On the Solutions to the Travelling Salesman Problem using Nature Inspired Computing Techniques

Hifza Afaq[1] and Sanjay Saini[2]
Department of Physics and Computer Science, Dayalbagh Educational Institute,
Agra, 282 010, India

Abstract
This paper attempts to bring forward various newly emerged natural computing techniques to a common platform. Six such techniques are compared among each other which have been used to solve a well known classical problem, the travelling salesman problem. The techniques discussed in this paper are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Bacterial Foraging Optimization Algorithm (BFOA), Gravitational Search Algorithm (GSA), Intelligent Water Drops (IWD), and River Formation Dynamics (RFD). In the end, some important results have been tabularized.

Keywords: *Ant Colony Optimization (ACO), Bacterial Foraging Optimization Algorithm (BFOA), Gravitational Search Algorithm (GSA), Intelligent Water Drops (IWD), Particle Swarm Optimization (PSO), River Formation Dynamics (RFD), Travelling Salesman Problem (TSP).*

1. Introduction

Natural systems are one of the rich sources of inspiration for inventing new intelligent systems. Swarm intelligence is one of the scientific fields that are based on natural swarms, such as ant colonies, bee colonies, group behavior of bacterium, rivers (group of water drops) etc. Evolutionary computation [9], neural networks [8], time adaptive self-organizing maps [22], ant colony optimization [7], bee colony optimization [18], particle swarm optimization [13], DNA computing [2] electromagnetism-like optimization [3] intelligent water drops [20] etc. are some of the techniques which are to some extent based on or takes inspiration from naturally occurring phenomena. Nature has inspired many heuristic algorithms to obtain reasonable solutions to complex problems. One of these algorithms is ant colony optimization (ACO) which is based on the food foraging behavior of real ant colonies. Ants leave their nest and move in random directions to search for the food. It has been observed that ants somehow find the shortest path from their nest to the food source. ACO proposed by Marco Dorigo mimics this path optimization strategy of the real ants. Particle Swarm Optimization (PSO) algorithm is a swarm based optimization algorithm. Particles search for the better solutions throughout the space by updating its position and velocity. This algorithm takes inspiration from natural swarms such as birds. Birds fly in coordination without communicating, in absence of any clear leader. In PSO, particles neither have a clear leader nor they communicate, but they influence each other to move in a better direction to find solution. Some of more recent techniques involve Bacterial Foraging Optimization Algorithm (BFOA). Bacterium search for the nutrient in a manner to maximize energy obtained per unit time. This foraging behavior of E.coli bacteria is the key idea for BFOA. River Formation Dynamics (RFD) is a heuristic optimization method based on imitating how water forms rivers by eroding the ground and depositing sediments. After drops transform the landscape by increasing/decreasing the altitude of places, solutions are given in the form of paths of decreasing altitudes. Decreasing gradients are constructed, and these gradients are followed by subsequent drops to compose new gradients and reinforce the best ones. On the other hand the Intelligent Water Drop (IWD) Algorithm mimics the path finding strategy of water drops. Natural rivers often find good paths among lots of possible paths in its ways from the source to destination. These near optimal or optimal paths are obtained by the actions and reactions that occur among the water drops and the water drops with the riverbeds. Gravitational search algorithm is based on the basic laws of motion given by Newton. In GSA all the solutions have mass, heavier an object is better it serves the problem as its solution. All the objects moved towards heavier objects obeying the gravitational laws. All these techniques somehow search the space for better solutions and so optimize the problem under consideration.

The travelling salesman problem is a classical problem which belongs to the class of NP-hard problems. It is one of the intensively studied problems. Scientists are still trying to explore the new ways which are more efficient to find the shortest route for a travelling salesman. The traveling salesman has to leave home, stop in *N* number of cities, each city visited exactly once except the first city which is visited twice, and returns home by traveling the shortest distance between all the cities. However, it is very hard, if not impossible, to find the shortest path for a

travelling salesman because the number of different routes that can be taken increases exponentially as the number of cities increases. A number of artificial Intelligence techniques have been implemented to find the optimal or near-optimal path by various researchers. Some of the approaches which have been used to solve TSP by researchers are Particle Swarm Optimization (PSO) [29], Genetic Algorithm (GA) [27], and Ant Colony Optimization (ACO) [5] in the recent past. On the other hand some newly emerging techniques have been also applied to solve the travelling salesman problem such as Intelligent Water Drop (IWD) algorithm [21], Bacterial Foraging Optimization Algorithm (BFOA) [12], River Formation Dynamics (RFD)[16], Gravitational Search Algorithm [1] etc. Some of the existing techniques that have been applied to travelling salesman problem are discussed briefly in section 2 namely ant colony optimization, intelligent water drops, particle swarm optimization and river formation dynamics. Section 3 discusses and compares the results of the work done by various researchers on travelling salesman problem. Conclusion and Future Scope are given in section 4 and 5 respectively.

2. Various Techniques to solve the Travelling Salesman Problem

The travelling salesman problem is a classical NP-hard problem. It is one of the most studied discrete optimization problems. TSP has many variations and a large number of applications.

Heuristics inspired by nature have been one of the most important and promising research fields in recent years. These heuristics, utilizes analogies with natural or social systems, are used to derive non-deterministic heuristic methods and obtain very good results in NP-hard combinatorial optimization problems [19].

2.1. Ant Colony Optimization

The inspiring source of ACO is the food foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During their return trip, ants deposit a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. The main principle behind these interactions is called stigmergy, or communication through the environment. An example is pheromone laying on trails followed by ants. Pheromone is a potent form of hormone that can be sensed by ants while traveling along trails. It attracts ants and therefore ants tend to follow trails that have high pheromone concentrations. This causes an autocatalytic reaction, i.e., one that is accelerated by itself. Ants attracted by the pheromone will lie more of the same on the same trail, causing even more ants to be attracted see Figure 1. This characteristic makes swarm intelligence very attractive for network routing, robotics, optimization etc. A number of extensions are proposed to the original ant algorithm. These algorithms performed better producing much improved results than the original ant algorithm.

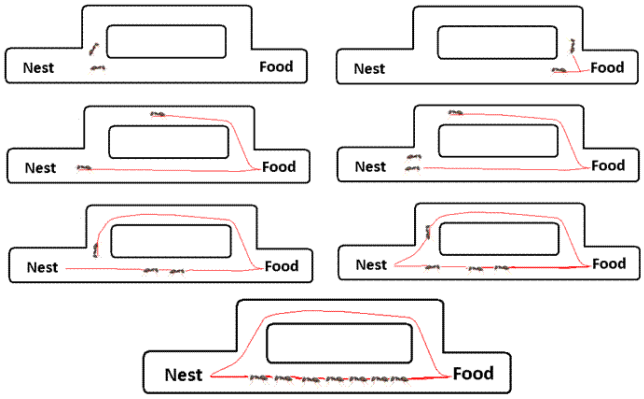

Figure 1 Optimization by Ant Colony.

Main characteristics of this model are positive feedback, distributed computation, and the use of a constructive greedy heuristic. The basic algorithm introduced by Marco Dorigo is given by following steps:

1. *Set parameters, initialize the pheromone trails*
2. *while (termination condition not met) do*
   a. *Construct ants solutions*
   b. *Apply local search*
   c. *Update pheromones*
   d. *end while*

2.2. Bacterial Foraging Optimization

The Bacterial Foraging Optimization Algorithm was proposed by K. M. Passino in 2002. Bacterium search for the nutrient in a manner to maximize energy obtained per unit time. The process, in which, bacterium moves in the medium by taking small steps while searching for nutrients, is called chemotaxis. BFOA mimics this chemotactic movement of bacteria in the problem space. In the process of foraging, E. coli (Escherichia coli) bacteria (bacteria considered by Passino in his original paper) undergo four stages, i.e., (i) Chemotaxis (ii) Swarming (iii)

Reproduction and (iv) Elimination and Dispersal. Bacterium can move in two ways (i) swim and (ii) tumble. E.coli has 8-10 flagella placed randomly on a cell body. When bacterium rotates its flagella counterclockwise the bacterium swims in the medium in order to move forward in a direction. On the other hand when it is rotated clockwise it tumbles to change the direction of movement (See Figure 2).

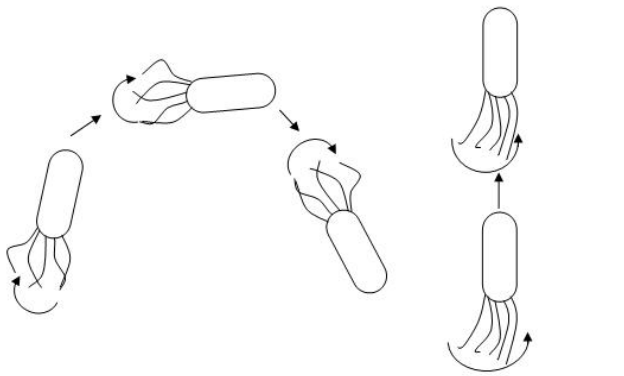

Figure 2 Bacterial Motion.

a. *Chemotaxis:* Biologically, bacterial chemotaxis is a complex combination of swimming and tumbling that keeps bacteria in places of higher concentration of nutrients. Bacteria while moving in the medium to explore and exploit the nutrients available in the medium, alters its motion between swim and tumble. It swims frequently as it approaches a nutrient gradient so as to move in the same direction to exploit the available nutrient. When it is required to change the direction of motion, to explore the medium for high concentration of nutrient gradient, the bacteria tumbles by moving its flagella clockwise. The motion patterns that the bacteria will generate in the presence of chemical attractants and repellents are called chemotaxis. A bacterium alternates between these two modes of motion for its entire lifetime. This alternation between the two modes enables the bacterium to move and search for nutrients.

b. *Swarming:* An interesting group behavior has been observed for several motile species of bacteria including E.coli When a group of E.coli is placed in the center of a semisolid agar with a single nutrient chemo-effector, they move out from the center in a traveling ring of cells by moving up the nutrient gradient created by consumption of the nutrient by the group.

c. *Reproduction:* E. coli asexually split into two to produce an exact copy of itself. The term reproduction means that each of the healthier bacteria (those optimizing the objective function) asexually split into two bacteria, which are then placed at a location from where a least healthy bacteria has been removed. This keeps the swarm size constant.

d. *Elimination and Dispersal:* The elimination and dispersal events can occur biologically such that bacteria in a region are killed or a group is dispersed into a new part of the environment due to some influence such as increase in temperature etc. This process have the effect of possibly destroying chemotactic progress because it could disperse the bacteria from a region of high concentration of nutrient gradient, but they also have the effect of assisting in chemotaxis, since dispersal may place bacteria near/in better regions. Dispersal leads to diversity; it also strengthen the ability of global optimization.

This phenomenon is simulated by liquidating some bacteria at random with a very small probability and new replacements are randomly initialized over the search space [4]. In search space, BFOA seek optimum value through the chemotaxis of virtual bacteria, and realize quorum sensing via assemble function between bacterial, and satisfy the evolution rule of the survival of the fittest to make use of reproduction operation, and use elimination-dispersal mechanism to avoiding falling into premature convergence.

2.3. Gravitational Search Algorithm

GSA is basically based on the Newtonian laws of gravity and motion. The main idea of the GSA is to consider an isolated system of masses, where every mass represents a solution to a certain problem It is like a small artificial world of masses obeying the Newtonian laws of gravitation and motion. Acoording to the Newton's law of gravitation, each particle attracts every other particle with a force directly proportional to the product of their masses and inversely proportional to the square of the distance between them. Each agent, which is a potential solution to the problem at hand, is considered as an object having some mass. Mass of the objects is calculated by a fitness function. Performance of these objects is measured by their masses, objects having higher mass values are nearer to the best solution. All these objects attract each other by the gravitational force, heavy objects attracts all other objects. So, this force causes a movement of all objects globally towards the heavy objects.

In GSA, each object has four specifications as stated by Abarghouei [1],

1. Position
2. Inertial mass
3. Active gravitational mass
4. Passive gravitational mass

The position of the mass corresponds to a solution of the problem, and its gravitational and inertial masses are determined using a fitness function. In other words, each mass presents a solution, and the algorithm is navigated by properly adjusting the gravitational and inertial masses. By lapse of time, we expect that masses be attracted by the heaviest mass. This mass will present an optimum solution in the search space. GSA can be summarized in steps 1 to 6.

1. *Generate initial population.*
2. *Evaluate fitness of each agent.*
3. *Update gravitational constant and objects having best and worst fitness.*
4. *Calculate mass and acceleration for each agent.*
5. *Update velocity and position of each agent.*
6. *If the termination criterion is met then stop else repeat steps from 2 to 5.*

2.4. Intelligent Water Drops

Intelligent Water drops Algorithm was introduced by Shah-Hosseini, H. in 2007 [20]. It is a population based constructive optimization algorithm which has been inspired from natural rivers and exploit the path finding strategies of rivers. A natural river often finds good paths among lots of possible paths in its ways from the source to destination. These near optimal or optimal paths follow from actions and reactions occurring among the water drops and the water drops with their riverbeds. In the IWD algorithm, several artificial water drops cooperate to change their environment in such a way that the optimal path is revealed as the one with the lowest soil on its links. The solutions are thus incrementally constructed by the IWD algorithm.

In the original IWD algorithm the water drops are created with two main properties:

1. Velocity
2. Soil

Both of above mentioned properties of IWD may change during its lifetime. The IWD begins its trip from a source to reach some destination with an initial velocity and zero soil. During its trip, an IWD travels in the environment from which it removes some soil and may gain some speed. This soil is removed from the path joining the two locations. IWD is supposed to flow in discrete steps. From its current location to its next location, its velocity is increased by the amount that is non-linearly proportional to the inverse of the soil between the two locations and the amount of soil added to the IWD is non-linearly proportional to the inverse of the time needed for the IWD to pass from its current location to the next location. Therefore, a path having less soil lets the IWD becomes faster than a path having more soil and the time interval is calculated by the laws of physics of linear motion. Thus, the time taken is proportional to the velocity of the IWD and inversely proportional to the distance between the two locations. An IWD prefers the paths having low soils than the paths having high soils.

The IWD algorithm as specified by Shah-Hosseini H. in [20] is as follows:

1. *Initialization of static parameters.*
2. *Initialization of dynamic parameters.*
3. *Spread the IWDs randomly on the nodes of the graph.*
4. *Update the visited node list of each IWD.*
5. *Repeat Steps a to d for those IWDs with partial solutions.*
   a. *For the IWD residing in node i, choose the next node j, which does not violate any constraints of the problem and is not in the visited node list of the IWD.*
   b. *For each IWD moving from node i to node j, update its velocity.*
   c. *Compute the soil.*
   d. *Update the soil.*
6. *Find the iteration-best solution from all the solutions found by the IWDs.*
7. *Update the soils on the paths that form the current iteration best solution.*
8. *Update the total best solution by the current iteration - best solution.*
9. *Increment the iteration number*
10. *Stops with the total best solution.*

2.5. Particle Swarm Optimization

Small birds fly in coordination and display strong synchronization in turning, initiation of flight, and landing without any clear leader. They follow certain rules for their movement. Simulation of a bird swarm was used to develop a particle swarm optimization (PSO) concept by Kennedy & Eberhart in 1995 [13]. Searching procedures by PSO can be described as follows:

1. *Each particle (or agent) evaluates the function to maximize at each point it visits in space.*

2. *Each agent remembers the best value it has found so far (pbest) and its co-ordinates.*
3. *Each agent know the globally best position that one member of the flock had found, and its value global best (gbest).*
4. *Using the co-ordinates of pbest and gbest, each agent calculates its new velocity and position.*

Here, pbest is the previous best position of each particle stored in its memory and gbest is the best position found by the swarm as a whole hence called global best. The velocity and position updates are given by Eq. (1) and Eq. (2).

$$\vec{v_i}(t+1) = w\vec{v_i}(t) + \phi_1 \otimes (\vec{p_i} - \vec{x_i}(t)) + \phi_2 \otimes (\vec{p_g} - \vec{x_i}(t)) \quad \ldots(1)$$

$$\vec{x_i}(t+1) = \vec{x_i}(t) + \vec{v_i}(t+1) \quad \ldots(2)$$

Here $v_i(t)$ is the velocity of i^{th} particle at a time t and $v_i(t+1)$ represents the velocity of i^{th} particle at a time t+1, p_i is the previous best position of i^{th} particle and p_g is the global best position of the swarm, $x_i(t)$ represents the previous position of the particle (position at time t). w is the inertia weight to limit the influence of the previous velocity on the velocity of the particle, The inertia factor (w) was not there in the original version of PSO proposed by Kennedy, it was later introduced to limit the effect of previous velocity on swarm. $\phi_1 = c_1 R_1$ *and* $\phi_2 = c_2 R_2$ where R_1 and R_2 are two random numbers such that $0<R_1<1$ and $0<R_2<1$. PSO equations have some parameters that are to be adjusted by the user. Many researchers and scientists focused their attention to different parameters used in PSO to produce better results. Pederson, M. E. H. [14] explains in Good Parameters for Particle Swarm Optimization that particle swarm optimization has a number of parameters that determine its behavior and efficacy in optimizing a given problem. Shi and Eberhart [24] in late 90's pointed out that for different problems, there should be different balances between the local search ability (pbest) and global search ability (gbest).

2.6. River Formation Dynamics

The River formation dynamics [17] is based on how water forms rivers by eroding the ground and depositing sediments. Local flat environment is provided at the beginning. The drops are moved on different paths. A drop selects different paths depending on some probability criteria. After drops transform the landscape by increasing or decreasing the altitude of places, solutions are given in the form of paths of decreasing altitudes. Decreasing gradients are constructed, and these gradients are followed by subsequent drops to compose new gradients and reinforce the best ones. The basic scheme of the RFD algorithm is as follows:

1. *Initialize drops*
2. *Initialize nodes*
3. *while (Not all drops follow same path and no other termination condition is met)*
   a. *Move drops*
   b. *Erode paths*
   c. *Deposit sediments*
   d. *Analyze paths*
   e. *end while*

3. Comparison of Techniques

Various techniques have been implemented to solve the travelling salesman problem. This section puts some emerging nature inspired techniques forward which have been applied to TSP namely BFOA [12], GSA [1], RFD [16] and IWD [21] which are new in comparison with ACO [5], [6], [10], [11] and PSO [23], [25], [28] these techniques have been discussed briefly in section 2.

A salesman is required to visit all the cities of a given map to complete his tour. In this tour every city is visited exactly once except the first city which is visited twice to complete the tour. The goal in the TSP is to find the tour with the minimum total length among all possible tours. Travelling salesman problem can be represented in the form of a graph having node set *N* and edge set *E* where *N* represents the number of cities and *E* represent the set of edges or paths between each pair of nodes, representing the distance between cities. A solution of TSP, which can be represented by a graph (*N*, *E*), is then an ordered set of *n* distinct cities. A TSP solution for an *n*-city problem may be represented by the tour $T = (1, 2, ..., n)$. The salesman travels from city 1 to 2, then from 2 to 3 and he continues this way until it gets to city *n*. He then returns to the first city 1. If the distance between two cities *i* and *j* is same in each opposite direction, it means the path length or cost in moving from city *i* to city *j* is same as the path length or cost in moving from city *j* to city *i*, then it forms an undirected graph and problem is symmetric travelling salesman problem. On the other hand, if these distances are not same or a path exists from city *i* to *j* but does not exist from *j* to *i* or vice versa then the problem is asymmetric travelling salesman problem.

Ant colony optimization has been applied to the travelling salesman problem for long. Dorigo and Gamberdella [6] applied ant colony system to both symmetric and asymmetric travelling salesman problem. According to them artificial ant is an agent which moves from city to city on a TSP graph and finds optimal path as the real ants

do. They applied Ant Colony System (ACS) approach to various standard benchmark functions to the symmetric and asymmetric TSPs. The result obtained show that ACS finds results which are at least as good as, and often better than, those found by then available techniques. They modified ACS to deal with bigger TSPs and they reported that the modified version improved the performance of ACS. Dorigo and Gambardella [5] applied ant colony approach to TSP and compared the results obtained with other heuristics available in the literature at that time. They tested the algorithm on two sets of TSP problems, first one comprising five randomly generated 50-city problems, while the second one comprised of three geometric problems (real geographic problems) that consists of 50 to 100 cities. By the results obtained they concluded that the ant colony system almost always offered the best performance when compared with the results obtained by the existing techniques. They also modified and tested the algorithm for bigger TSP. This algorithm was able to find good results for problems up to more than 1,500 cities. Also, the time to generate a tour grows only slightly more than linearly with the number of cities. Gambardella and Dorigo [10] hybridized ant system with Q learning and proposed Ant-Q algorithms. They applied this concept to get the solution for both symmetric and asymmetric instances of TSP. They performed experiments to investigate the function of Ant-Q. The results obtained by Ant-Q on symmetric TSP were competitive with those obtained by other heuristic approaches based on neural networks or local search. On the other hand results obtained by Ant-Q were very good when applied to some difficult asymmetric TSP's. Gambardella and Dorigo [11] developed ACS by modifying Ant-Q algorithm. They opted for a local trial updating policy to improve the performance of the system in term of speed and quality of results.

Zhong, W and Zhung, J. [29] modified original particle swam optimization algorithm [13] to solve the travelling salesman problem. In the original PSO, x_i represents the position of a particle, this position vector x_i represents a set of edges in the modified algorithm. All the edges connecting two cities have a probability to be selected. A velocity vector is used to update the position of particles. Zhong, W and Zhung, J. introduced a new parameter c_3 in Eq. (2) as given by Eq. (3).

$$\vec{x_i}(t+1) = \vec{v_i}(t+1) \oplus c_3 R_3 * \vec{x_i}(t) \quad \ldots(3)$$

where R_3 is a random number between 0 and 1 and c_1, c_2 in Eq. (1) and c_3 in Eq. (3) all are greater than 1. Zhong, W and Zhung, J. tested their algorithm with and without the effect of c_3 on six standard test functions of tsp library [26] namely Burma14, Eil51, Berlin52, Eil76, KroA100 and KroA200. They run the algorithm 50 times for each test function. By the results obtained it can be observed that C3DPSO (Discrete PSO with c_3 [29]) overcomes PSO without c_3 with better running length, lesser error and better success rates. For smaller number of nodes both C3DPSO and PSO without c_3 completes with almost same running lengths but the other three measures for C3DPSO are better. However, as the number nodes increases C3DPSO clearly outperforms the other. As stated by Zhong, W and Zhung [25], J. the mutation factor (c_3) play an important role in enhancing the searching ability of the algorithm. The modified algorithm gives better results in precision or computational cost. In another work Tasgetiren, Suganthan, & Pan applied a discrete version of PSO to the generalized TSP. The discrete particle swarm optimization algorithm introduced by them exploits basic features of continuous counterpart. To improve the quality of the solutions obtained they hybridized the PSO with a local search, variable neighbourhood descend algorithm. They tested this algorithm on a set of standard benchmark instances ranging from 51 to 442 nodes. The results obtained show that the discrete particle optimization algorithm is one of the best performing algorithms together with the GA and FST-Root algorithms for the generalized TSP (GTSP). Shi, Liang, Lee, Lu and Wang [23] proposed a particle swarm optimization based algorithm for the traveling salesman problem. They hybridized the proposed algorithm with crossover to accelerate the convergence speed of algorithm. They compared their algorithm with the other existing swarm intelligence algorithm for solving TSP and got better results. They extended the proposed PSO and applied it to generalized traveling salesman problem by employing the generalized chromosome. Authors tested the algorithm on 19 instances from the TSPLIB [26] library to validate the proposed algorithm. The results obtained show that the discrete PSO method proposed by Shi et al. [23] is effective in solving the generalized TSP. They also showed that this algorithm can also be used to solve larger problem.

Rabanal et al. [15] applied RFD to solve the travelling salesman problem. At the beginning of algorithm all nodes have same altitudes except the destination node, which is a hole. A drop starts from a node and take different paths, the choice of path is determined by the probability of the drop to take a given path. In the algorithm given by Rabanal et al. the probability to choose a particular path or edge instead of others is proportional to the gradient of the downward slope in the edge, which in turn depends on the difference of altitudes between both nodes and the edge distance either the cost of the edge. Drops are unleashed at the origin node and move around the flat environment until some of them fall in the destination node. This erodes

adjacent nodes, which creates new downward slopes, the process of erosion is propagated in this way. New drops are inserted in the origin node to transform paths and reinforce the erosion of promising paths. The RFD algorithm is tested on standard benchmark functions of tsp library [26] namely Eil51 and KroA100. In their paper [15] they compared the results obtained by RFD to the results obtained by the ant colony optimization algorithm. They state that RFD explores the graph more deeply than the ACO. Initially ACO provided better results than the proposed algorithm but as the time passes the performance of RFD becomes better than ACO. Performance of RFD is improved when number of nodes increases from 51 to 100.

Bacterial foraging algorithm proposed by Good and Sahin [12] when applied to the traveling salesman problem having fewer cities consistently finds the shortest path but algorithm starts to diverge from finding the optimal solution in the 14 city problem. The problem continues to become worse as the number of cites increases.

Shah-Hosseini, H. [20] applied intelligent water drop algorithm to solve the travelling salesman problem. In order to use the IWD algorithm for the TSP, the TSP problem as mentioned above is viewed as a graph (N, E). Each link of the edge set E has an amount of soil. An IWD can travel between nodes of the graph through these links and is able to change the amount of the soils on the links. Moreover, cities of the TSP are denoted by nodes of the graph, which hold the physical positions of cities. An IWD starts its tour from a random node and it visits other nodes using the links of the graph until it returns to the first node. The IWD changes the soil of each link that it flows on while completing its tour.

For the TSP, the constraint that each IWD never visits a city twice in its tour must be kept satisfied. Therefore, for the IWD, a visited city list V_c (IWD) is employed. This list includes the cities visited so far by the IWD. So, the next possible cities for an IWD are selected from those cities that are not in the visited list V_c (IWD) of the IWD. First of all, he initialized the static and dynamic parameters. Static parameters include number of water drops, number of cities and the Cartesian coordinates of each city. Number of water drops to be used in the algorithm is the only static parameter to be set by user, which the author set equal to number cities for the present problem, the rest are problem dependent. Some parameters such as a_v, b_v, c_v are to be set for updating velocity while some other such as a_s, b_s, c_s are to be set for updating soil. Soil and Velocity are initialized by the zero and *InitVel* respectively which are to be set by user and the tour length is initially set to infinity. Dynamic parameters include visited node list *Vc*(*IWD*), which is initially empty. A city is randomly selected (Each city has a p robability to be selected) for each IWD and the visited node list and velocity is updated. Soil and time taken by an IWD to move from a node to the other node is computed for each IWD. The soil is then updated. Finally when tour is complete the tour length is computed and a tour with minimum tour length is marked as T_M. The results obtained were competitive with the other existing techniques.

Abarghouei [1] proposed a hybrid approach to solve the travelling salesman problem. He combined gravitational search algorithm with genetic algorithm and found promising results. The results he obtained show that the approach he proposed outperforms all of the other methods he included in their study for e.g. PSO, ACO, GA etc. He implemented the algorithm on six famous instances of the TSP and compared the results obtained with the results obtained by applying Particle Swarm Optimization method. Also, he implemented the Genetic Algorithm on the same instances and showed that the GA often falls in the trap of local optima. The final results he obtained proved the superiority of the method he proposed to all the other considered methods, both in terms of accuracy and time.

Table 1 summarizes the results obtained by applying a number of techniques for different instances of TSP by

Table 1: Comparison of various techniques to solve TSP.

Problem	ACO			PSO		IWD	RFD
	ACO [15]	ACS [11]	ACS [5]	PSO [29]	C3DPSO [29]	IWD-TSP [21]	RFD [15]
Eil51	457.97[1] (454.42)	428.06[2] (426)	N/A	473.34[3] (443)	433.64[3] (427)	432.62 (428.98)	458.08[1] (441.9)
Eil76	N/A	N/A	N/A	626.94[3] (566)	551.72[3] (540)	558.23 (549.96)	N/A
KroA100	N/A	21,282[2] (21,341)	21,282 (21,285.44)	27,725.4[3] (24,343)	21,689.30[3] (21,296)	21,904.03 (21,407.57)	N/A

[1] Results are averaged over 30 runs.
[2] Results are averaged over 15 runs.
[3] Results are averaged over 50 runs.

various researchers. The average and best results obtained are given. The results obtained by various researchers are averaged over different number of runs; the best result obtained is enclosed within the brackets. It can be concluded from the results under consideration that results obtained by applying C3DPSO [29] and IWD [21] are far better than ACO [15], PSO [29] and RFD.

The results obtained by applying ACO [11], [5] are competitive with C3DPSO [29] and IWD [21] however former results are averaged over fewer runs.

4. Conclusion

Algorithms such as ant colony optimization (ACO) and particle swarm optimization (PSO) that have been applied to the travelling salesman problem by various researchers in recent past have been briefly discussed in the present text. Some newly emerged techniques such as bacterial foraging optimization algorithm (BFOA), intelligent water drops (IWD), gravitational search algorithm (GSA) and river formation dynamics (RFD) which have been applied by scientists to solve TSP are also explored. Various results obtained by applying these techniques to different instances of travelling salesman problem that include up to 100 city problems have been included for comparison. Some of the techniques included in the present text produced very good results while others not. Among the algorithm discussed in the present text BFOA [12] is one with worst results. GSA [1] when hybridized with GA produced good results. C3DPSO [29] and IWD [21] clearly outperformed PSO. ACS [5] is competitive with these techniques for larger number cities.

5. Future Scope

A number of algorithms that have applied by various scientists and researchers to the travelling salesman problem have been discussed in the present text. Among these algorithms BFOA is one with worst results. BFOA could be modified by incorporating new ideas or hybridized with some other techniques to produce better results for TSP. Techniques such as RFD which are not very good can be explored to get better results. Weak points of the algorithm are to be identified and improved by combing it with other techniques which are efficient in those tasks particularly. C3DPSO and IWD produced good results for TSP up to 100 city problems; these techniques can be applied to larger TSP instances. It is to be tested whether the idea of these kinds of techniques could be combined to give a better technique so as to give better results for more complex TSPs.

References

[1] Abarghouei, A. A. (2010). *A noval solution to travelling salesman problem using fuzzy sets, gravitational search algorithm, and genetic Algorithm.* Dissertation, Universiti Teknologi Malaysia, Faculty of Computer Science and Information Systems, Malaysia.

[2] Adleman, L. M. (1994). Molecular computation of solutions to combinatorial problem . *Science* , 1021–1023.

[3] Birbil, I., & Fang, S. C. (2003). An electro-magnetism-like mechanism for global optimization. *Journal of Global optimization , 25*, 263–282.

[4] Das, S., Biswas, A., Dasgupta, S., & Abraham, A. Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications.

[5] Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation , 1* (1).

[6] Dorigo, M., & Gamberdella, L. M. (n.d.). Ant colonies for the traveling salesman problem.

[7] Dorigo, M., & Stutzle, T. (2004). *Ant Colony Optimization.* Prentice-Hall.

[8] Farley, B., & Clark, W. A. (1954). Simulation of Self-Organizing Systems by Digital Computer. *IRE Transactions on Information Theory* , 76-84.

[9] Fraser, A. S. (1958). Monte Carlo analyses of genetic models. *Nature* , 208-9.

[10] Gambardella, L. M., & Dorigo, M. Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. *Proceedings of ML-95, Twelfth International Conference on Machine Learning.* Tahoe.

[11] Gambardella, L. M., & Dorigo, M. (1996). Solving Symmetric and Asymmetric TSPs by Ant Colonies. *IEEE Conference on Evolutionary Computation (ICEC'96).* Nagoya, Japan.

[12] Good, B., & Sahin, F. (2006). Bacterial Foraging Approach to Classic Traveling Salesman Problem. *2006 International Conference on Computational Science and Education.*

[13] Kennedy, J., & Eberhart, R. C. (1995). Particle Swarm Optimization. *In Proceedings of the Fourth IEEE Intrenational Confrence on N eural Networks* (pp. 1942-1948). Perth, Australia: IEEE Srvice Center.

[14] Pederson, M. E. (2010). *Good Parameters for Particle Swarm Optimization.* Technical Report, Hvass Laboratories.

[15] Rabanal, P., Rodriguez, I., & Rubio, F. (2009). Applying River Formation Dynamics to Solve NP-Complete Problems. In *Nature-Inspired Algorithms for Optimization* (pp. 333-368). Springer-Verlag.

[16] Rabanal, P., Rodriguez, I., & Rubio, F. (2008). Solving Dynamic TSP by Using River Formation Dynamics. *2008 Fourth International Conference on N atural Computation.*

[17] Rabanal, P., Rodriguez, I., & Rubio, F. (2007). Using River Formation Dynamics to Design Heuristic Algorithms. *Lecture Notes in Computer Science , 4618/2007*, pp. 163-177.

[18] Sato, T., & Hagiwara, M. (1997). Bee system: finding solution by a concentrated search. *IEEE Int. Conf. on Computational cybernatics and simulation*, (pp. 3954–3959).
[19] Serban, G., & Pintea, C.-M. (2004). Heuristics and Learning Approaches for Solving The Travelling Salesman Problem. 27-36.
[20] Shah-Hosseini, H. (2007). Problem Solving by Intelligent Water Drops. *Proc. IEEE Congress on Evolutionary Computation*, (pp. 3226-3231). Singapore.
[21] Shah-Hosseini, H. (2009). The intelligent water drops algoritm: a nature imspired swarm-based optimization algorithm. *Int. J. Bio-Inspired Computation , 1*, 71-79.
[22] Shah-Hosseini, H. (2006). The time adaptive self-organizing map in a neural network system based on artificial immune system. *Proc.IEEE world congress on computational intelligence,* (pp. 1007–1014). Canada: vancouver.
[23] Shi, X. H., Liang, Y. C., Lee, H. P., Lu, C., & Wang, Q. X. (2007). Particle swarm optimization-based algorithms for TSP and genrelized TSP. *Science Direct* , 169–176.
[24] Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizaer. *Proceedings of IEEE International Conference on Evolutionary Computation*, (pp. 69-73).
[25] Tasgetiren, M. F., Suganthan, P. N., & Pan, Q.-K. (n.d.). A Discrete Particle Swarm Optimization Algorithm for the Generalized Traveling Salesman Problem.
[26] *TSPLIB*. (n.d.). Retrieved from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
[27] Wang, L. Y., Zhang, J., & Li, H. (2007). An Improved Genetic Algorithm for TSP. *Machine Learning and Cybernetics*, *2*, pp. 925 - 928. Hong Kong.
[28] Zhi, X. H., Xing, X. L., Wand, Q. X., Zhang, L. H., Yang, X. W., Zhou, C. G., et al. (2004). A discrete PSO method for generalized TSP problem. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*, *4*, pp. 2378 - 2383.
[29] Zhong, W., & Zhang, J. (n.d.). A Noval Discrete Particle Swarm Optimization to Solve Travelling Salesman Problem.

Hifza Afaq: Hifza Afaq is Research Scholar in the Department of Physics and Computer Science, Dayalbagh Educational Institute (Deemed University), Dayalbagh, Agra, Uttar Pradesh, India. She received her Master degree in Physics with Specialization in Computer Science in 2007 from Dayalbagh Educational Institute. Currently she is pursuing her PhD in Nature Inspired Computing.

Sanjay Saini: Dr. Sanjay Saini is Assistant Professor in the Department of Physics and Computer Science, Dayalbagh Educational Institute (Deemed University), Dayalbagh, Agra, Uttar Pradesh, India. He received his Master degree in Mathematics with Specialization in Computer Science in 1993 and PhD in 2003 in the field of Mathematical Ecology and Dynamical Systems, both from Dayalbagh Educational Institute. The areas of his research interests are Mathematical Ecology, Dynamical Systems, Predator-Prey Modeling and Dynamics, Soft Computing, Optimization and recently Nature Inspired Computing.