Neurocomputing 151 (2015) 1227–1236

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A discrete invasive weed optimization algorithm for solving traveling salesman problem

Yongquan Zhou [a,b], Qifang Luo [a], Huan Chen [a], Anping He [a,b], Jinzhao Wu [a,b]

[a] College of Information Science and Engineering, Guangxi University for Nationalities, Nanning 530006, China
[b] Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning 530006, China

ARTICLE INFO

Article history:
Received 29 September 2013
Received in revised form
28 November 2013
Accepted 25 January 2014
Available online 8 November 2014

Keywords:
Discrete invasive weed optimization
Positive integer encode
3-Opt
I2Opt
TSP

ABSTRACT

The Traveling Salesman Problem (TSP) is one of the typical NP-hard problems. Efficient algorithms for the TSP have been the focus on academic circles at all times. This article proposes a discrete invasive weed optimization (DIWO) to solve TSP. Firstly, weeds individuals encode positive integer, on the basis that the normal distribution of the IWO does not change, and then calculate the fitness value of the weeds individuals. Secondly, the 3-Opt local search operator is used. Finally, an improved complete 2-Opt (I2Opt) is selected as a second local search operator for solve TSP. A benchmarks problem selected from TSPLIB is used to test the algorithm, and the results show that the DIWO algorithm proposed in this article can achieve to results closed to the theoretical optimal values within a reasonable period of time, and has strong robustness.

1. Introduction

The traveling salesman problem (TSP) is one of the most cited NP-hard combinatorial optimization problems because it is so intuitive and easy to understand but difficult to solve. It is a challenging optimization problem of significant academic value as it is often used as a benchmark problem when new solution approaches are developed. Intensive research efforts have therefore been directed toward the development of both exact and heuristic algorithms for the TSP [1]. In fact, the TSP and its variants have several important applications, such as, drilling of printed circuit boards, X-ray crystallography, computer wiring, vehicle routing and scheduling, control of robots, among others. Therefore, solving this class of problems is both of academic interest and practical importance, and consequently, it has been an important topic of active research. So searching for an efficient algorithm has an important theoretical and practical significance. From the graph theory point of view, TSP problem can be described as weighted graph:$G=(V,E,w)$, where $V$is vertex set, $|V|=n$ is the number of vertex, $E$is edge set, $w: E\rightarrow R^{+}$is weight function. The purpose of TSP is to find a Hamilton loop that all the vertices are visited once and only once. Let $T=<v_1,v_2,\cdots,v_n,v_1>$, and $\forall 1\le i\ne j\le n, v_i\ne v_j$, moreover $1\le i\le n, <v_i,v_{\mathrm{mod}(i,n)+1}>\in E$. Let $TSP(G)$be the set of all TSP loop. Define$w(T)=w(<v_1,v_n>)+\sum_{i=1}^{n-1}w(<v_i,v_{i+1}>)$, so TSP problem is finally to find a Hamilton loop T^* that make $w(T)$ minimal, that is to say $T^*=\arg\{w(T)\}\rightarrow \min, \forall T\in TSP(G)$.

No efficient algorithm exists for the TSP and all its relevant variants or problems of the same class. The need to quickly find good (not necessarily optimal) solutions to these problems has led to the development of various approximation algorithms such as metaheuristics. Metaheuristic algorithms have demonstrated their potential and effectiveness in solving a wide variety of optimisation problems and have many advantages over traditional algorithms. Two of the advantages are simplicity and flexibility. Metaheuristics are usually simple to implement, but they often can solve complex problems and can thus be adapted to solve many real-world optimization problems, from the fields of operations research, engineering to artificial intelligence. In addition, these algorithms are very flexible, and they can deal with problems with diverse objective function properties, either continuous, or discrete, or mixed. Such flexibility also enables them to be applied to deal a large number of parameters simultaneously.

At present, all the methods to solve TSP can be divided into two categories, one is exact methods that guarantee the optimal solution, and the other is approximation algorithm. The exact methods can guarantee to get the optimal solution, but with the expansion of the scale of the problem, the solving time required increase with an exponential, so it is difficult to apply to solve large scale problems. The common exact methods include dynamic programming method [2] (DM), branch and bound method [3]. While Approximation algorithm can get more

E-mail addresses: yongquanzhou@126.com (Y. Zhou), l.qf@163.com (Q. Luo), 15977776020@126.com (H. Chen), hapetis@gmail.com (A. He), Wu_jz@gmail.com (J. Wu).

http://dx.doi.org/10.1016/j.neucom.2014.01.078
0925-2312/

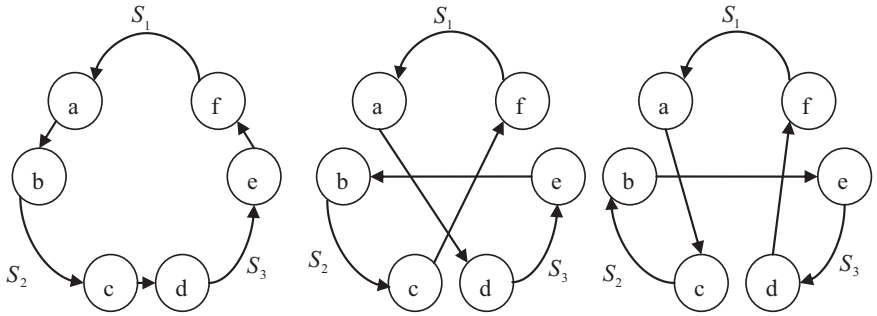


Fig.1. Schematic diagram of 3-Opt.

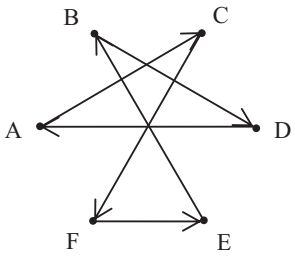


Fig. 2. Tour*ACFEBD*.

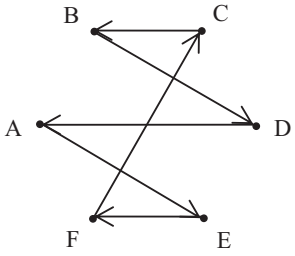


Fig. 3. Tour*AEFCBD*.

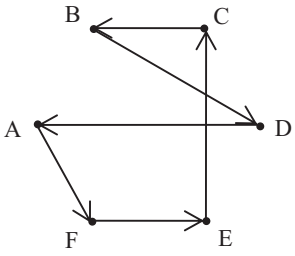


Fig. 4. Tour*AFECBD*.

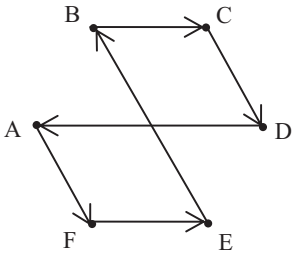


Fig. 5. Tour*AFEBCD*.

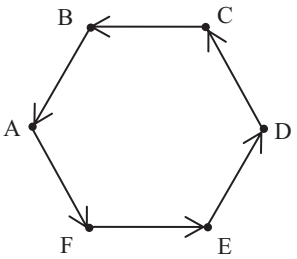


Fig. 6. Tour*AFEDCB*.

accurate solution in polynomial time, thus they are suitable for solving large-scale problems. Approximation algorithms can be divided into two categories also; one is local search algorithms which related to the problems' characteristics, such as 2-Opt [4], 3-Opt [5], LK [6], the LKH [7] and Inver-over [8]. These algorithms can make effective use of the relevant characteristics of the problem to find the local optimal solution of the problem, but with the scale of the problem increase, the computation will greatly increase. The other is heuristic optimization methods developed so far for searching nearly optimal solution in solving TSP such as ant colony algorithm [9] (ACO), genetic algorithm (GA) [10], simulated annealing algorithm (SA) [11], particle swarm optimization (PSO) [12], artificial neural networks (ANN) [13,24,34], and artificial immune algorithm (AIS) [14]. These algorithms do not depend on the problem itself, thus have strong global search capability, while easy to fall into local optimum. In recent years, many scholars combine local search with metaheuristic algorithms to produce a new hybrid algorithm for solving TSP, such as [15] genetic operators and LK are mixed; [16] proposed a method that combines the ant colony algorithm and mutation strategy; Samanlioglu et al. [17] proposed a method of combining genetic algorithm with a 2-Opt; Peng Gang et al. [18] proposed an improved complete 2-Opt (Complete 2-Opt, C2OPT), Yongquan zhou et.al proposed a discrete glowwom swarm algorithm (DGSO) [45], Xin-She Yang et.al proposed a discrete cuckoo search algorithm (DCS) [46] and the improved genetic algorithm is adopted, they can get satisfactory solution in fewer iterations. Besides, the above mentioned exact and heuristic algorithms, metaheuristic algorithms, have been applied successfully to the TSP by a number of researchers.

This article, a novel discrete invasive weed optimization (DIWO) to solve TSP is proposed. In the DIWO algorithm, Firstly, weeds individuals encode positive integer, on the basis that the normal distribution of the IWO does not change, and then calculate the fitness value of the weeds individuals. Secondly, the 3-Opt local search operator is used to weeds individuals. Finally, an Improve complete 2-Opt (I2Opt) is selected as a second local search operator for solve TSP. Benchmarks selected from TSPLIB are used to test the algorithm, and the results show that the algorithm proposed in this article can achieve to result closed to the theoretical optimal values within a reasonable period of time, and has strong robustness.

The rest of the article is organized as follows. In Section 2, the basic IWO is simply introduced. In Section 3 the DIWO, complete 2-opt, 3-Opt and I2Opt is proposed and explained in detail. Simulation and

Table 1
Parameters settings.

Parameter meaning	Variable	Value	Parameter meaning	Variable	Value
Lower bound of individual	*X_* min	-1000	The initial variance	*step_ini*	300
Upper bound of individual	*X_* max	1000	The final variance	*step_final*	0
The initial number of population	*G_SIZE*	2	The maximum number of seed generated	*seed_* max	10
The maximum number of population	*P_MAX*	4	The minimum number of seed generated	*seed_* min	1
The maximum iteration number	*iter_* max	1000	The nonlinear modulation index	*n*	4

Table 2
Computational results of DIWO algorithm for 20 TSP benchmark instances for TSPLIB.

Instance	Cities	Opt	Best (%)	Mean (%)	Std (%)	Time (sec)
Att48	48	33523	0.0021	0.0021	0	10.9
Eil51	51	426	0.6741	0.6999	0.3527	9.2
Berlin52	52	7542	0.0313	0.0313	0	9.8
St70	70	675	0.3125	0.3125	0	18.1
KroA100	100	21282	0.0161	0.0375	0.0210	91.0
KroB100	100	22141	0.6471	0.8816	0.1928	96.5
Pr107	107	44303	0.3096	0.4837	0.1598	143.4
Pr136	136	96772	0.2356	0.9400	0.4981	517
Chn144	144	30353	0.1016	0.8935	0.0103	451.2
KroA150	150	26524	0.7401	0.7780	0.0541	563.7
KroB150	150	26130	0.2789	0.3229	0.0666	524.2
D198	198	15780	0.4304	0.6691	0.3445	513.1
Tsp225	225	3859	0.4470	2.3949	2.8208	1872.4
Pr226	226	80369	0.0117	0.2238	0.1918	1127.6
A280	280	2579	0.4297	0.7679	0.6999	1639.5
Rd400	400	15281	1.7153	2.4229	0.8627	2516.9
Pcb442	442	50778	1.6137	2.1731	0.7989	3472.3
Att532	532	87550	1.2981	1.8737	0.8019	4634.4
Pr1002	1002	259045	2.6970	3.1873	2.6036	7054.7
Pr2392	2392	378032	2.4978	3.6192	3.1031	7836.4

comparisons are presented in Section 4. Finally, we end the article with some conclusions and future work in Section 5.

2. Invasive weed optimization algorithm

Invasive weed optimization algorithm (IWO) was proposed by Mehrabian and Lucas in 2006 [19], which is inspired from a common phenomenon in agriculture: colonization of invasive weeds. The algorithm has a simple structure, less parameters, strong robustness, easy to understand, and easy to program features. At present, as a new optimization method, IWO has been successfully applied to the training feed-forward neural networks [20], radial basis probabilistic neural networks [27,30,32,41,43], neural computing [36,37,39], constrained optimization of combustion at a coal-fired utility boiler [21], PID controller design [22], energy efficient trajectory planning [23], multi-user detection for MC-CDMA [35], tuning of auto-disturbance rejection controller [25], cooperative multiple task assignment of UAVS [26], design of non-uniform circular antenna arrays [42], design of encoding sequences for DNA [28], the piezoelectric actuator placement [29], image clustering [40], constrained engineering design [31], permutation flow-shop scheduling problem [44], and other issues.

In the basic IWO, weeds represent the feasible solutions of problems and population is the set of all weeds. A finite number of weeds is being dispread over the search area. Every weed produces new weeds depending on its fitness. The generated weeds are randomly distributed over the search space by normally distributed random numbers with a mean equal to zero. This process continues until maximum number of weeds is reached. Only the weeds with better fitness can survive and produce seed, others are being eliminated. The process continues until maximum iterations are reached or hopefully the weed with best fitness is closest to optimal solution.

The process is addressed in detail as follows:

Step 1: *Initialize a population*
A population of initial solutions is being dispread over the*D*dimensional search space with random positions.
Step 2: *Reproduction*
The higher the weed's fitness is, the more seeds it produces. The formula of weeds producing seeds is

$$weed_n = \frac{f - f_{\min}}{f_{\max} - f_{\min}}(s_{\max} - s_{\min}) + s_{\min} \quad (1)$$

where$f$is the current weed's fitness. $f_{\max}$and $f_{\min}$ respectively represent the maximum and the least fitness of the current population. $s_{\max}$and $s_{\min}$ respectively represent the maximum and the least value of a weed.
Step 3: *Spatial dispersal*
The generated seeds are randomly distributed over the*D*dimensional search space by normally distributed random numbers with a mean equal to zero, but with a varying variance. This ensures that seeds will be randomly distributed so that they abide near to the parent plant. However, standard deviation (σ) of the random function will be reduced from a previously defined initial value (σ_{init}) to a final value (σ_{final}) in every generation. In simulations, a nonlinear alteration has shown satisfactory performance, given as follows

$$\sigma_{cur} = \frac{(iter_{\max} - iter)^n}{(iter_{\max})^n}(\sigma_{init} - \sigma_{final}) + \sigma_{final} \quad (2)$$

where, $iter_{\max}$ is the maximum number of iterations, σ_{cur} is the standard deviation at the present time step and n is the nonlinear modulation index. Generally, n is set to 3.
Step 4: *Competitive exclusion*
After passing some iteration, the number of weeds in a colony will reach its maximum (*P_MAX*) by fast reproduction. At this time, each weed is allowed to produce seeds. The produced seeds are then allowed to spread over the search area. When all seeds have found their position in the search area, they are ranked together with their parents (as a colony of weeds). Next, weeds with lower fitness are eliminated to reach the maximum allowable population in a colony. In this way, weeds and seeds are ranked together and the ones with better fitness survive and are allowed to replicate. The population control mechanism also is applied to their offspring to the end of a given run, realizing competitive exclusion.

3. DIWO for solving TSP

Firstly DIWO is described. In order to remove the crossed boundary, 3-Opt and improved 2-Opt are used serially to solve TSP problem.

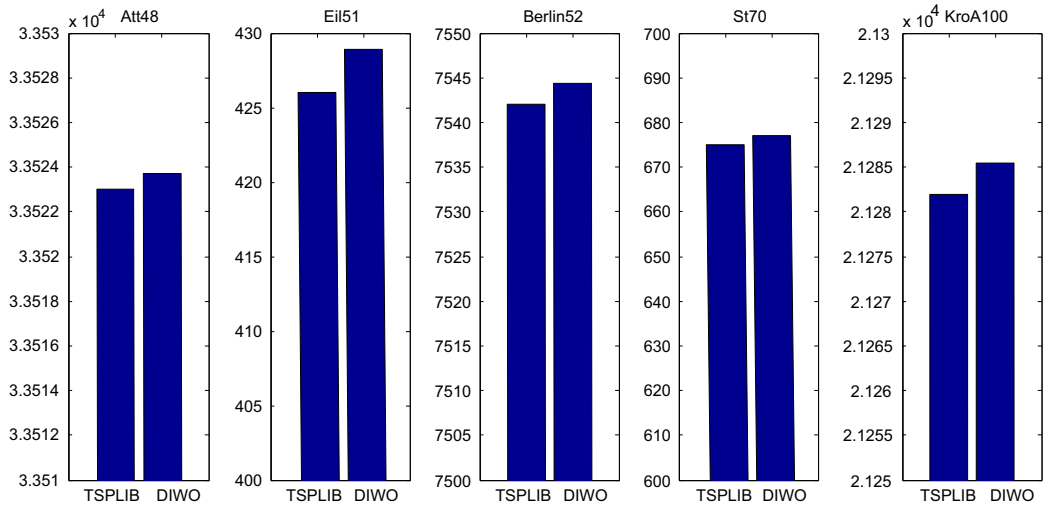


Fig. 7. Histogram about Att48, Eil51, Berlin52, St70, KorA100.

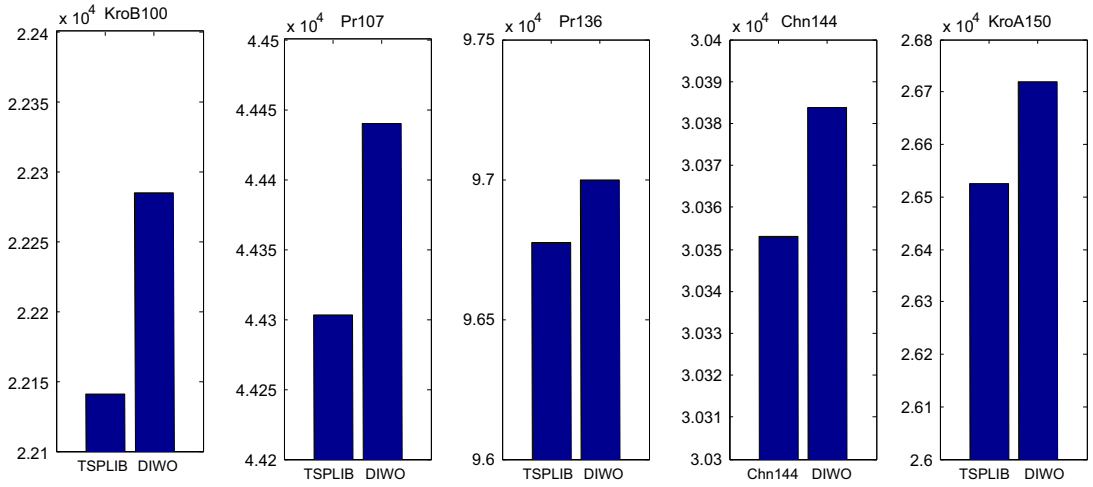


Fig. 8. Histogram about KroB100, Pr107, Pr136, Chn144, KroA150.

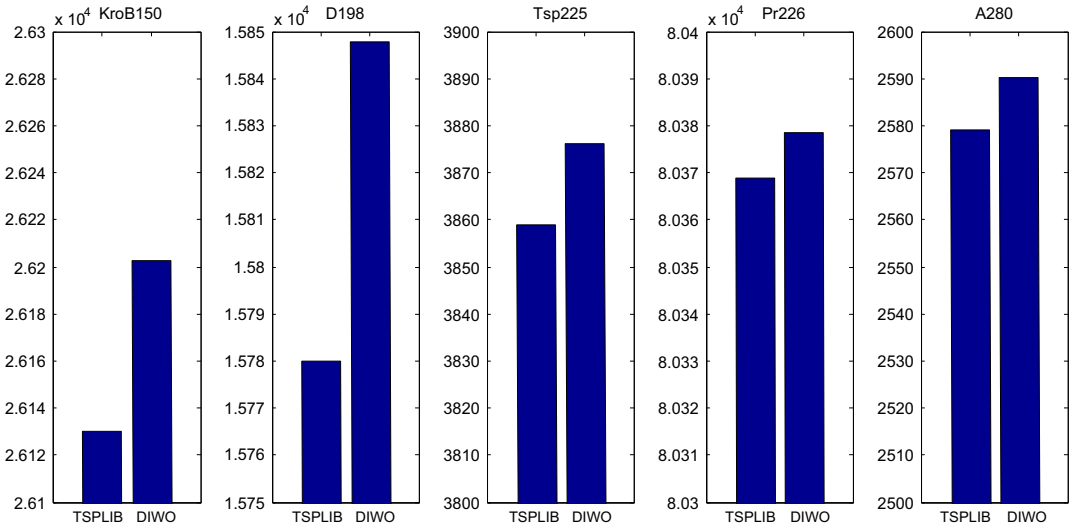


Fig. 9. Histogram about KroB150, D198, Tsp225, Pr226, A280.

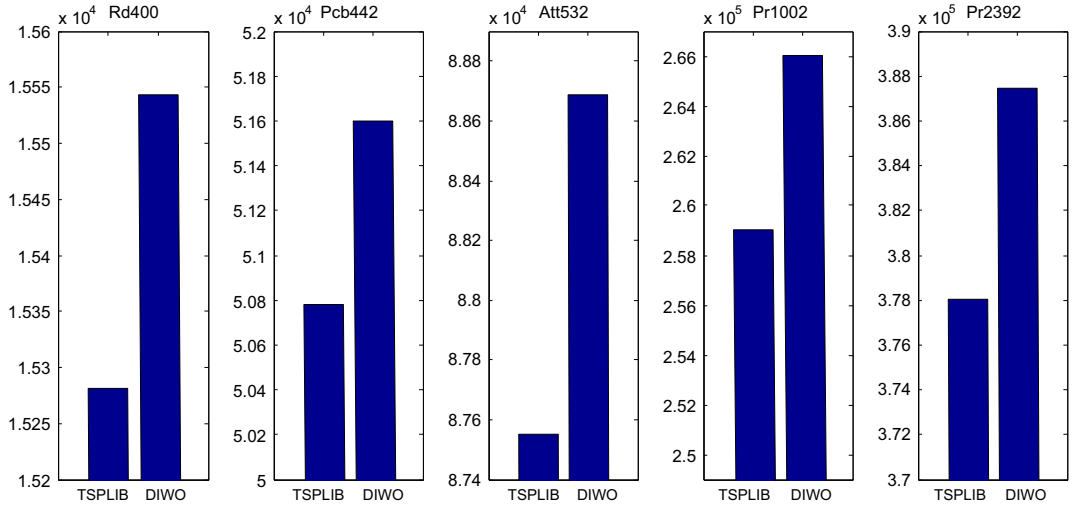


Fig. 10. Histogram about Rd400, Pcb442, Att532, Pr1002, Pr2392.

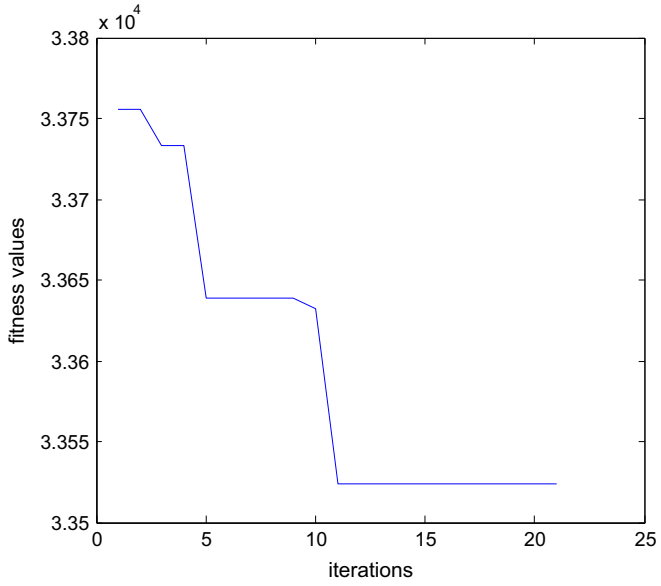


Fig. 11. Curve evolution diagram of Att48.

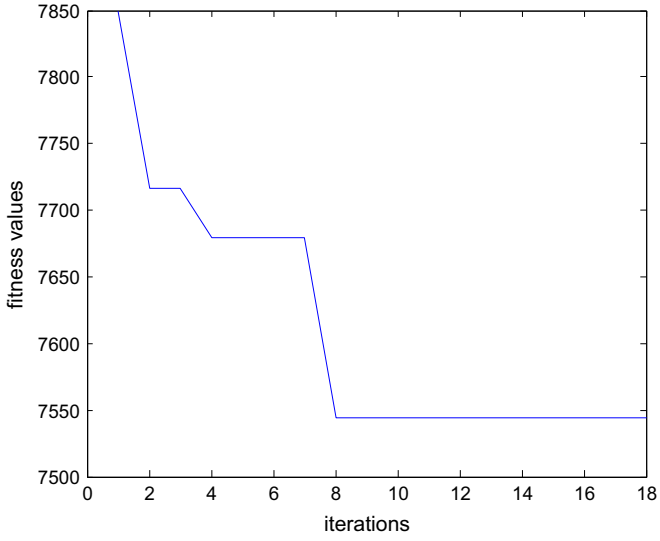


Fig. 13. Curve evolution diagram of Berlin52.

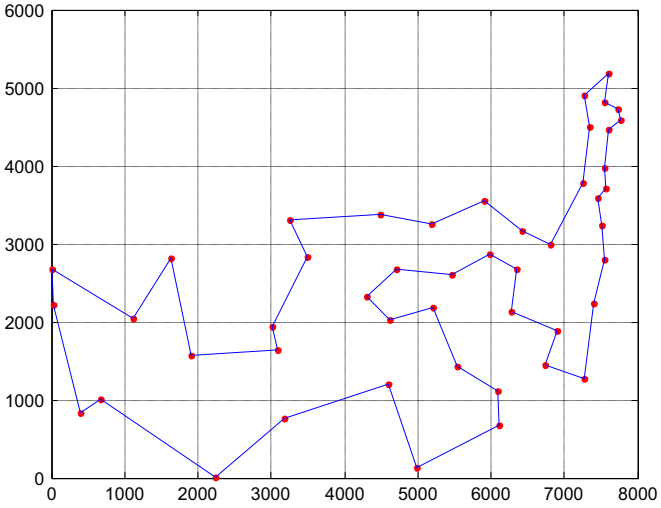


Fig. 12. Tour of Att48 obtained by DIWO .

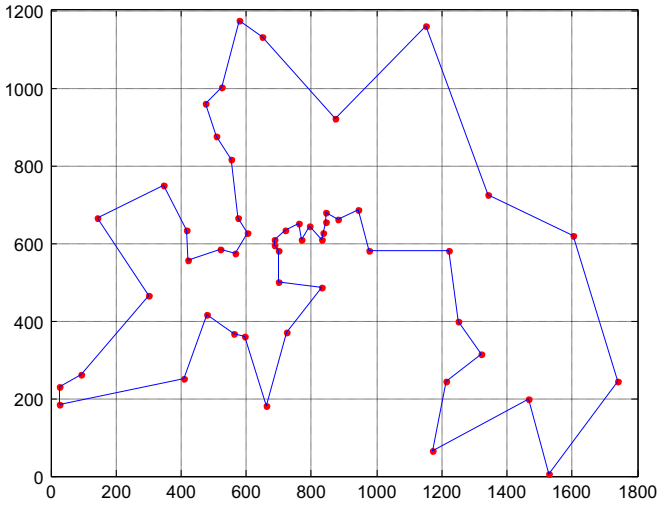


Fig. 14. Tour of Berlin52obtained by DIWO.

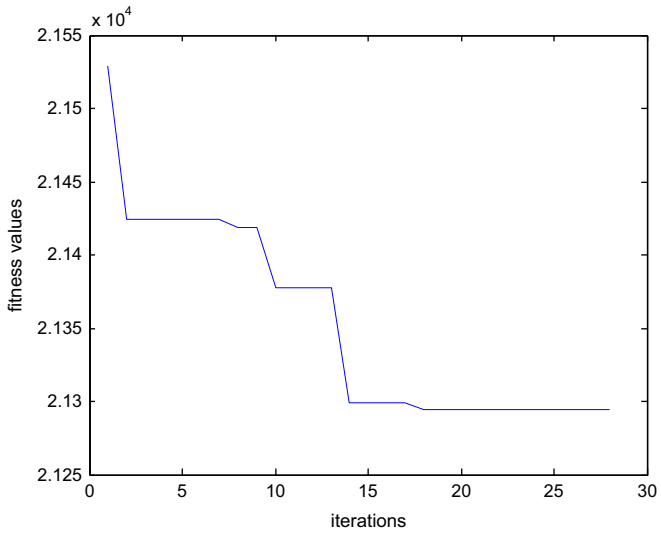


Fig. 15. Curve evolution diagram of Kroa100.

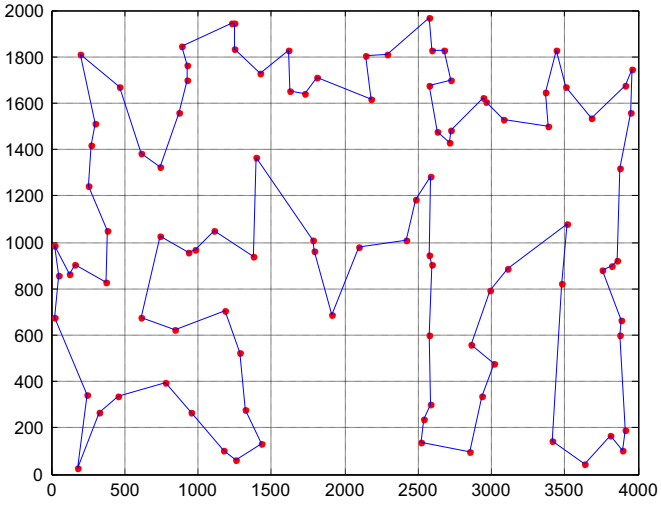


Fig. 16. Tour of Kroa100 obtained by DIWO.

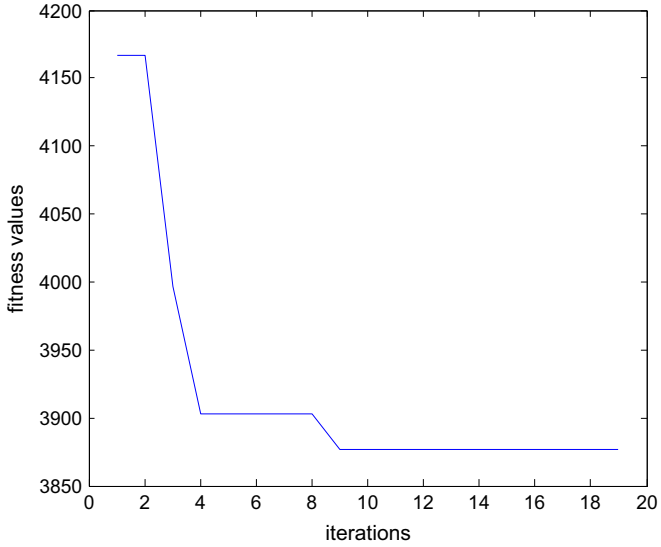


Fig.17. Curve evolution diagram of Tsp225.

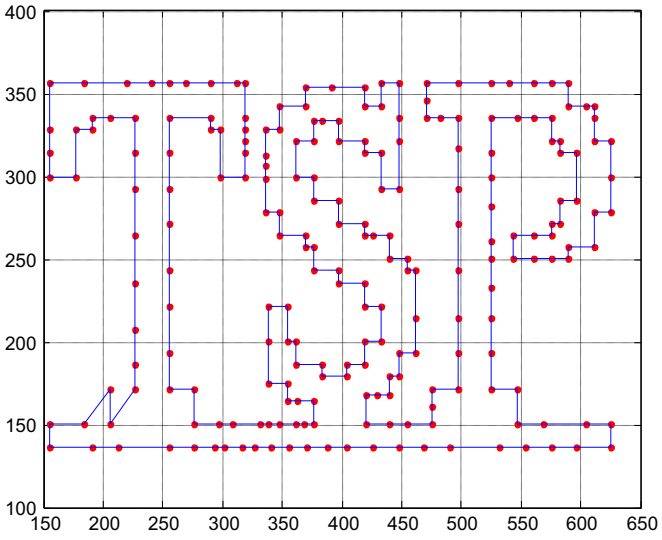


Fig. 18. Tour of Tsp225 obtained by DIWO.

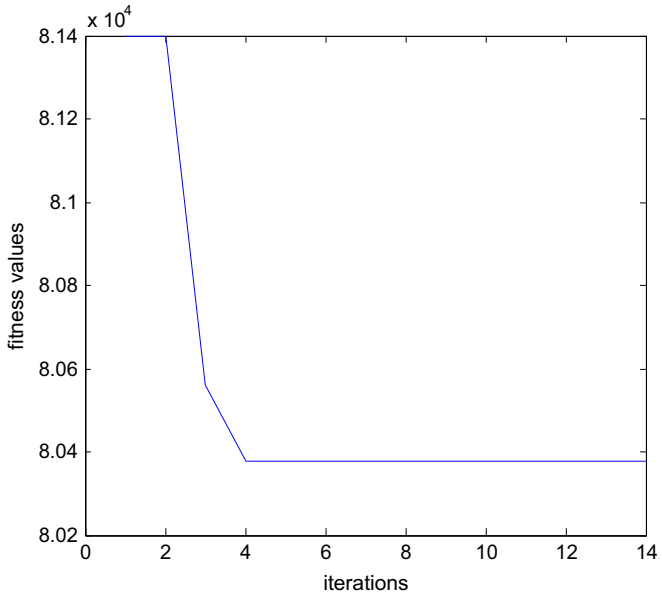


Fig. 19. Curve evolution diagram of Pr226.

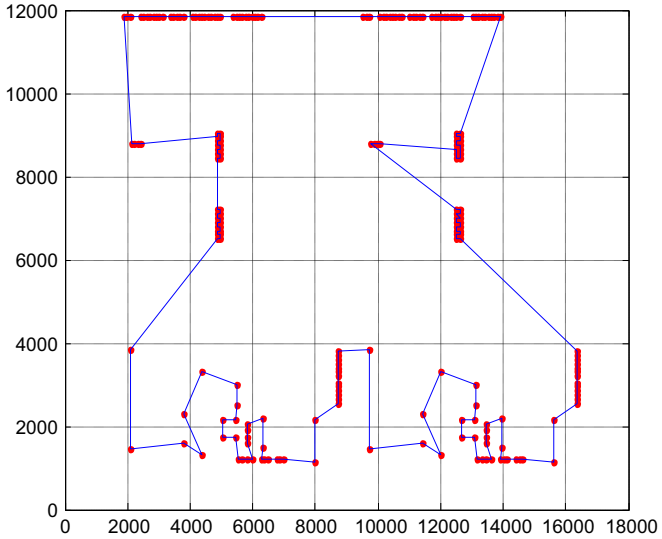


Fig. 20. Tour of Pr226 obtained by DIWO.

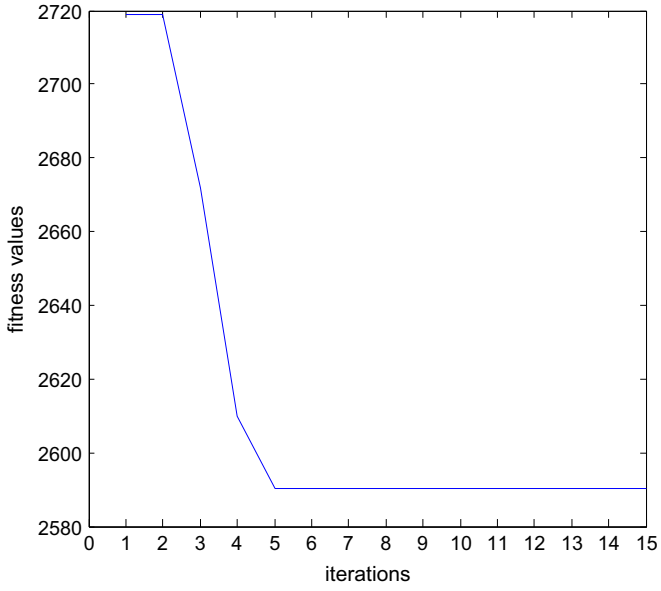


Fig. 21. Curve evolution diagram of A280.

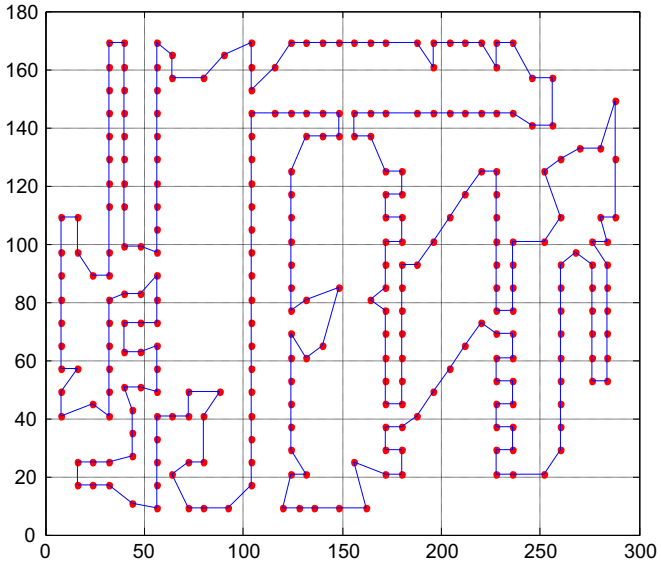


Fig. 22. Tour of A280 obtained by DIWO.

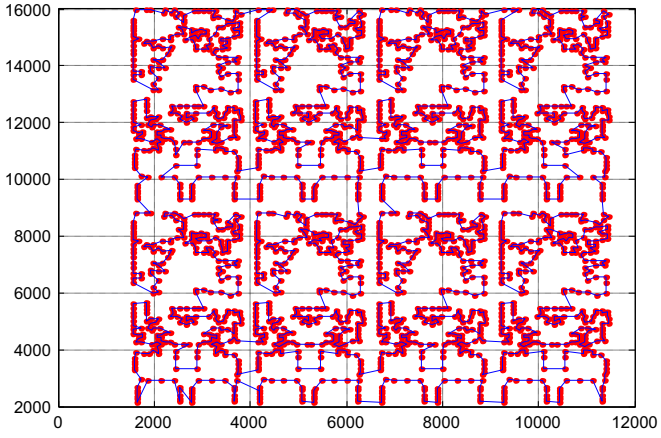


Fig. 23. Tour of Pr2392 obtained by DIWO.

Table 3
Comparison with other algorithms on 7 TSP benchmark instances from TSPLIB.

Instance	Cities	Opt	SWT[38]	ACS+2-Opt[33]	DCS[46]	DIWO
Eil51	51	426	430	431	426	**428**
Berlin52	52	7542	–	7573	7542	**7544**
St70	70	675	678.6	681	675	**677**
Pr107	107	44303	44303	44352	44303	**44440**
Tsp225	225	3859	3899.9	3951	3916	**3876**
Pr226	226	80369	–	80618	80386	**80378**
Pcb442	442	50778	52194.7	51788.4	–	**51597**

3.1. DIWO

The discrete invasive weed optimization algorithm can be described as follows.

Step 1: *Population is initialized with positive integer.*
A full array randomly produced with range 1 to *n_city* is acted as a weed individual, where *n_city* the number of cities. Other weed individuals are produced as the same way.
Step 2: *Reproductions.*
The number of individual seeds is calculated by the same method with IWO.
Step 3: *Positive integer spatial diffusion*
In IWO algorithm, all individuals of each generation have a diffuse step size which decrease by generation. Diffuse value is calculated by the step size according to normal distribution. The next generation is produced with the way that parent added to diffuse value. In DIWO the diffuse step size is calculated by the same formula (2), but the difference with IWO is that the diffusion value generated by the step value is rounded before added to the weeds individuals (parent). Then individuals generated modulo*n_city*, which limits the individual cross the border. To avoid emerging number 0, individuals generated are added one after modulo. At this moment, individual perhaps exist same number, which means one city is visited not only once. So, the way that serial number of unvisited city replaced same number is adopted.
For example, the i individual in t generation is $x_i^t = (3,1,2,5,4)$. The step size is 9, and the five diffusion values produced by $N(0,9)$ are $(-3.8931, -14.9903, 1.1280, 2.5891, -10.3182)$. After diffusion values being rounded, the values are $(-4, -15, 1, 2, -11)$. Thus $(4,1,3,2,3)$ is calculated after modulo 5; $(5,2,4,3,4)$ will be got when values add 1. This moment, the 4^{th} city is visited twice while the first city is not visited. So number 1 will randomly replace one number 4, the next generation is $(5,2,4,3,1)$ at last.
Step 4: *Competitive exclusion*
Same approach with IWO is adopted. After each iterate, only the first *P_MAX* outstanding individuals are reserved.

3.2. Complete 2-Opt and 3-Opt operations

The 2-Opt algorithm removes randomly two edges from the already generated tour, and reconnects the new two paths created. This is referred as a 2-Opt move. The reconnecting edge is valid only if it is shorter than the older. This is continued till further improvement is possible. The resulting tour is now two optimal.

The 3-Opt works in a similar fashion, but instead of removing the two edges it removes three edges. Randomly selected three edges in the tour$S$, we will get three path fragment S_1、S_2、S_3. So there are two kinds of reconnection, as shown in Fig. 1. Assumed that edge $<c,d><e,f>$ and $<a,b>$ are removed, then edge $<a,d><e,b>$ and $<c,f>$ will be reconnect or edge

$< a, c >$ $< b, e >$ and $< d, f >$ will be reconnected. For the former, the fact is that S_2 and S_3 are reversed; For the latter, in fact, S_2 is reversed and $S_3$is reversed. The original tour$S$ will be replaced only if the reconnection tour S' is superior to the original tour S.

3.3. Improve complete 2-opt

Because randomly select two edges cost a lot of time, so, an improved complete 2-Opt is proposed.

Basic process of improve complete 2-Opt (I2Opt) is as follows:>

(1) Select a tour $T = < v_1, \cdots, v_i, v_{i+1}, \cdots, v_j, v_{j+1}, \cdots, v_n > i = 1, j = i+2$.
(2) Select an edge of the tour $< v_i, v_{i+1} >, i \le n-2$ as the first edge.
(3) Select an edge of the tour $< v_j, v_{j+1} >, j \le n$ as the second edge. If $j = n$ then$j+1 = 1$.
(4) If $w(< v_i, v_j >) + w(< v_{i+1}, v_{j+1} >) < w(< v_i, v_{i+1} >) + w(< v_j, v_{j+1} >)$, then 2-Opt will be implemented. $< v_i, v_{i+1} >$ and $< v_j, v_{j+1} >$ will be deleted. $< v_i, v_j >$ and $< v_{i+1}, v_{j+1} >$ will be added. Reverse the part between cities $v_{i+1}, v_j, i = 1, j = i+2$ go to (2). Otherwise go to (5).
(5) $j = j+1$.If$j > n$, go to (6); otherwise go to (3).
(6) $i = i+1$.If$i \le n-2$, go to (2); otherwise terminate process.

To describe I2O pt, TSP containing six cities is exampled, as shown in Figs. 2–6. The process is as follows: starting from point A, and found the cross edge AC and BD, then AE and CB respectively replace the AC and BD, that is, conversion from Fig. 2 to Fig. 3. Starting from point A again, found the cross edge AE and FC, then AF and EC respectively replace AE and FC, that is, conversion from Fig. 3 to Fig. 4. From point A until point E, and found the cross edge EC and BD, EB and CD will replace EC and BD respectively, that is, the conversion from Fig. 4 to Fig. 5. Starting from point A until point E, and found the cross edge EB and DA, using ED and BA replace EB and DA respectively, that is, Fig. 5 transformed to Fig. 6. At this moment, there is no longer cross edge in the tour.

3.4. Pseudo code of DIWO solving TSP

Pseudo Code

```
Begin
        Discrete Initialize(visit_sequence_weed), pop_size
  =N;
        iter=1;
        While iter < iter_max
            visit_sequence_weed=3-Opt
  (visit_sequence_weed);
            visit_sequence_weed =I2Opt
  (visit_sequence_weed);
            For i=1:N
                fitness(i)= -F(visit_sequence_weed);
          End For
                BestFitness=max(fitness);
            WorstFitness=min(fitness);
            stepLength=(iter_max-iter)\widehatn*
  (stepLength_ini-stepLength_final)/(iter_max)\widehatn
            +stepLength_final;
                For i=1:N
          num=(seed_max-seed_min)*(fitness(i)-
  WorstFitness)/(BestFitness-WorstFitness)
          +seed_min;
          visit_sequence_seed =normrnd(0,stepLength,num);
        End For
      visit_sequence_weed=3-Opt(visit_sequence_weed);
      visit_sequence_seed =I2Opt(visit_sequence_seed);
```

Pseudo Code

```
      If num(weed, seed) > P_SIZE
        pop=selectBetter(weed, seed, P_SIZE); N=P_SIZE;
      Else
          pop= join(weed, seed); N= num(weed, seed);
      End If
      End While
      Output: select the best visit_sequence
  (visit_sequence_weed or visit_sequence_seed)
          F(visit_sequence);
      End
```

4. Simulation experiments and results analysis

Experimental test cases are all from the TSPLIB library http://www.iwr.uni-heidelberg.de/groups /comopt/software/TSPLIB95, except for Chn144. In addition, the optimal values provided by the library TSPLIB are rounded to integer. In order to compare with other algorithms, four decimal is placed. Each test case in the simulation run independently 20 times and each run stop the algorithm when the test results for 10 consecutive generations do not change.

4.1. Test platform and parameters settings

The experimental program testing platform as: Processor: CPU Intel Core i3–370, Frequency: 2.40GHz, Memory: 4GB, Operating system: Windows 7.0, Run software: Matlab7.6. Parameters settings are given by Table 1 as following

4.2. Evaluation formula of experimental results

Table 2 summarize the experiments results, where the first column shows the name of the instance, the column 'opt' shows the optimal solution length taken from the TSPLIB, the column 'best' shows the length of the best solution found by DIWO algorithm, the column 'Mean' shows the length of the worst solution found by DIWO algorithm, *Std* shows the standard deviation, and the column 'time' shows the average time used by DIWO. The percentage deviation of a solution to the best known solution (or optimal solution if known) is given by the formula (3):

$$\text{error} = \frac{\text{result} - \text{opt}}{\text{opt}} \times 100\% \tag{3}$$

4.3. Experimental results

Table 2 shows the results obtained by DIWO for solving some TSP instances. It can be seen from the table2 that the results of instances' city size less than 100 is close to the optimal results provided by the TSPLIB library and the deviation of the results is smaller. Such as the Att48, Berlin52, St70, the deviations are 0 which indicate that DIWO can obtain optimal results 100% with good robustness. In addition, from Table 2 it can be seen that, for TSP test instance that city size less than 400, DIWO can obtain results with error less than 1% within an average time of 30 minutes, and the deviation is smaller. However, with the scale of the problem expansion, running time of DIWO is rapidly increasing. The running time is more than 30 minutes, even up to several hours, but the quality of the solution is satisfactory, and the error between the optimal and obtained by DIWO is in the [1.5% , 3%].

In order to facilitate observation, the comparison histogram about results obtained by DIWO and the optimal is given by

Figs. 7–10. The first column is the optimal result and the second column is results obtained by DIWO. From the comparison histogram, results obtained by DIWO can be read intuitively. For example, from Fig. 7 it can be seen that Berlin 52's result obtained by DIWO is close to 7544. With regard to TSP instances that city size less than 100, it can be seen from histogram that results obtained by DIWO is close to the optimal, while the error of instances' city size more than 400 is in the [1.5% , 3%].

Figs. 11–22 gives some examples of the curve evolution diagram. It can be seen from the figure that DIWO can achieve convergence in shorter iteration, such as Att48, the convergence iteration is about 10, and the convergence iteration of Pr226 and A280 is about five. That is to say, the convergence speed of algorithm proposed in this article is fast. Figs. 13, 15, 17, 19, 21, and 23 are respectively the tour obtained by DIWO of Att48, Berlin52, KroA100, Tsp225, Pr226, and A280. Fig. 23 is the tour obtained by DIWO about Pr2392.

4.4. Comparison with other algorithms

In order to verify the effect of the algorithm proposed in this article, Recurrent Neural Networks with Soft Winner Takes All (SWT) in [38] and Ant Colony System with 2-Opt (ACS+2-Opt) in [33], Discrete version of the Cuckoo Search algorithm [45] are selected to compare with DIWO. Table 3 gives the comparison with other algorithms on 7 TSP benchmark instances from TSPLIB. Opt shows the optimal value. Boldface letter is the best one of the three results obtained by these algorithms, '—' means that no result is provided in the original references.

In Table 3, the experimental results of the DIWO algorithm are compared with among methods SWT, ACS+2-Opt and DCS. It can be seen clearly from Table 3 that DIWO outperforms the other two algorithms (SWT and ACS+2-Opt), be second only to DCS in solving all the five tested TSP instances (Eil51, Berlin52, St70, Pr107, and Tsp225), the tested TSP instances Pr226 is better than DCS algorithm. But for instance Pcb442, the DCS didn't find the optimal solution, and DIWO finds the optimal solution.

5. Conclusions

In this article, we have extended and improved the standard invasive weed optimization algorithm (IWO) through weeds individuals encode positive integer by reconstructing its population, a new discrete invasive weed optimization algorithm (DIWO) is proposed. The DIWO is discretized and then used to solve the traveling salesman problem (TSP). DIWO has been implemented and its performance has been tested on twenty benchmark TSP instances. The results show that the algorithm proposed in this article can achieve to result closed to the theoretical optimal values within a reasonable period of time, and it also has strong robustness. The next research content is how to get a satisfactory solution for large scale TSP problem in a relatively shorter time. Through the experiment, we got good results. Some strategies may increase the searching speed of DIWO algorithms for dynamic TSP, forecasting the change pattern of the cities and per-optimizing, doing more experiments including changing the city number, and etcetera. These will be our future work.

Acknowledgments

This work is supported by National Science Foundation of China under Grant No. 61165015 and No.61463007. Key Project of Guangxi Science Foundation under Grant No. 2012GXNSFDA053028, Key Project of Guangxi High School Science Foundation under Grant No. 20121ZD008, the Funded by Open Research Fund Program of Key Lab of Intelligent Perception and Image Understanding of Ministry of Education of China under Grant No. IPIU01201100.

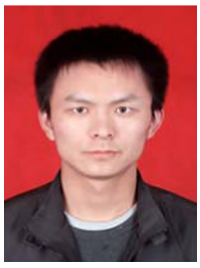
References

[1] Jiang He, H.u. Yan, Qiang Li, et al., Fat computational complexity and heuristic design for the TSP, J. Softw 20 (2009) 2344–2351.
[2] R.E. Bellman, S.E. Dreyfus, Applied Dynamic Programming, Princeton University Press, Princeton, New Jersey, 1962.
[3] E.L. Lawler, D.E. Wood, Branch and bound methods: a survey, Oper. Res 14 (1996) 699–719.
[4] G.A. Croes, A method for solving traveling salesman problem, Oper. Res 6 (1958) 791–812.
[5] S. Lin, Computer solutions of the traveling salesman problem, Bell Syst. Tech. J 44 (1965) 2245–2269.
[6] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res 21 (1973) 498–516.
[7] K. Helsgaun, An effective implementation of the lin-lemighan traveling salesman heuristic, Eur. J. Oper. Res 12 (2000) 106–130.
[8] T.a.o. Guo, Z. Michalewicz, Evolutionary algorithms for the TSP, Process of the 5th Parallel Problem Solving from Nature Conference (1998) 803–812.
[9] J.i. Junzhong, Zhen Huang, L.i.u. Chunnian, An ant colony algorithm based on multiple grain representation for the traveling salesman problems, J. Comput. Res. Dev 47 (2010) 434–444.
[10] H.U. Xiaobing, W.U. Shufan, J.U. Jiang, An advanced genetic algorithm for TSP, Comput. Technol. Automat 19 (2000) 34–38.
[11] S. Kirk Patrick, C.D. Gelatt Jr, M.P. Vecchi, Optimization by simulated annealing, Science 1220 (1983) 671–680.
[12] K. James, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, Proceedings of the IEEE Conference on Systems, Man, and Cybernetics (1997) 4104–4109.
[13] X.D. Liu, C.B. Xiu., A novel hysteretic chaotic neural network and its applications, Neurocomputing 70 (2007) 2561–2565.
[14] J. Huant, D. Cooke, Learning using an artificial immune system, J. Netw. Comput. Appl 19 (1996) 189–212.
[15] Merz P., Freisleben B. Genetic local search for the TSP: new results, Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, 1997, pp.159-163.
[16] J.H. Yang, X.H. Shi, M. Marchese., An ant colony optimization method for generalized TSP problem, Prog. Nat. Sci 18 (2008) 1417–1422.
[17] Samanlioglu F., Ferrell J.W.G., Kurz M.E. A memetic random key genetic algorithm for a symmetric multi-objective traveling salesman problem, IEEE T. Evolut. Comput. 5 (2006) 613-622.
[18] Gang Peng, Ichiro Iimura, Nakayama Shigeru, An evolutionary multiple heuristic with genetic local search for solving TSP, Int. J. Inform. Technol 14 (2008) 1–11.
[19] A.R. Mehrabian, C. Lucas, A novel numerical optimization algorithm inspired from weed colonization, Ecol. Inform 1 (2006) 355–366.
[20] Ritwik Giri, Aritra Chowdhury, Arnob Ghosh. A modified invasive weed optimization algorithm for training of feed-forward neural networks, IEEE International Conference on Systems Man and Cybernetics (SMC), Istanbul: IEEE, 2010, 3166-3173.
[21] Huan Zhao, Peihong Wang, Xianyong Peng, et al. Constrained optimization of combustion at a coal-fired utility boiler using hybrid particle swarm optimization with invasive weed, 2009 International Conference on Energy and Environment Technology (ICEET), Washington, DC, IEEE Computer Society, 2009, 564-567.
[22] D. Kundu, K. Suresh, S. Ghosh, et al., Designing fractional-order PIλDμ controller using a modified invasive weed optimization algorithm, World Congress on Nature & Biologically Inspired Computing (2009) 1315–1320.
[23] A. Sengupta, T. Chakraborti, A. Konar, Energy efficient trajectory planning by a robot arm using invasive weed optimization technique, Third World Congress on Nature and Biologically Inspired Computing (2011) 311–316.
[24] F.e.i. Han, Qing-Hua Ling, D.S. Huang, An improved approximation approach incorporating particle swarm optimization and a priori information into neural networks, Neural Comput. Appl 19 (2010) 255–261.
[25] Zhihua Chen, Shuo Wang, Zhonghua Deng Tuning of auto-disturbance rejection controller based on the invasive weed optimization, 2011 Sixth International Conference on Bio-Inspired Computing: Theories and Applications, 2011, 314–318.
[26] Mohsen Ramezani Ghalenoei, Hossein Hajimirsadeghi, Caro Lucas. Discrete invasive weed optimization algorithm: application to cooperative multiple task assignment of UAVS, 48th IEEE Conference on Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. 2009, 1695–1670.
[27] D.S. Huang, Horace H.S. Ip, Zheru Chi, A neural root finder of polynomials based on root moments, Neural Comput 16 (2004) 1721–1762.
[28] X. Zhang, Y. Wang, G. Cui, et al., Application of A novel IWO to the design of encoding sequences for DNA computing, Comput. Math. Appl 57 (2009) 2001–2008.
[29] A.R. Mehrabian, A. Yousefi-Koma, A novel technique for optimal placement of piezoelectric actuators on smart structure, J. Franklin Inst 348 (2011) 12–23.

[30] Shang Li, D.S. Huang, Ji-Xiang Du, Chun-Hou Zheng, Palmprint recognition using FastICA algorithm and radial basis probabilistic neural network, Neurocomputing 69 (2006) 1782–1786.
[31] S.u. Shoubao, Wang Jiwen, Zhang Ling, et al., An invasive weed optimization algorithm for constrained engineering design problems, J. Univ. Sci. Technol. China 39 (2009) 885–893.
[32] Ji-Xiang Du, D.S. Huang, Guo-Jun Zhang, Zeng-Fu Wang, A novel full structure optimization algorithm for radial basis probabilistic neural networks, Neurocomputing 70 (2006) 592–596.
[33] F.X. Le Louarn, M. Gendreau, J.Y. Potvin., Geni ants for the traveling salesman problem, Ann. Oper. Res 131 (2004) 187–201.
[34] D.S. Huang, Ji-Xiang Du, A constructive hybrid structure optimization methodology for radial basis probabilistic neural networks, IEEE T. Neural Netw 19 (2008) 2099–2115.
[35] Ho-Lung Hung, Chien-Chi Chao, Chia-Hsin Cheng. Invasive weed optimization method based blind multi-user detection for MC-CDMA interference suppression over multipath fading channel, IEEE International Conference on Systems Man and Cybernetics (SMC), 2010, 2145-2150.
[36] D.S. Huang, M.i. Jian-Xun, A new constrained independent component analysis method, IEEE T. Neural Netw 18 (2007) 1532–1535.
[37] Ji-Xiang Du, D.S. Huang, Xiao-Feng Wang, Xiao Gu, Shape recognition based on neural networks trained by differential evolution algorithm, Neurocomputing 70 (2007) 896–903.
[38] Paulo Henrique Siqueira, Maria Teresinha Arns Steiner, et al. Recurrent neural networks with the soft 'winner takes all' principle applied to the traveling salesman problem. 〈http://www.degraf.ufpr.br/docentes/paulo/publicacoes_arquivos/TSP_2010.pdf〉, 2012 (accessed 10.06.12).
[39] J.u.n. Zhang, D.S. Huang, Tat-Ming Lok, Michael R. Lyu, A novel adaptive sequential niche technique for multimodal function optimization, Neurocomputing 69 (2006) 2396–2401.
[40] Shoubao Su, J.i.e. Fang, Jiwen Wang, Image clustering method based on invasive weed optimization, J. South China Univ. Technol 36 (2008) 95–105.
[41] D.S. Huang, Wenbo Zhao, Determining the centers of radial basis probabilities neural networks by recursive orthogonal least square algorithms, Appl. Math. Comput 162 (2005) 461–473.
[42] Gourab Ghosh Roy, Swagatam Das, Prithwish Chakraborty, Design of non-uniform circular antenna arrays using a modified invasive weed optimization algorithm, IEEE T. Antennas Propag 59 (2011) 110–118.
[43] D.S. Huang, Radial basis probabilistic neural networks: model and application, Int. J. Pattern Recogn 13 (1999) 1083–1101.
[44] Huan Chen, Yongquan Zhou, Sucai He, Xinxin Ouyang, Peigang Guo, Invasive weed optimization algorithm for solving permutation flow-shop scheduling problem, J. Comput. Theor. Nanosci 10 (2013) 708–713.
[45] Yongquan Zhou, Zhengxin Huang, Hongxia Liu, Discrete glowworm swarm optimization algorithm for TSP problem, ACTA Electron. Sinica (in Chinese) 40 (2012) 1164–1170.
[46] Aziz Ouaarab, Belaıd Ahiod, Xin-She Yang, Discrete version of the cuckoo search algorithm for the traveling salesman problem, Neural Comput. Appl 24 (2014) 223–237.

Yongquan Zhou, Ph.D & Prof., received his MS degree in computer science from Lanzhou University, Lanzhou, China, in 1993 and the Ph.D degree in computation intelligence from the Xidian University, Xi'an, China, in 2006. He is currently a professor in Guangxi University for Nationalities. His research interests include computation intelligence, neural networks, and intelligence information processing et al. He has published 1 book, and more than 150 research papers in journals.

Qifang Luo, Associate Prof. received his BS degree from Guangxi University of School of Computer and Electronis Information, Guangxi, China, in 1993. He is currently research interest is in computation intelligence, neural networks.



Huan Chen, Ph.D., received his BS degree from Henan University of Science and Technology, Zhengzhou, China, in 2010. He is currently research interest is in computation intelligence, swarm intelligence algorithm.

Anping He, Ph.D & Associate Prof., received his Ph.D degree in computer science from Lanzhou University, Lanzhou, China, in 2006. He is currently research interest are in verification of integrated circuits, formalization method.

Jinzhao Wu, Ph.D & Prof., received his MS degree in computer science from Lanzhou University in 1991, and the Ph. D degree in Computation Science from the Peking University in 1995. His is currently a professor in Guangxi University for Nationalities. His research interests are in the areas of hybrid computation and Integrated circuit validation.