A Novel Hybrid Algorithm for Join Ordering Problem in Database Queries

Ali Safari Mamaghani[1], Kayvan Asghari[2], Fariborz Mahmoudi[3] and Mohammad Reza Meybodi[4]
Computer Engineering Department
Islamic Azad University of ([1]Bonab, [2] Khamene,[3] Qazvin)
[4]Amirkabir University of Technology
Iran

Abstract: - Optimizing the database queries is one of hard research problems. Exhaustive search techniques like dynamic programming is suitable for queries with a few relations, but by increasing the number of relations in query, much use of memory and processing is needed, and the use of these methods is not suitable, so we have to use random and evolutionary methods. The use of evolutionary methods, because of their efficiency and strength, has been changed in to a suitable research area in the field of optimizing the database queries. In this paper, a hybrid evolutionary algorithm has been proposed for solving the optimization of Join ordering problem in database queries. 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 speed of finding an answer has been accelerated and prevented from getting stuck in local minimums. The results of experiments show that hybrid algorithm has dominance over the methods of genetic algorithm and learning automata.

Key-Words: - Query, Join Operator, Learning Automata, Genetic Algorithms

1 Introduction

Relational data model has been introduced by Codd [1] and in recent years, relational database systems have been recognized as a standard in scientific and commercial applications. Work on join operator, because of its high evaluation cost, is the primary purpose of relational query optimizers. If queries be in the conversational manner, they will include fewer relations and we can do the optimization of these expressions by an exhaustive search. However, in case the number of relations is more than five or six relations, exhaustive search techniques will bear high cost regarding the memory and time. Queries with lots of joins are seen in new systems like deductive database management systems, expert systems, engineering database management systems (CAD/CAM), Decision Support Systems, Data mining, and scientific database management systems and etc. Whatever the reason, database management systems need the use of query optimizing techniques with low cost in order to counteract with such complicated queries. A group of algorithms which are searching for suitable order for performing the join ordering problem are exact algorithms that search all of state space and sometimes they reduce this space by heuristic methods [7]. One of these algorithms is dynamic programming method which at first introduced by Selinger et al [2, 9] for optimizing the join ordering in System-R. The most important disadvantage of this algorithm is that increasing the number of relations in queries needs much use of memory and processor. We can imply other exact algorithms like minimum selectivity algorithm [7], KBZ algorithm [10] and AB algorithm [11].Other algorithms named random algorithms have been introduced for showing the inability of exact algorithms versus large queries. Introduced algorithms in this field are iterative improvement [5, 6, 12], simulated annealing [5, 12, 13, 14], two-phase optimization [12], toured simulated annealing [8] and random sampling [15].According the nature of evolutionary algorithms and considering this matter that they are resistant and efficient in most of the cases and by considering the works done in this field, the most suitable choice for solving this problem is use of evolutionary algorithms. The first work in optimizing the join ordering problem has been done by Bennet et al [3]. In general, the algorithm used by them bears low cost in comparison with dynamic programming algorithm used for System-R. Other features of this algorithm are the capability to use in parallel architecture. Some other works have been done by Steinbrunn et al [7] that they have used different coding methods and genetic operators. Another sample of evolutionary algorithms used for

solving the join ordering problems is genetic programming which is introduced by Stillger et al [16]. CGO genetic optimizer has also been introduced by Mulero et al [17].

In this paper, a hybrid evolutionary algorithm has been proposed for solving the optimization of Join ordering problem in database queries. 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 speed of finding an answer has been accelerated and prevented from getting stuck in local minimums. The results of experiments show that hybrid algorithm has dominance over the methods of genetic algorithm and learning automata. The paper has been organized as follows: The second part defines join ordering problem in database queries. Part three provides brief account of learning automata and genetic algorithms. Part 4 explains the proposed hybrid algorithm and on basis of data analysis, part 5 provides an account of analysis, in other words it deals with results. The conclusion and implication will be presented in part 6.

2 The definition of problem

Query optimization is an action that during which an efficient query execution plan (qep) is provided and is one of the basic stages in query processing. At this stage, database management system selects the best plan from among execution plans, in a way that query execution bears the low cost, especially the cost of input/output operations. Optimizer's input 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 qep0 which belongs to S set, so that [3]:

$$cost(qep_0) = min_{qep \in S} cost(qep)$$

The execution plan for responding to query is the sequel of algebra relational operators applied on the database relations, and produces the necessary response to that query. Among the present relational operators, processing and optimizing the join operators which are displayed by symbol ∞ are difficult operations. Basically, join operator considers two relations as input and combines their tuples one by one on the basis of a definite criterion and produce a new relation as output. Since the join operator has associative and commutative features, the number of execution plans for responding to a query increases exponentially when the number of joins among relations increases. Moreover, database management system usually supports different methods of implementing a join operator for join processing and various kinds of indexes for achieving to relations, so that, these cases increase the necessary choices for responding to a query. Although all of the present execution plans for responding to a definite query, have similar outputs, but while generated inter-relation's cardinality aren't similar, thus generated execution plans have different costs. Thus 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 NP-hard problem [4].

3 Learning automata and genetic algorithms

Learning automata approach for learning involves determination of an optimal action from a set of allowable actions. It selects an action from its finite set of actions. The selected action serves as input to the environment which in turn emits a stochastic response from allowable responses. Statistically, environment response is dependent to automata action. Environment term includes a set of all external conditions and their effects on automata operation. For more information about learning automata, you can refer [18]. Learning automata have various applications such as: routing in communicative networks, image data compression, pattern recognition, processes scheduling in computer networks, queuing theory, accessing control on non-synchronic transmitting networks, assisting the instruction of neural networks, object partitioning and finding optimal structure for neural networks. For the query with n join operator, there are n! Execution plans. If we use learning automata for finding optimal execution plan, automata will have n! Actions, the large number of actions reduces speed of convergence in automata. For this reason, object migration automata proposed by Oomen and Ma [18]. Genetic algorithms operate on basis of evolution idea and search optimal solution from among a large number of potential solutions in populations. In each generation, the best of them is selected and after mating, produces new childes. In this process, suitable people will remain with greater possibility for next generation [9].

Proposed hybrid algorithm for solving join ordering problem

by combining genetic algorithms and learning automata and integrating gen, chromosome, actions and depth concepts, historical track of problem solving evolution is extracted efficiency and used in the search process. The major feature of hybrid algorithm is it's resistance versus nominal changes of responses. In other words, there is a flexible balance between efficiency of genetic algorithm, resistance of learning automata in hybrid algorithm. Generation, penalty and reward are some of features of hybrid algorithm. It has been explained in the following basic parameters of this algorithm.

Gen and Chromosome: in proposed algorithm, unlike classical genetic algorithm, binary coding or natural permutation representations aren't used for chromosomes. Each chromosome is represented by learning automata of object migration kind, so that each of gens in chromosome is attributed to one of automata actions, and is placed in a definite depth of that action. In these automata, $\underline{\alpha}=\{\alpha_1, \alpha_2, \dots, \alpha_k\}$ is set of allowed actions of automata. These automata have k actions (the number of actions of these automata equals with the number of join operators). If number u join from given query is placed in the m number action, in this case, number u join will be mth join of query will be executed. $\underline{\phi}=\{\phi_1, \phi_2, \dots, \phi_{KN}\}$ Is set of states and N is memory depth for automata. The set of automata states are divided to k subsets $\{\phi_1, \phi_2, \dots, \phi_N\}$, $\{\phi_{N+1}, \phi_{N+2}, \dots, \phi_{2N}\}$, …, $\{\phi_{(K-1)N+1}, \phi_{(K-1)N+2}, \dots, \phi_{KN}\}$, and join operators are classified on the basis of this matter that in which state they are. If number u join of query is placed in the set $\{\phi_{(j-1)N+1}, \phi_{(j-2)N+1}, \dots, \phi_{jN}\}$, in this case, number u join will be nth join that is executed. In a set of states of j action, $\phi_{(j-1)N+1}$ is called internal state and is called boundary state. The join which has located in $\phi_{(j-1)N+1}$ is called more certainty and the join which has located in ϕ_{jN} is called less certainty. The join state is changed as a result of giving reward or penalty, and after producing several automata generations with genetic algorithm, we can achieve optimal permutation, and this is the best choice for the problem. If a join is located in boundary state of an action, giving penalty causes a change in the action that join is located on it, and causes new permutations. Now consider the following query:

(A∞C) and (B∞C) and (C∞D) and (D∞E)

Each join operator has a clause for joining that is omitted for the simplicity of display, determines which tuples of joned relations emerge in the result relation. The above query is represented as graph in figure1. Capital letters are used for showing relations and Pi is used for show join operator.

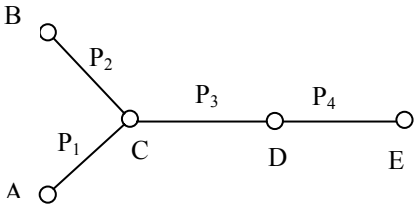


Fig 1. An example of a query graph

We consider a set of join operators like p1, p2, p3, p4 to show permutation of join operator executions with object migration automata, based on Tsetlin automata. This automata has four actions {a1,a2,a3,a4} (the same number of query joins) and depth 5. A set of states like {1,6,11,16,21,26} are internal status and a set of states like {5,10,15,20,25,30} are boundary states of learning automata. At first, each of query joins is in the boundary state of related action. In hybrid algorithm, each of chromosome's gen equals with one automata action, so we can use these two words instead of each other. Learning automata (chromosome) has four actions (gen), and each action has five internal states. Suppose in the first permutation, the order of join operator executions will be join 3, join 2, join 1 and join 4 respectively. The way of representing execution order by object migration automata is shown in figure 2. This matter is done on the basis of Tsetlin automata. At first, each of these joins is in the boundary state of related action.

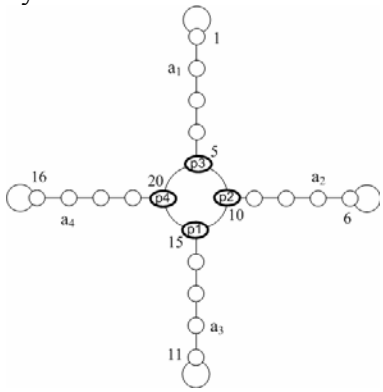


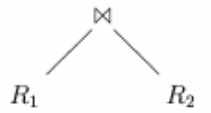
Fig 2. Display of joins permutation (p_3, p_2, p_1, p_4) by learning automata based on Tsetlin automata connections.

Fitness function: In genetic algorithms, fitness function is the feature for surviving chromosomes. The purpose of searching the optimized order of query joins is finding permutation of join operators, so that total cost of query execution is minimized in this permutation. One important point in computing fitness function is the number of references to the

disc. So, we can define the fitness function of F for an execution plan like qep as follows:

$$F(qep) = \frac{1}{\text{The number of references to disk}}$$

For computing the number of references to disc (the cost) of an execution plan, consider execution plan as a processing tree. The cost of each node as recursive form (down to up and right to left) is computing by adding the total number of achieved costs of the two child nodes, and needed cost for joining them in order to achieve the final result [7]. For example, we consider join R1∞R2:



In this case, the cost of evaluation equals to following amounts:

$$C_{total} = C(R_1) + C(R_2) + C(R_1 \infty R_2)$$

In which, $C(R_k)$ is the cost of achieving to child node, and is computed as follows:

$$C(R_k) := \begin{cases} 0 & \text{if } R_k \text{ is a base relation} \\ C(R_i \bowtie R_j) & \text{if } R_k = R_i \bowtie R_j \text{ is an intermediate result} \end{cases}$$

In a special manner, if R1 and R2 are both the base relations, the cost C_{total} equals with $C(R_1 \infty R_2)$ and according to this matter that we have used nested loops for join operator implementation, so the computation cost of join operator will be same as $C(R_1 \infty R_2) = b_{R_1} + b_{R_2}$ and b_{R_k} is numbers of blocks of R_k relation

Operators: Considering this that in hybrid algorithm, each chromosome is represented as learning automata; crossover and mutation operators are different from similar operators in simple genetic algorithms.

Selection operator: The selection used for this algorithm is roulette wheel.

Crossover Operator: In this operator, two parent chromosomes are selected and two gens i and j are selected randomly in one of the two parent chromosomes. Then these genes are selected in another parent. A set of gens with numbers between i and j are called crossover set. Then gens which have same numbers replace with each other in two crossover set (for example, i gen from first crossover set replaces with the i gen from the second crossover set. i+1 gen from first crossover set replaces with the i+1 gen from the second crossover set, and so on). Two new chromosomes are produced in this way which are so-called the children of two automata parents. In continue, the pseudo code of this operator is represented in figure 3. Since, n chromosome (automata) is used in this algorithm, and each automata has its own special characteristics (states, action and object like each action), we represent these features by automata name and point separator for more readability of pseudo code.

For example, for showing the join state of u from i automata, LA_i.State(u) has been used. In the aforementioned algorithm, $cost_i$(LA1) is the cost (the number of reference to disc) for joining the ith action of first learning automata.

```
Procedure Crossover ( LA1, LA2 )
 Generate two random numbers r1 and r2 between 1 to n
 r1 = Random *n; r2 = Random *n;
 r1 = Min(r1, r2 ), r2 = Max(r1, r2 )
 For i = r1 to r2 do
  If (cost_i( LA1) < cost_i( LA2) ) then
    j = Action of LA2 where
    LA2.Object(LA2.Action( j )) =LA1.Object(LA1.Action( i ));
Swap(LA2.Object(LA2.Action(i)),LA2.Object(LA2.Action(j)));
  Else
    j = Action of LA1 where
    LA1.Object(LA1.Action( j ))=LA2.Object(LA2.Action( i ));
Swap(LA1.Object(LA1.Action(i)),LA1.Object(LA1.Action(j)));
  Endif
 End For
End Crossover
```

Fig 3. Pseudo code of crossover operator

Mutation operator: For executing this operator, we can use different method which are suitable for work with permutations. For example in swap mutation, two actions (gens) from one automata (chromosome) are selected randomly and replaced with each other.The pseudo code of this operator is represented in figure 4.

```
Procedure Mutation (LA)
      i = Random *n; j = Random *n;

Swap(LA.Object(LA.Action(i)),LA.Object(LA.Action(j)));
End Mutation
```

Fig 4. Pseudo code of mutation operator.

Penalty and Reward operator

In each chromosome, evaluating the fitness rate of a gen which is selected randomaly, penalty or reward is given to that gen. as a result of giving penalty or reward, the depth of gen changes. Figure 5 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
```

Fig 5. Pseudo code of reward operator.

Pseudo code of penalty operator is shown in figure6. The manner of join move occurs in two different ways and is shown below.

a) The join be in a state except boundary state: giving penalty reduces the certainty of the join.

b) The join be in a boundary state: in this manner, we find a join from query, so that if we change place of two join in execution plan, the cost will decrease more. In this case, if found join be in a boundary state, the place of two joins is replaced, otherwise, at first, the determined join is transmitted into its boundary state, and then replacement takes place.

```
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 join appears in the boundary state
  bestcost = ∞ ;
  for U = 1 to n do
    Create QEP LA′ from LA by swapping u and U
    If costi( LA′) < bestError then
       bestcost = costi( LA′);
       bestjoin = U;
    End If
  End for
  LA.State(bestjoin) = LA.Action( bestjoin)*N;
  LA.State(u) = LA.Action(u)*N;
  Swap(LA.State(u),LA.State(bestjoin));
End Penalize
```

Fig 6. Pseudo code of penalty operator

The Pseudo code of hybrid algorithm is shown in figure 7

```
Function JoinOrdering(Query)
  //n=Number of Joins
  Create the initial population LA1 … LAn;
  EvalFitness();
  While ( Not (StopCondition)) do
    NewLA1=NewLA2= LA with minimum
Value of Cost;
    For i = 2 to n do
     Select LA1 ; Select LA2 ;
    If ( Random > 0.9 ) then Crossover (
LA1, LA2 );End If
    If (Random > 0.6 ) then
       Mutation ( LA1 ); Mutation ( LA2 );
    End If
     NewLAi+1 = LA1; NewLAi+2 = LA2 ;
i=i+2;
   End For
   For i = 0 to n do
     LAi = NewLAi;  u = Random *n;
    If (costu( LAi) <MeanCost ) then
Reward(LAi , u );
     Else Penalize(LAi , u );
     End If
   End For
   EvalFitness();
  End While
End JoinOrdering
```

Fig 7. The Pseudo code of hybrid algorithm for solving join ordering problem in database queries.

4 Experiment results

In this part, experiment results done by learning automata (LA) and genetic (GA) and hybrid (GALA) algorithms are presented. The results show that hybrid algorithm has dominance over the methods of learning automata and genetic algorithm. Figure 8 show the results of hybrid algorithm based on Tsetlin automata in comparison with learning automata and genetic algorithm. In the following diagrams, the vertical axis shows the average cost of executing plans that resulted by each algorithm and horizontal axis shows the number of joins in queries. The experiment results show that hybrid algorithm has better and suitable results in comparison with genetic algorithm, and the cost of executing plans with hybrid algorithm for a definite number of joins is less than genetic algorithm, but the results of hybrid algorithm based on Tsetlin automata in comparison with learning automata don't have appropriate improvement, and mostly results of these two algorithm are close to each other and sometimes hybrid algorithm had better efficiency. figures 9, 10 and 11 respectively show the results of hybrid algorithm based on Krinisky, Krylov and Ommen automata in comparison with genetic algorithm and learning automata. The experiments show that the dominance of hybrid algorithm over other methods, in other words the cost of execution plans obtained by hybrid algorithm based on Krinsky and Krylov and Oomen automata has been less than other algorithms in all states.

Comparing the results of hybrid algorithms with each other show the dominance of hybrid algorithm based on Krinsky automata over other. This matter is shown figure 12. So, we can say that from among algorithms, hybrid algorithm based on Krinsky is the most suitable way for solving join ordering problem in database queries.

5 Conclusion

In this paper, a hybrid evolutionary algorithm has been proposed for solving the optimization of join ordering problem in database queries. 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 speed of finding an answer has been accelerated and prevented from getting stuck in local minimums. The results of experiments show that hybrid algorithm has dominance over the methods of genetic algorithm and learning automata.

References:

[1] E.F. Codd, "A relational model of data for large shared data banks", CACM, 13(6): pages 377-387, 1970.

[2] P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price, "Access path selection in a relational database management system", In Proc. Of the ACM SIGMOD Conf. on management of Data, pages 23-34, Boston, USA, 1979.
[3] 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, pages 400-407, San Diego, USA, 1991.
[4] T. Ibaraki and T. Kameda, "Optimal nesting for computing N-relational joins", ACM Trans. on Database Systems, 9(3): pages 482-502, 1984.
[5] A. Swami and A. Gupta, "Optimization of large join queries", In Proc. Of the ACM SIGMOD Conf. on Management of Data, pages 8-17, Chicago, IL, USA, 1988.
[6] A.Swami, "Optimization of large join queries: Combining heuristics and combinational techniques", In Proc. Of the ACM SIGMOD Conf. on Management of Data, pages 367-376, Portland, OR, USA, 1989.
[7] M. Steinbrunn, G. Moerkotte, and A. Kemper, "Heuristic and randomized optimization for the join ordering problem", VLDB Journal: Very Large Data Bases, 6(3): pages 191-208, 1997.
[8] R. Lanzelotte, P. Valduriez, and M. Zait, "On the effectiveness of optimization search strategies for parallel execution spaces", In Proc. Of the Conf. on Very Large Data Bases (VLDB), pages 493-504, Dublin, Ireland, 1993.
[9] M. M. Astrahan et al, "System R: A relational approach to data management." ACM Trans. on Database Systems, 1(2) pages 97-137, 1976.
[10] R. Krishnamurthy, H. Boral, and C. Zaniolo, "Optimization of nonrecursive queries.", In Proc. of the Conf. on Very Large Data Bases (VLDB), pages 128-137, Kyoto, Japan, 1986.
[11] A. Swami and B. Iyer, "A polynomial time algorithm for optimizing join queries.", In Proc. IEEE Conf. on Data Engineering, pages 345-354, Vienna, Austria, 1993.
[12] Y. E. Ioannidis and Y. C. Kang, "Randomized algorithms for optimizing large join queries", In Proc. Of the ACM SIGMOD Conf. on Management of Data, pages 312-321, Atlantic City, USA, 1990.
[13] Y. Ioannidis and E. Wong, "Query optimization by simulated annealing", In Proc. Of ACM SIGMOD Conf. on the Management of Data, pages 9-22, San Francisco, CA, 1987.
[14] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing", Science, 220(4598): pages 671-680, 1983.
[15] C. Galindo-Legaria, A. Pellenkoft, and M. Kersten, "Fast, randomized join-order selection why use transformations", In Proc. Of the 20th Int. Conf. on Very Large Data Bases (VLDB), pages 85-95, Santiago, Chile, 1994.
[16] M. Stillger and M. Spiliopoulou, "Genetic programming in database query optimization", In Proc. Of the First Annual Conf. on Genetic Programming, pages 388-393, Stanford University, CA, USA, 1996.
[17] V. Muntes-Mulero, J. Aguilar-Saborit, C. Zuzarte, and J.-L. Larriba-Pey, "Cgo: a sound genetic optimizer for cyclic query graphs", In Proc. Of ICCS 2006, pages156-163, Reading, Springer-Verlag, UK, 2006.
[18] H. Beigy and M. R. Meybodi, "Randomized Las Vegas Algorithm for Graph Isomorphism", Proceedings of Third International Conference on Intelligent Data Engineering and Automated Learning, Manchester, UK, Aug.12-14, 2002.
[19] Y. Wang and, K. "Fan Genetic-Basic Search for Error-Correcting Graph Isomorphism", IEEE Transaction on Systems, Man. And Cybernetics-Par`t B: Cybernetics, Vol. 27, No. 4, August 1997.

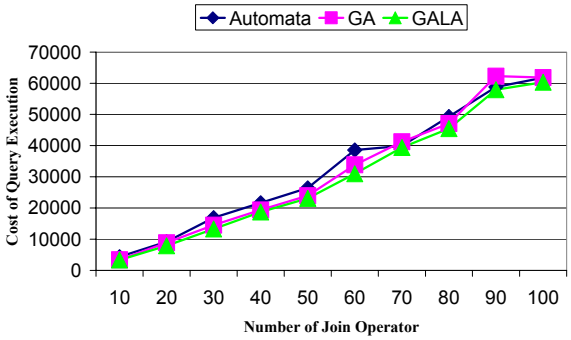


Fig 8: Comparison of averaged cost obtained from hybrid algoritm and learning automata based on Tsetline and genetic algorithm.

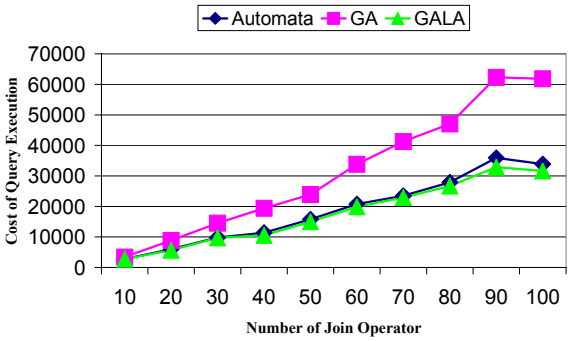


Fig 9: Comparison of averaged cost obtained from hybrid algoritm and learning automata based on Krinsky and genetic algorithm.

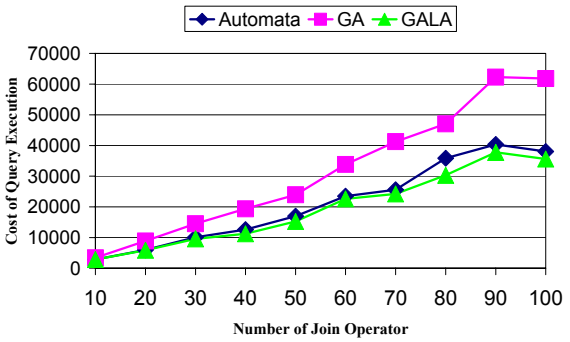


Fig 10: Comparison of averaged cost obtained from hybrid algoritm and learning automata based on Krylov and genetic algorithm.

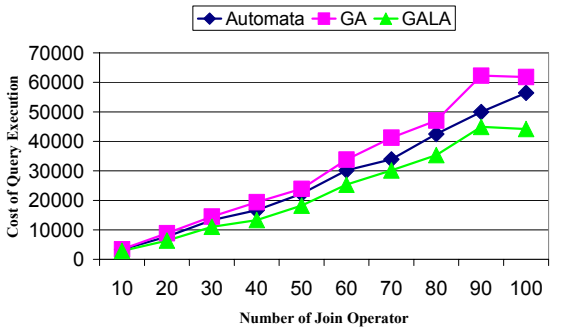


Fig 11: Comparison of averaged cost obtained from hybrid algoritm and learning automata based on Oomen and genetic algorithm.

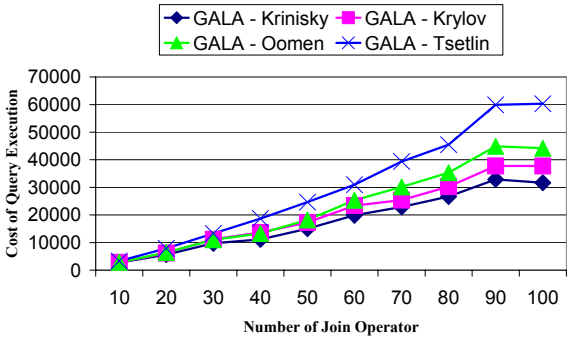


Fig 12: Comparison of averaged cost obtained from hybrid algoritms based on different Automata.