(j))= $LA_1$.Object($LA_1$.Action(i)));
Swap($LA_2$.State($LA_2$.Action(i), $LA_2$.State($LA_2$.Action(j));
end for
   End Procedure
```

Figure 5.The Pseudo Code of Crossover Operator

An illustration of this operator is given in figure 6. In the figure, LA_2 and LA_5 automata are selected by operator selection. Next, genes a2 and a3 are chosen at random, and the crossover set of $\{\alpha_{2,}\ \alpha_3\}$ is formed; finally, two new chromosomes were created by exchanging genes exist in crossover set.

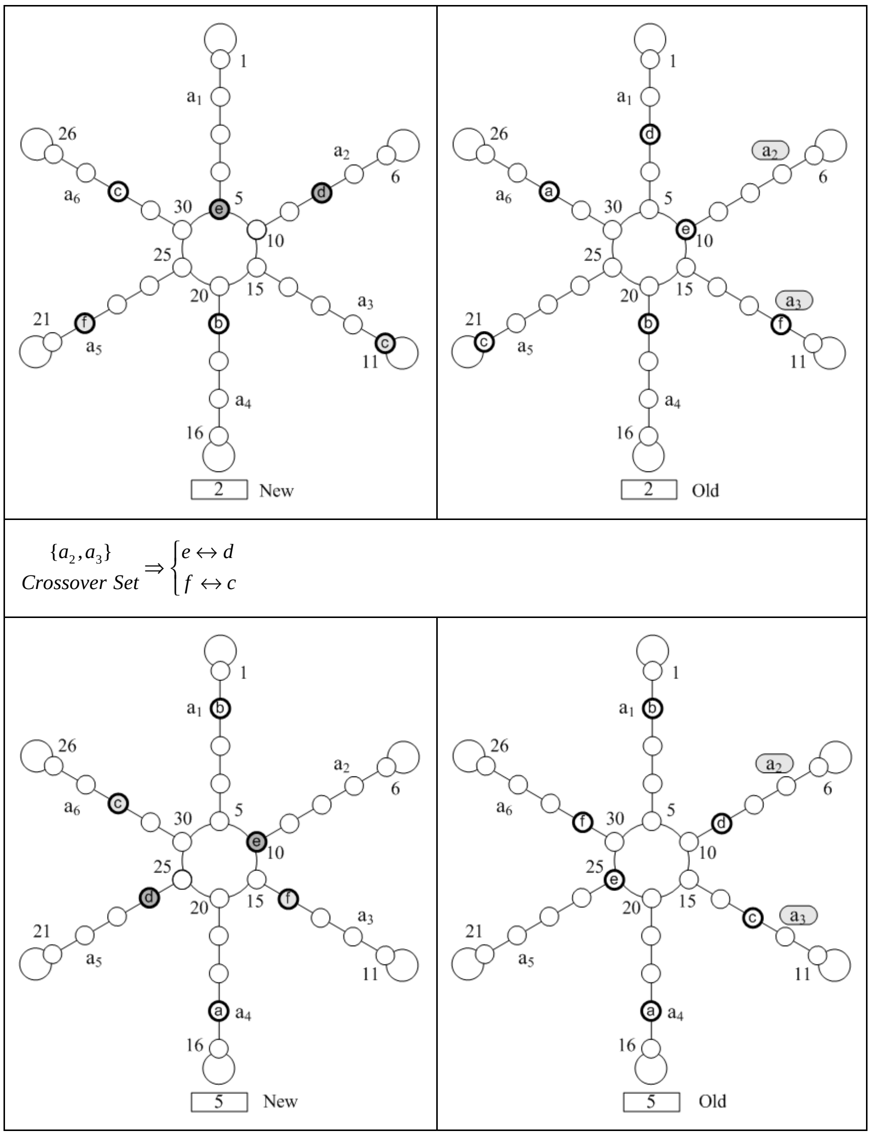

2 New
2 Old
$\{a_2, a_3\}$ Crossover Set $\Rightarrow \begin{cases} e \leftrightarrow d \\ f \leftrightarrow c \end{cases}$
5 New
5 Old

Figure 6: An illustration of the crossover operator

4.2.3 The Mutation Operator

The swap operator is used to mutate chromosomes into the forms that are suitable for learning automaton. The pseudo code of the operator is shown in figure 7.

```
Procedure Mutation (LA)
i = Random *n;
j = Random *n;
Swap(LA.Object(LA.Action(i)),LA.Object(LA.Action(j)));
End Mutation
```

Figure 7.The Pseudo code Of Mutation Operator

An example of mutation operator is clearly seen in figure 8.

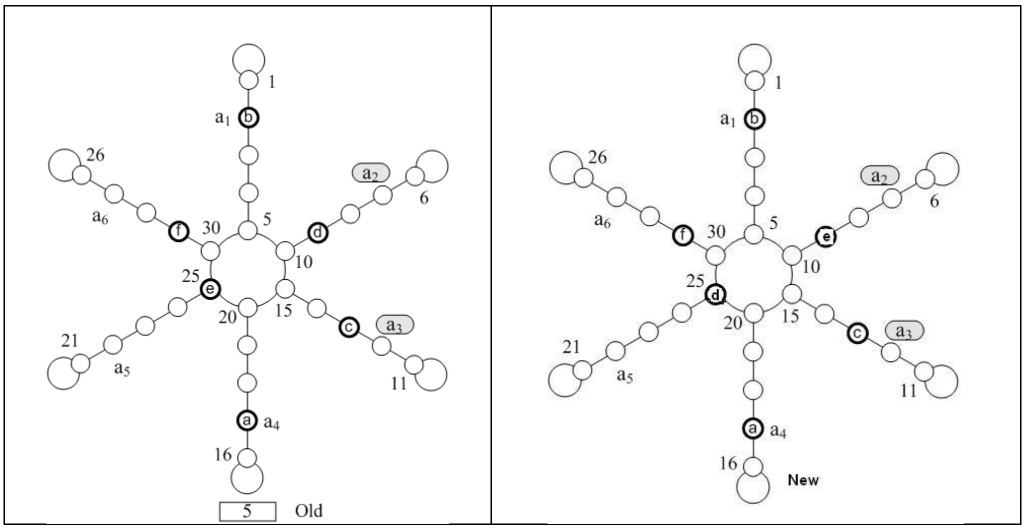


Figure 8.The way of mutation

4.3 The Penalty and Reward Operators

Just by coincidence, an object (i.e., node) of learning automaton is chosen, and subsequently it is either rewarded or penalized. Indeed, if the average weight of input and output edges of a specified node is less than threshold, the object will be rewarded and move toward more internal states; otherwise, it will be penalized and be placed in more boundary states. However, the value of threshold at each moment equals

$$\text{Threshould} = \frac{\text{Cost of the tour}}{\text{count of graphs nodes}}$$

It is important that taking either reward or penalty changes the state of node. If a node is placed in boundary state of an action, taking penalty will oblige object to go through the boundary state and change its action; therefore, new permutation will be created which is better than the previous tour. Figure 9 and 10 show the pseudo code of reward and penalty operators respectively.

```
Procedure Reward(u)
Begin
if (State(u)-1) mod N <> 0 then
Dec (State(u));
End Procedure
```

Figure 9.The Pseudo Code of Reward 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 for
until at least one node appears in the boundary state
bestTourLenght = ∞;
for U = 1 to n do
    Create permutation LA′ from LA by swapping u and U;
    if  Lenght(Specified Tour by LA′)<bestTourLenght then
        bestTourLenght = Lenght(Specified Tour by LA′);
        bestNode = U;
    end if
end for
LA.State(bestNode) = LA.Action(bestNode)*N;
LA.State(u) = LA.Action(u)*N;
Swap(LA.State(u),LA.State(bestNode));
End Procedure
```

Figure 10.The Pseudo Code of Penalty Operator

Figure 11 and 12 are instances of reward and penalty operators respectively.

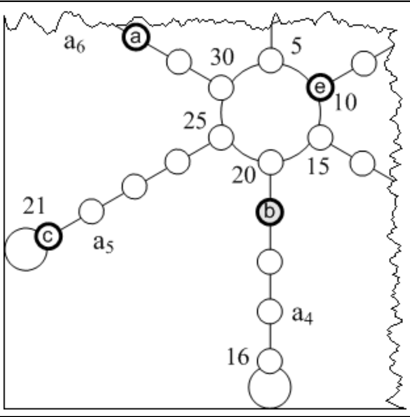
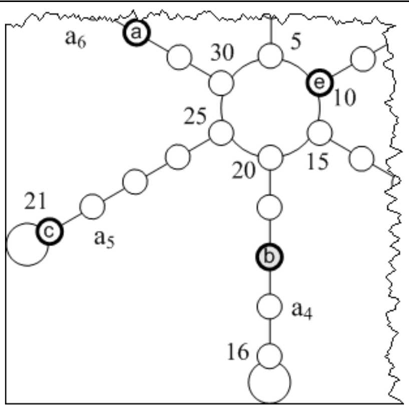
	
a) Status of b node before reward	b) Status of b node after reward

Figure 11.An example of Reward Operator

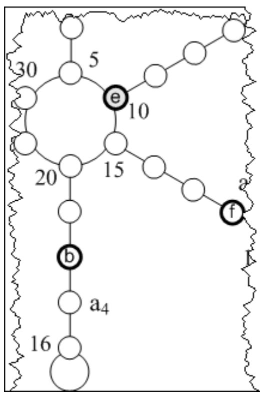
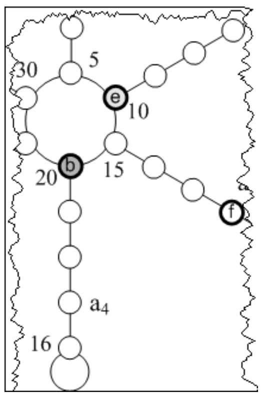
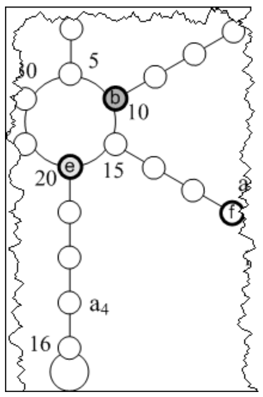
		
a) Status of b node before penalty	b) Coming toward boundary state	c) Exchange between b and e nodes

Figure 12.An instance of penalty Operator

Now, regarding the pervious descriptions, we show the algorithm used to solve the TSP in Figure 13. In fact, the algorithm consists of two phase; first stage is GA and the second stage includes learning automaton.

```
Function TSP_Solver(G) : TSP_Tour
Begin
n = Size of Population;    // n = |VG|
Create the initial population LA_1 … LA_n;
EvalFitness();
while( All (Length of Specified Tour By LA_i> Constant-Value) ) do

//stage one
NewLA_1 = NewLA_2 = LA with minimum Value of  Tour-Lenght;
for i = 2 to n do
Select LA_1;  Select LA_2 ;
if (Random > 1 - CrossoverRate) then
Crossover ( LA_1, LA_2 );
if (Random > 1 - MutationRate) then
Mutation ( LA_1 ); Mutation ( LA_2 );
NewLAi+1 = LA1;
NewLAi+2 = LA_2 ;
i=i+2;
end for
//stage two
for i = 0 to n do
LA_i = NewLA_i;
u = Random *n;
if ( Ju( LA_i) < threshold Threshold(LA_i )) then   Reward(LAi , u );
else   Penalize(LA_i , u );
end for
EvalFitness();
end while
End Function
//Threshold(LA_i) = Lenght( Specified Tour by LA_i ) / |V_G|;
   //Ju(LA_i) = (lenght of edge (u-1,u) in LA_i + lenght of edge (u,u+1) in LA_i) / 2;
```

***Figure 13.The Proposed Hybrid Algorithm to Solve the TSP*[21].**

4.4 Representation of Chromosomes for case study problems

The nature of case study problems is rather similar because most of them are permutation based problems; indeed, in this kind of problems, the solutions are represented as permutations; therefore, the proposed approach can be used to solve them.

Moreover, the structure of the algorithm is the same, but the encoding of diverse problems can be a little different. However, all of these problems can be encoded as an object migration learning automaton, but their encodings slightly contrast with each other in regard to the definition of objects and actions.

4.4.1 Encoding of theJoin Ordering Problem

In this problem the set$V= \{V_1,V_2, ..., V_n\}$ is an array of Join operators. These operators move over different states of automaton and create new various permutations of join operators.

In the automaton, $\alpha= \{\alpha_1,\alpha_{2,...,}\alpha_k\}$ is a set of allowed actions of automaton which has k actions; in fact, the number of actions equals the number of join operators. Also, if the join u of given query is placed on the action m, it will be mth join operator of the query that should be executed. For more information refer to [18 and 19].

4.4.2 Encoding of the Total Weighted Tardiness Scheduling Problem

Similarly, $V= \{V_1,V_2, \dots, V_n\}$ is the set of jobs that should be scheduled to execute. The transfer of jobs over varied states results in different time tableting.

Like the previous problem, α is the set of admissible actions of learning automaton, with k actions which equals the number of jobs to be scheduled. Besides, the job u will be the mth job which be executed in case it is located on the action m [20].

4.4.3 Encoding of Hamiltonian Cycle Problem

Definition of objects and actions of automaton in Hamiltonian cycle problem is exactly the same as TSP problem's mentioned before. Refer to [22] for further details.

4.4.4 Encoding Of Graph Bandwidth Minimization Problem

In the Graph Bandwidth Minimization problem, V and α are the set of labels and nodes of given graph respectively. Take for example; the label u is placed on action m, in that way the label u will be assigned to mth node of the graph [23].

4.4.5 Encoding of the Software Clustering Problem

Although the software clustering is not a permutation based problem, the chromosomes can be encoded as learning automaton in which the objects lying on actions are graph nodes [26].

In spite of Bunch Genetic Method [24], it does not have the number of related cluster; it contains the number of another node of graph [25]. In fact, the arrangement of objects on the actions is defined as same clustering relations. This permutation is mapped with a decoding algorithm to a clustering. The algorithm in figure 14 represents decoding of such a permutation based Encoding. According to the figure, if the cell value is greater than cell index, index will be located in the new cluster; otherwise, it will be placed in the same cluster as cell value. In order to understand better, note figure 15.

```
Decode Chromosome- Algorithm
Array C holds a permutation of graph nodes 1 to n
For node i=1    to   N  do
if C[i]>=i   Create a new cluster and assign node i to it.
Else
Allocate i to the same cluster as node C[i].
   End Decode Chromosome.
```

Figure 14. The Decoding Algorithm for the Permutation Based Encoding [25]

For example, the object number 5 is located in action number 2; therefore, the node number 2 will be located in a new cluster. Node number 1 has already been located in cluster number 1; as a result, node number 2 will be located in cluster number 2.

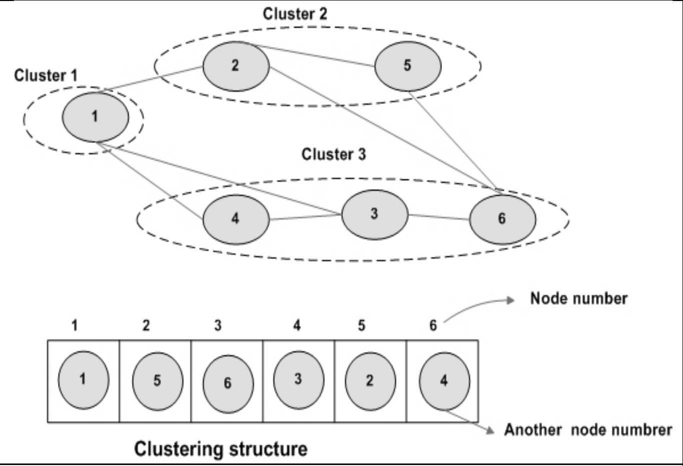
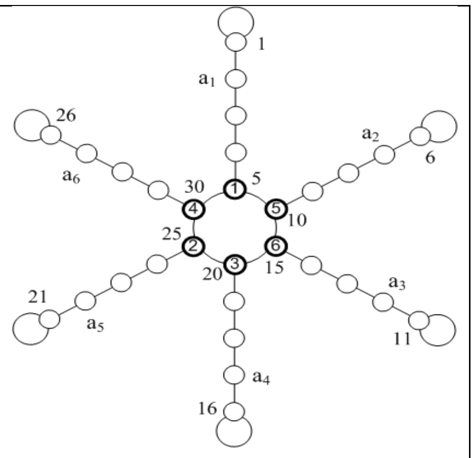
 Cluster 2 Cluster 1 Cluster 3 Node number Another node numbrer Clustering structure	 1 a1 26 a2 6 a6 30 5 10 25 20 15 21 a3 a5 11 a4 16
a) Clustering structure	b) Mapping on the Object migration automaton

Figure 15. An example of Permutation based Encoding for The Software Clustering Problem

4.4.6 Encoding Of the Data Fragment Allocation Problem in Distributed Systems

We allocate n data fragments to m sites using an automaton in which V= $\{V_1,V_{2,\ ...,}\ V_n\}$ is the set of site's numbers. Additionally, the automaton has n actions; indeed, Theα= $\{\alpha_1,\alpha_{2,...}\ ,\alpha_k\}$ is set of n data fragments in distributed system. The uth fragment will be allocated to mth site of the distributed system in case the object u is placed in action m.

Since the structure of this problem is different from the previous problems, so we can use k-point crossover and simple mutation operator. For more information refer to [27and 28].

5. Experimental Results

In this section, the implementation results of the hybrid algorithm are shown, and compared with some well-known algorithms. Analysis of experimental results is divided into two sections; the first part is the evaluation of fitness values which are obtained by diverse algorithms. Another section is dedicated to assessment of algorithms' stability.

5.1 The Evaluation of Fitness Values for Case Study Problems

The comparison of different algorithms for solving the case study problems is done in this section. The compression criterion is value of fitness function.

5.1.1 The Evaluation of Algorithms' fitness value to solve the Software Clustering Problem

In order to examine the algorithms, a group of module dependency graphs has been used. These software graphs are created by code analyzer tools such as CIA [29]; in other words, CIA applies the source codes of software systems to produce their MDG graphs. Also, the MDG is input of algorithms. A list of software systems with their descriptions is shown in Table1.

Table 1. Description of software systems

SOFTWARE SYSTEM	MODULE IN MDG	EDGES IN MDG	SYSTEM DESCRIPTION
Compiler	13	32	A small Compiler
Boxer	20	29	A graph designing system
Mini-Tunis	20	57	A simple operating system
FSS	16	51	A file system service
Ispell	24	103	An open source spell checker

The Bunch-Genetic algorithm [24], DAGC-Genetic algorithm [11], Hill climbing approach [11], learning Automata based Algorithm and GALA, which is the proposed hybrid algorithm are approaches which are applied to compare. Also, since most of the used algorithms are random approaches, each algorithm has been executed 5 times on a specified instance, so the average value of Basic MQ is shown in figure 16.

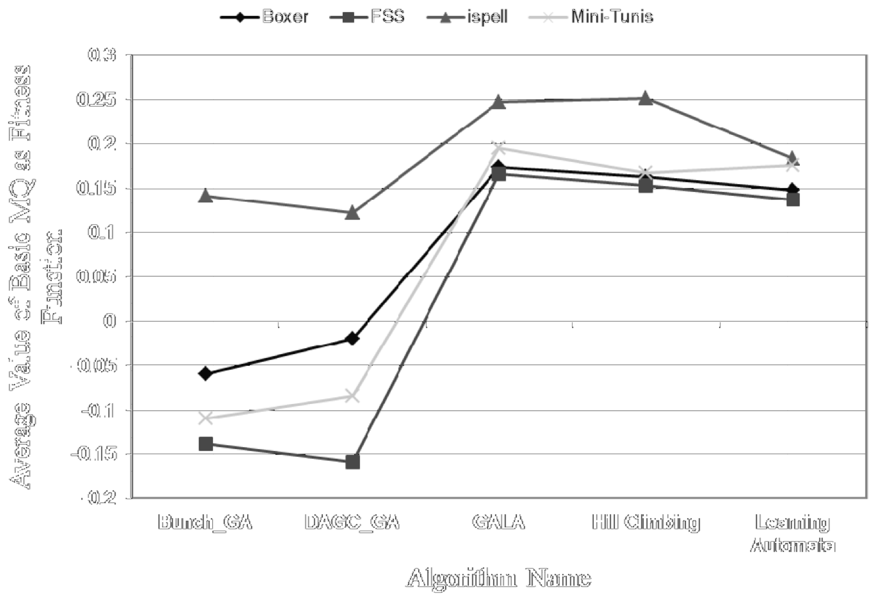


Figure 16: The comparison of Basic MQ among a number of algorithms

Figure 16 reveals the amounts of fitness function of Basic MQ which range from -1 to 1 in different algorithms. It is clearly seen that the GALA algorithm has the highest value of MQ in almost all of the instances. Also, it has noticeable difference with other approaches, especially genetic algorithms.

5.1.2 The Evaluation of algorithms' fitness value to solve the Bandwidth minimization problem

A number of instances from Harwell-Boeing Sparse Matrix Collection [32] are used by diverse algorithms to evaluate their fitness values which are bandwidth of graphs. The dimensions of these instances ranges from 30 to 200; besides, all of instances have been obtained from linear systems or other scientific and engineering problems. The graphs used are as follow: ash85.mtx, bcsstk04.mtx, fs_183_1.mtx, Ins_131.mtx and west0156.mtx.

Also, the discussed algorithms are hybrid GA-HC algorithm [23], a heuristic GPS method [23] , SA-DJ which is simulated annealing approach [23], Learning Automata based and GALA algorithm. Results are shown in figure 17.

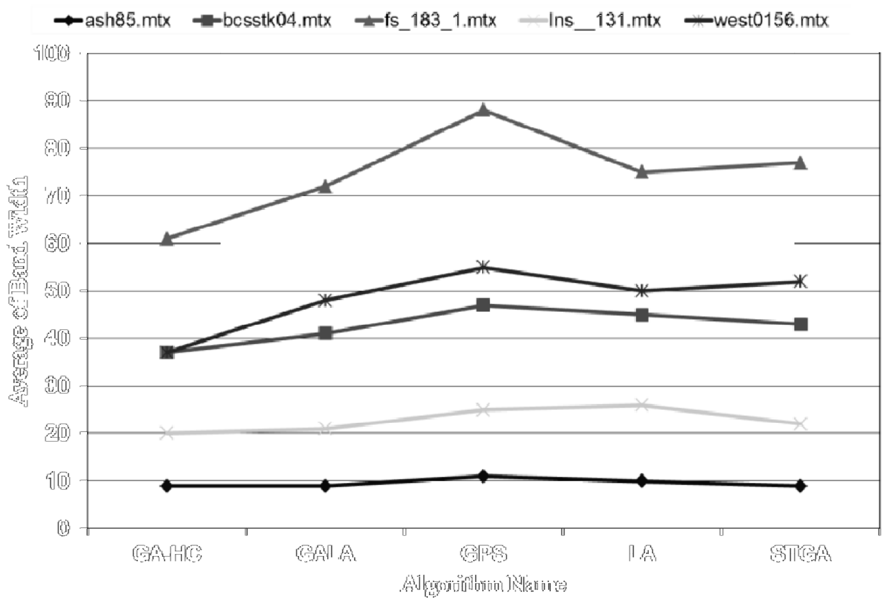


Figure 17: Performance comparison of the algorithms to solve the BMP problem

Generally, constructing a labeling with low bandwidth is the objective of BMP problem. According to the figure 17, GALA and GA-HC are two domineering approaches since they can generate suitable solutions with the lowest value of bandwidth. It should be considered that GAHC is one of the famous methods to solve the BMP problem. Therefore, GALA can be an effective algorithm to solve the problem.

5.1.3 The Evaluation of Algorithms' Fitness Value to Solve the Data Allocation Problem

The algorithms applied to compare with the new algorithm to solve the Data Allocation Problem are as follows: the neighborhood random search algorithm (RS) [28] and two genetic algorithms, the Corcoran and the Ishfaq Algorithm [13]. In order to compare the quality of generated solutions, different databases with varied data fragments are rendered. Also, the number of sites ranges from 25 to 100, and the number of sites is supposed to be 20. Figure 18 provides some information about them.

In general, the objective of data allocation algorithms is to determine an assignment of fragments to different sites in a way that minimize the total data transfer cost incurred in executing a set of queries. According to figure 18, there are remarkable declines in LAGA and OMA algorithms; in other words, these two algorithms create the most suitable allocations among other approaches although GALA in most of the cases generates slightly better solutions than OMA.

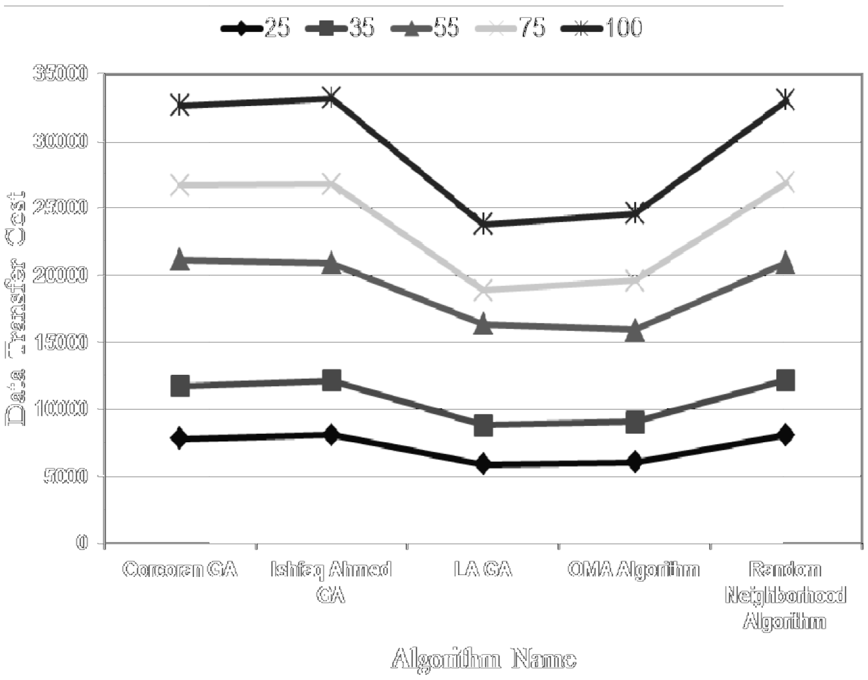


Figure 18. The comparison of different algorithm considering data transferring cost

5.1.4 The evaluation of algorithms' fitness value to solve the Total Weighted Tardiness Scheduling Problem

A set of benchmark instances for solving the problem is available at OR Library. Also, it contains the best known results which have yet been acquired. An instance from OR library, which has 100 jobs, used to compare the efficiency of varied methods. The algorithms compared with hybrid GALA approach are as follows: Descent Method, GA, learning Automaton, Memetic algorithm, Greedy approach, Iterated Dynasearch Neighborhood and Optimal solution [20], which is best known.

The goal is to find a permutation (schedule) of jobs which minimizes the total weighted tardiness of jobs executions. Like the previous sections, Figure 19 points out that LAGA algorithm creates the nearest solutions to the results of optimal solutions, which have yet been the best known results.

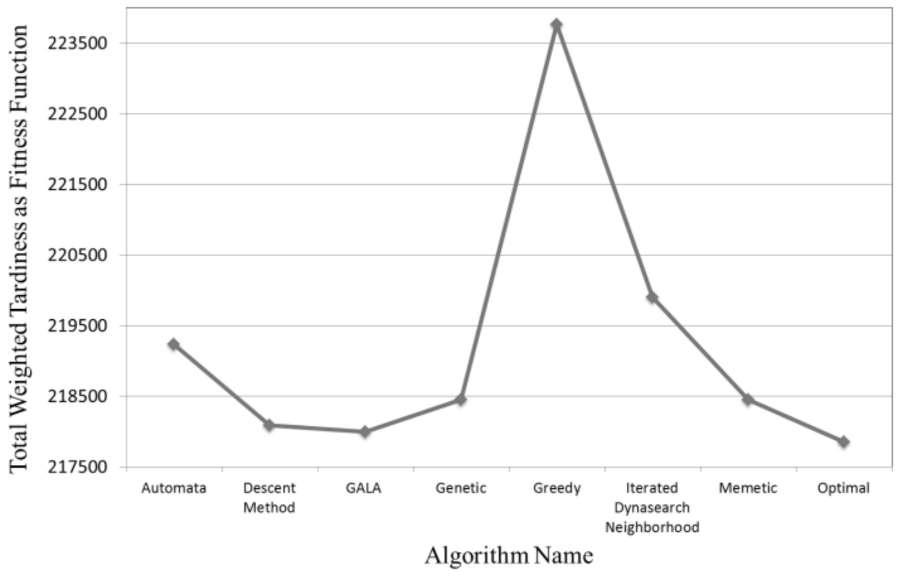


Figure 19. Performance comparison of the algorithms to solve the Total Weighted Tardiness Scheduling Problem

5.1.5 Evaluation of algorithms' fitness value to solve the Hamiltonian Cycle problem

For solving Hamiltonian cycle problem, different problem instances with various edge densities are examined. Some of these problems are got from the TSP library [21], and others are produced according to the previous papers. The concept of edge density is the ratio of number of edge in current graph to number of edges in complete graph.

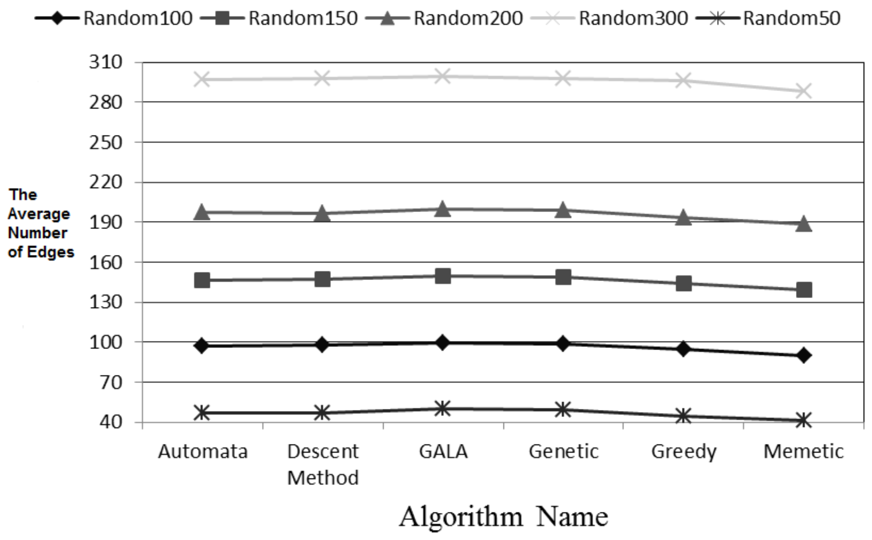


Figure 20.Performance comparison of the algorithms to solve the Hamilton Cycle problems with the edge density of 0.15.

Generally, the goal of the problem is to find a permutation of graph nodes which maximizes the number of visited edges. In the experiment, five problems including 50,100, 150, 200 and 300 nodes with different values of density are produced. The figure 19 indicates that in most of the cases, the hybrid algorithm finds the greatest number of edges.

5.1.6 Evaluation of algorithms' fitness value to solve the TSP Problem

In this experiment, some standard instances from TSP library and some random generated instances are used. The instances contain 22 to 280 nodes and the algorithms are GA, NN heuristic, Greedy approach, minimum spanning tree, Learning Automata and the hybrid algorithm. The lengths of TSP tours found by the diverse algorithms have been shown in figure 21. The results show that the tours found by Hybrid algorithm had the minimum length.

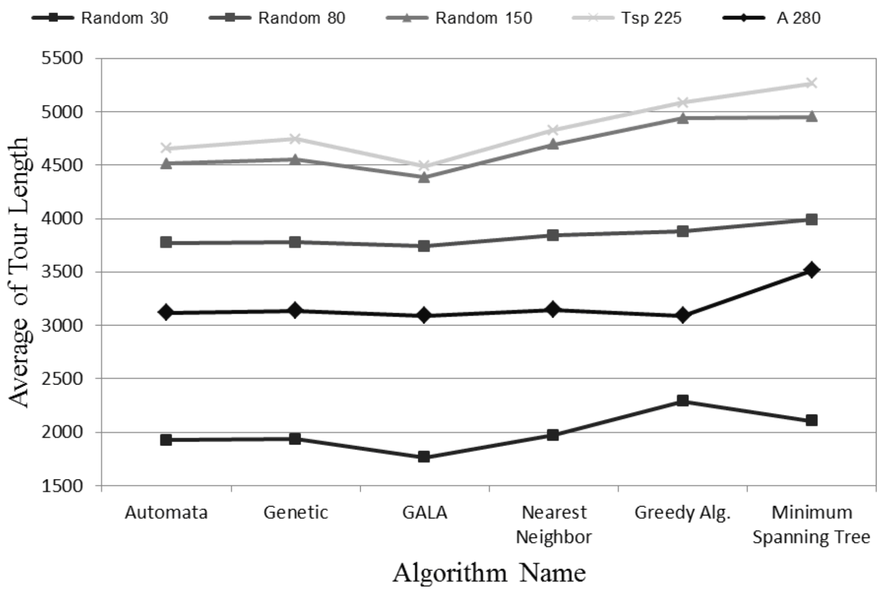


Figure 21: performance comparison of the algorithms to solve the TSP problem

5.1.7 Evaluation of algorithms' fitness value to solve the Join Ordering Problem

To assess the efficiency of the algorithms for solving the Join Ordering Problem, some problems with 20-100 join operators have been generated by coincidence. Figure 22 demonstrates the cost of different query execution plans which are gained by the varied algorithms which are GA, learning Automaton and GALA. The results indicate that GALA has considerably superiority over its rivals since it creates plans with the least values of execution cost.

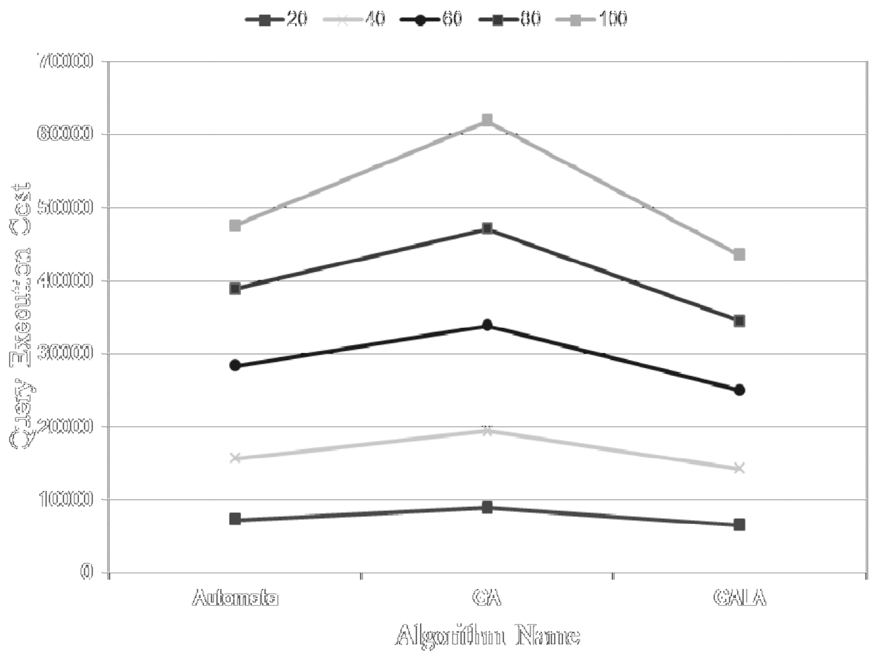


Figure 22. Performance comparison of the algorithms to solve the Join ordering problem

5.2 Evaluation of algorithms' stability

This section is dedicated to stability of algorithms, so the value of standard deviation is measured for diverse methods. Figures 23 and 24 shows this comparison for Software Clustering and Hamiltonian Cycle problem, respectively.

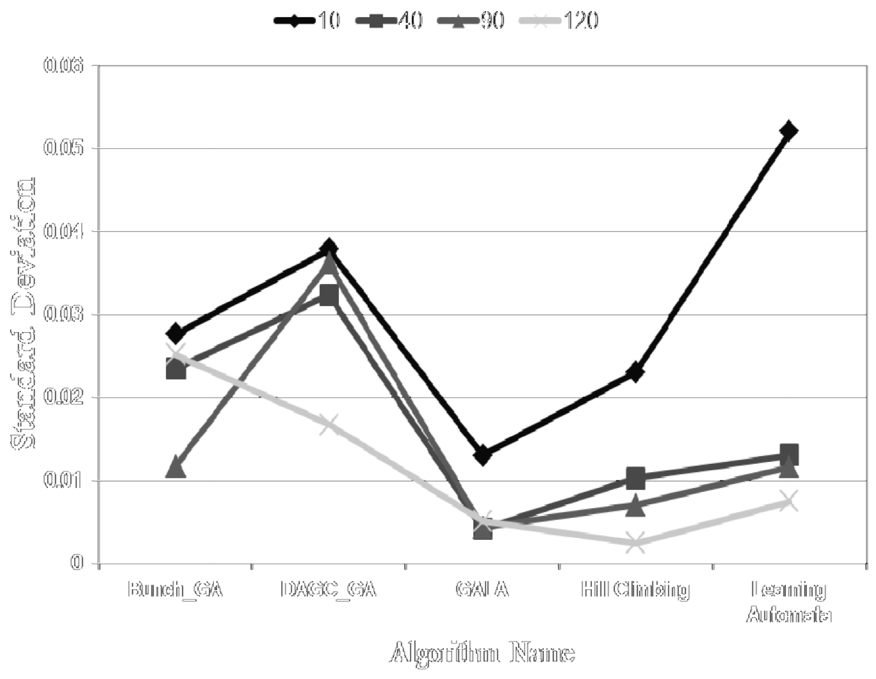


Figure 23. Stability of the algorithms to solve the software clustering problem

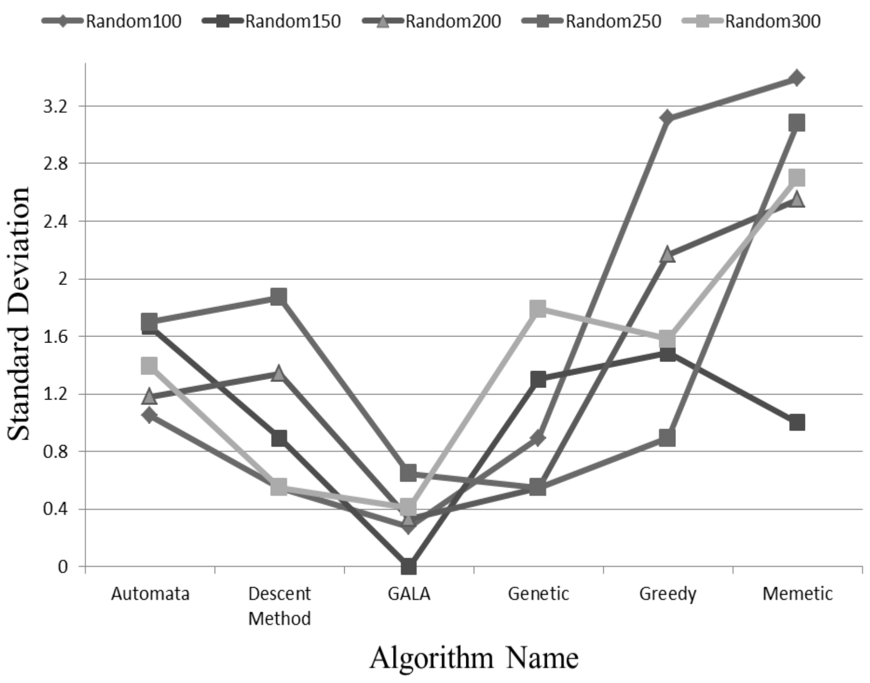


Figure 24. Stability of the algorithms to solve the Hamiltonian Cycle problem

The Comparison results indicate that the values of the criterion in GALA are substantially less than others.

The less standard division is the more stability is; that, the less standard division results in figures of different runs being closed, practically the same; consequently, The GALA can be considered as a reliable and stable technique to solve the case study problems.

6. Conclusions

To conclude, informed approaches such as various heuristic methods- techniques based on simple learning and other repetitive algorithms- have their own weaknesses like getting stuck in local optimum, So in this paper, a newly developed algorithm is proposed which is based on learning automaton and GA combination; in fact the algorithm is prohibited from falling in local optimal solutions, and mostly generates better solutions.

Furthermore, another positive point of this hybrid algorithm is its remarkable stability because of standard division it's results is less than other algorithms'. It means that the results are almost similar; thus, we can consider the hybrid algorithm as a reliable technique to solve the case study problems. Eventually, it seems that regarding variety of studied problems and experiments, the new GALA algorithm can be considered as a precious approach in optimization problems and can be used as a suitable tool to solve the other NPproblems in the future.

7. References

[1] K. Bennet, M. C. Ferris, and Y. E. Ioannidis, A genetic algorithm for database query optimization, In Proc. Of the Fourth Intl. Conf. on Genetic Algorithms, pp. 400-407, San Diego, USA, 1991.

[2] T. Ibaraki and T. Kameda, Optimal nesting for computing N-relational joins, ACM Trans. on Database Systems, 9(3): pages 482-502, 1984.
[3] E. Sevinc and A. Cosar, An Evolutionary Genetic Algorithm for Optimization of Distributed Database Queries. The Computer Journal,v.54, pp.1-15, 2011.
[4] J. K. Lenstra, A. H. G. RinnooyKan, and P. Brucker, Complexity of machine scheduling problems, Annals of Discrete Mathematics, Vol. 1, pp. 343-362, 1977-
[5] J. Du, J.Y.T. Leung, Minimizing total tardiness on one processor isNP-hard, Math Operations Research, Vol. 15, pp.483–495, 1990.
[6] Haddad, H &Nematollahi, M, Minimizing total weighted tardiness for the single machine scheduling problem with dependent setup time and precedence constraints.Management Science Letters, 2(2), 517-524, 2012.
[7] K. A. De Jong, and W.M. Spears, Using genetic algorithms to solve NP-complete problems, proceeding of Third International Conference on Genetic Algorithms and their Applications, pp. 124-132, Kaufmann, 1989.
[8] S. OveisGharan, A. Saberi, M. Singh, A Randomized Rounding Approach to the Traveling Salesman Problem, FOCS, 2011.
[9] C. H. Papadimitriou, The NP-completeness of the bandwidth minimization problem, Computing, Vol. 16, pp. 263-270, 1976.
[10] M. Garey, R. Graham, D. Johnson and D. E. Knuth, Complexity results for band-width minimization, SIAM Journal of Applied Mathematics, Vol. 34, pp. 477-495, 1978.
[11] B. S. Mitchell, A Heuristic Search Approach to Solving the Software Clustering Problem, Ph.D Thesis, Drexel University, Philadelphia, 2002.
[12] D. Doval and S. Mancoridis, Automatic Clustering of Software Systems using a Genetic Algorithm, Proceedings of the IEEE Int. Conf. On Software Tools and Engineering Practice (STEP'99), 1999.
[13] A. Corcoran and J. Hale, A Genetic Algorithm for Fragment Allocation in a Distributed Database System, ACM, pp.247-250, 1994.
[14] W.W. Chu, Optimal file allocation in a multiple computer system, IEEE Transactions on Computers, vol. C-18, no. 10, pp. 885–889, 1969.
[15] Narendra K, Thathachar M A L. Learning Automata: An Introduction, Prentice Hall, 1989.
[16] Oommen B J, Ma D C Y. Deterministic Learning Automata Solutions to the equip partitioning problem. IEEE Trans. on Computers, 1998.
[17] M. Rezapour, Solving Graph Isomorphism problem Using Learning Automata, M. Sc. Thesis, Amirkabir University of technology, Tehran, Iran, 2004.
[18] Asghari, K., Safari Mamaghani, A. and Meybodi, M. R., An Evolutionary Approach for Query Optimization Problem in Database, Proceedings of International Joint Conferences on Computers, Information and System Sciences, and Engineering (CISSE2007), University of Bridgeport, England, 2007.
[19] Asghari, K., Safari Mamaghani, A., Mahmoudi, F. and Meybodi, M. R., A Relational Databases Query Optimization using Hybrid Evolutionary Algorithm, Journal of Computer and Robotics, Vol. 1, No. 1, Islamic Azad University of Qazvin, Qazvin, Iran, 2008.
[20] Asghari, K. and Meybodi, M. R., Solving Single Machine Total Weighted Tardiness Scheduling Problem using Learning Automata and Combination of it with Genetic Algorithm The Third Iran Data Mining Conference (IDMC2009), Iran, Tehran, 2009.
[21] Zarei, B., Meybodi, M. R. and Abbaszadeh, M., A Hybrid Method for Solving Traveling Salesman Problem, Proceedings of 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), IEEE Computer Society, pp.394-399 , Melbourne, Australia, 2007.
[22] Asghari, K. and Meybodi, M. R., Searching for Hamiltonian Cycles in graphs using Evolutionary Methods, Proceedings of the second Joint Congress on Fuzzy and Intelligent Systems, MalekAshtar University of Technology, Tehran, Iran, 28-30 October, 2008.
[23] Safari Mamaghani, A. and Meybodi, M. R., A learning automaton based approach to solve the graph bandwidth minimization problem,Proceeding of the 5th International Conference on Application of Information and Communication Technologies(AICT2011),pp. 1-5, Baku, Azerbaijan ,12-14 October ,2011.

[24] S. Mancordis, B. S. Mitchell, Y. Chen and E. R. Gansner, Bunch: A clustering tool for the recovery and maintenance of software system Structures, Proceedings of Int. Conf. of Software Maintenance, pp. 50-59, 1999.

[25] S. Parsa, and O. Bushehrian, A New Encoding Scheme and a Framwork to investigate Genetic Clustering Algorithms, Journal of Research and Practice in Information Technology, Vol. 37, No. 1, pp. 127-143, 2005.

[26] Safari Mamaghani, A. and Meybodi, M. R., Clustering of Software Systems using New Hybrid Algorithms, Proceedings of the IEEE Ninth International Conference on Computer and Information Technology (CIT 2009), Xiamen, China, pp. 20-25, 2009.

[27] Safari Mamaghani, A., Mahi, M. and Meybodi, M. R., A Learning Automaton based Approach for Data Fragments Allocation in Distributed Database Systems, Proceedings of the 10th IEEE International Conference on Computer and Information Technology (CIT 2010), pp. 8-12, Bradford, West Yorkshire, UK, 2010.

[28] Safari Mamaghani, A., Mahi, M., Meybodi, M. R. and HosseinzadehMoghaddam, M., A Novel Evolutionary Algorithm for Solving Static Data Allocation Problem in Distributed Database Systems, Proceedings of the 2nd International Conference on Network Applications, Protocols and Services (NETAPPS2010), , pp. 14-19, AlorSetar, Malaysia, 2010.

[29] Y. Chen, E. Gansner and E. Koutsofios, A C++ Data Model Supporting Reachability Analysis and Dead Code Detection, Proceedings of 6th European Software Engineering Conf. and 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering, 1997.

[30] Hassan I. Abdalla, A New Data Re-Allocation Model for Distributed Database Systems, International Journal of Database Theory And Application (IJDTA), Volume 5, Issue 2, June, 2012.

[31] T Dokeroglu, A. Cosar, Parallel Genetic Algorithms for the Optimization of Multi-Way Chain Join Queries of Distributed Databases, VLDB, PhD Workshop, Istanbul, 2012.

[32] V. Campos, E. Piñana and R. Martí,Adaptive Memory Programming for Matrix Bandwidth Minimization, Annals of Operations Research 183, pp. 7-23 ,2011.