Solution of Travelling Salesman Problem using Intelligent Water Drops Algorithm

Sevda DAYIOĞLU GÜLCÜ\textsuperscript{a}, Şaban GÜLCÜ\textsuperscript{b} and Humar KAHRAMANLI\textsuperscript{c}
Computer Engineering Department
Selcuk University\textsuperscript{a,c}, Necmettin Erbakan University\textsuperscript{b}
Alaaddin Keykubat Campus, Selçuklu, Konya
TURKEY
sevdadayioglu@gmail.com, sgulcu@konya.edu.tr, hkahramanli@selcuk.edu.tr

Abstract: - A lot of intelligent systems have been developed by inspiring from natural events. One of them is the rivers. A natural river often finds good paths among lots of possible paths in its ways. These optimal or near optimal paths are obtained by the actions that happen among the water drops in the rivers. The intelligent water drops (IWD) algorithm is a swarm-based optimization algorithm inspired from observing natural water drops. In this paper, the IWD algorithm is applied to find solutions of the travelling salesman problem that is one of the well-known discrete optimization problems.

Key-Words: - Intelligent Water Drops, Travelling Salesman Problem, Meta-heuristic, Optimization, Swarm-based Intelligent System, Nature-inspired Optimization Algorithm

1 Introduction
A lot of intelligent systems have been developed by inspiring from natural events. Swarm-based intelligent system is one of them. It is related to natural swarms such as birds, ants, fishes and rivers. Some of the swarm-based techniques are artificial immune systems [1], ant colony optimization [2], particle swarm optimization [3] and intelligent water drops [4]. These algorithms have been successfully applied in a lot of areas of science and engineering.

Travelling Salesman Problem (TSP) is one of the well-known discrete optimization problems. The TSP is to pass once from distance known points and to travel each point with minimum cost by condition of turning back to starting point. Various methods have been used in solution of the TSP until today such as genetic algorithm [5,6,7], tabu search [8], artificial immune system [9,10] and ant colony optimization. Intelligent Water Drops (IWD) algorithm is also one of these methods, too.

The IWD algorithm is a meta-heuristic method which works for a lot of combinatorial optimization problems. The IWD algorithm is based on the dynamic of river systems and actions that happen among the water drops in rivers. The natural water drops are used to develop the IWD and thus the IWDs cooperate together to reach a better solution for a given problem.

In this study, the TSP is solved using intelligent water drops algorithm and results are analyzed.

This paper is organized as follows: travelling salesman problem and intelligent water drops algorithm are briefly presented in Section 2 and Section 3 respectively. Experiment results are analyzed in Section 4. The paper concludes in Section 5.

2 Travelling Salesman Problem
TSP is an optimization problem. It is formulated in 1930 for the first time. The aim of the TSP is to find the shortest path of a travelling salesman who started from a city and stop by in each city only once and come back to the starting city [11].

For the solution of the TSP, a lot of algorithms have been developed until today. The IWD algorithm is one of the keys of this problem.

In the TSP when \( n \) quantity city is given, travelling salesman is aimed to travel each city only one time and create a minimum length closed tour. In this study travelling salesman is used as water drop and city is used as node.

3 Intelligent Water Drops Algorithm
The IWD algorithm was firstly introduced in 2007 by Shah-Hosseini [12]. The IWD algorithm is based on the dynamic of river systems and actions that happen among the water drops in rivers. In the IWD algorithm, IWDs have two main properties: velocity and soil [4].
Both properties change during the runtime of the algorithm. An IWD moves from a source to a destination. The velocity and the soil of the IWD are initially zero. During its travel, it moves in the environment from which it removes some soil and it may acquire some speed. From its current position to its next position, the velocity of IWD is increased by the amount nonlinear proportional to the inverse of the soil between the two positions. Therefore, an IWD in a path with less soil becomes faster than an IWD in a path with more soil.

There is a mechanism in IWD to select the path to its next location or step. In this mechanism, the IWD prefers the paths having low soils rather than the paths having high soils. This behaviour of path selection is implemented by imposing a uniform random distribution on the soils of the available paths. Then, the probability of the next step to be selected is inversely proportional to the soils of the available paths. Therefore, paths with lower soils have higher chance to be selected by the IWD.

The IWD algorithm represents the TSP in the form of a graph \((N, E)\). \(N\) represents the node set, namely cities. \(E\) represents the edge set, namely distances between cities. Then, each IWD begins creating its solution stepwise by travelling between the cities until the IWD finally completes its solution. One iteration of the algorithm is complete when all IWDs have completed their solutions.

After each iteration, the iteration-best solution \(T^B\) is found. Then, it is used to update the total-best solution \(T^{IB}\). The amount of soil on the edges of the \(T^B\) is reduced based on the quality of the solution. 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 algorithm stops when it reaches the maximum number of iterations \(\text{iter}_{\text{max}}\) or the \(T^{IB}\) reaches the expected quality.

The procedure of the IWD for the TSP can be described as follows:

**Step 1.** Initialization of static parameters. Static parameters are unchangeable during the runtime of the algorithm. The graph \((N, E)\) of the problem is given to the algorithm. The quality of the total-best solution \(T^{IB}\) is initially set to the worst value: \(q(T^{IB}) = -\infty\).

The maximum number of iterations \(\text{iter}_{\text{max}}\) is specified by the user.

The iteration count \(\text{iter}_{\text{count}}\) is set to zero. The number of water drops \(N_{\text{WD}}\) is set to a positive integer value, which is usually set to the number of nodes \(N\) of the graph.

For velocity updating, the parameters \(a_v, b_v\) and \(c_v\) are used.

For soil updating, the parameters \(a_s, b_s\) and \(c_s\) are used.

To update local soil, parameter \(\rho_s\) is used. It is a small positive number between 0 and 1.

To update the global soil, parameter \(\rho_{\text{WD}}\) is used. It is a small positive number between 0 and 1.

The parameter \(\text{InitSoil}\) is the initial soil on each path. The soil of the path between every two nodes \(i\) and node \(j\) is set by \(\text{soil}(i, j) = \text{InitSoil}\). The parameter \(\text{InitVel}\) is the initial velocity of each IWD.

These parameters are specified by users. They should be tuned empirically for the application.

**Step 2.** Initialization of dynamic parameters. Dynamic parameters are reinitialized at the end of each iteration of the algorithm. Each IWD has a visited node list \(V_i(\text{IWD})\) which is initially empty: \(V_i(\text{IWD}) = \{ \}\). The velocity of each IWD is set to \(\text{InitVel}\). All IWDs initially have zero amount of soil.

**Step 3.** Spread the IWDs randomly over the cities as their first visited nodes.

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

**Step 5.** Repeat Steps 5.1 to 5.4 for those IWDs with partial solutions.

**Step 5.1.** 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 \(V_i(\text{IWD})\) of the IWD, using the probability function in Eq. (1):

\[
P_i^{\text{IWD}}(j) = \frac{f(\text{soil}(i,j))}{\sum_{k\in V_i(\text{IWD})} f(\text{soil}(i,k))}
\]

**Step 5.2.** For each IWD moving from node \(i\) to node \(j\), update its velocity using Eq. (4).
\[ \text{vel}^{\text{IWD}}(t+1) = \text{vel}^{\text{IWD}}(t) + \frac{a_v}{b_v + c_v \cdot \text{soil}^2(i,j)} \]  

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

**Step 5.3.** For the IWD moving on the path from node \( i \) to node \( j \), compute the soil \( \Delta \text{soil}(i,j) \) that the IWD loads from the path using Eq. (5).

\[ \Delta \text{soil}(i,j) = \frac{a_s}{b_s + c_s \cdot \text{time}^2(i,j; \text{vel}^{\text{IWD}}(t+1))} \]  

where \( \text{time}(.) \) is defined as

\[ \text{time}(i,j; \text{vel}^{\text{IWD}}(t+1)) = \frac{\text{HUD}(j)}{\text{vel}^{\text{IWD}}(t+1)} \]

where the heuristic undesirability \( \text{HUD}(j) \) is defined appropriately for the given problem.

**Step 5.4.** Update the soil \( \text{soil}(i,j) \) of the path from node \( i \) to node \( j \) traversed by that IWD using Eq. (7). Then update the soil \( \text{soil}^{\text{IWD}} \) that the IWD carries using Eq. (8).

\[ \text{soil}(i,j) = (1 - \rho_n) \cdot \text{soil}(i,j) - \rho_n \cdot \Delta \text{soil}(i,j) \]  

\[ \text{soil}^{\text{IWD}} = \text{soil}^{\text{IWD}} + \Delta \text{soil}(i,j) \]  

**Step 6.** Find the iteration-best solution \( T^B \) from all the solutions \( T^{\text{IWD}} \) found by the IWDs using Eq. (9).

\[ T^B = \arg \max_{T^{\text{IWD}}} q(T^{\text{IWD}}) \]  

where \( q(.) \) is the quality function. Here the maximum is considered as the best solution.

**Step 7.** Update the soils on the paths that form the current iteration-best solution \( T^B \) by Eq. (10).

\[ \text{soil}(i,j) = (1 + \rho_{\text{soil}}) \cdot \text{soil}(i,j) - \rho_{\text{soil}} \cdot \text{soil}^{\text{IWD}} \]  

\[ \rho_{\text{soil}} = \frac{1}{N_{IB} - 1} \cdot \text{soil}^{\text{IWD}} \quad \forall (i,j) \in T^B \]  

where \( N_{IB} \) is the number of nodes in the solution \( T^B \).

**Step 8.** Update the total best solution \( T^{TB} \) by the current iteration-best solution \( T^B \) using Eq. (11).

\[ T^{TB} = \begin{cases} T^B & q(T^B) \geq q(T^{IB}) \\ T^{IB} & \text{otherwise} \end{cases} \]  

**Step 9.** Increment the iteration number by \( \text{iter}_{\text{count}} = \text{iter}_{\text{count}} + 1 \). Then, go to Step 2 if \( \text{iter}_{\text{count}} \) is less than \( \text{iter}_{\max} \).

**Step 10.** The algorithm stops and returns the total-best solution \( T^{TB} \).

The important point is that the IWD algorithm can find the optimal solution if the iteration number is enough large.

### 4 Experiment Results

In this section, we present computational results of IWD algorithm. In this study Berlin52, Eil51, Eil76, Eil101, KroA100, KroC100, Pr76, Lin105, St70, Ulysses22 travelling salesman problems [13] were solved by using IWD algorithm. Experiments were tested with a computer which has Intel i5 3.2GHz cpu, 2GB ram. While solving these problems with IWD algorithm, there are some initial parameters which must be used. These parameters are; \( a_v, b_v, c_v \) for velocity updating, \( a_s, b_s, c_s \) for soil updating, \( \text{iter}_{\max} \) for the maximum number of iterations, \( N_{\text{IWD}} \) for the number of water drops, parameter \( \text{InitSoil} \) for the initial soil on each path, \( \text{InitVel} \) for the velocity of each IWD, \( \rho_{\text{soil}} \) for the global soil updating and \( \rho_n \) for the local soil updating. These used parameter and their values are shown in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_v )</td>
<td>1</td>
</tr>
<tr>
<td>( b_v )</td>
<td>0.01</td>
</tr>
<tr>
<td>( c_v )</td>
<td>1</td>
</tr>
<tr>
<td>( a_s )</td>
<td>1</td>
</tr>
<tr>
<td>( b_s )</td>
<td>0.01</td>
</tr>
<tr>
<td>( c_s )</td>
<td>1</td>
</tr>
<tr>
<td>( \text{iter}_{\max} )</td>
<td>1000</td>
</tr>
<tr>
<td>( \text{InitSoil} )</td>
<td>10000</td>
</tr>
<tr>
<td>( \text{InitVel} )</td>
<td>200</td>
</tr>
<tr>
<td>( \rho_{\text{soil}} )</td>
<td>0.9</td>
</tr>
<tr>
<td>( \rho_n )</td>
<td>0.9</td>
</tr>
<tr>
<td>( N_{\text{IWD}} )</td>
<td>number of cities</td>
</tr>
</tbody>
</table>

The number of water drops \( N_{\text{IWD}} \) was assessed as the equal amount to the number of cities \( N_c \) in problems. The stopping criterion was determined as 1000 iteration. The algorithm was run 50 times for
each TSP instance in order to show the consistency of the results.

Euclidian is used as distance measure for the distance between the nodes. According to Euclidian distance measure \[14\], the distance between the node \(i\) and node \(j\) is calculated by Eq. (12).

\[
d_{ij} = \left( (x_i - x_j)^2 + (y_i - y_j)^2 \right)^{1/2} \tag{12}
\]

The comparisons between our obtained results and optimum results for Berlin52, Eil51, Eil76, Eil101, KroA100, KroC100, Pr76, Lin105, St70, Ulysses22 problems are shown in Table 2.

Table 2: Comparison between optimum length and IWD

<table>
<thead>
<tr>
<th>Problem Name</th>
<th>Number of City</th>
<th>IWD</th>
<th>Optimum tour length</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berlin52</td>
<td>52</td>
<td>7542</td>
<td>7542</td>
<td>0</td>
</tr>
<tr>
<td>Eil51</td>
<td>51</td>
<td>426</td>
<td>426</td>
<td>0</td>
</tr>
<tr>
<td>Eil76</td>
<td>76</td>
<td>540</td>
<td>538</td>
<td>0.37</td>
</tr>
<tr>
<td>Eil101</td>
<td>101</td>
<td>639</td>
<td>629</td>
<td>1.59</td>
</tr>
<tr>
<td>KroA100</td>
<td>100</td>
<td>21429</td>
<td>21282</td>
<td>0.69</td>
</tr>
<tr>
<td>KroC100</td>
<td>100</td>
<td>20816</td>
<td>20749</td>
<td>0.32</td>
</tr>
<tr>
<td>Lin105</td>
<td>105</td>
<td>14393</td>
<td>14379</td>
<td>0.10</td>
</tr>
<tr>
<td>Pr76</td>
<td>76</td>
<td>109608</td>
<td>108159</td>
<td>1.34</td>
</tr>
<tr>
<td>St70</td>
<td>70</td>
<td>676</td>
<td>675</td>
<td>0.15</td>
</tr>
<tr>
<td>Ulysses22</td>
<td>22</td>
<td>72</td>
<td>72</td>
<td>0</td>
</tr>
</tbody>
</table>

Error value which is indicated in Table 2 is required by Eq. (13). Error value refers to how close the obtained results by IDW are to the optimum tour lengths.

\[
Error = \frac{X - Y}{Y} \times 100 \tag{13}
\]

X: Tour length which is found by IWD
Y: Optimum tour length

The lowest error rate in solved problems was observed in Berlin52, Eil51 and Ulysses22 problems with 0% error rate and the highest rate was observed in Eil101 with 1.59 rate. Table 2 shows that the tours obtained by IWD are equal or quite close to the known optimum tour length.

We also tested these standard problems with ant colony optimization algorithm in order to compare with IWD. Ant amount parameter was assessed as the equal amount to the number of cities in problems. Stopping criteria was determined as 1000 iteration like in IWD. The algorithm was run 50 times for each TSP instance. The results of ACO and IWD are shown in Table 3. Table 3 shows that the tours obtained by IWD algorithm are better than the tours obtained by ACO algorithm.

Table 3: Comparison between IWD and ACO

<table>
<thead>
<tr>
<th>Problem Name</th>
<th>Number of City</th>
<th>IWD</th>
<th>ACO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berlin52</td>
<td>52</td>
<td>7542</td>
<td>7549</td>
</tr>
<tr>
<td>Eil51</td>
<td>51</td>
<td>426</td>
<td>438</td>
</tr>
<tr>
<td>Eil76</td>
<td>76</td>
<td>540</td>
<td>553</td>
</tr>
<tr>
<td>Eil101</td>
<td>101</td>
<td>639</td>
<td>674</td>
</tr>
<tr>
<td>KroA100</td>
<td>100</td>
<td>21429</td>
<td>22413</td>
</tr>
<tr>
<td>KroC100</td>
<td>100</td>
<td>20816</td>
<td>21031</td>
</tr>
<tr>
<td>Lin105</td>
<td>105</td>
<td>14393</td>
<td>14649</td>
</tr>
<tr>
<td>Pr76</td>
<td>76</td>
<td>109608</td>
<td>113774</td>
</tr>
<tr>
<td>St70</td>
<td>70</td>
<td>676</td>
<td>696</td>
</tr>
<tr>
<td>Ulysses22</td>
<td>22</td>
<td>72</td>
<td>75</td>
</tr>
</tbody>
</table>

5 Conclusion

One of the approaches used for solution of Travelling Salesman Problem is Intelligent Water Drops algorithm. In IWD, behaviours of the water drops are used and paths are followed on nodes (cities) and the shortest path is tried to be reached.

In this study, well-known travelling salesman problems (Berlin52, Eil51, Eil76, Eil101, KroA100, KroC100, Pr76, Lin105, St70, Ulysses22) which are frequently used in literature, were solved by the IWD algorithm. We also tested these standard problems with the ant colony optimization algorithm in order to compare with IWD. Both algorithms are heuristic methods. According to experiment results, it is seen that optimum or nearly optimum results can be found by the IWD algorithm in a short time and the IWD gives the most proper solution than the ACO.

References:


