A Hybrid Method for Solving Traveling Salesman Problem

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

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

Mortaza Abbaszadeh
Dept. of Computer Engineering, Islamic Azad University, Soufian Branch, Iran
M_A138@yahoo.com

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 reaching to 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 the superiority of hybrid algorithm over LA and GA.

1. Introduction

Graphs are powerful tools used widely in various applications. One of the 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 … can be modeled by TSP.

TSP is generalization of famous Hamiltonian Cycle problem. 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 $(u,v) \in E$ has non-negative cost $C(u,v)$. The salesman should start with an origin and visit all the nodes once and return to the origin in a way that it minimized the total cost of tour.

In some problems, the increase of their dimension lead to 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.

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 reaching to answer increases remarkably using LA and GA simultaneously in search process, and it also prevents algorithm from being trapped in local minimums.

2. 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 termination condition is established.

3. Learning automata

Learning in LA is choosing an optimal action from a set 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 set 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.

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

Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)
0-7695-2841-4/07 $ 9.00 © 2007 IEEE

reduces the convergence speed of automata. For this reason Object Migrating Automata (OMA) was proposed by Oomenn and Ma.

4. 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. 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 of chromosome's genes is attributed to the one of the 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.

$\underline{\phi} = \{\phi_1, \phi_2, \ldots, \phi_{KN}\}$ 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 $\{\phi_1, \phi_2, \ldots, \phi_N\}$, $\{\phi_{N+1}, \phi_{N+2}, \ldots, \phi_{2N}\}$,… and $\{\phi_{(K-1)N+1}, \phi_{(K-1)N+2}, \ldots, \phi_{KN}\}$, and graph nodes is classified on the basis of its status. If node u from graph is in the status set $\{\phi_{(j-1)N+1}, \phi_{(j-1)N+2}, \ldots, \phi_{jN}\}$, 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.

For instance, consider complete graph in figure 1 that includes 6 nodes.

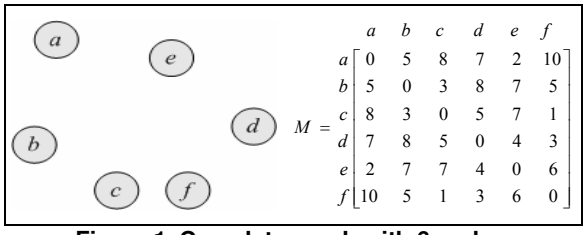

Figure 1. Complete graph with 6 nodes

Consider permutation $< c, b, f, d, a, e >$ from graph figure 1. This permutation has been shown by a LA with similar connections to Tsetline automata in figure 2. 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 graph nodes is placed at the border status of each relative action.

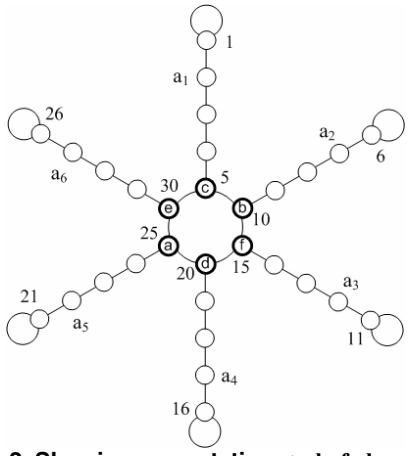

Figure 2. Showing permutation <c, b, f, d, a, e> by a LA with similar connections to Tsetline automata

Initial population:

Supposing that the number of population's members is n. n-1 population's members 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 in which it has the most similarity with the final answer.

As an example, the formation of initial population for graph in figure 1 with the assumption n=6 is explained in 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 6^{th} permutation not visited nearest neighbor method is used. Assuming that the start node is a, the 6^{th} permutation is $< a, e, d, f, c, b >$. The initial population resulting from graph in figure 1 has been shown in figure 3. At the beginning each node is at the border status of its action.

Fitness function:

In GAs fitness function is indicator of chromosomes survive. So that in TSP the fitness of an automata defined as follow:

$f(LA_i) = 1 / \textit{Lenght of Specified Tour by } LA_i$

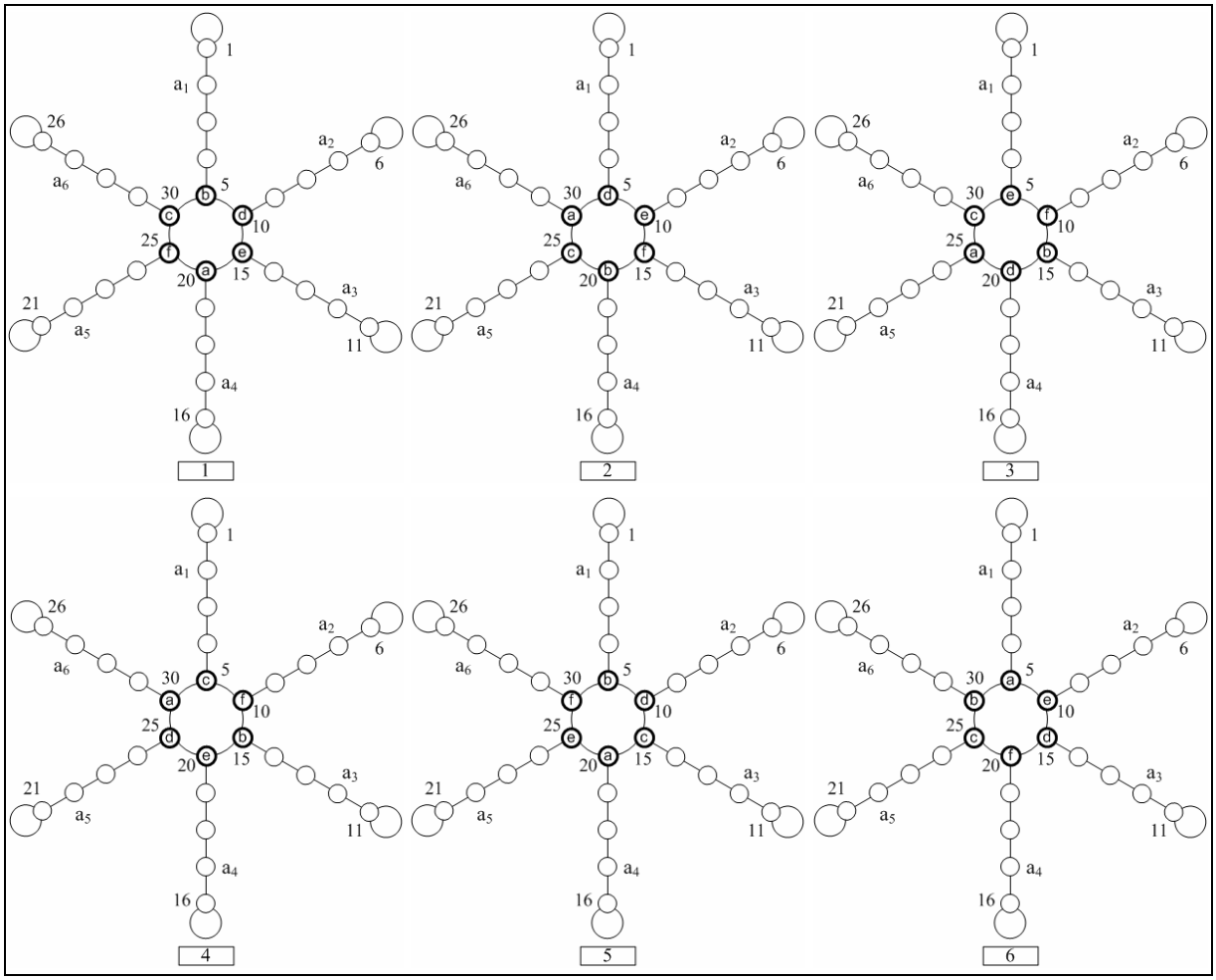

Figure 3. Initial population for graph in Figure 1

Operators:

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

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.

Crossover operator: To do this operator one of the methods: Partially Mapped Crossover, Ordered Crossover, Cycle Crossover, and New Crossover which are appropriate to work with permutations can be used. 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. The result of this process is two chromosomes which are called the two parent automata's children.

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 4 two new chromosomes are created as the result of replacing appropriated actions at replacement distance.

Mutation operator: To do this operator one of the methods: Swap Mutation, Insertion Mutation, Inversion Mutation, and Scramble Mutation which are appropriate to work with permutations can be used. For instance in Swap Mutation two actions (genes) from an automata (chromosome) are selected randomly and replaced.

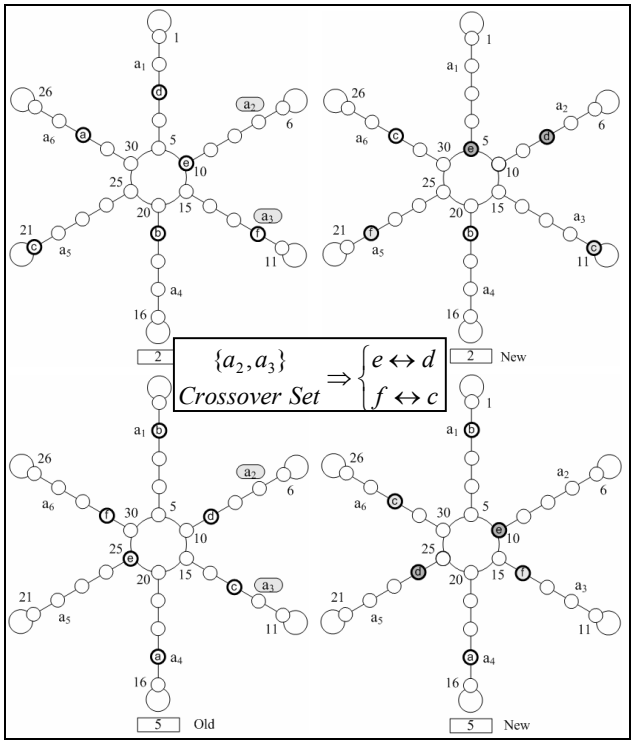

Figure 4. The manner of doing New-Crossover oper ator

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 the result of rewarding or penalizing. If a gene is placed at the border status of an action, penalizing it lead to changing that gene's action 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.

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 from b)/2) is less than threshold value (threshold value is determined comparatively and its value at any moment equals in: totals tour cost / node numbers) this node are rewarded and move to the more interior (importance) 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 5.

If the average cost of incoming and outgoing edges to a node is more than threshold value, the resulting tour won't be appropriate and it will be penalizing. The movement of thus node for 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 and move to the more outer status of this action.
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 border status of its action and then they will be replaced. The movement of this node is shown in figure 6.

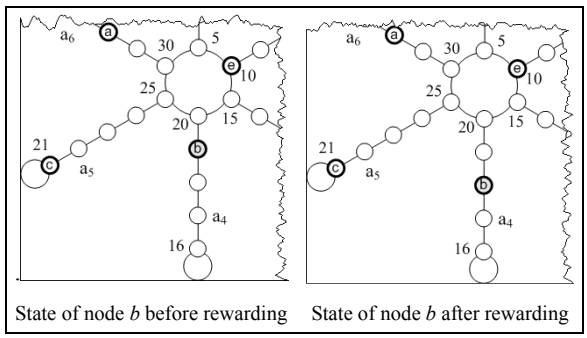

Figure 5. Rewarding node b

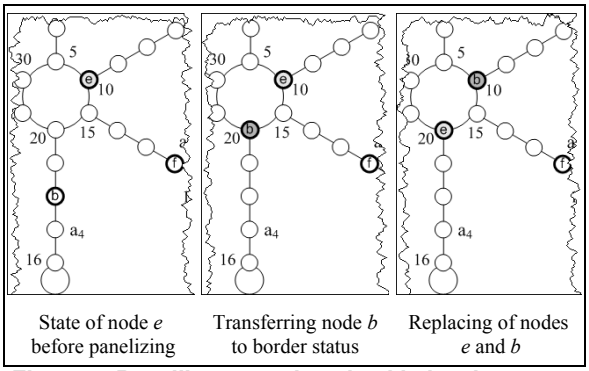

Figure 6. Penaliizng a node palced in border status

5. Experimental results

In this section the Experimental results of TSP solver algorithms which were implemented on the basis of LA, GA and hybrid algorithm are shown. These results indicate remarkable improvement of hybrid algorithm compared with methods based on LA and GA. In the experiments size of graphs (taken from the TSPLIB) 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.

Diagrams 1 to 7 shows briefly comparison of hybrid algorithm and other algorithms used to solve TSP. As the results 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 crossover rates both in terms of execution time and tour length (cost).

6. Conclusion and suggestions

Graphs, specially labeled graphs are very powerful and widely used tools in computer applications; one of most important problems in graphs theory is finding TSP tour. Researchers worked on this problem more than two decades. 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 by clustering graph nodes and execution of hybrid algorithm on each cluster independently better results can be achieved. And also using multi-population GA can improve results.

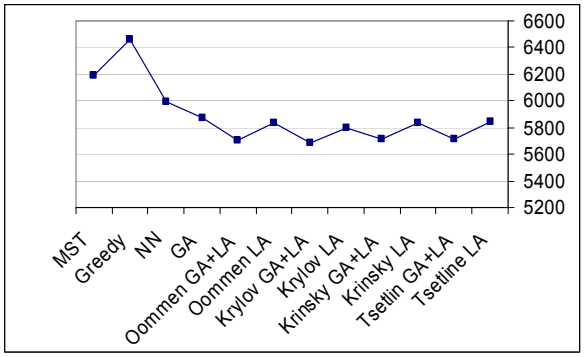

Diagram 1. Average tour length acquired by various algorithms

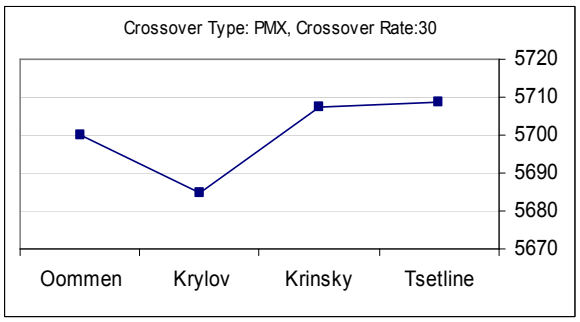

Diagram 2. Average tour length acquired by Hybrid Alg. using Partially Mapped Crossover with rate 30

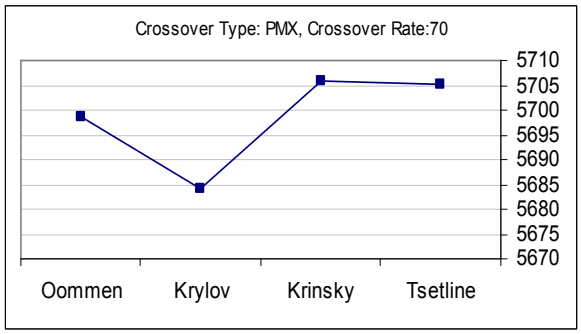

Diagram 3. Average tour length acquired by Hybrid Alg. using Partially Mapped Crossover with rate 70

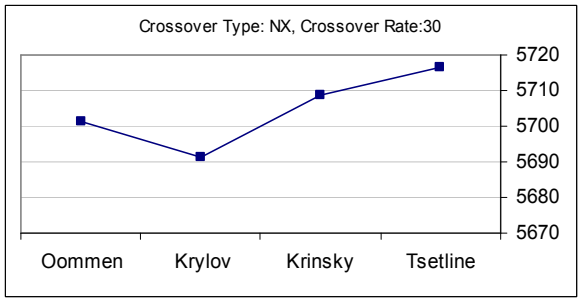

Diagram 4. Average tour length acquired by Hybrid Alg. using New Crossover with rate 30

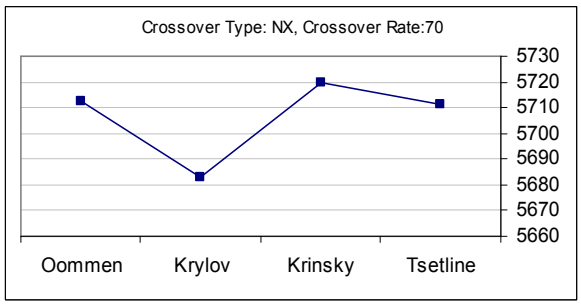

Diagram 5. Average tour length acquired by Hybrid Alg. using New Crossover with rate 70

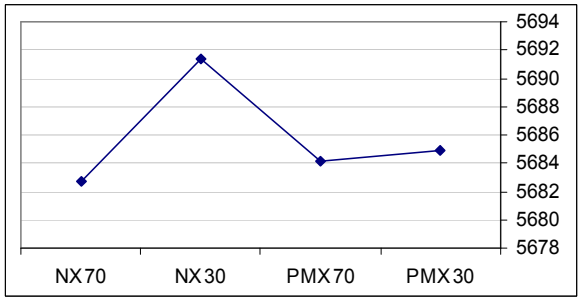

Diagram 6. Average tour length acquired by Hybrid Alg. based on Krylov automata with applying variou s crossover methods and crossover rates

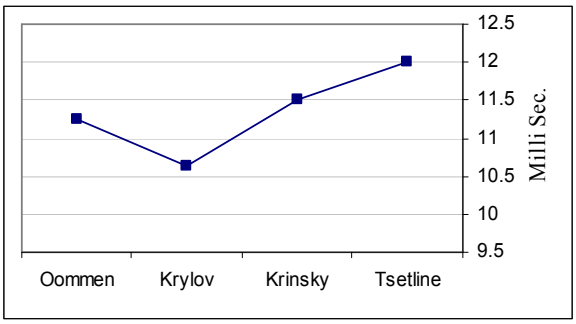

Diagram 7. Average requried time for Hybrid Alg. Based on Tsetline, Krinsky, Krylov and Oommen automatas

7. References

[1] 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.

[2] 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.

[3] 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.

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

[5] Meybodi, M. R and Beigy, H.,"Solving Graph Isomorphism Using Learning Automata", *Dept. Of Computer Engineering,* Amirkabir University, Tehran, Iran, 2001.

[6] Meybodi, M. R and RezapoorMirsaleh, M.,"A Hybrid Method (GA+LA) for Solving Graph Isomorphism", Dept. Of Computer Engineering, Amirkabir University, Tehran, Iran, 2003.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

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

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

[15] 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.

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

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

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