# Solving Traveling Salesman Problem by Modified Intelligent Water Drop Algorithm

R. Kesavamoorthy M.E.,
Assistant Professor, Faculty of Information Technology,
P.S.R. Rengasamy College of Engineering for Women,
Sivakasi, Tamilnadu, India

D.ArunShunmugam M.E.,
Assistant Professor, Faculty of Computer Science and Engineering,
P.S.R. Engineering College,
Sivakasi, Tamilnadu, India

L.ThangaMariappan M.E.
Assistant Professor, Faculty of Computer Science and Engineering,
V.S.B. Engineering College,
Karur, Tamilnadu, India

## ABSTRACT
In this paper, a modification to the new problem solving algorithm called "Intelligent Water Drops" or simply IWD algorithm has been proposed. This algorithm is based on the dynamic of river systems and the actions that water drops do in the rivers. Based on the observation on the behavior of natural water drops, artificial water drops are developed which possesses some of the remarkable properties of the natural water drops. These ideas are embedded into the proposed algorithm for solving the Traveling Salesman Problem or the TSP. Here a local Heuristic function has been added to the original IWD algorithm, which measures the undesirability of an IWD to move from one node to another. Also it is suggested that, after a few number of iterations, the soils of all paths of the graph of the given TSP are reinitialized again with the initial soil except the paths of the total-best solution which are given less soil than initial soil. The modified IWD algorithm finds better tours and hopefully escapes local optimums.

## Keywords
water drops; Traveling Salesman Problem; heuristic; tour; local optimum**;**

## 1. INTRODUCTION
Scientists are beginning to realize more and more that nature is a great source for inspiration in order to develop intelligent systems and algorithms. In the field of computational Intelligence, especially Evolutionary Computation and Swarm-based systems, the degree of imitation from nature is surprisingly high and it leads to the edge of developing and proposing new algorithms and/or systems, which partially or fully follow nature and the actions and reactions that happen in a specific natural system or species.

Among the most recent nature-inspired swarm-based optimization algorithms is the Intelligent Water Drops (IWD) algorithm. IWD algorithm imitate some of the processes that happen in nature between the water drops of a river and the soil of the river bed. The IWD algorithm was first introduced in [1] in which the IWDs are used to solve the TSP.

In the TSP, a map of cities is given to the salesman and he is required to visit every city only once one after the other to complete his tour and return to its first city. The goal in the TSP is to find the tour with the minimum total length among all such possible tours for the given map.

A TSP is represented by a graph (N, E) where the node set N denotes the n cities of the TSP and the edge set E denotes the edges between cities. Here, the graph of the TSP is considered a complete graph. Thus, every city has a direct link to another city. It is also assumed that the link between each two cities is undirected. So, in summary, the graph of the TSP is a complete undirected graph.

## 2. RELATED WORKS
One of the famous swarm-based optimization algorithms has been invented by simulating the behavior of social ants in a colony. Ants can find the shortest path from their nest to a food source or vice versa with its own intelligence. Other social insects generally show such complex and intelligent behaviors are bees and termites. Ant Colony Optimization or ACO algorithm and Bee Colony Optimization or BCO algorithm are among the swarm-based algorithms imitating social insects for optimization.

Evolutionary Computation is another system, which has been inspired from observing natural selection and reproduction systems in nature. Genetic algorithms or GA are among the most famous algorithms in this regard.

Artificial Immune Systems or simply AIS follow the processes and actions that happen in the immune systems of vertebrates. Clonal Selection Algorithms, Negative Selection Algorithms, and Immune Network Algorithms are among the most common techniques in the field of AIS.

Another swarm-based optimization algorithm is the Particle Swarm Optimization. PSO uses a swarm of particles, which each one has position and velocity vectors and they move near together to find the optimal solution for a given problem. In fact, PSO imitates the processes that exist in the flocks of birds or a school of fish to find the optimal solution.

## 3. NATURAL WATER DROPS
Flowing water drops are observed mostly in rivers, which form huge moving swarms. The paths that a natural river follows have been created by a swarm of water drops. For a swarm of water drops, the river in which they flow is the part of the environment that has been dramatically changed by the swarm and will also be changed in the future.

One of the attractive features of a water drop flowing in a river is its velocity. It is assumed that each water drop of a river can also carry an amount of soil. This soil is usually transferred from fast parts of the path to the slow parts. As the fast parts get deeper by being removed from soil, they can hold more volume of water and thus may attract more water. The removed soils, which are carried in the water drops, are unloaded in slower beds of the river.

Assume an imaginary natural water drop is going to flow from one point of a river to the next point in the front. Three obvious changes such as an increase in the velocity and soil of the water drop and a decrease in the soil of the river's bed between the two points will happen during this transition.

In fact, an amount of soil of the river's bed is removed by the water drop and this removed soil is added to the soil of the water drop. Moreover, the speed of the water drop is increased during the transition.

It was mentioned above that a water drop has also a velocity. This velocity plays an important role in removing soil from the beds of rivers. The following properties are assumed for a flowing water drop:

1) A high speed water drop gathers more soil than a slower water drop.
2) The velocity of a water drop increases more on a path with low soil than a path with high soil.
3) A water drop prefers a path with less soil than a path with more soil.

## 4. INTELLIGENT WATER DROPS

Based on the aforementioned statements, Intelligent Water Drops were developed, which possesses a few remarkable properties of a natural water drop. This Intelligent Water Drop, IWD for short, has two important properties:

1) The soil it carries, denoted by soil(IWD).
2) The velocity that it posses, denoted by velocity(IWD).

For each IWD, the values of both properties, soil(IWD) and velocity(IWD) may change as the IWD flows in its environment. From the engineering point of view, an environment represents a problem that is desired to be solved. A river of IWDs seeks an optimal path for the given problem.

Each IWD is assumed to flow in its environment from a source to a desired destination. In an environment, there are numerous paths from a given source to a desired destination. The location of the destination may be unknown. If the location of the desired destination is known, the solution to the problem is obtained by finding the best (often the shortest) path from the source to the destination.

However, there are cases in which the destination is unknown. In such cases, the solution is obtained by finding the optimum destination in terms of cost or any other desired measure for the given problem.

An IWD moves in discrete finite-length steps in its environment. From its current location i to its next location j, the IWD velocity, velocity (IWD), is increased by an amount Δvelocity(IWD), which is nonlinearly proportional to the inverse of the soil between the two locations i an j, soil(i,j)

$$\Delta velocity(IWD) \propto^{NL} \frac{1}{soil(i,j)} \quad (1)$$

Here, nonlinearly proportionality is denoted by ∝NL. One possible formula is given below in which the velocity of the IWD denoted by $vel^{IWD}(t)$ is updated by the amount of soil, soil(i, j) between the two locations i and j:

$$\Delta vel^{IWD}(t) = \frac{a_v}{b_v + c_v . soil^{2\alpha}(i,j)} \quad (2)$$

Here, the, $a_v$, $b_v$, $c_v$ and α are user-selected positive parameters.

Moreover, the IWD's soil, soil(IWD), is increased by removing some soil of the path joining the two locations i an j. The amount of soil added to the IWD, Δsoil(IWD)= Δsoil(i,j), is inversely (and nonlinearly) proportional to the time needed for the IWD to pass from its current location to the next location denoted by time(i,j; IWD).

$$\Delta soil(IWD) = \Delta soil(i,j) \propto^{NL} \frac{1}{time(i,j;IWD)} \quad (3)$$

One suggestion for the above formula is given below in which time(i, j;$vel^{IWD}$ ) is the time taken for the IWD with velocity $vel^{IWD}$ to move from location i to j. The soil added to the IWD is calculated by

$$\Delta soil(i,,j) = \frac{a_s}{b_s + c_s . time^{2\theta}(i,j; vel^{IWD})} \quad (4)$$

Here, the parameters, $a_s$, $b_s$, $c_s$ and θ are user-selected positive numbers.

The duration of time for the IWD is calculated by the simple laws of physics for linear motion. Thus, the time taken for the IWD to move from location i to j is proportional to the velocity of the IWD, velocity (IWD), and inversely proportional to the distance between the two locations, d(i,j). More specifically:

$$time(i,j;IWD) \propto^{L} \frac{1}{velocity(IWD)} \quad (5)$$

where $\propto^L$ denotes linear proportionality. One such formula is given below, which calculates the time taken for the IWD to travel from location i to j with velocity $vel^{IWD}$:

$$time(i,j;vel^{IWD}) \propto^{L} \frac{HUD(i,j)}{vel^{(IWD)}} \quad (6)$$

where a local heuristic functionHUD(. , .) has been defined for a given problem to measure the undesirability of an IWD to move from one location to the next.

Some soil is removed from the visited path between locations i and j. The updated soil of the path denoted by soil(i,j) is proportional to the amount of soil removed by the IWD flowing on the path joining i to j, Δsoil(i,j)= Δsoil(IWD). Specifically:

$$soil(i, j) \propto^{L} \Delta soil(i, j) \quad (7)$$

One such formula has been used for the IWD algorithm such that soil(i, j) is updated by the amount of soil removed by the IWD from the path i to j.

$$soil(i, j) = \rho_o . soil(i, j) - \rho_n . \Delta soil(i, j) \quad (8)$$

where $\rho_o$ and $\rho_n$ are often positive numbers between zero and one. In the original IWDalgorithm for the TSP [1], $\rho_o = 1 - \rho_n$.

The soil of the IWD denoted by $soil^{IWD}$ is added by the amount soil(i, j) as shown below:

$$soil^{IWD} = soil^{IWD} + \Delta soil(i, j) \quad (9)$$

To implement the behavior of path choosing, a uniform random distribution is used among the soils of the available paths such that the probability of the IWD to move from location i to j denoted by p(i,j;IWD) is inversely proportional to the amount of soils on the available paths.

$$p(i,j;IWD) \propto {}^{L}\ soil(i,j) \tag{10}$$

The lower the soil of the path between locations i and j, the more chance this path has for being selected by the IWD located on i. One such formula based on (10) has been used in which the probability of choosing location j is given by

$$p(i,j;IWD) = \frac{f(soil(i,j))}{\sum_{k \notin vc(IWD)} f(soil(i,k))} \tag{11}$$

$$\text{where } f(soil(i,j)) = \frac{1}{\epsilon_s + g(soil(i,j))}$$

The constant parameter $\epsilon_s$ is a small positive number to prevent a possible division by zero in the function f (.) . The set $V_C$(IWD) denotes the nodes that the IWD should not visit to keep satisfied the constraints of the problem.

The function g(soil(i, j)) is used to shift the soil(i, j) of the path joining nodes i and j toward positive values and is computed by

$$g(soil(i,j)) = \begin{cases} soil(i,j) & \text{if } \min_{k \notin vc(IWD)}(soil(i,l)) \geq 0 \\ soil(i,j) - \min_{k \notin vc(IWD)}(soil(i,l)) & \text{else} \end{cases} \tag{12}$$

where the function min(.) returns the minimum value of its arguments.

The IWDs work together to find the optimal solution to a given problem. The problem is encoded in the environment of the IWDs, and the solution is represented by the path that the IWDs have converged to.

## 5. MODIFIED INTELLIGENT WATER DROP (MIWD) ALGORITHM

The MIWD algorithm employs a number of IWDs to find the optimal solutions to a given problem. The problem is represented by a graph (N, E) with the node set N and edge set E. This graph is the environment for the IWDs and the IWDs flow on the edges of the graph.

Each IWD begins constructing its solution gradually by traveling between the nodes of the graph along the edges until the IWD finally completes its solution denoted by TIWD. Each solution TIWD is represented by the edges that the IWD has visited. One iteration of the IWD algorithm is finished when all IWDs complete their solutions.

After each iteration, the iteration-best solution $T^{IB}$ is found. The iteration-based solution $T^{IB}$ is the best solution based on a quality function among all solutions obtained by the IWDs in the current iteration. $T^{IB}$ is used to update the total-best solution $T^{TB}$. The total-best solution $T^{TB}$ is the best solution since the beginning of the IWD algorithm, which has been found in all iterations.

For a given problem, an objective or quality function is needed to measure the fitness of solutions. Consider the quality function of a problem to be denoted by q (.). Then, the quality of a solution $T^{IWD}$ found by the IWD is given by q ($T^{IWD}$). Therefore, the iteration best solution $T^{IB}$ is given by:

$$T^{IB} = \arg\max_{\forall IWDs} q(T^{IWD}) \tag{13}$$

It should be noted that at the end of each iteration of the algorithm, the total-best solution $T^{TB}$ is updated by the current iteration-best solution $T^{IB}$ as follows:

$$T^{TB} = \begin{cases} T^{TB} & \text{if } q(T^{TB}) \geq q(T^{IB}) \\ T^{IB} & \text{otherwise} \end{cases} \tag{14}$$

At the end of each iteration of the IWD algorithm, the amount of soil on the edges of the iteration-best solution $T^{IB}$ is reduced based on the quality of the solution. One such mechanism to update the soil(i, j) of each edge (i, j) is:

$$soil(i,j) = \rho_s \,.\, soil(i,j) - \rho_{IWD} \,.\, 1/(N_{IB}-1) \,.\, soil^{IWD} \quad \forall (i,j) \in {}_{IB}T^{IB} \tag{15}$$

where $soil_{IB}{}^{IWD}$ represents the soil of the iteration-best IWD. The iteration-best IWD is the IWD that has constructed the iteration-best solution $T^{IB}$ at the current iteration. $N_{IB}$ is the number of nodes in the solution $T^{IB}$ .$\rho_{IWD}$ is the global soil updating parameter, which should be chosen from[0,1]. $\rho_s$ is often set as (1 + $\rho_{IWD}$).

Then, the algorithm begins another iteration with new IWDs but with the same soils on the paths of the graph and the whole process is repeated. The MIWD algorithm stops when it reaches the maximum number of iterations $iter_{max}$ or the total-best solution $T^{TB}$ achieves the expected quality demanded for the given problem.

The IWD algorithm as expressed in the following ten steps:

1) Initialization of static parameters:
   a) The graph (N, E) of the problem is given to the algorithm, which contains $N_C$ nodes.
   b) The quality of the total-best solution TTB is initially set to the worst value: $q(T^{TB}) = -\infty$
   c) The maximum number of iterations max iter is specified by the user and the algorithm stops when it reaches $iter_{max}$.
   d) The iteration count $iterc_{ount}$ which counts the number of iterations, is set to zero.
   e) The number of water drops $N_{IWD}$ is set to a positive integer value. This number should at least be equal to two. However, $N_{IWD}$ is usually set to the number of nodes $N_C$ of the graph.
   f) Velocity updating parameters are $a_v$ , $b_v$ and $c_v$. Here, $a_v = c_v = 1$ and $b_v = 0.01$
   g) Soil updating parameters are $a_s$, $b_s$, and $c_s$. Here, $a_s = b_s = 1$ and $c_s = 0.01$
   h) The local soil updating parameter is $\rho_n$. Here, $\rho_n = 0.9$ except for the AMT, which is $\rho_n = -0.9$.
   i) The global soil updating parameter is $\rho_{IWD}$. Here, $\rho_{IWD} = 0.9$.
   j) The initial soil on each edge of the graph is denoted

by the constant InitSoil such that soil of the edge between every two nodes i and j is set by soil(i,j) = InitSoil .Here, InitSoil = 10000.

k) The initial velocity of each IWD is set to InitVel . Here, InitVel = 200.

2) Initialization of dynamic parameters:

a) Every IWD has a visited node list $V_C$(IWD), which is initially empty: $V_C$(IWD) = { }.

b) Each IWD's velocity is set to InitVel.

c) All IWDs are set to have zero amount of soil.

3) Spread the IWDs randomly on the nodes of the graph as their first visited nodes.

4) Update the visited node list of each IWD to include the nodes just visited.

5) Repeat steps 5.a to 5.d for those IWDs with partial solutions.

a) For the IWD residing in node i, choose the next node j, which doesn't violate any constraints of the problem and is not in the visited node list vc(IWD) of the IWD, using the $p_i^{IWD}(j)$ which is given in (11). Then, add the newly visited node j to the list $V_C$(IWD).

b) For each IWD moving from node i to node j, update its velocity $vel^{IWD}(t)$ by,

$$vel^{IWD}(t+1) = vel^{IWD}(t) + \frac{a_v}{b_v + c_v . soil^2(i,j)} \quad (16)$$

where $vel^{IWD}(t+1)$ is the updated velocity of the IWD.

c) For the IWD moving on the path from node i to j, compute the soil, $\Delta soil(i,j)$ that the IWD loads from the path by the following equation, where the heuristic Undesirability HUD(j) is defined appropriately for the given problem in (4) and (6).

d) Update the soil that IWD carries $soil^{IWD}$ by using (9) and also update the soil soil(i, j) of the path from node i to j traversed by that IWD by

$$soil(i, j) = (1-\rho_n).soil(i, j) - \rho_n.\Delta soil(i, j) \quad (17)$$

6) Find the iteration-best solution $T^{IB}$ from all the solutions $T^{IWD}$ found by the IWDs using (13)

7) Update the soils on the paths that form the current iteration-best solution $T^{IB}$ by

$$soil(i,j) = (1+\rho_{IWD}).soil(i,j) - \rho_{IWD} . 1/(N_{IB}-1) . soil^{IWD} \quad \forall (i,j) \in T^{IB} \quad (18)$$

8) Update the total best solution $T^{TB}$ by the current iteration-best solution $T^{IB}$ using (14)

9) Increment the iteration number by $Iter_{count} = Iter_{count} + 1$. Then, go to step 2 if $Iter_{count} < Iter_{max}$.

10) The algorithm stops with the total-best solution $T^{TB}$.

## 6. CONVERGENCE PROPERTIES OF THE MIWD ALGORITHM

Let the graph (N, E) represents the graph of the given problem. This graph is assumed to be a fully connected graph with $N_C$ nodes. Let $N_{IWD}$ represents the number of IWDs in the IWD algorithm. In the soil updating of the algorithm, two extreme cases are considered:

1) Case (i): Only those terms of the IWD algorithm, which increase soil to an edge (arc) of (N,E), are considered.

2) Case (ii): Only those terms of the IWD algorithm, which decrease soil to an edge (arc) of (N,E), are considered.

For each case, the worst-case is followed.

For case (i), the highest possible value of soil that an edge can hold after m iterations, $soil(edge_{max})$, will be:

$$soil(edge_{max}) = ((\rho_s\rho_o)^m IS_o) \quad (19)$$

where the edge is denoted by $edge_{max}$. $IS_o$ is the initial soil of an edge (i, j), which is denoted by InitSoil in the IWD algorithm. $\rho_o$ is used in (8)and $\rho_s$ is used in (15).

For case (ii), the lowest possible value of soil for an edge is computed and is denoted by $edge_{min}$, after m iterations,

$$soil(edge_{min}) = \left[ m(\rho_{IWD} - \rho_{IWD.} N_{IWD}) \left[ \frac{a_s}{b_s} \right] \right] \quad (20)$$

where $N_{IWD}$ is the number of IWDs. Soil updating parameters $a_s$ and $b_s$ are defined in (4). $\rho_{IWD}$ is the global soil updating parameter used in (18). $\rho_n$ is the local soil updating parameter used in (17).

Based on (19) and (20), the following proposition is stated:

### 6.1 Proportion 1

The soil of any edge in the graph (N,E) of a given problem after m iterations of the MIWD algorithm remains in the interval [ $soil(edge_{min})$, $soil(edge_{max})$ ]. The probability of finding any feasible solution by an IWD in iteration m is $(P_{lowest})^{NC-1}$ where the probability of any IWD, going from node i to node j, is always bigger than $P_{lowest}$.

Since there are $N_{IWD}$ IWDs, then the probability P(s;m) of finding any feasible solution s by the IWDs in iteration m is:

$$P(s;m) = N_{IWD} (P_{lowest})^{NC-1} \quad (21)$$

Then at the end of M iterations of the algorithm,

$$P(s;M) = 1 - \prod_{m=1}^{M} ( 1 - P(s;m) ) \quad (22)$$

Because $0 < P(s;m) \leq 1$, then by making M large enough, it is concluded that:

$$\lim_{M \to \infty} \prod_{m=1}^{M} ( 1 - P(s;m) ) = 0 \quad (23)$$

Therefore, the following proposition is true.

## 6.2 Proportion 2

If P(s;M) represents the probability of finding any feasible solution s within M iterations of the IWD algorithm. As M gets larger, P(s;M) approaches to one:

$$\lim_{M\to\infty} P(s;M)) = 1 \qquad (24)$$

Knowing the fact that the optimal solution s * is a feasible solution of the problem, from above proposition, the following proposition is concluded.

## 6.3 Proportion 3

The IWD algorithm finds the optimal solution s * of any given problem with probability one if the number of iterations M is sufficiently large.

It is noticed that the required M to find the optimal solution s * should be decreased by careful tuning of parameters of the IWD algorithm for the given problem.

## 7. MIWD FOR THE TRAVELING SALESMAN PROBLEM(TSP)

A solution of the TSP having the graph (N, E) is an ordered set of n distinct cities. For such a TSP with n cities, there are $(n-1)!/2$ feasible solutions in which the global optimum(s) is sought.

A TSP solution for an n-city problem may be represented by the tour $T = (c_1, c_2, ..., c_n)$. The salesman travels from city $c_1$ to $c_2$, then from $c_2$ to $c_3$, and he continues this way until it gets to city $c_n$. Then, he returns to the first city $c_1$, which leads to the tour length TL(.), defined by:

$$TL(c_1, c_2, ..., c_n) = \sum_{i=1}^{n} d(c_i, c_{i+1}) \qquad (25)$$

such that $c_{n+1} = c_1$ and d(.,.) is the Euclidean distance. The goal is to find the optimum tour $T^* = (c^*_1, c^*_2, ..., c^*_n)$ such that for every other feasible tour T :

$$\forall T : TL(T^*) \le TL(T) \qquad (26)$$

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. One possible local heuristic for the TSP, denoted by $HUD_{TSP}$ (i, j), has been suggested as follows:

$$HUD_{TSP}(i, j) = \| C(i) - C(j) \| \qquad (27)$$

where c(k) denotes the two dimensional positional vector for the city k . The function ||.|| denotes the Euclidean norm. The local heuristic HUD (i,j) TSP measures the undesirability of an IWD to move from city i to city j . For near cities i and j, the heuristic measure HUD(i , j) becomes small whereas for far cities i and j , the measure HUD(i , j) becomes big. It is reminded that paths with high levels of undesirability are chosen fewer times than paths with low levels of undesirability. In the IWD algorithm, the time taken for the IWD to pass from city i to city j , is proportional to the heuristic $HUD_{TSP}$ (i , j).

A modification to the IWD-TSP has been proposed in which finds better tours and hopefully escape local optimums. After a few number of iterations, say $N_I$, the soils of all paths (i, j) of the graph of the given TSP are reinitialized again with the initial soil InitSoil except the paths of the total-best solution $T^{TB}$, which are given less soil than InitSoil. The soil re – initialization after each $N_I$ iterations is expressed in the following equation:

$$soil(i,j) = \begin{cases} \alpha_I \Gamma_I \, InitSoil & \text{for every } (i,j) \in T^{TB} \\ InitSoil & \text{otherwise} \end{cases} \qquad (28)$$

where $\alpha_I$ is a small positive number chosen here as 0.1. $\Gamma_I$ denotes a random number, which is drawn from a uniform distribution in the interval [0, 1]. As a result, IWDs prefer to choose paths of $T^{TB}$ because less soil on its paths is deposited.

## 8. EXPERIMENTAL RESULTS

Here, the proposed Modified IWD algorithm has been tested for solving the TSP, by generating artificial problems. The results are shown in the Fig.1.

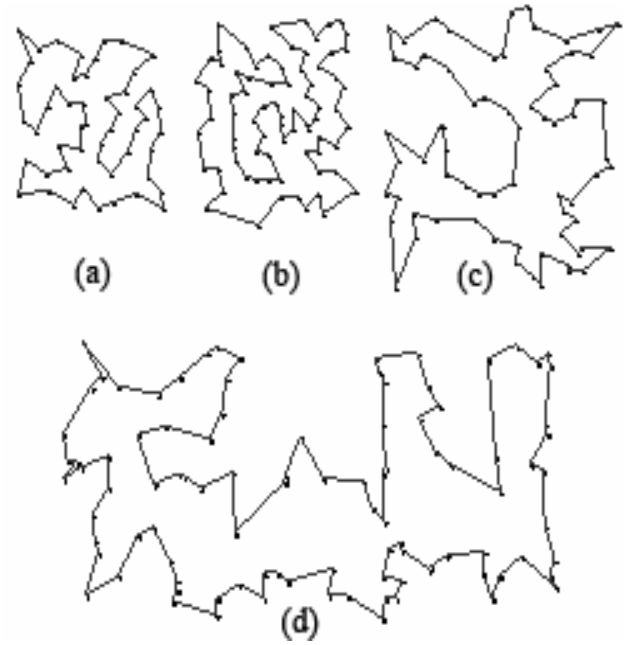


Fig 1. Best tours of MIWD_TSP for (a) 50 cities (b) 60 cities (c) 70 cities (d) 100 cities

The modified IWD algorithm finds better tours and hopefully escapes local optimums. It should be mentioned that the IWD algorithm converges fast to optimum solutions. Moreover, each iteration of the IWD algorithm is computationally light.

In the ACO algorithm, an ant cannot remove pheromones from an edge whereas in the IWD algorithm, an IWD can both remove and add soil to an edge.

In the IWD algorithm, the changes made on the soil of an edge are not constant and they are dependent on the velocity and soil of the IWDs visiting that edge. In contrast, in the ACO algorithm, each ant deposits a constant amount of pheromone on the edge.

Besides, the IWDs may gain different velocities throughout an iteration of the IWD algorithm whereas in ACO algorithms the velocities of the ants are irrelevant.

## 9. CONCLUSION AND FUTURE ENHANCEMENTS

The experiments indicate that the IWD algorithm is capable to find optimal or near optimal solutions. However, there is an open space for modifications in the standard IWD algorithm, embedding other mechanisms that exist in natural rivers. Also some new better local heuristics can be invented that fit better with the given problem.

The IWD can also be used for solving n-queen problem, multi-dimensional knapsack problem and even for automatic multilevel thresholding in images.

As a consequence, further research can focus on the points for amplification of strengths and eliminating the weaknesses. The IWD algorithm demonstrates that the nature is an excellent guide for designing and inventing new nature-inspired optimization algorithms.

## 10. REFERENCES

[1] Haibin Duan, Senqi Liu, and Xiujuan Lei (2008), Air robot path planning based onIntelligent Water Dropsoptimization, IEEE World Congress on ComputationalIntelligence, pp. 1397 – 1401

[2] Hamed Shah_Hosseini (2007), Problem solving by intelligent water drops, Evolutionary Computation, 2007. CEC 2007. IEEE Congress, pp. 3226 – 3231

[3] Hamed Shah-Hosseini (2008), Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem, International Journal of Intelligent Computing and Cybernetics, Vol. 1 Iss: 2, pp.193 – 212

[4] Hamed Shah_Hosseini (2009), The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm, InternationalJournalof Bio-Inspired Computation Volume 1, No.1/2, pp.71 – 79

[5] Kamkar, Iman, Akbarzadeh-T, Mohammad-R., Yaghoobi, and Mahdi (2010), Intelligent water drops a new optimization algorithm for solving the Vehicle Routing Problem, Systems Man and Cybernetics (SMC), pp.4142 –4146

[6] Liu Wei and Zhou Yuren (2010), An Effective Hybrid Ant Colony Algorithm for Solving the Traveling Salesman Problem, IntelligentComputation Technology and Automation (ICICTA), Volume 1, pp. 497 – 500

[7] Mohammad Reza Bonyadi, Mostafa Rahimi Azghadi and Hamed Shah-Hosseini (2008), Population-Based Optimization Algorithms for Solving the Travelling Salesman Problem, Travelling Salesman Problem, pp. 202 – 236

[8] Wei Zhou, Yuanzong Li (2010), An improved genetic algorithm for multiple traveling salesman problem, Informatics in Control, Automation and Robotics (CAR), Volume 1, pp. 493 – 495

[9] Yunming Li (2010), Solving TSP by an ACO-and-BOA- based hybrid algorithm, Computer Application and System Modeling (ICCASM), Volume: 12, pp. 189 – 192

[10] Zhong Liu and Lei Huang (2010), A mixed discrete particle swarm optimization for TSP, Advanced Computer Theory and Engineering (ICACTE), Volume 2, pp. 208 – 211