Problem Solving by Intelligent Water Drops

Hamed Shah_Hosseini

Abstract—**In this paper, we propose a new problem solving algorithm called "Intelligent Water Drops" or IWD algorithm which is based on the processes that happen in the natural river systems and the actions and reactions that take place between water drops in the river and the changes that happen in the environment that river is flowing. It is observed that a river often chooses an optimum path regarding the conditions of its surroundings to get to its ultimate goal which is often a lake or sea. These ideas are embedded into the proposed algorithm for solving the Traveling Salesman Problem or the TSP. The IWD algorithm is tested with artificial and standard TSP problems and the experimental results demonstrate that it is a very promising problem solving algorithm and deserves more research to improve it and/or to adapt it to other engineering problems.**

I. Introduction

The creatures and natural systems which are working and developing in nature are one of the interesting and valuable sources of inspiration for designing and inventing new systems and algorithms in different fields of science and technology. Evolutionary Computation [1], Neural Networks [2], Time Adaptive Self-Organizing Maps [3], Ant Systems [4], Particle Swarm Optimization [5], Simulated Annealing [6], and DNA Computing [7] are among the problem solving techniques inspired from observing nature.

Here, we propose a problem solving algorithm based on the dynamic of river systems and the actions that water drops do in the rivers. The ideas that are taken from natural water drops are used in order to develop artificial water drops. The artificial water drops are then adapted for solving the TSP problems. In the TSP, a map of cities is given to the salesman and he has to visit all the cities only once to complete a tour such that the length of the tour is the shortest among all possible tours for this map. It is known that the TSP is an NP-hard problem [8] and is often used for testing the optimization algorithms.

The next section reviews some processes that occur in a river which involves the water drops of the river. Section III proposes intelligent water drops based on the ideas of natural water drops. Section IV introduces the IWD algorithm which uses intelligent water drops for solving the TSP. Experimental results with the proposed IWD algorithm for artificial and standard TSPs form section V. Concluding remarks are the final section of the paper.

H. Shah_Hosseini is with the Electrical and Computer Engineering Department, Shahid Beheshti University, Tehran, Iran, (e-mail: h_shahhosseini@sbu.ac.ir, tasom2002@yahoo.com).

II. Natural Water Drops

In nature, we often see water drops moving in rivers, lakes, and seas. As water drops move, they change their environment in which they are flowing. Moreover, the environment itself has substantial effects on the paths that the water drops follow. Consider a hypothetical river in which water is flowing and moving from high terrain to lower terrain and finally joins a lake or sea. The paths that the river follows, based on our observation in nature, are often full of twists and turns. We also know that the water drops have no visible eyes to be able to find the destination (lake or river). If we put ourselves in place of a water drop of the river, we feel that some force pulls us toward itself (gravity). This gravitational force as we know from physics is straight toward the center of the earth. Therefore with no obstacles and barriers, the water drops would follow a straight path toward the destination, which is the shortest path from the source to the destination. However, due to different kinds of obstacles in the way of this ideal path, the real path will have to be different from the ideal path and we often see lots of twists and turns in a river path. In contrast, the water drops always try to change the real path to make it a better path in order to approach the ideal path. This continuous effort changes the path of the river as time passes by. One feature of a water drop is the velocity that it flows which enables the water drop to transfer an amount of soil from one place to another place in the front. 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. There are other mechanisms which are involved in the river system which we don't intend to consider them all here.

In summary, a water drop in a river has a non-zero velocity. It often carries an amount of soil. It can load some soil from an area of the river bed, often from fast flowing areas and unload them in slower areas of the river bed. Obviously, a water drop prefers an easier path to a harder path when it has to choose between several branches that exist in the path from the source to the destination.

III. Intelligent Water Drops

Based on the observation on the behavior of water drops, we develop an artificial water drop which possesses some of the remarkable properties of the natural water drop. This Intelligent Water Drop, IWD for short, has two important

1-4244-1340-0/07/$25.00 ©2007 IEEE

properties:

1) The amount of the soil it carries now, Soil(IWD).

2) The velocity that it is moving now, Velocity(IWD).

The values of the both properties may change as the IWD flows in its environment. This environment depends on the problem at hand. In an environment, there are usually lots of paths from a given source to a desired destination, which the position of the destination may be known or unknown. If we know the position of the destination, the goal is to find the best (often the shortest) path from the source to the destination. In some cases, in which the destination is unknown, the goal is to find the optimum destination in terms of cost or any suitable measure for the problem.

We consider an IWD moving in discrete finite-length steps. From its current location to its next location, the IWD velocity is increased by the amount nonlinearly proportional to the inverse of the soil between the two locations. Moreover, the IWD's soil is increased by removing some soil of the path joining the two locations. The amount of soil added to the IWD is inversely (and nonlinearly) proportional to the time needed for the IWD to pass from its current location to the next location. This duration of time is calculated by the simple laws of physics for linear motion. Thus, the time taken is proportional to the velocity of the IWD and inversely proportional to the distance between the two locations.

Another mechanism that exists in the behavior of an IWD is that it prefers the paths with low soils on its beds to the paths with higher soils on its beds. To implement this behavior of path choosing, we use a uniform random distribution among the soils of the available paths such that the probability of the next path to choose is inversely proportional to the soils of the available paths. The lower the soil of the path, the more chance it has for being selected by the IWD.

IV. Intelligent Water Drops for the TSP

In this section, we specifically express the steps for solving the TSP. The first step is how to represent the TSP in a suitable way for the IWD. For the TSP, the cities are often modeled by nodes of a graph, and the links in the graph represent the paths joining each two cities. Each link or path has an amount of soil. An IWD can travel between cities through these links and can change the amount of their soils. Therefore, each city in the TSP is denoted by a node in the graph which holds the physical position of each city in terms of its two dimensional coordinates while the links of the graph denote the paths between cities. To implement the constraint that each IWD never visits a city twice, we consider a visited city list for the IWD which this list includes the cities visited so far by the IWD. So, the possible cities for an IWD to choose in its next step must not be from the cities in the visited list.

In the following, we present the proposed Intelligent Water Drop (IWD) algorithm for the TSP:

1. Initialization of static parameters: set the number of water drops N_{IWD}, the number of cities N_c, and the Cartesian coordinates of each city *i* such that $\mathbf{c}(i) = [x_i, y_i]^T$ to their chosen constant values. The number of cities and their coordinates depend on the problem at hand while the N_{IWD} is set by the user. Here, we choose N_{IWD} to be equal to the number of cities N_c. For velocity updating, we use parameters $a_v = 1000$, $b_v = .01$, and $c_v = 1$. For soil updating, we use parameters $a_s = 1000$, $b_s = .01$, and $c_s = 1$. Moreover, the 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 $soil(i, j) = InitSoil$. The initial velocity of IWDs is denoted by the constant $InitVel$. Both parameters $InitSoil$ and $InitVel$ are also user selected. In this paper, we choose $InitSoil = 1000$ and $InitVel = 100$. The best tour is denoted by T_B which is still unknown and its length is initially set to infinity: $Len(T_B) = \infty$. Moreover, we should specify the maximum number of iterations that the algorithm should be repeated or some other terminating condition suitable for the problem.

2. Initialization of dynamic parameters: For every IWD, we create a visited city list $V_c(IWD) = \{\}$ set to the empty list. The velocity of each IWD is set to $InitVel$ whereas the initial soil of each IWD is set to zero.

3. For every IWD, randomly select a city and place that IWD on the city.

4. Update the visited city lists of all IWDs to include the cities just visited.

5. For each IWD, choose the next city *j* to be visited by the IWD when it is in city *i* with the following probability:

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

such that $f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))}$ and $g(soil(i,j)) = \begin{cases} soil(i,j) & if \min_{l \notin vc(IWD)} (soil(i,l)) \geq 0 \\ soil(i,j) - \min_{l \notin vc(IWD)} (soil(i,l)) & else \end{cases}$.

Here ε_s is a small positive number to prevent a possible division by zero in the function $f(.)$. Here, we use $\varepsilon_s = 0.01$. The function $\min(.)$ returns the minimum value among all available values for its argument. Moreover, $vc(IWD)$ is the visited city list of the IWD.

6. For each IWD moving from city *i* to city *j*, update its velocity as follows

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

such that $vel^{IWD}(t+1)$ is the updated velocity of the IWD. $soil(i,j)$ is the soil on the path (link) joining the current city i and the new city j. With formula (2), the velocity of the IWD increases less if the amount of the soil is high and the velocity would increase more if the soil is low on the path.

7. For each IWD, compute the amount of the soil, $\Delta soil(i,j)$, that the current water drop IWD loads from its the current path between two cities i and j:

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s . time\left(i,j;vel^{IWD}\right)} \quad (3)$$

such that $time\left(i,j;vel^{IWD}\right) = \frac{\| \mathbf{c}(i) - \mathbf{c}(j) \|}{\max\left(\varepsilon_v, vel^{IWD}\right)}$ which computes the time taken to travel from city i to city j with the velocity vel^{IWD}. Here, the function $\mathbf{c}(.)$ represents the two dimensional positional vector for the city. The function $\max(.,.)$ returns the maximum value among its arguments, which is used here to threshold the negative velocities to a very small positive number $\varepsilon_v = 0.0001$.

8. For each IWD, update the soil of the path traversed by that IWD using the following formulas:

$$\begin{aligned} soil(i,j) &= (1-\rho).soil(i,j) - \rho.\Delta soil(i,j) \\ soil^{IWD} &= soil^{IWD} + \Delta soil(i,j) \end{aligned} \quad (4)$$

where $soil^{IWD}$ represents the soil that the IWD carries. The IWD goes from city i to city j. The parameter ρ is a small positive number less than one. Here we use $\rho = 0.9$.

9. For each IWD, complete its tour by using steps 4 to 8 repeatedly. Then, calculate the length of the tour $Tour^{IWD}$ traversed by the IWD, and find the tour with the minimum length among all IWD tours in this iteration. We denote this minimum tour by T_M.

10. Update the soils of paths included in the current minimum tour of the IWD, denoted by T_M:

$$soil(i,j) = (1-\rho).soil(i,j) + \rho.\frac{2.soil^{IWD}}{N_c(N_c-1)} \quad \forall (i,j) \in T_M \quad (5)$$

11. If the minimum tour T_M is shorter than the best tour found so far denoted by T_B, then we update the best tour by

$$T_B = T_M \quad and \quad Len(T_B) = Len(T_M) \quad (6)$$

12. Go to step 2 unless the maximum number of iterations is reached or the defined termination condition is satisfied.

13. The algorithm stops here such that the best tour is kept in T_B and its length is $Len(T_B)$.

It is reminded that it is also possible to use only T_M and remove step 11 of the IWD algorithm. However, it is safer to keep the best tour T_B of all iterations than to count on only the minimum tour T_M of the last iteration.

V. Experimental Results

In this section, we test the proposed intelligent water drops for solving the TSP. At first, artificial problems are generated and tested by the IWDs. Then, some standard TSP problems are used for evaluating the performance of the IWDs.

The cities of the artificial problems are points on a circle which are equally spaced. Each point on the perimeter of the circle represents a city in the TSP. The first experiment involves 30 cities on the circle. The initial parameters of the IWD algorithm are set according to the steps 1 and 2 of the algorithm. At the initial, each link between two cities has the same amount of soil denoted by $InitSoil$. As the time passes, the links will have different amount of soils and the algorithm prefers links with less soil to links with more soil. The Intelligent Water Drops are randomly spread over the cities. In this experiment, we use 30 IWDs. Therefore, the number of cities and the number of IWDs are equal here.

Fig.1(a) shows the best tour found by the IWDs after one iteration for the TSP problem with 30 cities on the circle. When all IWDs complete one complete tour of themselves in the proposed algorithm, we say one iteration of the algorithm has been passed. In the algorithm, the best tour found so far is always kept. The best tour at some other iterations are also shown in Figs. 1(b)-(f) after two, three, four, five, and 12 iterations, respectively. After 12 iterations, the algorithm converges and no change in the best tour (the same as the minimum tour in this experiment) is observed. In fact, the global optimum tour has been found by the IWDs for this TSP problem.

For this experiment, the length of the best tour versus the iteration is shown in Fig. 2. As it is seen, it converges with a sharp descending curve to the global optimum.

However, it is not always guaranteed that the global optimum is found in each run of the algorithm. Sometimes the algorithm falls into a good local optimum. The next experiment shows such a good local optimum. To make it more visible, we use fewer cities on the circle and run the algorithm for a 10-city problem several times to reach a case in which the algorithm doesn't reach the global optimum and falls into a local optimum. Such a case is shown in Fig. 3 in which we see a small self-crossing in the best tour obtained by the algorithm. The algorithm converges to this local optimum after four iterations. However, this local optimum does not usually occur for the algorithm. Moreover, this local self-crossing can be removed by some simple heuristics [9].

Another experiment is with 100 cities located on the perimeter of the circle. This experiment is to test the ability of the IWD algorithm in dealing with more cities. Four best tours obtained by the IWD algorithm are shown in Fig. 4 for first, second, fifth, and 13th iterations. The algorithm converges at the 13th iteration with the length 320. The global optimum tour has the length 314 and is very close to the tour length obtained by the algorithm. The tour found by

the IWD algorithm is a very good local optimum. As Fig. 4 shows the converged tour has no self-crossing in the middle of the circle. Therefore, only small self-crossing happens in local neighborhood of cities, which again can be removed by some appropriate local heuristics.

Fig. 5 shows the relation of the length of the minimum tour versus its iteration obtained by the IWD algorithm for the 100-city problem of Fig. 4. Almost all parts of the curve is descending, except for the iteration 8 which shows a slight increase in the length of minimum tour in contrast to its previous iteration. However, after this short ascending, the curve follows its general downward movement. This property demonstrates that the proposed IWD is able to go upward to get rid of some local optimum in order to get to better optimums. This property makes the proposed algorithm more appealing. We see such a property in some other strong optimization algorithms such as simulated annealing and genetic algorithms.

In the following, the proposed IWD algorithm is tested by some standard TSP problems [10]. The eil51 problem is a 51-city problem with the known optimum tour length 426. The IWD algorithm is tested with eil51 and it gives the average tour length 470 over 10 runs. An example run of the algorithm for eil51 is shown in Fig. 6 after one, two, 15, and 50 iterations. The IWD algorithm finally gets to the optimum length 471 which is close to the global optimum tour with the length 426.

The IWD algorithm is also tested by eil75 and kroA100 which are 75-city and 100-city TSP problems, respectively. The converged tours for eil75 and kroA100 obtained by the IWDs are shown in Figs. 7 and 8, respectively. The algorithm obtains a tour with the length 559 after 300 iterations in contrast to the length 538 of the global optimum tour of eil75. Moreover, the IWD algorithm reaches to the tour with the length 23156 after 1500 iterations which the result tour is close to the length 21282 of the global optimum tour of kroA100.

To become sure that the results obtained by proposed IWD algorithm in the aforementioned experiments are not found by chance, we develop some experiments here to statistically demonstrate the reproducibility and power of the algorithm in finding at least good local optimums. The problem of Fig. 4 is used here in which we have 100 cities on a circle. We apply the IWD algorithm to that problem for 20 independent runs. In each run, the algorithm is continued for 50 iterations and then the best tour length is identified. As shown in Fig. 9, in each run of the IWD algorithm, the best tour length is close enough to the global optimum tour length 314.10 denoted by the dotted line.

Now, we examine the performance of the IWD algorithm as the problem size increases. The cities are considered to be on a circle but the number of cities is increased by five from 10 to 100. For each case, the IWD algorithm runs 10 times and the average best tour length is calculated. These average best tour lengths versus the number of cities of the TSP are depicted in Fig. 10. The dotted lines show the global optimum tour length whereas the solid lines show the average best tour lengths. It is seen that the IWD algorithm performs so well in reaching close enough to the global optimums no matter how big are the size of the problems.

It should be mentioned that the IWD algorithm converges fast and needs moderate or few iterations to converge to good optimum solutions in comparison to other optimization algorithms [11]. Moreover, each iteration of the IWD algorithm is computationally light.

If we look at the ant colony based algorithms [11], we see that the ants change the pheromone deposits on the paths they travel. Every ant leaves a certain amount of pheromone in each path it follows. A similar role is played in the proposed IWD algorithm by the water drops. The amounts of soils are changed by the water drops. However, in contrast to the ants, these changes are not constant and are dependent on the velocity and soil of the water drop traversing the path. Moreover, the water drops may have different velocities whereas in ant colony based algorithms the speeds of ants are not considered.

VI. Conclusion

Some properties that exist in natural water drops flowing in rivers are adopted in an algorithm here for solving optimization problems. In this paper, the proposed IWD algorithm is designed to solve the TSP. The IWD algorithm is experimented by artificial and some benchmark TSP environments. The proposed algorithm converges fast to optimum solutions and finds good and promising results. This research is the beginning of using water drops ideas to solve engineering problems. So, there is much space to improve and develop the IWD algorithm.

References

[1] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, 2003.

[2] S. Haykin, *Neural Networks*, Prentice-Hall, second edition, 1999.

[3] H. Shah-Hosseini, "The time adaptive self-organizing map is a neural network based on Artificial Immune System," In *Proc. IEEE World Congress on Computational Intelligence*, Vancouver, Canada, July 2006, pp. 1007-1014.

[4] M. Dorigo and T. Stutzle, Ant Colony Optimization, Prentice-Hall, 2004.

[5] Russ C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," In *Proc. Sixth Intl. Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, pp. 39–43.

[6] S. Kirkpatrik, "Optimization by simulated annealing: quantitative studies," *Journal of Statistical Physics*, vol. 34, 1984, pp. 975-986.

[7] Leonard M. Adleman, "Molecular computation of solutions to combinatorial problem," *Science*, 1994, pp. 1021–1023.

[8] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[9] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Syst. Journal*, vol. 44, 1965, pp. 2245-2269.

[10] TSP Library, Available: http://www.informatik.uni-heidelberg.de/groups/comopt/software/TSPLIB95/STSP.html

[11] E. Bonabeau, M. Dorigo, and G. Theraultz, *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, 1999.

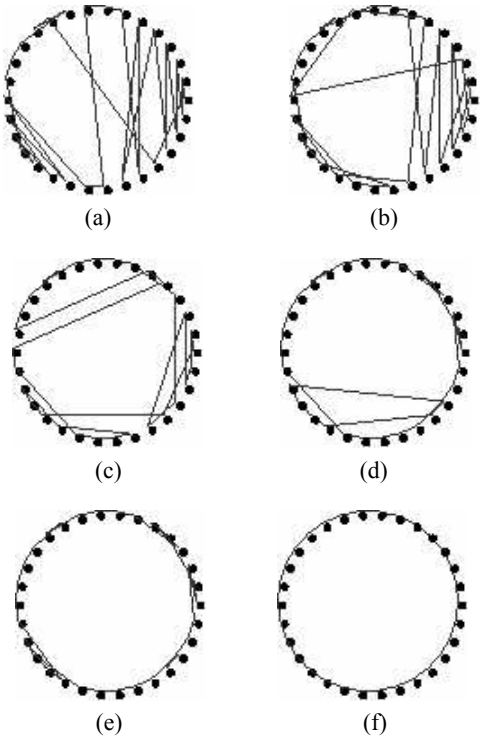

Fig. 1. The best tours found using the proposed IWD algorithm for a 30 city TSP problem. (a) The best tour after one iteration. (b) The best tour after two iterations. (c) The best tour after three iterations. (d) The best tour after four iterations. (e) The best tour after five iterations. (f) The best tour after 12 iterations which in fact is the global optimum tour in this case.

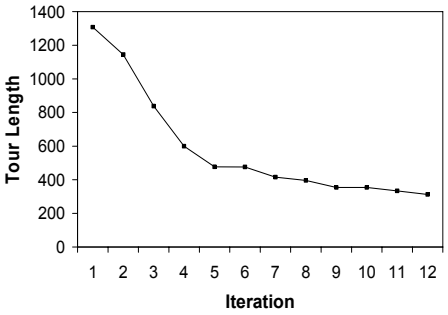

Fig. 2. The length of the best tour (or the minimum tour) versus the iteration which is found by the IWD algorithm for the 30-city problem of Fig.1.

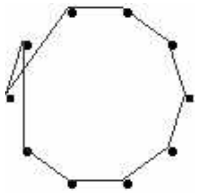

Fig. 3. An example of a case in which the proposed IWD converges to a good local optimum. It is seen a small self-crossing in the best tour obtained by the algorithm for the 10-city TSP problem.

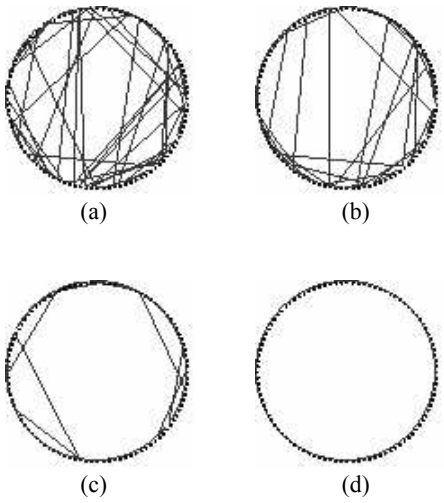

Fig. 4. The best tours found using the proposed IWD algorithm for a 100 city TSP problem. (a) The best tour after one iteration. (b) The best tour after two iterations. (c) The best tour after five iterations. (d) The best tour after 13 iterations which is the tour that the algorithm converges to with the total tour length 320. This length is very close to the global optimum tour length 314.

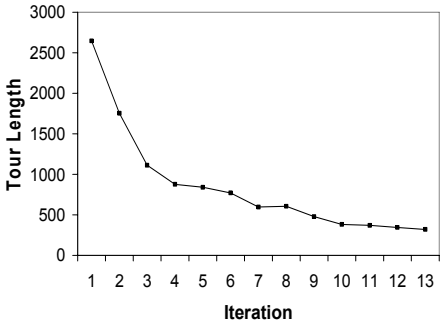

Fig. 5. The length of the minimum tour versus the iteration found by the IWD algorithm for the 100-city problem of Fig. 4. The algorithm converges to a very good local optimum. A slight upward movement at iteration 8 reveals that the algorithm can jump out of some local optimums to move to better optimums hopefully to the global optimum.

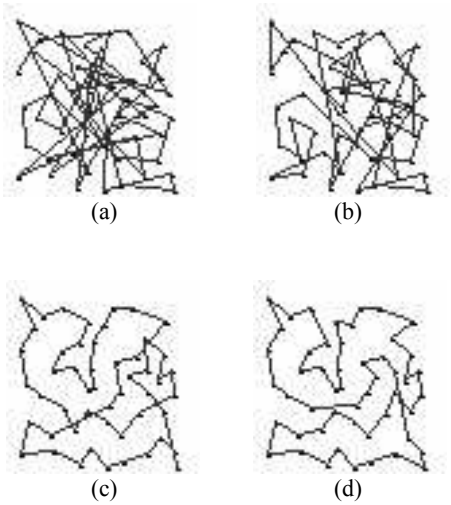

Fig. 6. The best tours found using the proposed IWD algorithm for the 51-city TSP problem eil51. (a) The best tour after one iteration. (b) The best tour after two iterations. (c) The best tour after 15 iterations. (d) The best tour after 50 iterations which is the tour that the algorithm converges to with the total length 471. This length is quite close to the global optimum tour length 426.

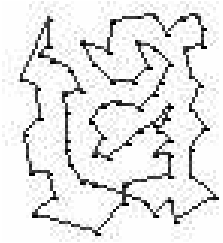

Fig. 7. The best tour found by the proposed algorithm after 300 iterations for the 76-city problem eil76. The algorithm gets a good local optimum with the tour length 559 which is quite close to the global optimum 538.

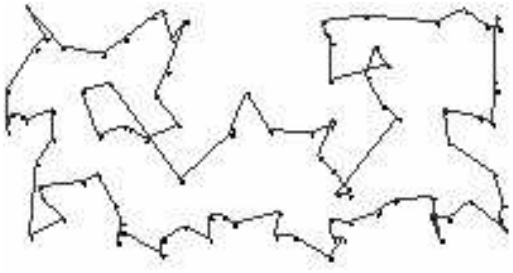

Fig. 8. The best tour found by the proposed algorithm after 1500 iterations for the 100-city problem kroA100. The algorithm gets a good local optimum with the tour length 23156 which is quite close to the global optimum 21282.

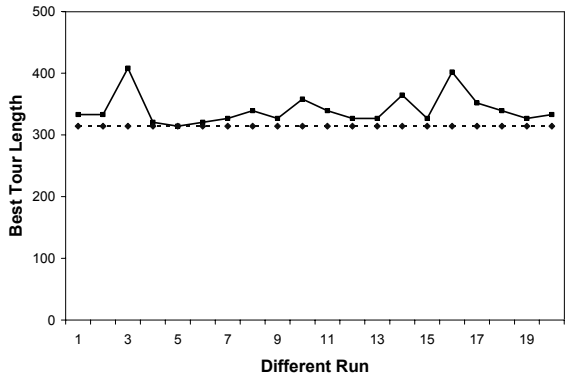

Fig. 9. The length of best tour of the IWD algorithm in 20 independent runs after 50 iterations for the 100-city problem of Fig. 4. The global optimum tour length is 314.10 and is shown by the dotted line. As it is seen, the tour lengths of the proposed algorithm denoted by the solid lines are mostly very close to the shortest tour length.

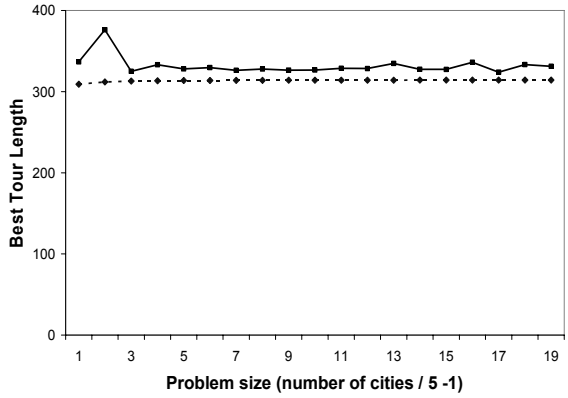

Fig. 10. The average length of the best tours of the IWD algorithm in 10 independent runs for the TSP problems in which the cities are on a circle. The number of cities is increased from 10 to 100 by the value of five, and in each case the best average tour length over 10 runs is depicted. The dotted lines show the global optimum tour length whereas the solid lines are the best tour lengths obtained by the IWD algorithm.