A Hybrid Method for Solving Traveling Salesman Problem

Bager Zarei
Dept. of Computer Engineering,
Islamic Azad University Branch of Shabestar, Iran
Zarei_Bager@yahoo.com

M.R. Meybodi
Dept. of Computer Engineering & Information Technology,
Amirkabir University, Tehran, Iran
MMeybodi@aut.ac.ir

Abstract—**One of the important problems in graphs theory is TSP. Both learning automata and genetic algorithms are search tools which are used for solving many NP-Complete problems. In this paper a hybrid algorithm is proposed to solve TSP. This algorithm uses both GA and LA simultaneously to search in state space. It has been shown that the speed of finding answer increases remarkably using LA and GA simultaneously in search process, and it also prevents algorithm from being trapped in local minimums. Experimental results show that superiority of hybrid algorithm over LA and GA**.

Keywords: Traveling Salesman Problem (TSP), Learning Automata (LA), Genetic Algorithm (GA)

1. Introduction

Graphs are powerful tools used widely in various applications. One of important problems in graphs theory is TSP. Many of general applications such as sonet network rings design, power cables and airplanes path, vehicles routing and etc can be modeled by TSP.

TSP is generalization of famous Hamiltonian Cycle. General form of this problem was proposed by Karl Menger for the first time in 1930, and later was promoted by Hassler Whitney and Merrill Flood. Suppose that we have a complete graph in which every edge $(u,v) \in V$ has non-negative cost $C(u,v)$. In this problem, graph vertices are equivalent to cities, graph edges are equivalent to path between cities and cost of edges are equivalent to length of path between cities. The salesman should start with an origin and visit all the cities once and return to the origin in a way that it minimized the total cost of tour. If the graph is symmetric the problem is called symmetric TSP and if the graph is asymmetric the problem is called asymmetric TSP.

One method for solving this problem is fining all the possible permutations of visiting order of cities and calculating the cost of each; the least costly permutation is the solution. The number of possible permutations for a graph with n node is $n!$. Since there is n node in every permutation, the required time for creating of permutation and calculate their length (cost) equals $n \times n!$. Instead of using deterministic method heuristic methods such as greedy algorithm, not visited nearest neighbor algorithm and minimum spanning tree can be used. These algorithms can lead to a desirable solution in a reasonable time, but in these algorithms the located tour are not very appropriate. For instance in greedy algorithm and not visited nearest neighbor algorithm the last added edge has usually more length.

Both learning automata and genetic algorithms are search tools which are used for solving many NP-Complete problems such as objects classifying, graph partitioning, keyboard optimization, finding optimal structures for neural networks and etc. In this paper a hybrid algorithm is proposed to solve TSP. This algorithm uses both GA and LA simultaneously to search in state space. It has been shown that the speed of finding answer increases remarkably using LA and GA simultaneously in search process, and it also prevents algorithm from being trapped in local minimums. Experimental results show that superiority of hybrid algorithm over LA and GA.

This paper has been organized as following. Section 2 explains TSP. A brief explanation of GA and LA has been included in section 3 and 4. Section 5 is devoted to hybrid algorithm that is used for solving TSP. In section 6 experimental results are shown and section 7 includes conclusion. References are in section 8.

2. Definition of the problem

A weighted graph is show in triad $G = (V, E, \alpha)$ in which V is the non-empty set of nodes, $E \subset V \times V$ is the set of edges and $\alpha : V \to R_V$ is a function of V to R_V that R_V is set of edges weight.

In some problems, the increase of their dimension results in the exponential increase of the time required to solve them. These problems are combinational optimization problems that their solution time is not polynomial. TSP is one of them in which solving it not only means finding the best tour in comparison to the previously known tours, but also proving the fact that there is no tour with less cost than the found tour.

Regarding the fact that TSP is a combinational optimization problem, the purpose is finding permutation of nodes, so that the considered permutation cost is the least, or in other words the specified tour length is the least.

3. Genetic Algorithms

Genetic Algorithms which act on the basis of evaluation in nature search for the final solution among a population of potential solution. In every generation the fittest of that generation selected and after reproduction produce a new set of children. In this process the fittest individuals will survive more probably to the next generations.

At the beginning of algorithm a number of individuals (initial population) are created randomly and the fitness function is evaluated for all of them. If we do not reach to the optimal answer, the next generation is produced with selection of parents based on their fitness and the children mutate with a fixed probability then the new children fitness

is calculated and new population is formed by substitution of children with parents and this process is repeated until the conclusion condition is established.
The most advantages of this algorithm compared with common methods are: parallel search instead of serial search, not requiring any additional information such as problem solving method, in-deterministic of algorithm, easy implementation and reaching to several choices.
GA uses several operators, each of which have different types and can be implemented using different methods.

4. Learning automata

Learning in LA is choosing an optimal action from a series of allowable automata actions. This action is applied on a random environment and the environment gives a random answer to this action of automata from a series of allowable answers. The environment's answer depends statistically on automata action. The term environment includes a set of outside conditions and their effect on automata operation. Connection of an automata with the environment is shown in the following figure**.**

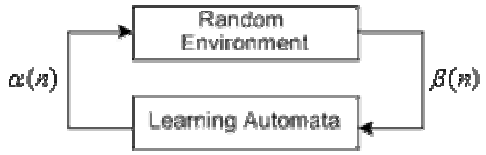


Figure 1.Connection of LA with random environment

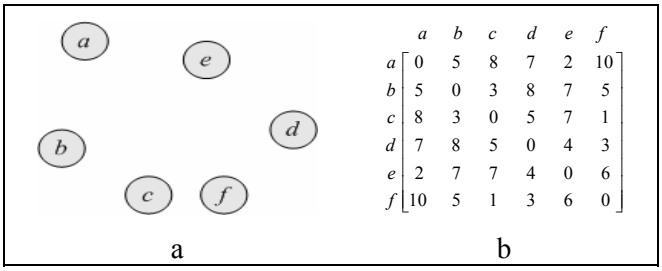
LA has many applications. Some of them are: routing in communication networks image compressing, pattern recognization, process scheduling in a computer network, queuing theory, access control in asynchronous transfer networks, help in teaching neural network, objects classifying and finding the optimal structure for neural networks.
For a graph with n nodes there are n! permutations of nodes and if LAs used to solve TSP, automata should have n! actions. Large number of actions reduces the convergence speed of automata. For this reason object migrating automata was proposed by Oomenn and Ma.

5. Hybrid searching algorithm to solve TSP

With the combination of the GA and LA, and also gene, chromosome, action and depth, the historic record of the evolutionary of problem's solution was extracted effectively and used in the search process.
Resisting against the superficial changes of answers is the most important characteristic of hybrid algorithm. In other words, there is a flexible balance between the effectiveness of GA and stability of learning automata in hybrid algorithm. Self recovery, reproduction, penalty and reward (guidance) are characteristic hybrid algorithm. In the following section, the main parameters of this algorithm will be explained.

Gene and chromosome:
Unlike classic GA, the proposed algorithm does not use binary coding for chromosomes. Each chromosome is shown by a learning automata of the object migrating type in a way that each gene in a chromosome is attributed to an automata's actions and is placed at a certain depth of that action.

In this automata $\underline{\alpha} = \{\alpha_1,...,\alpha_k\}$ is the set of allowable actions for the LA. This automata has *k* action (action numbers of this automata is equal to the graph nodes). If node *u* from graph is placed at action *m*, node *u* will be the m^{th} city in visiting order.
$\underline{\phi} = \{\phi_1, \phi_2, ..., \phi_{KN}\}$ is the set of status and *N* is the depth of memory for automata. The status set of this automata is portioned to the *k* subset $\{\phi_1, \phi_2, ..., \phi_N\}$, $\{\phi_{N+1}, \phi_{N+2}, ..., \phi_{2N}\}$,... and $\{\phi_{(K-1)N+1}, \phi_{(K-1)N+2}, ..., \phi_{KN}\}$, and graph nodes is classified on the basis of its status. If node *u* from graph is in the status set $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, ..., \phi_{jN}\}$, the node *u* will be j^{th} city in visiting order. In the status set of action *j*, state $\phi_{(j-1)N+1}$ is called inside state and state ϕ_{jN} is called border state. The node placed at state $\phi_{(N-1)N+1}$ is the most significant node and the node placed at state ϕ_{jN} is the least significant node.
As the result of rewarding or penalizing an action, the status of dependant node to it will be change. If a node is placed at the border status of an action, panelizing it lead to changing of action that node depend it, and this changing lead to creation a new permutation. For instance, consider complete graph in figure 2.a that includes 6 nodes. Figure 2.b shows the adjacency matrix of graph.



	a	b	c	d	e	f
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

Figure 2.Complete graph with 6 nodes

Consider permutation $< c, b, f, d, a, e >$ from graph figure 2. This permutation has been shown by a LA with similar connection to Tsetline automata in figure 3. This automata has 6 actions $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6\}$ (equal to number of graph nodes) and depth of 5. status set {1,6,11,16,21,26} are inside status and status set {5,10,15,20,25,30} are border status of automata. At the beginning each graph nodes is placed at the border status of each relative action. In hybrid algorithm each gene of chromosome equals to an automata's action, so we can use this terms interchangeably. This LA (chromosome) has 6 actions (gene) and each action has 5 inside state.

Initial population:
Supposing that the number of population's member is *n*. *n-1* population member can be produced by creating *n-1* random permutation. To produce the last member of the population

we use the not visited nearest neighbor method. We call this permutation approximation permutation. The last added member to the population has the most similarity with the final answer.

As an example, the formation of initial population for graph in figure 2 with the assumption $n=6$ is explained as following. The first 5 member of the population are created by these $<b,d,e,a,f,c>$, $<d,e,f,b,c,a>$, $<e,f,b,d,a,c>$, $<c,f,b,e,d,a>$, and $<b,d,c,a,e,f>$ random permutations. To create the 6th permutation not visited nearest neighbor method is used. Assuming that the start node is a, the 6th permutation is $<a,e,d,f,c,b>$. The initial population resulting from graph in figure 2 has been shown in figure 4. At the beginning each node is at the border status of its action.

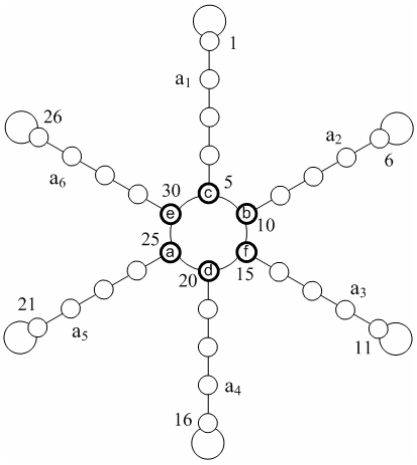


Figure 3.Showing permutation $<c,b,f,d,a,e>$ by a LA with similar connection to Tsetline Automata

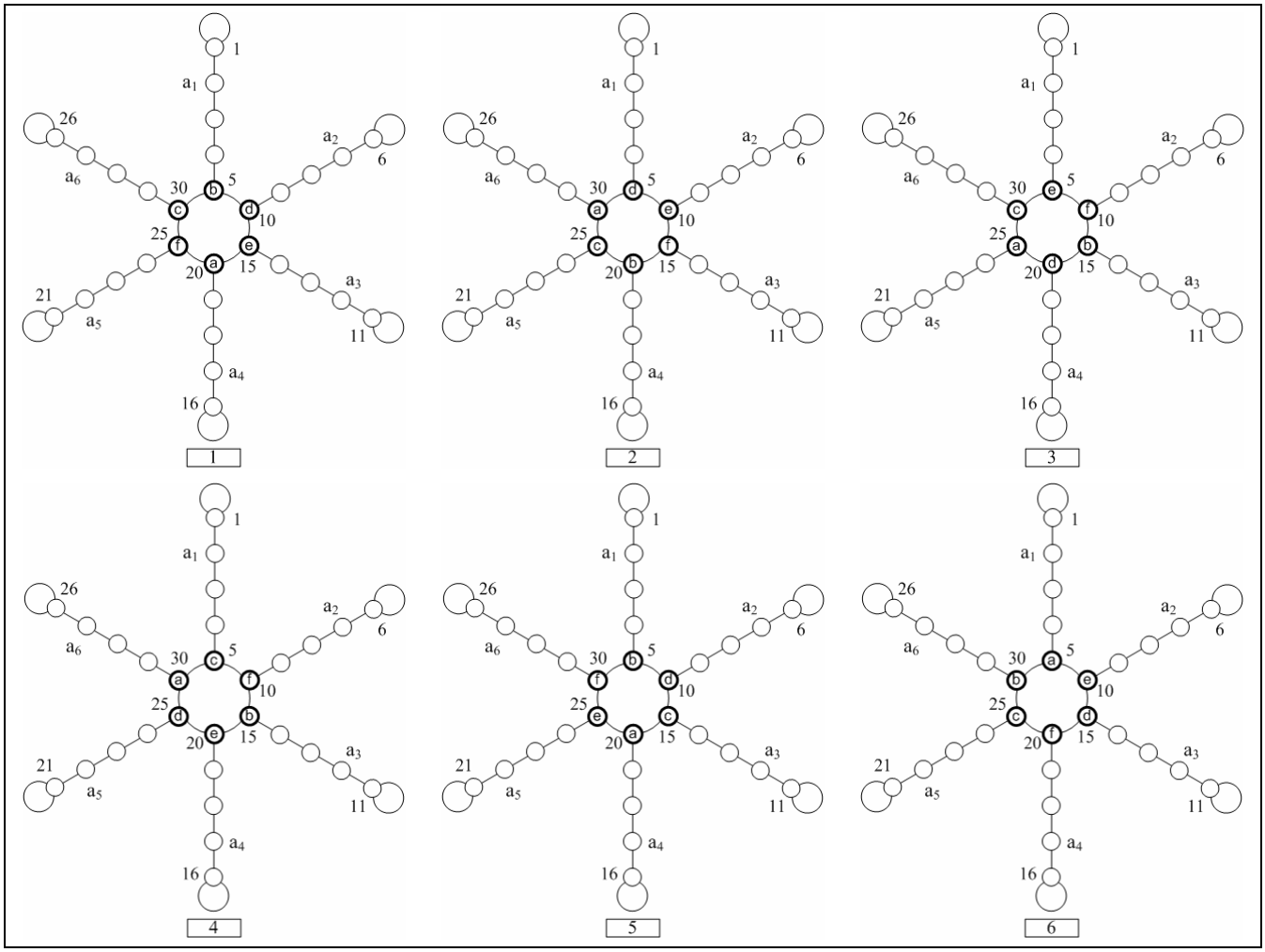


Figure4.Initial population for grapg in figure 2

Fitness function:

In GAs fitness function is indicator of chromosomes survive. In TSP the purpose is finding a permutation (tour) such as σ in which the cost is minimum. So that fitness function $f(\sigma)$ in TSP is defined as following. The fitness function pseudo code is displayed at figure 5.

Procedure EvalFitness()
Begin
  for i = 1 **to** n **do**
    $f(LA_i) = 1 / Lenghtof\ Specified\ Tour\ By\ LA_i$
End Procedure

Figure 5.Fitness function pseudo-code

Operators:

Since in hybrid algorithm, the chromosome is shown in the form of LA, crossover and mutation operators are not similar to genetic traditional operators.

A) Selection operator: In order to choose LAs (chromosomes) for crossover or mutation operators, one of these methods can be used: ranking selection, roulette wheel selection and tournament selection.

B) Crossover operator: To do this operator one of methods which is appropriate to work with permutations can be used: Partially Mapped Crossover, Ordered Crossover, Cycle Crossover, and New Crossover. In this paper only the proposed method namely New Crossover will be explained. In this method two parent chromosome are chosen and genes i, j are selected randomly in one of these chromosomes. Then these two genes are selected in another parent chromosome. We call the set of genes with number between i and j the crossover set. Then the genes with the same number are replaced with one another in two crossover set. (for example gene number i from the first crossover set with gene number i from the second crossover set, gene number i+1 from the first crossover set with gene number i+1 from the second crossover set, …). The result of this process is two chromosomes which are called the two parent automata's children.

In figure 6 this operator is shown. Since in this algorithm *n* automats (chromosomes) are used and each automaton has its exclusive characteristics (status, action and object appropriate for each action), we show these characteristics with the name of automata as prefix and point separator in order to have more readability for pseudo-code. For instance, for showing the position of node u from automat i LAi.state(u) has been used.

For instance LA_2 and LA_5 automatas from the previously formed population are selected randomly. Then with the random selection of two a_2 and a_3 places, crossover set $\{a_2,a_3\}$ achieved, and finally according to figure 7 two new chromosomes are created as the result of replacing appropriated actions at replacement distance.

```
Procedure Crossover (LA1, LA2)
Begin
    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 (Ji(LA1) < Ji(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)));
        end if
        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)));
        end else
    end for
End Procedure
// Ji(LAk) = (Weight of Edge (i-1,i) in LAk + Weight of Edge (i,i+1) in
LAk)/2 - (Length of  Specified Tour By LAk/n);
```

Figure 6.NewCrossover operator's pseudo-code

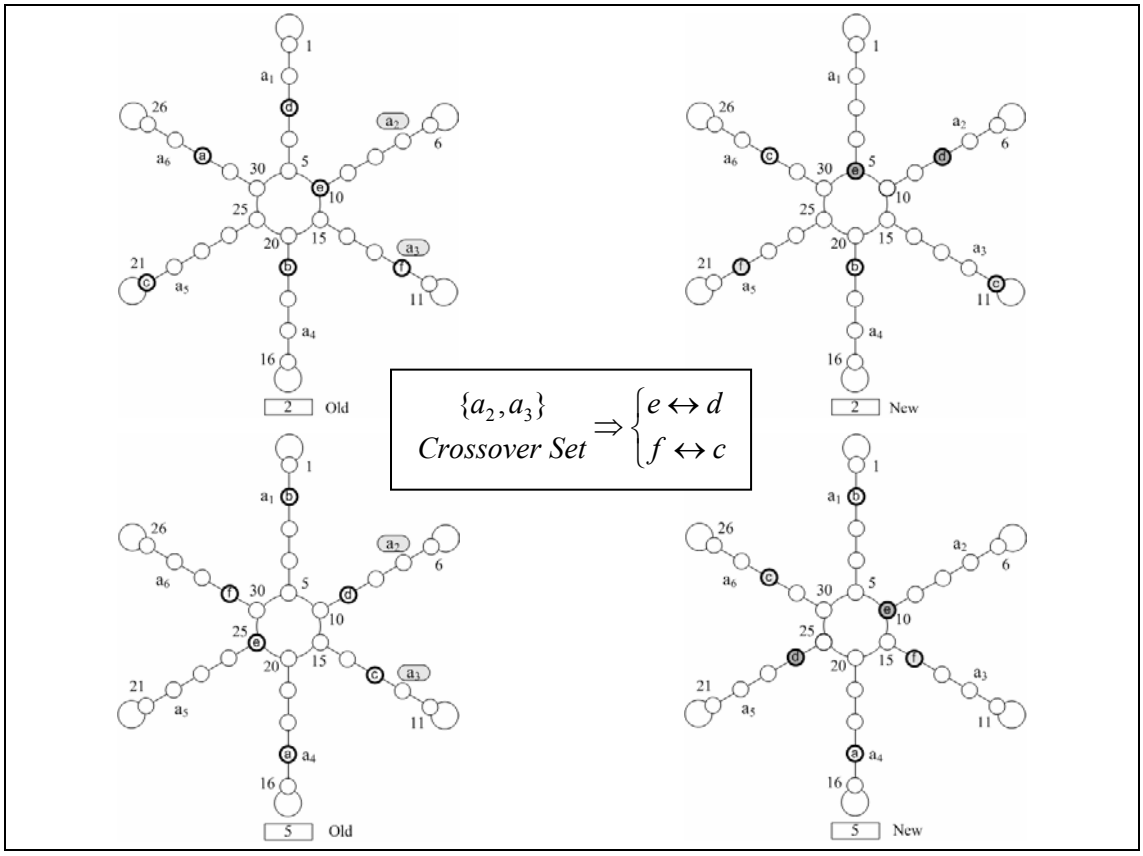


Figure 7.Doing NewCrossover operator

C) One of the methods: Swap Mutation, Insertion Mutation, Inversion Mutation, and Scramble Mutation, which are appropriate to work with permutation, can be used to do this operator. For instance in Swap Mutation two actions (gene) from an automata (chromosome) are selected randomly and replaced. In figure 8 this operator is shown.

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

Figure 8.Mutation operator's pseudo-code

E) Penalty and reward: Since each chromosome has been displayed as a LA, after examining fitness of a gene (node or action) which are selected randomly, that gene will be rewarded or penalized. State of a gene in the relative action status set will be changed as result of rewarding or penalizing. If a gene is placed at the border status of an action, finalizing it results in changing that gene's actions and consequently creating a new permutation. The rate of this operator should be low, because this operator is a random search operator, and if it is applied with high rate, it will reduce the effectiveness of algorithm. Reward and penalty operator varies according to LA type. Figure 9 shows *u* node reward operator of LA automata with connections similar to Tsetline automata.

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

Figure 9.Node *u* reward operator of LA automata with connections similar to Tsetline automata

For instance, in automata with connections similar to Tsetline automata if node *b* is at the status set {16,17,18,19,20}, and the average cost of incoming and outgoing edges to node *b* ((cost of incoming edge to *b* + cost of outgoing edge to *b*)/2) is less than threshold value (threshold value is determined comparatively and its value at any moment equals totals tour cost divided by node numbers) this node are rewarded and move to the more interior status of this action. If node *b* is at inside status (status number 16) and is rewarded, it will remain in the same status. The movement of this node is shown in figure 10.

If the fitness value of a node is more than threshold value, the resulting tour won't be appropriate and it will be penalizing. *u* action penalty operator of LA automata with connections similar to Tsetline automata is shown in figure 11.

.

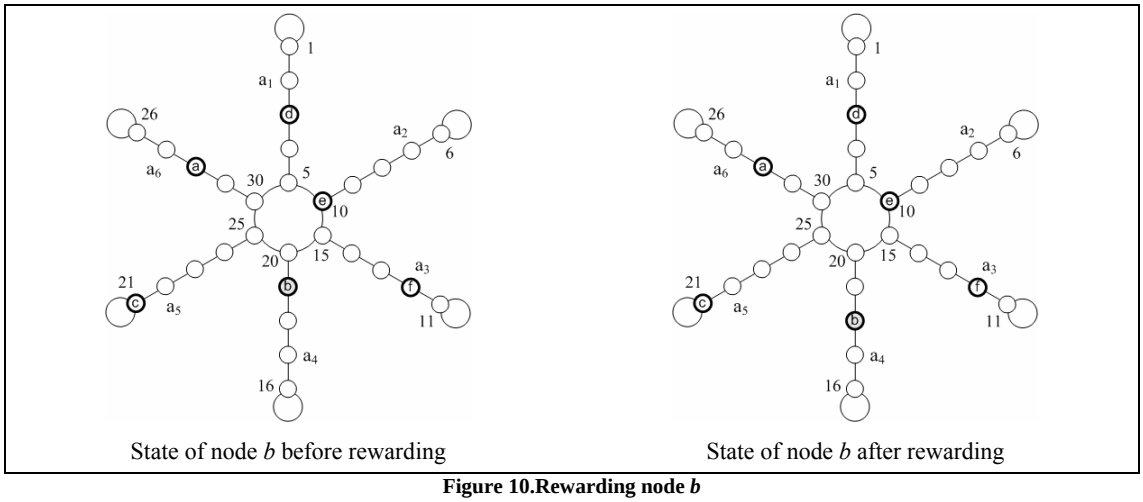


Figure 10.Rewarding node *b*

```
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 11.*u* node penalty operator of LA automata with connections similar to Tsetline automata

The movement of this node in two different ways as follows:

a) The node is at a status other than the border status: Panelizing this node reduces the importance of this node. Movement of this node is shown in figure 12.
b) The node is in border status: in this case we find a graph node in a way that if two nodes are replaced by each other in the relative permutation, we will have the maximum reduction in tour cost. In this case if the located node is in border status, two node are replaced by each other; otherwise, first the specified node will be returned to its border status action and then they will be replaced. The movement of this node is shown in figure 13.

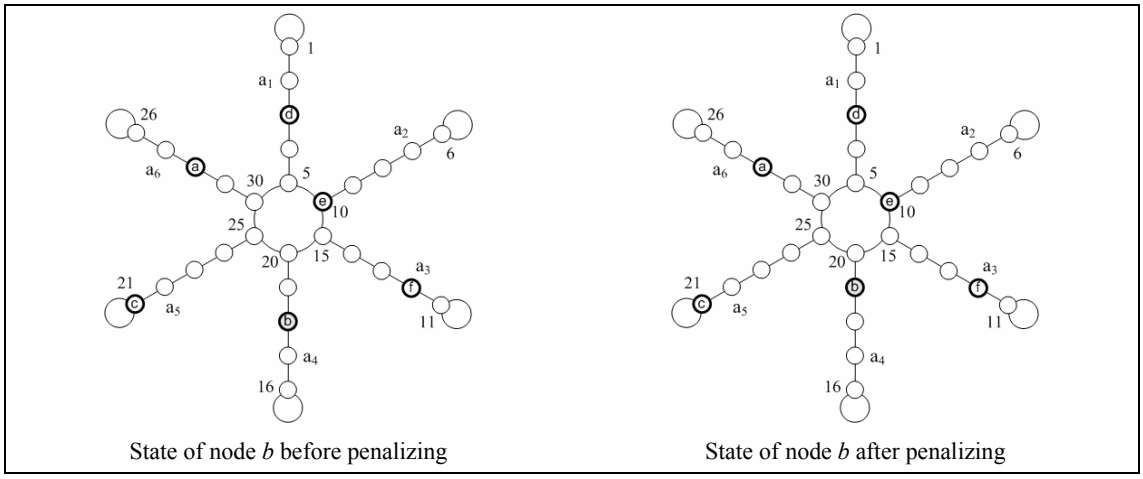


Figure 12.Penaliizng a node palced in a status other than border status

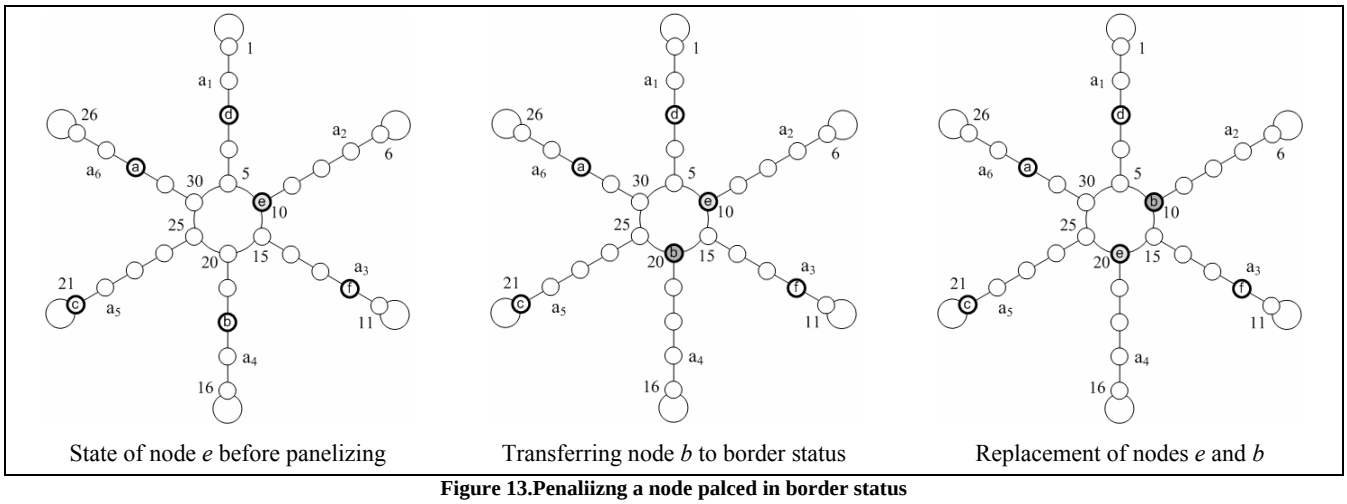


Figure 13.Penaliizng a node palced in border status

Hybrid algorithm pseudo-code is as follows.

```
Function TSP_Solver(G) : TSP_Tour
Begin
    n = Size of Population;    // n = |V_G|
    Create the initial population LA_1 … LA_n;
    EvalFitness();
    while( All (Length of Specified Tour By LA_i > Constant-Value) ) do
        NewLA_1 = NewLA_2 = LA with minimum Value of  Tour-Lenght;
        for i = 2 to n do
            Select LA1;  Select LA2 ;
            if (Random > 1 - CrossoverRate) then
                Crossover ( LA1, LA2 );
            if (Random > 1 - MutationRate) then
                Mutation ( LA1 ); Mutation ( LA2 );
            NewLA_{i+1} = LA_1;
            NewLA_{i+2} = LA_2 ;
            i=i+2;
        end for
        for i = 0 to n do
            LA_i = NewLA_i;
            u = Random *n;
            if ( J_u( LA_i) <threshold Threshold(LA_i )) then
                Reward(LA_i , u );
            else
                Penalize(LA_i , u );
        end for
        EvalFitness();
    end while
End Function
//Threshold(LA_i) = Lenght( Specified Tour by LA_i ) / |V_G|;
//J_u(LA_i) = (lenght of edge (u-1,u) in LA_i + lenght of edge (u,u+1) in LA_i) / 2;
```

Figure 14.Hybrid algorithm to solve TSP

6. Experimental results

In this section the Experimental results of TSP solver algorithms which was implemented on the basis LA, GA and hybrid algorithm are shown. These results indicate a remarkable improvement of hybrid algorithm compared with method base on LA and GA. In the experiments size of graphs was considered from 22 to 280 node and number of iterations from 50 to 500. In hybrid algorithm and LA depth of 1, 4,7,10 and 15 were put to test. Also in hybrid algorithm and GA mutation rate and mutation method was 25%, and swap mutation respectively, population size was equal to the number of graph nodes and ranking method chose for selecting chromosomes.

Tables 1 to 5 shows briefly a comparison of hybrid algorithm and other algorithms used to solve TSP. as the result indicate hybrid algorithm based on krylov automata with New Crossover combination method and crossover rate of 70 acts better than other algorithms and other crossover methods and rates. In diagram 1 to 7 result of tables 1 to 5 are shown graphically.

On the basis of following reasons the execution time of hybrid algorithm can't compare with other TSP solvers.

a) in some cases GA and LA may not be converged in finding a tour with the same length as the located tour by hybrid algorithm, and on the other hand, because these algorithms are random it is impossible to determine when these algorithms will converge and when they will diverge –if not converged, the average time for them will be limitless– while hybrid algorithm converges in most cases.

b) Greedy, not visited nearest neighbor and minimum spanning tree algorithms can't locate a tour with the same length located by hybrid algorithm. If the required settings are done in a way that hybrid algorithm becomes a tour that has the same length as these algorithms, then execution time of it will be shorter than these algorithms.
c) Hybrid algorithm, GA and LA are random algorithms and their execution time will be different each time.

Generally if the required settings are done for GA and LA in a way that their tour length –if convergent– is equal to that of hybrid algorithm, then their execution time will be loner than that of hybrid algorithm. Therefore in table 6 and diagram 8 only the required average time for hybrid algorithm based on Tsetline, Krylov, Krinsky and Oomenn automatas has been shown. As it can be seen, the required time for hybrid algorithm based on Krylov is the shortest of all.

To sum up, we can say that hybrid algorithm based on Krylov automata indicates better operations than other algorithms both in terms of execution time and tour length.

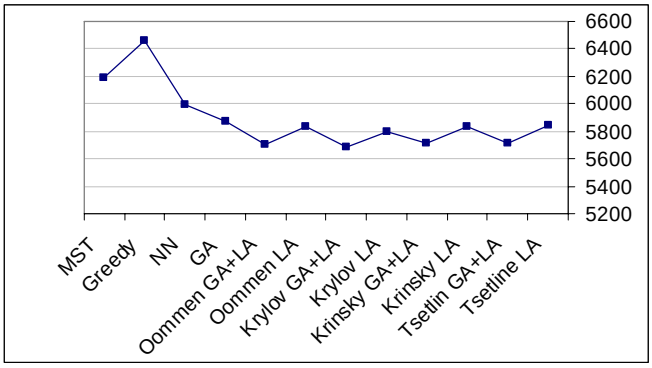


Diagram 1.Average tour length (cost) acquired by different algorithms

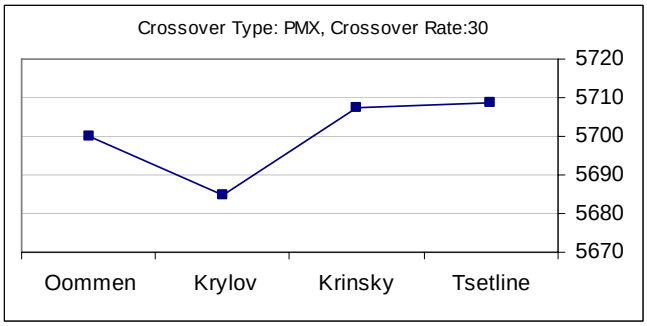


Diagram 2.Average tour length (cost) acquired by Hybrid Alg. using Partially Mapped Crossover with rate 30

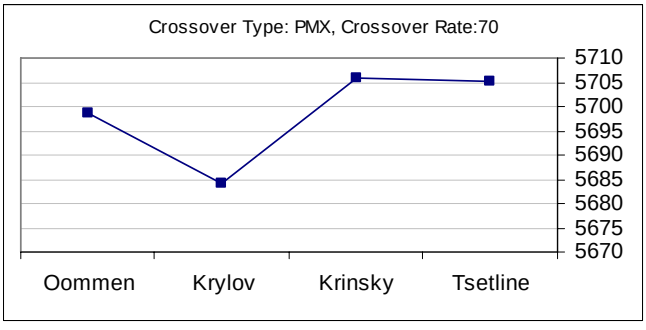


Diagram 3.Average tour length (cost) acquired by Hybrid Alg. using Partially Mapped Crossover with rate 70

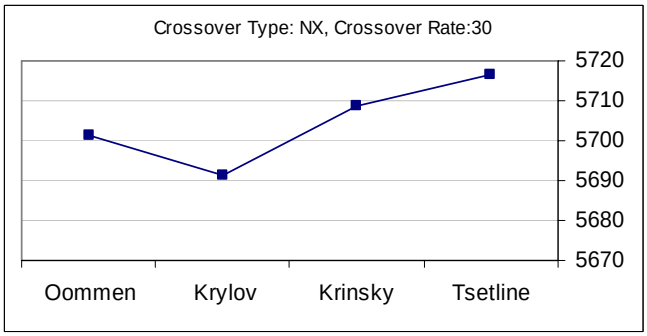


Diagram 4.Average tour length (cost) acquired by Hybrid Alg. using New Crossover with rate 30

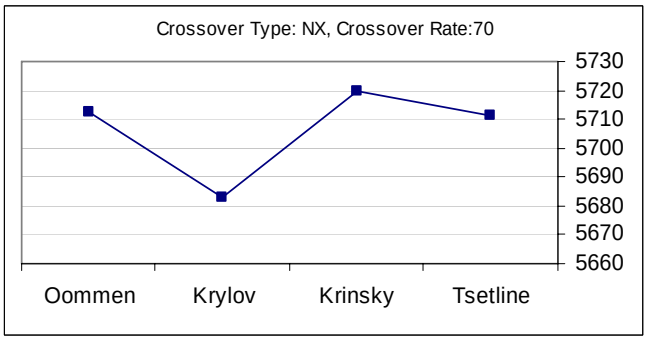


Diagram 5.Average tour length (cost) acquired by Hybrid Alg. using New Crossover with rate 70

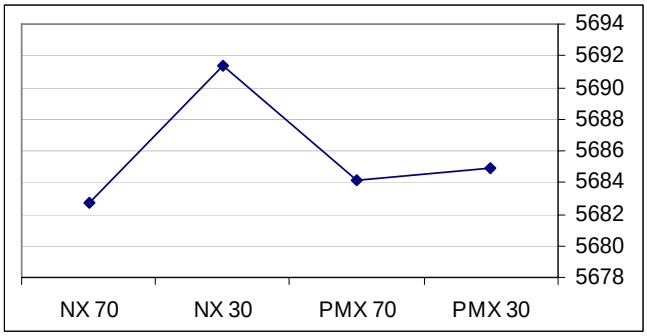


Diagram 6.Average tour length (cost) acquired by Hybrid Alg. Based on Krylov automata with applying different crossover methods and crossover rates

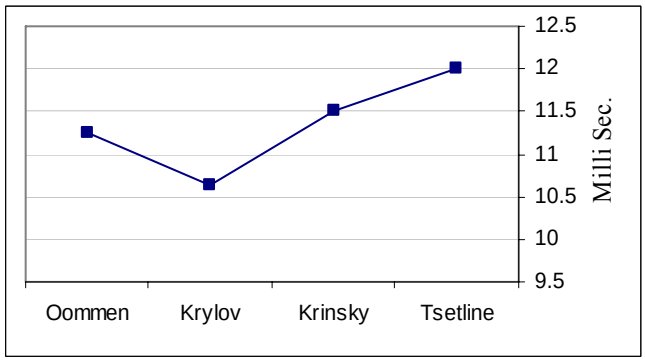


Diagram 7.Average requried time for for Hybrid Alg. Based on Tsetline, Krinsky, Krylov and Oommen automatas

7. Conclusion and suggestions

Graphs, specially labeled graphs are very powerful and widely used tools in computer applications; one of them

most important problems is finding TSP tour. More than five decades of research has been done in this problem and since there is still no polynomial algorithm for solving this problem, the researches in this field is continuing. Optimal algorithms for solving this problem can be found by using appropriate search methods and combining them. Also reach better results by clustering graph nodes, execution hybrid algorithm on each cluster independently and using multi-population GA.

8. References

[۱] میبدی، محمد رضا و بیگی، حمید. "حل مساله تناظر گراف توسط آتوماتاهای یادگیر". دانشکده مهندسی کامپیوتر. دانشگاه صنعتی امیرکبیر. تهران. ایران. ۱۳۷۹.

[۲] میبدی، محمد رضا و رضاپور میرصالح، مهدی. "یک روش ترکیبی (GA+LA) برای حل مساله تناظر گراف". دانشکده مهندسی کامپیوتر. دانشگاه صنعتی امیرکبیر. تهران. ایران. ۱۳۸۲.

[3] D. S. Johnson, and L. A. McGeoch, "Experimental Analysis of Heuristics for the STSP", *in The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, 2002, Boston, 369-443.

[4] D. S. Johnson, G. Gutin, L. A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich, "Experimental Analysis of Heuristics for the ATSP", in *The Traveling Salesman Problem and its Variations*, G. Gutin and A. Punnen, Editors, Kluwer Academic Publishers, 2002, Boston, 445-487.

[5] D. S. Johnson, "A Theoretician's Guide to the Experimental Analysis of Algorithms", To appear in *Proceedings of the 5th and 6th DIMACS Implementation Challenges*, M. Goldwasser, D. S. Johnson, and C. C. McGeoch, Editors, American Mathematical Society, Providence, 2002.

[6] J. Cirasella, D.S. Johnson, L.A. McGeoch, and W. Zhang, "The Asymmetric Traveling Salesman Problem: Algorithms, Instance Generators, and Tests", in *Algorithm Engineering and Experimentation, Third International Workshop, ALENEX 2001*, Lecture Notes in Computer Science, Vol. 2153, Springer, Berlin, 2001, 32-59.

[7] M. Grötschel, and O. Holland, "Solution of Large-Scale Symmetric Traveling Salesman Problems", *Mathematical Programming* 51, 1991, 141-202.

[8] M. Padberg, and G. Rinaldi, "A Branch-And-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", *SIAM Review* 33, 1991, 60-100.

[9] M. Jünger, G. Reinelt, and G. Rinaldi, "The Traveling Salemsan Problem", in: *Handbooks in Operations Research and Management Science, Volume 7* (M.O. Ball, T. Magnanti, C.L. Monma, and G. Nemhauser, eds), Elsevier Science B.V., 1995, 225-330.

[10] P. Moscato, and M.G. Norman, "An Analysis of the Performance of Traveling Salesman Heuristics on Infinite-Size Fractal Instances in the Euclidean Plane", Oct. 1994.

[11] Sanjeev Arora, "Nearly Linear Time Approximation Schemes for Euclidean TSP and other Geometric Problems", January 1997, (added to TSPBIB on May 2, 1997).

[12] P. Merz, and B. Freisleben, "Genetic Local Search for the TSP: New Results", in *Proceedings of the 1997 IEEE*.

[13] B. Freisleben, and P. Merz "A Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems", appeared in *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation,* Nagoya, Japan, 1996, 616-621.

[14] B. Freisleben, and P. Merz, "New Genetic Local Search Operators for the Traveling Salesman Problem", in *Proceedings of the 4th Conference on Parallel Problem Solving from Nature - PPSN IV,* (H.-M. Voigt, W. Ebeling, I. Rechenberg, H.-P. Schwefel, eds.), Volume 1141 of Lecture Notes in Computer Science, 1996, 890-899.

[15] L.F. Escudero, and M.T. Ortuno, "On Due-Date Based Valid Cuts for the Sequential Ordering Problem", *Volume* 5, *No.* 1, 1997, 159-165.

[16] Beigy, H. and Meybodi, M. R. "Optimization of Topology of Neural Networks Using Learning Automata", *Proc. Of 3th Annual Int. Computer Society of Iran Computer Conf. CSICC-98*, Tehran, Iran, pp. 417-428, 1999.

[17] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Reading, MA: Addition-Wesley, 1989.

[18] Mars, P. and Narendra. K. S., and Chrystall, M., "Learning Automata Control of Computer Communication Networks", *Proc. Of Third Yale workshop on Application of Adaptive Systems Theory*, Yale University, 1983.

[19] Hashim, A.A., Amir, S.and Mars, p. "Application of Learning Automata to Data Compression, In Adaptive and Learning Systems", K. S. Narendra (Ed), New York: Plenum Press, pp. 229-234, 1986.

[20] Narendra, K. S. and Thathachar, M. A. L., "Learning Automata: An Introduction", Prentice-hall, Englewood cliffs, 1989.

[21] Meybodi, M. R. and Lakshmivarhan, S., "A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters", *proc. Of Third Yale Workshop on Applications of adaptive System Theory*, Yale University, pp. 106-109, 1983.

[22] Meybodi, M. R. and Beigy, H., "New Class of Learning Automata Based Scheme for Adaptation of Backpropagation Algorithm Parameters", *Proc. Of EUFIT-98*, Sep. 7-10, Achen, Germany, pp. 339-344, 1998.

[23] Oommen, B. J. and Ma, D. C. Y., "Deterministic Learning Automata Solution to the Keyboard Optimization Problem", IEEE Trans. On Computers, Vol. 37, No. 1, pp. 2-3, 1988.

[24] Beigy, H. and Meybodi, M. R., "Optimization of Topology of neural Networks Using Learning Automata", *Proc. Of 3th Annual Int. Computer Society of Iran Computer Conf. CSICC-98*, Tehran, Iran, pp. 417-428, 1999.

[25] K. Bryant, "Genetic Algorithms and the Traveling Salesman Problem", Thesis, Harvey Mudd College, Dept. of Mathematics, 2000.

[26] F. Busetti, "Genetic Algorithm Overview".

[27] E. Cantu-Paz, "A Survey of Parallel Gentic Algorithms", IlliGAL Reprot No. 97003, May 1997.

[28] Oommen, B. J., Valiveti, R. S., and Zgierski, J. R., "An Adaptive Learning Solution to the Keyboard Optimization Problem", IEEE Trans. On Systems. Man. And Cybernetics, Vol. 21, No. 6, pp. 1608-1618, 1991.

[29] http://www.tsp.gatech.edu.