ISSN:0975-9646

RBGCA- Bee Genetic Colony Algorithm for Travelling Salesman Problem

Hemant Nagpure , Rohit Raja

Department of Computer Science & Engineering,

SSCET,Bhilai

Abstract- **Finding the shortest route is a challenging job which visiting each member of a collection of locations and returning to starting point is an NP-hard problem. It is also known as Travelling salesman problem, Travelling Salesman Problem (TSP) is a NP-hard problem in combinatorial optimization studied in operation research and theoretical computer science. It is used as a benchmark for many optimization technique. The goal of TSP is to find one path that can travel between all the nodes (instances) of the graph just once (Hamiltonian tour) in the smallest tour, that is, smallest Euclidian distance. In our work we present a hybrid version of Evolutionary algorithm to solve TSP problem. In this method we extend Artificial Bee Colony operators with Genetic Algorithm i:e Employed Bees and Onlooker Bees and Scout Bees to improve the solution space named as Real Bee Genetic Colony Algorithm (RBGCA). Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution. Artificial Bee Colony (ABC) is an optimization algorithm based on the intelligent behavior of honey bee swarm. In the proposed method we extend three operators of ABC with GA for local search strategy. The experimental results show that compared to original GA our RBGCA model can reach broader domains in the search space and show improvements in both precision and computational time.**

Keywords **—— ABC, Artificial Bee Colony, GA, Genetic Algorithm, TSP, Travelling Salesman Problem, Optimization.**

I. INTRODUCTION

The selection of a best element from several set of existing alternatives referred as optimization problem. In many such problems, exhaustive search is not feasible. It has important applications in several fields, including artificial intelligence, machine learning, mathematics, and software engineering. Most of such problems are considered NP-hard i:e they cannot be solved to optimality in polynomial computational time. Some common problems involving optimization are the travelling salesman problem, the minimum spanning tree problem etc.

The travelling salesman problem (TSP) is one of the well-known and extensively studied problems in discrete or combinational optimization and asks for the shortest roundtrip of minimal total cost visiting each given city (node) exactly once[4]. TSP is an NP-hard problem and it is so easy to describe and so difficult to solve. Graph theory defines the problem as finding the Hamiltonian cycle with the least weight for a given complete weighted graph. It is widespread in engineering applications and some industrial problems such as machine scheduling, cellular manufacturing and frequency assignment problems can be formulated as a TSP.

There are various met heuristics and approximation algorithms, which quickly yield good solutions, have been devised [5] [6] [7]. Modern methods can find solutions for extremely large problems (millions of cities) within a reasonable time. Several algorithms employing metaheuristic approaches such as Genetic Algorithm (GA) [5], Ant Colony Optimization (ACO) [11], Particle Swarm Optimization (PSO) [7] or Bee Colony Optimization (BCO) [6] [8] were applied to solve TSP. There are few hybridized method also present which solves the TSP. Sarayut and Siripom [9] proposed ACO-GA and ACO-PSO which are the hybrid methods. These methods perform based on the adjustment of a parameter Qo. If Qo is high, the local search would be geared towards exploitation, and when Qo is low, the search is geared towards exploration. Wong, et al. [8] applied BCO to solve TSP. These algorithms yield solutions with lower precisions on average. To improve upon these results the ABC algorithm [10], an effective algorithm already being applied to several optimization applications [11], is focused.

In this paper we extend ABC and GA to the area of combinatorial problems. Proposed method basically hybrid the Real ABC and GA to the three basic operators of Bee colony for finding out the optimality. To validate the performance of proposed method TSP is used in our experiment.

Genetic Algorithm is applied during the solution update and before each successful updating employed bees are applied and after the update onlooker and scout bees are applied. This in turn produces a modification on the position (solution) in her memory for finding a new path chain and tests the nectar amount (fitness value) of the new solution. Experimental results conclude that proposed method (RBGCA) gives better and efficient result than the classical GA.

The organization of the paper is as follows section II gives brief introduction on Genetic algorithm. Artificial Bee Colony optimization is explained in III section, Section IV gives the introduction of TSP. Section V describes Real Bee Genetic Colony Algorithm; Proposed Methodology (RBGCA) is explained in section VI. Section VII outlines the Experimental setup and result, Section VIII gives Conclusion.

II. GENETIC ALGORITHM

Genetic Algorithm transforms a population of individual objects, each with an associated fitness value, into a new generation of the population using the Darwin principle of individual of reproduction and survival of the fittest and naturally occurring genetic operation such as a cross over

(recombination) and mutation. Each individual in the population represents a possible solution to a given problem. The genetic algorithm attempts to find a very good or best solution to the problem by genetically breeding the population of individuals. The GA is inspired by the principles of genetics and evolution, and mimics the reproduction behavior observed in biological populations. The GA employs the principal of "survival of the fittest" in its search process to select and generate individuals (design solutions) that are adapted to their environment (design objectives/constraints). Therefore, over a number of generations (iterations), desirable traits (design characteristics) will evolve and remain in the genome composition of the population (set of design solutions generated each iteration) over traits with weaker undesirable characteristics. The GA is well suited to and has been extensively applied to solve complex design optimization problems because it can handle both discrete and continuous variables and nonlinear objective and constrain functions without requiring gradient information.

The literature includes many versions of the Genetic Algorithm (GA). In this work, real coded GA with tournament selection, low probability mutation rate is employed to solve the problem. GA has three operators of reproduction, crossover, and mutation. Reproduction is devised to inherit good-working individuals from generation to generation. If an individual has a relatively high fitness value, it has more chances to obtain its offspring in the next generation by a fitness-based selection rule. This operator is an artificial version of natural selection. In the crossover phase, two individuals in a present population are randomly selected, and they exchange their bits from the crossing site determined by another random number. During the mutation process, every bit has an equal probability to be replaced by its complement number. This is an imitation of a natural crossover and is very rare in progressing GA as well.

Real Coded Genetic Algorithm (RCGA) has strings of real numbers for the parameters to be searched the total number of parameters are k.

S_r = {1.52, .856, 15.23, …….., 6.93}

Where S_r represents concatenated strings of **RCGA, r**espectively. Since chromosome representation in RCGA is different from that of Binary Coded Genetic Algorithm**,** a crossover operator should be modified. Thus far, many kinds of crossover operators, including simple crossover, arithmetical crossover [13], **BLX-CY** crossover [12], and modified simple crossover [14], have been suggested. In this paper we have taken linear Crossover.

III. Artificial Bee Colony

Artificial bee colony (ABC) Algorithm is an optimization algorithm based on the intelligent behavior of honey bee foraging. This model was introduced by Dervis Karaboga in 2005, and is based on inspecting the behaviors of real bees on finding nectar amounts and sharing the information of food sources to the other bees in the hive. These specialized bees try to maximize the nectar amount stored in the hive by performing efficient division of labour and self-organization . The three agents in Artificial Bee Colony are:

- **Employed Bee:** Employed bees visited the food source and gather information about food source location and the quality. Employed bees have memory, so they know the places they have visited before and the quality of food there. Employed bees performs the local search and try to exploit the neighboring locations of the food source and search the best places of foods in the surrounding areas of the present value.
- **Onlooker Bee:** Onlooker bees are bees that are waiting on the dance area to decide which food source is better. This decision is made on the basis of information provided by employed bees. Onlooker bees perform the global search for discovering the global optimum. Scout bees do a random search for the food.
- **Scout Bee:** Scout bees do a random search for the food. Scout bees discovers the new area which are uncovered by the employed bees, these bees are completely random in nature and their operation of search. Scout bees avoid the search process to get trapped in local minima.

In ABC algorithm each food source position represents a candidate solution of optimization problem. In optimization problem each solution is associated with the fitness value on the basis of fitness value it is decided that which solution is better. So the nectar amount of a food source corresponds to the fitness value of the associated solution in ABC algorithm. The number of employed bees or the onlooker bees is equal to the number of solutions in the population. The ABC algorithm generates a random solution or initial population of size NF, where NF denotes the size of population or total number of food source. Each solution is represents the position of food source and denoted as xij, where i represents a particular solution (i=1,2,…..,NF) and each solution is a D-dimensional vector so j represents a particular dimension of a particular solution (j=1,2,….,D). After initialization of random solution employed bees start their searching. Employed bees search the food source near the previous food source, if the generated new solution is better than the previous solution than new solution replaces the old one. The comparison of food sources or solutions is done on the basis of fitness value or nectar amount of food source.

After all employed bees complete the search process; they share the nectar information of food sources (solutions) and their position information with onlooker bees. Now onlooker bee chooses a food source depending on the probability value Pi associated with the food source. Probability value for each food source is calculated by following equation (1):

$$P_i = \frac{f_i}{\sum_{n=1}^{NF} f_n} \quad (1)$$

Where fi is the fitness value of the solution *i* or the nectar amount of food source evaluated by employed bee and *NF* is the number of food source. So after the evaluation of the food source by the employed bees the probability value for each food source is determined which is used by onlooker bees.

To produce the candidate solution from the previous solution artificial bee uses the following equation (2):

$$V_{ij}=X_{ij}+\emptyset_{ij}(X_{ij}-X_{kj}) \quad (2)$$

Where *j* is a index for dimension (*j=1,2,…..,D*), *k* is a index which represents particular individual or solution from the population (*k=1,2,3…..,NF*), and *i* is also a index represents a particular solution (*i=1,2,…..,NF*). The difference between *i* and *k* is that *k* is determined randomly and value of *k* has to be different from *i*. *Øij* is a random number between [-1,1]. It controls the production of the neighbor food positions around *xij*.. The difference between the parameters of the *xij* and *xkj* decreases, the perturbation on the position *xij* decreases, too. Thus, as the search approaches to the optimum solution in the search space, the step length is reduced. After the production of candidate solution *vij* , its fitness value is calculated and then it is compared with the fitness of *xij*. If the new candidate solution has equal or better nectar or fitness than the old, it is replaced with the old one in the memory. Otherwise, the old is retained. If a solution is not improved further through a predetermined number of cycles then that food source is assumed to be exhausted. Exhausted food source is replaced by new food source generated by scout bees.

IV. TRAVELLING SALESMAN PROBLEM

Given a collection of cities and the cost of travel between each pair of them, the travelling salesman problem, or TSP for short, is to find the cheapest way of visiting all of the cities and returning to your starting point. In the standard version we study, the travel costs are symmetric in the sense that travelling from city X to city Y costs just as much as travelling from Y to X.

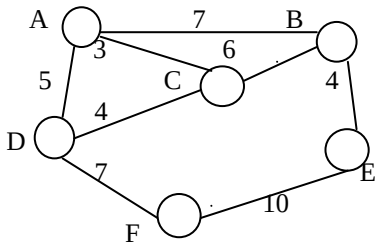

Figure 1: Graph Representation of TSP

In figure 1 each node in graph represents a city and each weighted edges in graph represents path from one city to other city. Weight is the cost to travel to reach city. Fitness function of the TSP will be

$$Tc = \sum_{k=1}^{d} Cij \quad (3)$$

Where Cij is cost associated of path i to j and k is for the no of city or dimension.

V. REAL BEE GENETIC COLONY ALGORITHM

Genetic algorithm (GA) and Bee colony optimization both are population based heuristic search technique used for optimization problems. GA is an effective optimization technique for both continuous and discrete optimization problem. A problem with GAs is that the genes from a few comparatively highly fit (but not optimal) individuals may rapidly come to dominate the population, causing it to converge on a local maximum. Once the population has converged, the ability of the GA to continue to search for better solutions is effectively eliminated: crossover of almost identical chromosomes produces little that is new. Only mutation remains to explore entirely new ground, and this simply performs a slow, random search. For better exploration of the local search space during the search process and to avoid convergence on local maximum problem employed and onlooker bees of ABC are added. Both operators are main operator of ABC algorithm. Employed bee operators use the property of solution (dimension) and generate new solutions. It may be possible that the generated solution is better than the existing one. If generated is better than it will remove the current solution. Onlooker bee chooses the solution on the basis of probability associated to individual solution and produces new solution on the property of the selected solution. Fitness of both the solutions are again compared the best one replaces the worst solution.

In this work we have chosen linear crossover as a crossover operator. Linear crossover is one of the earliest operator in real coded crossover it generates three solutions from two parents and the best two offsprings replace parents.

VI. METHODOLOGY

In this paper we have proposed a solution for Travelling Salesman problem using ABC and GA with Employed and onlooker and scout bee of ABC (RBGCA). For solving any optimization problem we have to first formulate the problem according to optimization problem. In this case first we formulate the Travelling salesman problem according to our proposed RBGCA. Next subsection describes how we formulate the Travelling salesman problem.

A. Representation

To solve the problem, representation of the individual and fitness value is required. Real coded bee Genetic colony algorithm is based on population (candidate solution) and each population have its own fitness value according to which it is compared from others, so we have to first represent the travelling salesman problem in terms of RBGCA.

In TSP, we have a set of city, path from one city to other and cost associated to the path as input, which informs that which path is to be operated on which city and in which order as output. RBGCA is based on population concept and each individual in population represents a solution, in case of TSP, solution is a sequence of path traversing each city with least cost. So we have to first formulate each individual of RBGCA. Our proposed RBGCA algorithms generates the real value vector for an individual but the TSP problem is a discrete optimization problem so we have to map the generated real value vector in to discrete values that used for a TSP problem. We have used a SPV rule (shortest position value) to map the generated real value vector to discrete value vector. For every generated individual T_{id} of RBGCA we have generated another vector of discrete value S_{ij} called sequence vector using SPV rule. The output for TSP problem is an optimal set of sequence of cities through which salesman has to travel. The sequence vector S_{ij} generated for an individual used as

sequence for salesman to travel the cities. We represent dimension of an individual as a number of city. The dimension value of each individual contains the real values from the search space. Each individual is represented by T_{id} = {t_{i1},t_{i2},t_{i3},…….t_{id}} and for each individual there is a Sequence vector S_{id} = {Si_1 , Si_2, Si_3, ...Si_d} where *i* is the particular individual and *d* represents the dimension index, is calculated using RBGCA with SPV rule. Sequence vector of each individual makes transformation on the basis of individual real value vector T_{id} = {t_1,t_2,t_3,…….t_d}. The individual gets it value random initially and the associated sequence vector is transformed by using SPV rule on individual. Crossover operator of real coded genetic algorithm is applied to each individual and updated individual is generated, associated sequence vector is generated for updated individual. In our approach we have taken linear crossover as crossover operator and for mutation uniform mutation is chosen. The individual or real value vector has real value which is transformed to discrete value. Smallest position value i.e. SPV rule is used to find a permutation corresponding position $T_{id.}$ The Individual T_{id} has continuous values. By using the SPV rule this continuous position value can be converted to discrete value transformation S_{id}= [s_{i1}, s_{i2},…….s_{id}]. S_{id} is the sequence of path of *i* individual in the processing order with respect to the *d* dimension. These newly generated individuals are then being updated by employed bee and onlooker bees of Bee colony algorithm and again the transformation of updated particles to the sequence vector is done. After each modification of an individual the sequence vector associated with that individual is calculated.

For ex. if we have 6 cities in Figure 1 then we have dimension value as 6. Based on SPV rules, the continuous position or individual convert to a transformation of sequences S_{id}, which is a sequence of city implied by the individual T_{id}. Individual T_{id} is calculated using RBGCA={4.83,-0.55,1.90,3.46,1.05,2.87,-0.28,0.19,4.0,2.28}

Then using SPV rule transformation of T_{id} to S_{id}, we have S_{id} value as {9,0, 4, 7,3,6,1,2,8,5}. These values are sorted and their indexes are kept in a vector which will be used for accessing cost associated to path which is stored in 2 dimensional arrays in priory.

B. Fitness Function

After representation of each individual first we have to calculate fitness value of each individual. On the basis of fitness value we determine the optimal solution. In case of TSP optimal solution is the minimization the value of equation (2). Equation (2) is the addition of the travelling cost from one city to another in a path to travel the whole city. For calculating the fitness value first we have to calculate the cost matrix. Cost matrix contains the cost of travel from one city to another. Our main objective is to minimize the fitness value, an individual who have the minimum fitness value is considered as the optimal solution.

C. Real Bee Genetic Colony

To solve the Travelling salesman problem we have used the Real Genetic Bee Colony Algorithm. We set an initial population by selecting random starting values from the search space than the sequence vector associated with each individual is calculated; sequence vector is a member of the set of x! Sequences; where x is the total number of city. After getting the initial population and associate initial sequence vector we calculate fitness value of each individual, according to equation (2).In the next step these individuals are visited by employed bees of Bee colony. These operator used for local search to avoid local max problem. In this phase produce new candidate solution and after that it produce the candidate sequence on the basis of candidate solution and compute the fitness of individual and if the fitness of new candidate solution is better than the exsisting solution then it replace the older solution and its sequence vector and calculate the probability for each individual. After that if crossover criteria is satisfied, then crossover operation performed over two randomly selected individuals and as a result a new individual and sequence is generated replacing the worst individuals and sequences based on fitness value. After that the onlooker bee phase apply which is also the operator of bee colony and it also generates the new vector and generates a new sequence vector . Then the cost of this offspring is calculated. Using the sequence and its cost from the cost matrix the fitness value of the each individual is calculated.

Algorithm 1: Travelling Salesman Problem using RBGCA

[Initialization Phase]
for p=0 to population size **do**
for d=0 to dimension size **do**
Randomly initialize particle
Using SPV rule a sequence vector is generated
end for d
Compute fitness of that particle
end for s

Repeat
[Employed Bee Phase]
for i=0 to max no of employed bee **do**
for d= 0 to dimension **do**
produce new candidate solution
produce candidate sequence on the basis of candidate solution
end for d
Compute fitness of individual
if fitness of new candidate solution is better than the exsisting solution replace the older solution and its sequence vector.
end for i
Calculate the probability for each individual.
[Update Phase]
[Crossover Operator Phase]
if crossover criteria is met then
Select two random individuals from current population for crossover operation
Apply crossover operation to generate new individuals new offspring generated from parents as a result of crossover.
New set of sequence vector is generated for new offspring
Compute the cost for that offspring

Compute the fitness of updated individual
Replace the worst parent and associated sequence individual with new best offspring and its sequence vector if it is better Update individuals

[Onlooker Bee Phase]

for i=0 to max no of onlooker bee **do**
choose food source on the basis of probability
for d= 0 to dimension **do**
produce new candidate solution
produce candidate sequence on the basis of candidate solution
end for d
compute fitness of individual
if fitness of new candidate solution is better than the existing solution replace the older solution and its sequence vector.
end for i

[Scout Bee Phase]

If any food source exhausted
Then replace it by random position generated by scout
memorize the best solution and sequence found so far

until(stopping criteria is not met**)**

VII. Experimental Setup And Results

A. Experiential Setup

For every algorithm there are some control parameters which are used for its efficient working. Hence, there are some control parameters for Real Coded Genetic Bee Colony Algorithm also.

The first control Parameter is Maximum cycle number: Maximum number of cycles (MCN) equals to the maximum number of generation, we have taken the result for 1500, 2000
and 2500 MCN value. The next parameter in our experiment is maximum number of population and we have taken its value to be 30 and 60. Another control parameter is number of runs and we have taken its value in our experiment as 30. It must be noted that each run contains maximum cycle number, which is 1500, 2000, 2500 in our experiment. The fourth control parameter is Dimension and it depends upon the number of city. In this experiment we are using the feature of linear crossover operator in the RBGCA algorithm. The control parameter for Crossover operator is Probability. Therefore we need to find the value of this parameter also. Its value can range from 0.1 to 0.9.

B. Experimental Result

In this section we analyze the result obtained by our algorithm. To test the efficiency of our algorithm results of RBGCA is compared with real coded Genetic algorithm and results. In a TSP we already have the information about the number of cities, path, and the cost associated with path that will be taken by a salesman to complete task. We just need to find the sequence of path which will provide us the optimal results. We conducted the experiment by varying the number of cities as well as varying the cost associated with path and then we compared our results with that of GA. In particular, we have taken two cases in which we have taken different number of cities and cost.

Experiment 1: Here, we are assuming there are 30 cities. Following are the execution time (in units) taken by RBGCA and GA.

TABLE I
DISTANCE CALCULATED BY GA AND RBGCA FOR 30 CITIES

No.of City	Cycles per evaluation	Means of 30 runs GA	Means of 30 runs RBGCA
30	1500	981.8	317.2
30	2000	966.63333	300.4
30	2500	950.5	329.233333

The sequence generated by GA is: 0, 25, 28, 23, 14, 18, 19, 22, 16, 9, 5, 6, 24, 15, 4, 11, 13, 21, 12, 27, 1, 20, 2, 7, 3, 17, 29, and 10,26,8.

The sequence generated by RBGCA is:4, 27, 9, 26, 2, 13,
16,15,6,10,24,5,17,3,7,11,14,23,8,29,21,20,19,22,0,18,12,25,28,28

Experiment 2: Here, we are assuming there are 60 cities. Following are the execution time (in units) taken by RBGCA and GA.

TABLE II
DISTANCE CALCULATED BY GA AND RBGCA FOR 60 CITIES

No.of City	Cycles per evaluation	Means of 30 runs GA	Means of 30 runs RBGCA
60	1500	2294.2	812.244322
60	2000	2328.366667	801.7
60	2500	2294.2	798.333333

The sequence generated by GA is: 0, 35, 34, 17, 9, 51, 59, 58, 19, 55, 13, 14, 33, 20, 39, 8, 56, 57, 45, 44, 2, 58, 12, 53, 25, 48, 38, 43, 21, 5, 4, 46, 6, 10, 36, 41, 10, 15, 18, 49, 32, 24, 50, 22, 52, 54, 11, 29, 47, 23, 56, 40, 27, 30, 42, 31, 7, 37, 1, 3.

The sequence generated by RBGCA is: 50, 40, 57, 21, 49, 7, 51, 11, 31, 13, 8, 16, 6, 18, 37, 46, 39, 44, 52, 22, 27, 20, 45, 33, 55, 41, 3, 17, 14, 23, 15, 32, 19, 25, 48, 10, 0, 24, 35, 28, 38, 12, 54, 47, 1, 26, 59, 42, 50, 9, 58, 43, 56, 53, 2, 36, 4, 34, 29,30.

The sequences generated by the algorithms shows the path for salesman to travel. The sequence shows the city number through which salesman travel one by one. For e.g. in sequence:0, 22, 5, 17, 20, 27, 12, 18, 14, 28, 25, 11, 16, 15, 29, 21, 19, 2, 4, 1, 3, 7, 8, 23, 13, 24, 10, 26, 9, 6.

Salesman starts from city 0 than move on to city 22 and then city 5 and so on it last visits the city 6 and goes back to 0. The above sequence is generated by RBGCA for 30 cities.

From the Table I and II it is clear that our proposed RBGCA performs better than the real coded genetic algorithm in every case.

VIII. Conclusion

It can be concluded from the results TABLE I and II that proposed RBGCA performs better than the existing GA algorithm. There is no specific value for crossover probability for which we can obtain best results for TSP. It depends upon number of cities and path. The procedure

followed in TSP consists of the generation of the population according to the algorithm, then the path sequence and cost associated with the path sequence are generated, each individual and the path sequence vector set are updated using crossover, employed, onlooker and scout operators. It is repeated again and again till the maximum number of cycles.

As future work we have the intention to apply other types of nature inspired algorithms to the Travelling salesman problem, comparing their results with the ones accomplished by the Real Bee Genetic Colony Algorithm. In future many crossover operators can be applied with RBGCA to solve the Travelling Salesman Problem. We also conclude that the merging of bee colony optimization algorithm's operator with

real coded genetic algorithm improves the real coded genetic algorithm performances. We can also use our proposed algorithm RBGCA to solve various optimization problems.

REFERENCES

[1]. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm :By Dervis Karaboga · Bahriye Basturk Received: 31 May 2006 / Accepted: 12 February 2007 / Published online: 13 April 2007 © Springer Science+Business Media B.V. 2007.

[2]. L. P. Wong, M. Y. H. Low and C. S. Chong, "Bee Colony Optimization with Local Search for Travelling Salesman Problem," International Journal on Artificial Intelligence Tools (IJAIT), vol. 19, pp. 305-334, 2010.

[3]. D. Karaboga and B. Akay, "Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization," pp. 1-6

[4] A Hybrid Artificial Bee Colony Algorithm for Flexible Job Shop Scheduling Problems: By J. Li, Q. Pan, S. Xie, S.Wang(Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. VI (2011), No. 2 (June), pp. 286-29

[5] G. Zhao, W. Luo, H. Nie, C. Li, "A Genetic Algorithm Balancing Exploration and Exploitation for the Travelling Salesman Problem," in Proceedings of the 2008 Fourth International Conference on Natural Computation, 2008, pp. 505-509.

[6] Dervis Karaboga · Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm" J Glob Optim (2007), pp 459-471

[7] W.-L. Zhong, l Zhang, W.-N. Chen, "A Novel Discrete Particle Swarm Optimization to Solve Traveling Salesman Problem," in Proc. IEEE Int. Conf. Evol. Comput. (CEC), 2007, pp. 3283-3287.

[8] L.-P. Wong, M.Y. Hean Low, C.S. Chong, "A Bee Colony Optimization Algorithm for Traveling Salesman Problem," Second Asia International Conference on Modelling & Simulation, 2008, pp. 818-823.

[9] S. Nonsiri, S. Supratid, "ModifYing Ant Colony Optimization," IEEE Conference on Soft Computing in Industrial Applications, 2008, pp. 95-100.

[10] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey, Technical Report-TR06, 2005.

[11] D. Karaboga, B. Akay, "A Survey: Algorithms Simulating Bee Swarm Intelligence," Artificial Intelligence Review., vol. 31, pp. 68-85, 2009.

[12] Z. Michalewicz, "Genetic algorithm + data structure = evolution pmgram",Springer-Verlag, Inc., Heidelberg, Berlin, 1996.

[13] G-. G. Jin, S-. R. Joo, "A Study on a Real-Coded Genetic Algorithm,"Journal of Control, Automation, and Systems Engineering, vol. 6, no. 4, pp. 268-274, April 2000.

[14] L. J. Eshelman, R. A. Caruana, and J. D. Schaffer, "Bases in the crossover landscape," Pmc. 3rd Int. Con$ on Genetic Algorithms, J. Schaffer(Ed.), Morgan Kaufmann Publishers, LA, pp.10-19, 1989.

[15] Chaotic Bee Swarm Optimization Algorithm for Path Planning of Mobile Robots Jiann-Horng Lin and Li-Ren Huang Department of Information Management I-Shou University, Taiwan 2009

[16] V.Singh,D.Singh R.Tiwari, A.Shukla, "RGBCA -Genetic Bee Colony Algorithm for Travelling Salesman Problem"IEEE Information & Communication Technologies(WICT),2011pp.1002-1008

[17]. P. W. Tsai1, J. S. Pan1, B. Y. Liao1, and S. C. Chu, "Enhanced Artificial Bee Colony Optimization," International Journal of Innovative Computing, Information and Control, vol. 5, pp. 1-14, Dec. 2009.