International Journal of Computer Theory and Engineering, Vol.2, No.6, December, 2010
1793-8201

# A Novel Genetic Algorithm for GTSP

Zaheed Ahmed, Irfan Younas and Muhammad Zahoor

***Abstract*— The Generalized Travelling Salesman Problem (GTSP) is a special instance of the well-known travelling salesman problem which belongs to NP-hard class of problems. In the GTSP problem which is being addressed in this research we split the set of nodes (e.g. cities) into non-overlapping subsets; where the optimal solution is a minimum cost tour visiting exactly one node from each subset. In this paper a genetic algorithm with new and innovative way of generating initial population is presented. Concepts like cluster segmentation, partially greedy crossover, greedy insert mutation and enhanced swap mechanisms are also introduced. An initial analysis of the proposed algorithm shows enhanced results in terms of optimality and computational time as compared to existing approaches.**

***Index Terms*— Generalized travelling salesman problem, genetic algorithms, greedy insert mutation, partially greedy crossover.**

## I. Introduction

The well known travelling salesman problem (TSP) is a problem to find the minimum cost tour of ‘n’ cities or nodes where each city must be visited exactly once. It is established that the TSP is one of the top ten problems addressed by researchers [1]. Generalized travelling salesman problem (GTSP) is a variant of classical TSP having a number of real world applications e.g. postal and vehicle routing [2], computer file sequencing [3], scheduling clients through welfare agencies [4].

The GTSP can be stated as: Given a set of clusters, where distances between them is known, calculate the minimum cost -tour starting from a given node in a cluster, passing through all the other clusters and returning to the first cluster. Note here that each cluster is comprised of several nodes and only one node is visited from each cluster.

Formally GTSP can be defined as [5, 6]: A domain of cities $C = \{c_1, c_2 \ldots c_n\}$ partitioned into ‘m’ subsets $\{s_1, s_2 \ldots s_m\}$ such that $s_1 \cup s_2 \ldots \cup s_m = C$ and $s_i \cap s_j = \phi$ where $i \neq j$. The round tour ‘T’ must contain exactly one city from each subset where the sum of Euclidean distances ($\sum$ d) are minimised. For example the Euclidean distance ‘*d*’ between two points ‘$p_1$’ and ‘$p_2$’ located in two dimensional space with coordinate ($x_1$, $y_1$) and ($x_2$, $y_2$) respectively, can be computed by (1):

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}$$

The TSP is categorised as a NP-hard problem and so is the GTSP. In addressing NP-hard class of problems many approaches have been considered, such as Simulated Annealing and Tabu Search, but Genetic Algorithms (GAs) have shown to be amongst the best approaches as the whole search space need not be traversed in obtaining the global minimum.

A number of genetic algorithms are available in literature for TSP like problems [1], [5], [6], [8]-[12]. Unlike some Meta-heuristics approaches where a single candidate solution is maintained and all efforts are deployed to improve the solution quality in an efficient manner, the GA maintains a solution set of size ‘N’ and puts less effort on individual solutions. Furthermore on each generation/iteration local improvement heuristics and genetic operators i.e. reproduction, crossover and mutation are applied to improve the solution set quality.

In GTSP, natural selection method is known as roulette wheel [1] which transforms the superior individuals from one generation to the next. Note that genetic operators and local improvement heuristics for TSP and GTSP are uniformly applied on both types of problems with slight modifications. For TSP and GTSP some standard crossover operators are: partially mapped crossover (PMX), random crossover (RX), cyclic crossover (CX), the ordered crossover (OX) [13], modified ordered crossover (MOX) [1] and modified rotational ordered crossover (mrOX) [8]. Among the well known mutation operators worth mentioning are: Simple inversion mutation (SIM) and insertion mutation (ISM) [14]. To enhance the power of GA, local improvement heuristics i.e. well-known ‘2-opt’, integrated local search [12], knowledge based multiple inversion and ‘swap’ procedures [1], [5] are also applied. Numerous variations have been cited in literature on the basis of the conventional GTSP [8].

This study presumes symmetric variant that means distance from city ‘$c_i$’ to city ‘$c_j$’ and vice versa are equal. It is further required that the solution contains exactly one city from each cluster. The remaining paper is organized as follows: section II discusses the related work; and proposed innovative GA for GTSP is presented in Section III. Section IV contains the cost-benefit analysis and discussions. Finally concluding remarks with future directions are summarized in section V.

## II. Related Work

Since the invention of the genetic approach by John Holland and his students in 1970s [15], the classical TSP is attempted by many researchers utilizing GA in a number of ways [1], [5]. However, to the best of our knowledge and

Manuscript received March 03, 2010.

Z. Ahmed has completed his MS from University Institute of Information Technoloty, University of Arid Agriculture Rawalpindi, Pakistan (phone: +92-345-5990300; e-mail: zaheed1@hotmail.com ).

I. Younas is full Professor at HITEC University Taxila Cantt., Taxila, Pakistan. (e-mail: iyounas101@gmail.com).

M. Zahoor obtained the degree of MS in Computer Science from UIIT, Arid Agriculture University, Rawalpindi, Pakistan. (e-mail: zahoor_51@yahoo.com).

according to Tasgetiren, Suganthan, Pan and Liang [12], evolutionary algorithms are rarely employed to address GTSP. So far the best performing evolutionary algorithms for GTSP available in literature are: *random key GA* for GTSP by Snyder & Daskin [5], *mrOX GA* by Silberholz & Golden [8] and a GA for GTSP by Tasgetiren, Suganthan, Pan and Liang [12].

A random key GA [5] provides a solution for GTSP that overcomes the problem of traditional key methods. Sometime traditional encoding method can produce infeasible offspring due to integer string crossover. Random keys are utilized to encode the solution and this encoded solution guarantees the feasibility of the offspring. Initial population in this algorithm is generated on the basis of random selection of genes from each cluster. Here "Level-1" improvement heuristic is applied on each solution in the population. A somewhat new concept of immigration is used rather than traditional mutation, where the concept of generating new chromosome from scratch rather than mutating the genes of existing chromosomes.

In the mroX [8] a traditional GA encoding scheme for TSP is applied, and two local improvement operations are used i.e. famous '2-opt' and 'swap procedure' which operates similar to random key GA [2]. Initial population of 50 chromosomes is generated by uniformly selecting one node from each cluster. It is to be noted that in [8] the classical TSP Ordered Crossover (OX) operator has been slightly modified according to GTSP requirements. Here a rotational component is added to modify ordered crossover (rOX). The rOX rotates the genes selected from second parent and thereafter all possible permutations are evaluated to find the best fitting permutation in combination with genes already selected from first parent. Furthermore, rOX is modified (mrOX) by adding another rotational component for side genes only. The mrOX increases the computational time but it is useful in generating diversity in the population.

Another best performing GA for GTSP has been developed by Tasgetiren, Suganthan, Pan and Liang [12]. Permutations of clusters and corresponding node from each cluster with Euclidean distance between adjacent nodes are used as part of the encoding scheme. Here the so-called 'Two-cut PTL crossover' operator is used which always produces a distinct child. The mutation operation is fairly simple where a randomly selected node is replaced with another node of its own cluster. The insertion method is similar to the random-key GA [5]. Operations such as Two-opt and iterated local search (ILS) are applied for the purpose of local improvement heuristics. Traditional ILS is modified in a way that search counter is re-initiated every time a better result is found.

All the optimal and efficient evolutionary algorithms discussed in this section have a common problem of generating the initial population randomly. A randomly generated chromosome requires more computation time to converge to desired level of solution optimality. In addition the whole cluster is considered for mutation and two-opt swapping procedures, both of which increase the search space and consequently the CPU time.

## III. Proposed GA using Cluster Segmentation

In this research we propose a new technique where the domain in divided into natural clusters and furthermore each cluster is subdivided into a number of segments containing nodes. Only nodes from active segments will be used for tour generation as explained below.

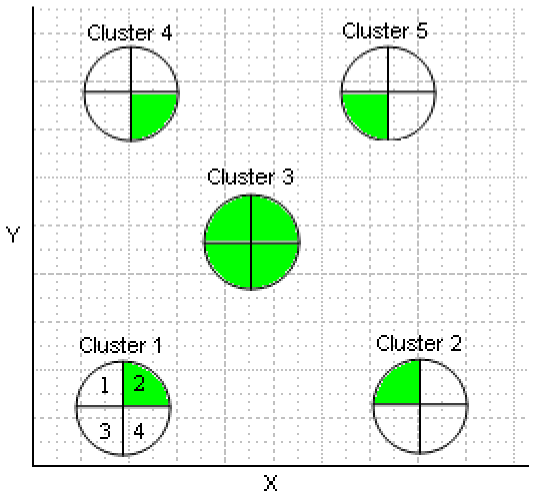

Figure 1: Cluster representation with active/inactive segments.

### A. Cluster Division and Segments Classification Method

To generate the initial population we propose to split each cluster into four segments. Each segment is quarter portion of the cluster having ID range 1-4 as depicted in fig. 1 (cluster 1). This cluster segmentation decreases the search space during GA operations. Each segment is either active or inactive depending on the location of cluster in domain. This classification is accomplished by an algorithm given in this section. If any of the active segments is empty then both of its adjacent segments are marked as active. Adjacent segment can be either top/bottom or left/right but diagonal segment is not considered as an adjacent.

```
for $j = 1$ to $|C|$
for $k = 1$ to 4
        Set $(S_k)$ = inactive
for $i = 1$ to $|C|$
for $m = i$ to $|C|$
            if $((C_i.x < C_j.x)$ && $(C_i.y < C_j.y))$
                    Set $(C_i.S_2$ = active)
                    Set $(C_j.S_3$ = active)
            if $((C_i.x < C_j.x)$ && $(C_i.y > C_j.y))$
                    Set $(C_i.S_4$ = active)
                    Set $(C_j.S_1$ = active)
            if $((C_i.x > C_j.x)$ && $(C_i.y > C_j.y))$
                    Set $(C_i.S_3$ = active)
                    Set $(C_j.S_2$ = active)
            if $((C_i.x > C_j.x)$ && $(C_i.y > C_j.y))$
                    Set $(C_i.S_2$ = active)
                    Set $(C_j.S_3$ = active)
            if $((C_i.x > C_j.x)$ && $(C_i.y < C_j.y))$
                    Set $(C_i.S_1$ = active)
                    Set $(C_j.S_4$ = active)
```

### B. Encoding/Decoding Scheme

One of the key elements of the GA is solution representation. In classical TSP the traditional encoding scheme is to represent the solution with integer permutations. An integer can be assigned to each of the node as {1, 2 … n}. This scheme is fairly simple but having problem of infeasible children during one and two point crossover. Infeasibility problem can be handled in many different ways i.e. via special repair algorithms or random key encoding scheme. However, in this research infeasible child problem is out of question due to the nature of our partially greedy crossover operator (section 3.4). The encoding scheme suggested by Tasgetiren, Suganthan, Pan and Liang [15] is used as the base of the solution representation in this GA. To handle the GTSP solutions correctly we have used both the cluster number and corresponding node as well. Clusters are represented as $C_i$ where 'i' is an integer from 1 to number of clusters (n) and all the nodes in a cluster are assigned an unsigned integer from 1 to number of nodes in that cluster (m). For example consider a chromosome as:

$C_1.2 \rightarrow C_4.1 \rightarrow C_3.3 \rightarrow C_2.2 \rightarrow C_5.4$

This chromosome contains the information that the cluster tour sequence is: $C_1$, $C_4$, $C_3$, $C_2$, $C_5$ and return to $C_1$ again. In cluster $C_1$ and $C_2$ the 2nd node is visited while 3rd node from $C_3$, 1st node from $C_4$ and 4th node from cluster $C_5$ respectively will be visited.

### C. *Initial Population Generation*

In this section we introduce a novel strategy to generate the initial population. Each cluster is divided into four segments as shown in fig. 1. We have already presented an algorithm (see section III-A) to mark each active segment of the cluster as shaded segment (see fig. 1). Then we consider each cluster as a node in the 2-D space. Initial population will be generated in 3 simple steps:

1) Assume each cluster as a node and find two distinct and best sequences of clusters to visit by using classical TSP algorithm.
2) Substitute each cluster sequence with its own node corresponding to the active segment and maintain two pools A and B corresponding to both of the sequences produced in step 1.
3) Generate initial population by applying (2) iteratively until population size N.

### D. *Genetic Operators*

The chromosome population for the GA is set to N=50 chromosomes. N can however vary depending on size and complexity of the problem. In each pool A and B, 25 chromosomes are maintained in every generation constituted as follows: 60 percent is produced by applying the crossover operator, 25 percent are maintained through reproduction while remaining 15 percent are generated by *immigration. Greedy Insert Mutation* operator is applied to generate diversity in population. *Enhanced swapping procedure* is applied on each chromosome as a source of local improvement heuristics which is explained in next section.

1) Reproduction

For reproduction purposes, proposed method selects the *superior* chromosomes. Here fitness function F, in (2), is the sum of distance among each pair of cluster nodes in a sequence:

$$F(t) = \sum_{i=1}^{C-1} d_{c_j.n,\, c_k.n} + d_{c_m.n,\, 1.n} \tag{2}$$

Where t, d, c, and n are tour, Euclidean distance, cluster number and node number respectively. This strategy carries over the best chromosomes to the subsequent generation which ensures that every new generation converges towards the desired solution

2) Partially Greedy Crossover

Generally, crossover, mutation and local improvement heuristics (stated in section 1) are applied discretely to increase population diversity and speedup the process of its convergence toward the desired solution. The Partially Greedy Crossover (PGC) operator picks two best solutions from parent generation and produces a child that inherits genetic material from both parents in the following way: First gene of the child can be inherited from either of the parents with 0.5 probabilities each. Remaining genes of the solution are chosen greedily i.e. the genes can belong to any of the parents having minimum cost with respect to the gene picked previously.

3) Greedy Insert Mutation

Insert mutation (ISM) operator [12] is one of the superior performers that can be used to enhance the population diversity. ISM operator selects a gene randomly in a chromosome and replaces it with a gene belonging to same cluster. However in the Greedy Insert Mutation (GISM) method proposed in this study the substitution of a gene with another gene of the same cluster as well as the replacement will only take place if that minimizes the overall cost of the tour. Well-known 2-opt improvement can also be applied separately in level-I and level-II improvements as in [5] but to some extent this functionality is automatically achieved using GISM.

4) Immigration

Immigration is achieved by producing a new chromosome by applying the technique explained in section 3.3 (Step 2, 3).

### E. *Enhanced Swapping Procedure*

Local improvement heuristics, swap and 2-opt are applied widely on each solution in population to improve the GA performance. We have included the greed in knowledge based neighborhood swapping (KBNS) operator encoded by Ray, Bandyopadhyay and Pal [1]. Although good results were produced by the KBNS, however there is major drawback in its swapping procedure. In fact KBNS is partially knowledge based because there is chance that a swap may produce a child having more cost than the parent. During swapping process only swapped position is considered to reduce the cost while swapping position may increase the cost of the solution.

The swap procedure presented here is applied on each chromosome for each generation but if swap does not occur during two consecutive generations then it will be disabled for all successive generations.

The Enhanced swapping procedure works in two steps:

*Step 1: Calculate the cost* 'd' *for node or city* 'c' *as* (3)*:*

$$\begin{aligned}
d_1(c_i) &= d(i-1,i) + d(i,i+1)\\
d_2(c_j) &= d(j-1,j) + d(j,j+1)\\
d'_1(c_i) &= d(i-1,j) + d(j,i+1)\\
d'_2(c_j) &= d(j-1,i) + d(j,i+1)
\end{aligned} \tag{3}$$

*Step 2:* Pick the pair of cities or nodes that reduces the overall cost of the tour on the basis of the following condition given in (4):

$$d_1'(c_i) + d_2'(c_j) < d_1(c_i) + d_2(c_j) \tag{4}$$

If this condition holds then swap both cities indexed 'i' & 'j' for example as given below.

A tour t: 2→4→1→5→3→6, having Euclidean distances among each pair of cities are given in table I. Since we are dealing with symmetric distance that is why only the upper half of table I needs to be populated.

TABLE 1: SYMMETRIC DISTANCES AMONG DIFFERENT CITIES

| City | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1 | 0 | 18 | 20 | 20 | 16 | 22 |
| 2 | - | 0 | 10 | 15 | 9 | 13 |
| 3 | - | - | 0 | 19 | 24 | 11 |
| 4 | - | - | - | 0 | 16 | 12 |
| 5 | - | - | - | - | 0 | 27 |
| 6 | - | - | - | - | - | 0 |

Following indexes are selected as 'i' and 'j':

| i-1 | i | i+1 | j-1 | j | j+1 |
|---|---|---|---|---|---|
| 2 | 4 | 1 | 5 | 3 | 6 |

$$d_1(c_i) = 15 + 20 = 35$$
$$d_1(c_j) = 24 + 11 = 35$$
$$d'_1(c_i) = 10 + 20 = 30$$
$$d'_2(c_j) = 16 + 12 = 28$$

After this calculation step-2 will be applied as:
$30+28 < 35+35$

This condition is true so swap will occur and resultant tour will be as:

| i-1 | i | i+1 | j-1 | j | j+1 |
|---|---|---|---|---|---|
| 2 | 3 | 1 | 5 | 4 | 6 |

The final novel genetic algorithm (nGA) for GTSP is illustrated by a flow diagram depicted in fig. 2. Input to the nGA are clusters having nodes information e.g. town containing information about the location of houses in it. Each cluster is divided into four segments and initially all segments are inactivated. The segment activation algorithm marks the required segments as active and leaves all others as inactive one. Classical TSP algorithm is utilized to find two optimal cluster sequences for population pools A and B (See section III-C for details). Each of the cluster sequences is substituted by a node from its active segments and the substituted sequence is known as a chromosome or solution. Both pools are generated in this fashion. The next step is to evaluate each chromosome against a fitness function and check for exit criteria. If the exit criteria is not fulfilled then reproduction and immigration are applied which transform and generate the individuals respectively. Remaining population is created by utilizing (Partially Greedy Crossover) PGC. When population pools become equal to the population size 'N' then each individual will pass through the GISM and enhanced swap procedure. Finally, the fitness function is again evaluated and checked against exit criteria. The process from fitness evaluation to enhanced swap procedure application falls under the loop until exit criteria is met.

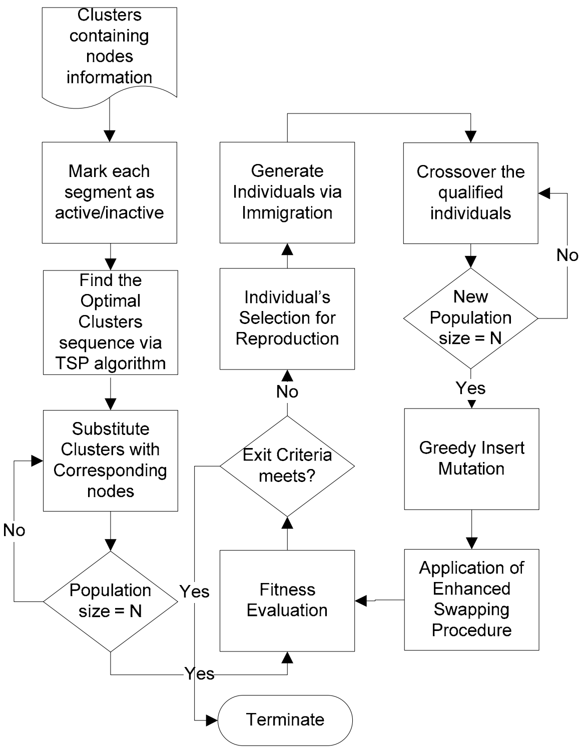
Clusters containing nodes information
Mark each segment as active/inactive
Find the Optimal Clusters sequence via TSP algorithm
Substitute Clusters with Corresponding nodes
No
Population size = N
Yes
Yes
Generate Individuals via Immigration
Individual's Selection for Reproduction
No
Exit Criteria meets?
Yes
Fitness Evaluation
Terminate
Crossover the qualified individuals
No
New Population size = N
Yes
Greedy Insert Mutation
Application of Enhanced Swapping Procedure

Figure 2. Overall novel Genetic Algorithm (nGA)

### *F. Termination Criterion*

Five consecutive populations do not improve or 50 iterations reach then termination condition meets. Exit criteria parameters can easily be changed according to the problem nature.

## IV. COST-BENEFIT ANALYSIS OF NGA

### *A. Search Space*

In existing literature the superior performing GA for GTSP [5], [8], [12] considers the whole search space for chromosome creation, genetic operations and local improvements. What is different in our approach is the partitioning of the cluster into segments as depicted in fig. 1 and only the relevant segments are activated for chromosome generation, application of evolutionary operators and local search improvements. The Algorithm constructed (See section III-A) for marking the segment as active/inactive takes only the O ($C^2$). By using the concept of segmentation given here we have gained up to 75% decrease in search space in a cluster and ultimately the CPU time.

### *B. Initial Population*

Conventionally in GA for any problem initial population is always generated randomly. Randomly generated chromosome needs much more CPU time to converge onto the final acceptable solution. In nGA initial population is generated by substituting the clusters with a node belonging to its active segments. Even though it may appear that more time is consumed to generate the initial population but in reality this is not the case because available heuristics are applied on benchmarks [16] having at most 89 clusters problem [8]. Hence in the presented algorithm we have to run TSP on at most 89 nodes, and that will find the near optimal or may be optimal cluster sequence in shorter time.

### *C. Genetic Operations & Improvements*

In this nGA, 40% of the population is constructed via reproduction and immigration while remaining 60% is generated by crossover. Although the PGC operation is taking fair computation time; this operator is integrating the functionality of ISM operator and '2-opt' local improvement heuristic as well. *Enhanced swapping* procedure is only consuming little CPU time due to the fact that initial population is generated in an intelligent fashion and there is a very little chance that swapping can take place. Hence most probably it will be disabled after 2 - 3 generations (See section III-E).

## V. Conclusions And Future Directions

A novel genetic approach is presented to solve the symmetric GTSP. An intelligent mechanism has been introduced for initial population generation so that the individual takes less time to converge on acceptable solution. PGC, GISM and enhanced swap procedures are defined which looks more promising to produce fairly simple, efficient and optimal solution. Cost-benefit analysis of the proposed study show more efficient use of CPU time and the solution quality is also significantly better. Furthermore it is quite easy to extend nGA to incorporate the asymmetric class and to handle the GTSP problem where different numbers of cities are required to be visited from different clusters.

## Acknowledgment

We would like to acknowledge Iftikhar Ahmed, Habiba Mobeen and Sadaf Khalid Rana for their moral support, editorial reviews and useful suggestions during this research work.

## References

[1] S. S. Ray, S. Bandyopadhyay and S. K. Pal, "New Operators of Genetic algorithms for Traveling Salesman Problem," Proc. 17th International Conference on Pattern Recognition (ICPR-04), 23-26 August 2004, Cambridge, UK, vol. 2, pp. 497-500.
[2] G. Laporte, A. Asef-Vaziri, and C. Sriskandarajah, "Some applications of the generalized travelling salesman problem," Journal of the operational research society 47 (12), 1996, pp. 1461–1467.
[3] A. Henry-Labordere, "The record balancing problem—A dynamic programming solution of a generalized travelling salesman problem," Revue Francaise D Informatique De Recherche Operationnelle 3 (NB2), 1969, pp. 43–49.
[4] J. P. Saksena, Mathematical model of scheduling clients through welfare agencies, Journal of the Canadian Operational Research Society, 1970, 8: 185-200.
[5] L. V. Snyder, and M. S. Daskin, A random-key genetic algorithm for the generalized traveling salesman problem, European Journal of Operational Research, 2006, pp. 38-53.
[6] H. Huang, X. Yang, Z. Hao, C. Wu, Y. Liang, and X. Zhao, Hybrid chromosome genetic algorithm for generalized travelling salesman problems, Springer-Verlag Berlin Heidelberg, 2005, pp. 137-140.
[7] Y. K. Hwang, and N. Ahuja, Gross motion planning - a survey, ACM Computing Surveys. 1992, 24(3):219–291.
[8] J. Silberholz, and B. Golden, "The Generalized Traveling Salesman Problem: a new Genetic Algorithm approach," Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies, 2007, pp. 165–181.
[9] D. Chao, C. Ye, and H. Miao, Two-level genetic algorithm for clustered traveling salesman problem with application in large-scale TSPs. Tsinghua Science and Technology, 2007, Volume 12, No.4, pp. 459-465.
[10] B. F. Al-Dulaimi and H. A. Ali, "Enhanced Traveling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA)", Proceeding of the world Academy of Science, Engineering and Technology", Rome 25th -27th May 2008. pp. 296-302.
[11] V. M. Kureichick, A. N. Melikhov, V. V. Miagkikh, O.V. Savelev, and A. P. Topchy, "Some new features in genetic solution of the traveling salesman problem." Proceedings of ACEDC, 1996.
[12] M.F. Tasgetiren, P.N. Suganthan, Q. Pan, and Y. Liang, "A genetic algorithm for the generalized traveling salesman problem", in Proc. IEEE Congress on Evolutionary Computation, 2007, pp. 2382-2389.
[13] J. J. Schneider, and S. Krikpatrick, Stochastic Optimization, Springer Berlin Heidelberg, 2006, pp. 415-422.
[14] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for travelling salesman problem: A review of representations and operators," Artificial Intelligence Review. 1999, 13: 129-170.
[15] M. Melanie, An Introduction to Genetic Algorithms, MIT Press, 1996.
[16] G. Reinelt, TSPLIB—A travelling salesman problem library, ORSA Journal on Computing, 1996, 4: 134-143.

**Z. Ahmed** received his BS in Information Technology from Allama Iqbal Open University, Islamabad, Pakistan in 2006 and recently he has completed his MS in Computer Science from University of Arid Agriculture Rawalpindi, Pakistan. His research interests are in the areas of evolutionary algorithms, fuzzy logic, formal methods and software testing.

**I.Younas** received his Ph.D. in 1997 from Southampton University, United Kingdom. Prior to this he obtained MSc and BSc degrees in Electrical Engineering from the Technical University of Denmark. After his doctorate degree he served at universities in the UK. In addition he has been a consultant for various software houses. His research interests are in computational modeling and more recently in evolutionary computation and optimization using neural networks and evolutionary algorithms.

**M. Zahoor** has completed his MS (CS) from University Institue of Information Technology (PMASAAUR). Before MS he obtained BS degree from Allama Iqbal Open University, Islamabad, Pakistan. His research interests include, designing algorithms, application of fuzzy logic and software testing techniques.