Implementation of Intelligent Water Drops Algorithm to Solve Graph-Based Travelling Salesman Problem

Roli Bansal, Hina Agrawal, Hifza Afaq and Sanjay Saini

Abstract The travelling salesman problem (TSP) is one of the most sought out NP-hard, routing problems in the literature. TSP is important with respect to some real-life applications, especially when tour is generated in real time. The objective of this paper is to apply the intelligent water drops algorithm to solve graph-based TSP (GB-TSP). The intelligent water drops (IWD) algorithm is a new meta-heuristic approach belonging to a class of swarm intelligence-based algorithm. It is inspired from observing natural water drops that flow in rivers. The idea of path finding of rivers is used to find the near-optimal solution of the travelling salesman problem (TSP).

Keywords Intelligent water drops (IWD) · Travelling salesman problem (TSP) · Swarm intelligence · Graph-based TSP (GB-TSP)

1 Introduction

Soft computing is a term applied to a field within computer science which is used to obtain near-optimal solutions to NP-complete problems in polynomial time. Swarm intelligence is a relatively new field of soft computing [1] which is inspired by nature. Swarm intelligence is based on the collective behavior of decentralized, self-

R. Bansal (✉) · H. Agrawal · H. Afaq · S. Saini
Department of Physics and Computer Science, Dayalbagh Educational Institute, Agra, India
e-mail: dei.rolibansal@gmail.com

H. Agrawal
e-mail: hinaagrawal29@gmail.com

H. Afaq
e-mail: hifza.afaq@gmail.com

S. Saini
e-mail: sanjay.s.saini@gmail.com

B. V. Babu et al. (eds.), *Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28–30, 2012*, Advances in Intelligent Systems and Computing 236, DOI: 10.1007/978-81-322-1602-5_15,

organized systems. It refers to algorithms such as ant colony optimization (ACO) [9], particle swarm optimization (PSO) [4], artificial bee colony [3], bat algorithm [11], and many more. Intelligent water drops (IWD) is an upcoming swarm intelligence-based algorithm. IWD was proposed by Hamed Shah-Hosseini [7] in 2007. IWD algorithm is based on the dynamics of river system, action, and reactions that takes place among the water drops in rivers [6]. A water drop prefers the path having low soil to the path having high soil. In this way, it finds best possible path for itself. This behavior of IWD is used to optimize the tour for travelling salesman problem (TSP).

In the TSP [2], a map of cities is given to the salesman and he is required to visit all the cities to complete his tour such that no city is visited twice except the city it starts with, which it has to visit in the end again to complete its tour. In real-life situation, all cities may not be completely interconnected with direct paths and so paths may not exist between some of the cities, so the map of cities is not completely interconnected. The goal in the TSP is to find the tour with the minimum total length, among all possible tours for the given map.

2 Behavior of Intelligent Water Drops

The behavior of the natural water drop is observed with some properties such as

- Velocity (with which a drop moves)
- Soil (which is carried by the drop)

IWD is based on these two properties of water drops. These properties change with time according to the environment of water drop when IWD flows from source to destination. Initially, IWD has zero amount of soil and nonzero velocity. It carries more soil with high velocity and unloads the soil with low velocity of IWD.

A water drop prefers an easier path to a harder path in an obvious way when it has to choose between several paths that exist in the path from source to destination. Each IWD has a number of possible paths to choose from when it goes from one position to another position. It chooses the path with the low soil and maximum probability.

Every IWD flows in finite length steps from one location to another location. The IWD's velocity depends on the soil between two locations. The IWD's velocity increases on less soil path and decreases on high soil path. Thus, IWD's velocity is inversely proportional to the soil between two locations. An IWD removes some amount of soil from the path and carries it while travelling through that path. It removes the soil from the path depending upon the time it takes to cover the distance between two locations. More soil is removed from the path if the time taken by IWD is high and vice versa. Thus, IWD's soil is inversely proportional to time taken in travelling from current location to next location. This time is calculated by the simple laws of physics linear motion.

3 Solving GB-TSP Using IWD

In this section, steps for solving the graph-based TSP (GB-TSP) are discussed. For geographical problems, where location of cities is given by their Cartesian coordinates and path from any node to any other node exists necessarily (which is simply the Euclidean distance between the two nodes), solution using IWD to such TSP has been given by Hamed Shah-Hosseini as MIWD [5]. In our case, a GB-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 paths between the two cities. The considered graphs are non-complete, i.e., it is not necessary that a direct edge exists between every pair of nodes. In fact, we consider graphs where a direct edge may not exist between certain two nodes. This formulation of the problem is much more realistic than the earlier problems. The cost associated with the edges represents distance between cities. However, for the sake of simplicity and similarity with earlier problems, in this paper, the location of cities is given by their Cartesian coordinates and the distance between them is their Euclidean distance. For GB-TSP, we start with a graph which is not completely interconnected. For this, we create an adjacency matrix depicting distances between cities, and then, we remove the edges where there is no path between the cities. In this way, we have a subset of edge set which represent a graph which is not completely interconnected. A solution of the GB-TSP having the graph (N, E) is then an ordered set of n distinct cities.

Now, the following IWD strategy for the GB-TSP is used. Each link of the edge set E has an amount of soil. An IWD visits nodes of the graph through the links. The IWD is able to change the amount of soil on the links. An IWD starts its tour from a random node. The IWD changes the soil of each link that it flows on while completing its tour.

Since there are no standard problems available where the graph is not complete, we have created such test graphs for our experiments. For this, we convert a completely connected graph into a non-complete graph. This conversion of complete graph into non-complete graph is done using the X nearest neighbor (XNN) algorithm [8]. In this method, the links, depicting distances, from one node to all other nodes are taken, and then, a certain percentage of those links are dropped. The dropped links are those which are the largest in that set of links. We repeat the same process for all the nodes in our graph. By converting the complete graph into a non-complete graph, the search space of the problem is reduced.

This algorithm takes a complete graph and drops a given percentage of links from it. For our experiments, a few standard problems are considered and 20 % links are dropped from it.

The IWD algorithm that is used for the GB-TSP is as follows:

1. Initialization of static parameters:
   - Set the number of water drops N_{IWD}, the number of cities N_C, and the Cartesian coordinate of each city i such that $c(i) = [x_i, y_i]^T$ to their chosen constant values. The number of cities and their coordinate values depends on the prob-

lem at hand, while the N_{IWD} is set by the user. We choose N_{IWD} to be equal to or greater than the number of cities.
- Set the parameter number of neighbor cities called neighbor_city. For instance, if we have to drop 20 % links from each node, then neighbor_city is 80 % of $N_C - 1$.
- Parameters for velocity updating: $a_v = c_v = 1$ and $b_v = 0.01$.
- Parameters for soil updating: $a_s = c_s = 1$ and $b_s = 0.01$.
- Initial soil on each link is denoted by the constant InitSoil such that the soil of the link between every two cities i and j is set by soil (i, j) = InitSoil. Here, we choose InitSoil = 10,000.
- Initial velocity of IWD is denoted by the constant InitVel. Velocity of each drop with which they start their tour. Here, InitVel = 200.
- The best tour with minimum tour length (Len(T_B)) is denoted by T_B. Initially, it is set as Len(T_B) = infinity.
- The termination condition is met when maximum number of iterations is reached.

2. Initialization of dynamic parameters:

   - For every IWD, we create an empty visited city list V_c(IWD) = {}.
   - Initially, each IWD has velocity equal to InitVal and soil equal to zero.

3. Non-complete graph is generated using XNN algorithm [8] along with its adjacency matrix.

   - Initialize the number of neighbors of each city with the constant neighbor_city.
   - Create the adjacency matrix for new city links.

4. For every IWD, select a city randomly and place that IWD on that city.
5. Update the visited city lists of all IWDs to include the cities just visited.
6. Select the next city:

   - For each IWD, choose the next city j to be visited by IWD when it is in city i with the probability $P_i^{\text{IWD}}(j)$ as given in (1).

$$P_i^{\text{IWD}}(j) = \frac{f(\text{soil}(i, j))}{\sum_{k \notin vc} f(\text{soil}(i, k))} \tag{1}$$

such that $f(\text{soil}(i, j)) = 1/\varepsilon_s + g(\text{soil}(i, j))$
where

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

Here, $\varepsilon_s = 0.01$. Where vc(IWD) is the visited city list of the IWD.

The probability depends on the soil between the path. IWD selects the city with maximum probability.

7. Update the soil and velocity:

   - An IWD in city i wants to go to next city j; then, the amount of soil on this path, i.e., soil (i, j) is used to update the velocity as given by (2).

$$\text{vel}^{\text{IWD}}(t+1) = \text{vel}^{\text{IWD}}(t) + \frac{a_v}{b_v + c_v \cdot \text{soil}(i, j)} \tag{2}$$

   - Each IWD, carries some amount of the soil, $\Delta\text{soil}(i, j)$, that the current IWD alters in its current path while travelling between the cities i and j is given by (3).

$$\Delta\text{soil}(i, j) = \frac{a_s}{b_s + c_s \cdot \text{time}(i, j; \text{vel}^{\text{IWD}})} \tag{3}$$

   where time taken to travel from city i to city j with velocity vel^{IWD} is given by $\text{time}(i, j; \text{vel}^{\text{IWD}}) = c(i) - c(j)/\max(\varepsilon_v, \text{vel}^{\text{IWD}})$

   - For each IWD, update the soil of the path traversed by that IWD by removing certain soil from the path as in (4)

$$\text{soil}(i, j) = (1 - \rho) \cdot \text{soil}(i, j) - \rho \cdot \Delta\text{soil}(i, j) \tag{4}$$

   Here, $\rho = 0.9$.
   Update the soil of each IWD by adding soil removed from the path in present soil of the IWD

$$\text{soil}^{\text{IWD}} = \text{soil}^{\text{IWD}} + \Delta\text{soil}(i, j) \tag{5}$$

   $\text{soil}(i, j)$ represents the soil of the path between i and j. $\Delta\text{soil}(i, j)$ represents the soil that IWD carries from that path, and soil^{IWD} represents total soil carried by the drop.

8. Each IWD completes its tour by using steps 5 to 8 repeatedly. Then, length of the tour (Tour^{IWD}) traversed by the IWD is calculated. Then, the tour with minimum length among all IWD tours in this iteration is found. Test the correctness of the minimum path from the adjacency matrix.
9. If iteration best tour (current minimum tour) exists then:

   - Update the soil of the paths included in the current minimum tour of the IWD denoted by T_M by (6).

$$\text{soil}(i, j) = (1 - \rho) \cdot \text{soil}(i, j) + \rho \cdot \frac{2 \cdot \text{soils}^{\text{IWD}}}{N_c(N_c - 1)} \forall (i, j) \in T_M \tag{6}$$

- If the minimum tour T_M is shorter than the best tour T_B found so far, then T_B is updated by (7).

$$T_B = T_M \text{and Len}\,(T_B) = \text{Len}\,(T_M) \tag{7}$$

10. Otherwise discard the current minimum tour.
11. Go to step 2 unless the maximum number of iterations is reached.
12. If the maximum number of iterations is reached, then the algorithm stops with the best tour T_B with tour length Len(T_B).

4 Experimental Result

In this section, we present computational results obtained. We evaluated the performance of IWD algorithm for some TSP benchmark problems form TSPLIB [10]. We applied IWD algorithm on self-generated network like a pentagon. Firstly, a five node layout is taken as a complete graph, which then is converted to a non-complete graph using XNN algorithm. This network and its optimal path are shown in Fig. 1.

We also apply this on some benchmark problems such as eil51, eil76, st70, and kroA100 after converting them to non-complete graphs.

The experimental result of benchmark problem is shown in the Table 1.

Table 1 Experimental results for benchmark problems (10 run)

Problems	Optimum length	Average length by IWD
Eil51	426	445
Eil76	538	550
St70	675	748
Kroa100	21,282	24,344

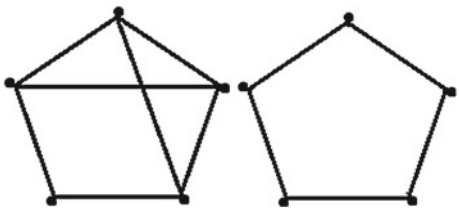

Fig. 1 *Left* the non-complete graph and *right* the optimal path

5 Conclusion

The intelligent water drops algorithm or the IWD algorithm is one of the recent bio-inspired swarm-based optimization algorithms. It gives the optimal solution to various optimization problems. The experimental results show that this algorithm is capable of finding the near-optimal solution. In this paper, we apply IWD on graph-based TSP (non-complete graphs) which gives the near-best optimal solution. We used XNN algorithm to convert the complete TSP graph to non-complete graph. GB-TSP represents the real-life transportation problem.

The IWD algorithm can also be used for solving multiple knapsack problem, *n*-Queen problem, multilevel thresholding problem, etc.

References

1. Bonabeau, E., Dorigo, M., Theraultz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
2. Greco, F.: Travelling Salesman Problem. In-Teh is Croatian Branch of I-Tech Education and Publishing KG, Vienna, Austria (2008)
3. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput **214**(1), 108–132 (2009)
4. Kennedy, J., Eberhart, R.C.: Particle swarm optimization, pp. 1942–1948. Proceedings of the Fourth IEEE International Conference on Neural Networks, IEEE Service Center, Perth, Australia (1995)
5. Shah-Hosseini, H.: Optimization with the Nature-Inspired Intelligent Water Drops Algorithm. In: Sentos E.W (ed.) Evolutionary Computation, p. 572. I-Tech, Vienna, Austria (2009).
6. Shah-Hosseini, H.: Problem solving by intelligent water drops. Proceedings on IEEE Congress on Evolutionary Computation, pp. 3226–3231. Singapore (2007).
7. Shah-hosseini, H.: The intelligent water drops algorithm : a nature-inspired swarm-based optimization algorithm. Compu. Eng. **1**(1), 71–79 (2009)
8. Taba, M.S.: Solving Traveling Salesman Problem With a Non-complete Graph. Waterloo, Ontario, Canada (2009)
9. Toksari, M.D.: Ant colony optimization for finding the global minimum. Appl. Math. Comput. **176**(1), 308–316 (2006)
10. TSPLIB. (n.d.). Retrieved 08 23, 2012, from http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
11. Yang, X.-S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010) 284, 65–74 (2010).