Complex & Intelligent Systems (2023) 9:1809–1821
https://doi.org/10.1007/s40747-022-00884-6

ORIGINAL ARTICLE

A collaborative neurodynamic optimization algorithm to traveling salesman problem

Jing Zhong[1] · Yuelei Feng[1] · Shuyu Tang[1] · Jiang Xiong[1] · Xiangguang Dai[1] · Nian Zhang[2]

Received: 24 May 2022 / Accepted: 20 September 2022 / Published online: 13 October 2022

Abstract

This paper proposed a collaborative neurodynamic optimization (CNO) method to solve traveling salesman problem (TSP). First, we construct a Hopfield neural network (HNN) with $n \times n$ neurons for the n cities. Second, to ensure the convergence of continuous HNN (CHNN), we reformulate TSP to satisfy the convergence condition of CHNN and solve TSP by CHNN. Finally, a population of CHNNs is used to search for local optimal solutions of TSP and the globally optimal solution is obtained using particle swarm optimization. Experimental results show the effectiveness of the CNO approach for solving TSP.

Keywords Combinatorial optimization problems · Collaborative neurodynamic optimization · Hopfield neural network · Traveling salesman problem

Introduction

The traveling salesman problem (TSP) is to find a route to travel each city once and return to the starting city. The best route is a feasible route of a minimum total distance of a given city list. TSP can be regarded as a classic combinatorial optimization problem. The related optimization theory can also be used to some similar problems including the quadratic assignment problem and the scheduling problem [1]. It is well known that TSP is a NP-hard optimization problem, which was discussed and studied by extensive researchers [2–5]. Some classic optimization methods, including Nearest Neighborhood Search, Simulated Annealing, and Genetic Algorithm, were proposed to solve TSP.

In the past decade, some optimization theory based on neural network was emerged to solve optimization problems. Hopfield [6] first used the networks of several neurons as a powerful computational model to solve the complexity problem. In the seminal paper of Hopfield, two types of Hopfield neural network models (i.e., the continuous HNN and the discrete HNN) were proposed. The two neural network modes were used to solve linear programming problems and combinatorial optimization problems [7–9]. After that, numerous neural network models were developed to solve various optimization problems, including linear and nonlinear programming [7,10–13], generalized convex optimization problems (e.g., [14,15]), minimax optimization problems (e.g., [16]), distributed optimization problems (e.g., [17]), and combinatorial optimization (e.g., [18]).

Because of the computational complexity of TSP, the above-mentioned neural network methods fall into a local solution easily. Recently, collaborative neurodynamic opti-

Jing Zhong, Yuelei Feng, Shuyu Tang, Jiang Xiong, Xiangguang Dai, and Nian Zhang have contributed equally to this work.

✉ Jiang Xiong
xiongjiang@sanxiau.edu.cn

Jing Zhong
zjing@sanxiau.edu.cn

Yuelei Feng
120211609@stumail.sanxiau.edu.cn

Shuyu Tang
120201623@stumail.sanxiau.edu.cn

Xiangguang Dai
daixiangguang@sanxiau.edu.cn

Nian Zhang
nzhang@udc.edu

1 Key Laboratory of Intelligent Information Processing and Control of Chongqing Municipal Institutions of Higher Education, Chongqing Three Gorges University, Bai'an Dam, WanZhou, Chongqing 404120, China

2 Department of Electrical and Computer Engineering, University of the District of Columbia, Washington, DC 20008, USA

mization (CNO) approaches are very popular for solving the combinatorial optimization problems [19–21]. Compared with traditional neural networks, CNO can search the global solution of a given problem. In the CNO, several neurodynamic models in a parallel mode are used to search the local solutions of the optimization problem and the searching process are repeated by the initialization of initial states until the global solution is achieved. Theory and experiments were presented to prove the convergence of CNO approaches and the effectiveness in searching the global optima of combinatorial optimization problems [22].

In this paper, the CNO method is proposed for solving TSP based on continuous Hopfield networks (CHNs). First, we reformulate the TSP into a quadratic unconstrained binary optimization (QUBO) problem [23] by converting the penalty functions into equality constraints. Second, we propose a population of CHNs to search the local solution of TSP. Third, we reinitialize the initial states of each CHN by employing Particle Swarm Optimization (PSO) and repeat the step 2 until the global solution of TSP is achieved. Our achievements of this paper are

- Combining CHNs and PSO, this paper proposed a CNO algorithm to search the global solution of the TSP.
- Experimental results of four benchmark datasets are presented to demonstrate the superior performance of the CNO approach than the existing TSP algorithm based on CHNs.

Related works

Continuous Hopfield network

Continuous Hopfield network (CHN) is a archetypal feedback network, where all neurons are both inputs and outputs in the CHN. Suppose that there are n neurons in the CHN and each neuron connects with each other. The states of all the neurons can be denoted by $u = [u_1, \dots, u_n]$. CHN can update all the neurons synchronously by the following form:

$$\begin{aligned} &u(t+1) = u(t) + \frac{\mathrm{d}u}{\mathrm{d}t} \triangle t, \\ &v = g(u), \end{aligned} \tag{1}$$

where $\triangle t$ and $v \in \{0, 1\}^n$ denote a constant and a state vector, respectively. $\frac{\mathrm{d}u}{\mathrm{d}t}$ is decided by the following equation:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{u}{\tau} + Tv + I, \tag{2}$$

where $T \in R^{n\times n}$ and $I \in R^{n\times 1}$ denote a symmetric matrix and a bias matrix, respectively. The $g(u_i)$ of Eq. (1) is expressed as follows:

$$v_i = g(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{u_0}\right)\right), \quad u_0 > 0,\ i = 1, 2 \dots n, \tag{3}$$

where u_0 is a positive constant. To satisfy the convergence property of CHN synchronous, two conditions should be satisfied. First, any neuron should not exist a self feedback. Second, the connecting weight between neurons T_{ij} and T_{ji} should be the same.

In general, the energy function [24] of CHN is described by

$$E = -\frac{1}{2} v^t T v - (i^b)^t v. \tag{4}$$

For Eq. (3), there are two updating modes (i.e., asynchronous or synchronous). The asynchronous mode means that each neuron v_i can be updated sequentially. The synchronous mode can update all the neurons simultaneously. The two update modes have been extensively studied in [6,25–27]. In this paper, we use the synchronous mode. The T of Eq. (4) should satisfy the following two conditions: (1) the values of the diagonal elements should be zeros; (2) T should be symmetric.

The initial value of v are initialized randomly. Therefore, the CHN can achieve different local optimal solutions by different initial values. In other words, CHN cannot search for a global optimal solution. In the following subsection, we introduce Particle Swarm Optimization to search for a global optimal solution.

Particle swarm optimization

Particle swarm optimization (PSO) is a popular metaheuristic optimization algorithm [28–34], which is often used to solve NP-hard problems. PSO is first proposed by Kennedy and Eberhart [35], which simulates the bird flock searching for food. PSO provides a searching procedure by a population of individuals. Each individual called the particle can change its position (state) with time. While searching a multidimensional space, each particle re-adjusts its position (state) by a new velocity which is computed by its own and its neighboring's flying experience.

Suppose that x and v denote a particle position (state) and its velocity in a searching space, respectively. $x_i = (x_{i1}, x_{i2}, \dots x_{ij})$ represents the ith particle in the d-dimensional space. $\mathrm{pbest}_{ij} = (\mathrm{pbest}_{i1}, \mathrm{pbest}_{i2}, \dots, \mathrm{pbest}_{ij})$ denotes the best previous position of the ith particle. gbest is the global optimal position searched by all particles in the group. $v_{ij} = (v_{i1}, v_{i2}, \dots v_{ij})$ represents the velocity of the ith particle. The velocity and position of the particle are calculated in terms of the following formula:

$$v_{ij}^t = w v_{ij}^{t-1} + c_1 r_1 (\mathrm{pbest}_{ij} - x_{ij}^{(t-1)})$$

$$+c_2 r_2(\text{gbest}_{ij} - x_{ij}^{(t-1)}), \tag{5}$$

$$x_{ij}^t = x_{ij}^{t-1} + x_{ij}^t, \tag{6}$$

where c_1 and c_2 denote the acceleration speeds, r_1 and r_2 denote random numbers in [0, 1], and w is a positive constant called the inertia weight. However, this mode cannot be used to optimize discrete variable problems [36]. Kennedy and Eberhart [37] proposed another PSO algorithm to address this problem. The updated velocity of the particle x_{id} is expressed as follows:

$$v_{id} = v_{id} + \varphi(\text{pbest}_{id} - x_{id}) + \varphi(\text{gbest}_{id} - x_{id}), \tag{7}$$

$$x_{id} = \begin{cases} 1, & \text{if rand}() < \text{sig}(v_{id}) \\ 0, & \text{otherwise}, \end{cases} \tag{8}$$

where the definition of x_{id}, v_{id}, pbest and gbest are given in the beginning. According to Eq. (8), x_{id}, pbest_{id} and gbest_{id} can be normalized to 0 or 1. $\text{sig}(v_{id})$ is a transformation limiting function, which constrains x_{id} to [0, 1], by $\frac{1}{1+\exp(-v_{id})}$. rand$(\cdot)$ can generate random numbers in [0, 1].

Problem formulations

Problem formulation

In [38], the energy function form of TSP is

$$\begin{aligned} E(v) = {} & \frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{j\neq i}^{n} v_{xi}v_{xj} + \frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n}\sum_{y\neq x}^{n} v_{xi}v_{yj} \\ & + \frac{C}{2}\left(\sum_{x=1}^{n}\sum_{i=1}^{n} - n\right)^2 \\ & + \frac{D}{2}\sum_{x=1}^{n}\sum_{y\neq x}^{n}\sum_{i=1}^{n} d_{xy}v_{xi}(v_{y,i+1} + v_{y,i-1}), \end{aligned} \tag{9}$$

where A, B, C and D are positive constants. The first three terms of Eq. (9) are constraints, and the last term is the objective function. Some explanations of Eq. (9) are given as follows:

- For the first term, each row has exactly one 1 or the values of each row are all zeros.
- For the second term, each column has only one 1 or the values of each column are all zeros.
- For the third term, the matrix v should has 1 for n times. Therefore, each row or each column appears only one once.

Table 1 The permutation matrix v

Sequence city name	1	2	3	4	5
cn_A	0	1	0	0	0
cn_B	0	0	0	1	0
cn_C	1	0	0	0	0
cn_D	0	0	0	0	1
cn_E	0	0	1	0	0

- The last item depicts the total path that may be taken through these cities. According to the first three constraints, only one path is a local optimum solution.

Note: d_{xy} denote the distance between city x and city y, and v_{xi} denote whether the city x is passed. v_{yi} denote whether the city y is passed.

TSP can be mapped into the state vector of the neural network and expressed by a permutation matrix. Suppose that n cities are needed to visit. Each row and column must has one 1 once, and the rests are zeros. A local optimum solution of TSP can be expressed by a permutation matrix in Table 1.

In Table 1, cn_A, cn_B, cn_C, cn_D, and cn_E denote different city name; the sequences 1, 2, 3, 4, and 5 denote the path sequence. The permutation matrix v concludes that the salesman visits cn_C → cn_A → cn_E → cn_B → cn_D → cn_C, successively.

Problem reformulation

To simplify the form of Eq. (9), Sun and Zheng [39] make some improvements. Next, Eq. (9) can be rewritten as follows:

$$\min \sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{i=1}^{n} v_{xi}d_{xy}v_{y,i+1} \tag{10a}$$

$$\text{s.t.} \sum_{x=1}^{n} v_{xi} = 1, \quad x = 1,\ldots,n, \tag{10b}$$

$$\sum_{i=1}^{n} v_{xi} = 1, \quad i = 1,\ldots,n, \tag{10c}$$

$$v_{xi} \in \{0,1\}, \quad i, j = 1,\ldots n, \tag{10d}$$

where d_{xy} is the distance of cities x and y, n is the number of cities, and $v_{xi} = 1$ denotes that the city x is visited in the ith time.

Equation (10a) is the total distance of an effective path, and the constraints in (10b) and (10c) denote that a salesman enters and leaves a city only once. The Euclidean distance is used to measure the distance of cities x and y, where d_{xy} is symmetric.

Equation (10d) can be rewritten by the Lagrange multiplier method as follows:

$$E(v) = \frac{D}{2}\sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{i=1}^{n} v_{xi} d_{xy} v_{y,i+1} + \frac{A}{2}\sum_{x=1}^{n}\left(\sum_{i=1}^{n} v_{xi} - 1\right)^2 + \frac{A}{2}\sum_{i=1}^{n}\left(\sum_{x=1}^{n} v_{xi} - 1\right)^2, \tag{11}$$

where A and D are positive penalty parameters.

The partial derivative of Eq. (11) is expressed as follows:

$$\frac{\mathrm{d}U_{xi}}{\mathrm{d}t} = -\frac{\mathrm{d}E}{v_{xi}} = -D\sum_{x=1}^{n}\sum_{y=1}^{n}\sum_{i=1}^{n} d_{xy} v_{y,i+1} - A\left(\sum_{x=1}^{n} v_{xi} - 1\right) - A\left(\sum_{i=1}^{n} v_{xi} - 1\right). \tag{12}$$

Algorithmic design

The solution of TSP is based on the CHN and PSO, and the details of procedure are described as follows: (1) Initialize the population (i.e., given multiple initial solutions of the Hopfield neural network); (2) CHN is used to optimize Eq. (11) using Eqs. (1), (3), and (12) and obtain several feasible solutions; (3) these feasible solutions are reset to the initial solutions using PSO; (4) the above steps are repeated until the global optimal solution is obtained or the maximum number of iterations is reached.

Note: pop denotes the population size, and $\ln(\cdot)$ denote the logarithmic function. p denote the number of the class. $\triangle t$, A, D, and u_0 are the parameters of CHN.

Algorithm 1 describes our proposed algorithm in detail.

Algorithm 1 CHN_PSO

Require: Sample matrix $X^{n\times m}$, initial states $v^{n\times p\times pop} \in \{0, 1\}$, population size N, termination criterion M, $c1$, $c2$, random number matrix $u^{n\times p}$, $\triangle t$, A, D, $iter$ and u_0
Require: Initialize states by $[v^{(1)}(0), \ldots, v^{(N)}(0)] = u_0 \times ln(n-1) + u$, $v \in \{0, 1\}^{(np+p)\times N}$;
Require: Initialize the pbest by $[v^{(1)*}, \ldots, v^{(N)*}] \leftarrow [v^{(1)}(0), \ldots, v^{(N)}(0)]$;
Require: Initialize the gbest by $v^{0*} \leftarrow \arg\min[f_p(v^{(1)}(0)), \ldots, f_p(v^{(N)}(0))]$
Require: Initialize $init_v^{(i)} \in R^{(np+p)\times N}$ to zeros, $i = 1, \ldots, N$; $k \leftarrow 1$;
Ensure: v^*;
while $k < M$ **do**
  while $i < N$ **do**
    while $j = 1 < iter$ **do**
      Update neuronal states of each batch v by Eqs. (1), (3) and (12).
    end while
    $\hat{v}^{(i)} \leftarrow$ all neuronal states $v^{(i)}$;
    if then $f_p(\hat{v}^{(i)}) < f_P(v^{(i)^*})$
      $v^{(i)*} \leftarrow \hat{v}^{(i)}$
    end if
  end while
  $v^{k*} \leftarrow \arg\min\left\{f_p(v^{(1)*}), \ldots, f_p(v^{(N)*})\right\}$;
  if then $f_p(v^{(k-1)*}) = f_p(v^{k*})$
    $k = k + 1$;
  else[$k \leftarrow 1$;]
  end if
  Generate r_1,$r_2 \in R^{(np+p)\times N}$ randomly in [0, 1];
  while do $i = 1$ to N
    Update $v^{(i)}$ according to Eqs. (7) and (8);
  end while
end while
$v^* \leftarrow v^{k*}$;
return v^*.

Experiment

Experiment set

In the paper, our proposed CHN_PSO approach is used to measure performance on the att48, ulysses16, ulysses22, and burma14. The parameters of our algorithm refer to Table 2. A, D, u_0, iter, and $\triangle t$ are the parameters of CHN; N, M, c_1, and c_2 denote the parameters of PSO.

Experiments are performed on windows 10 with Intel(R) Core(TM) i5-1035G1 CPU @ 1.00 GHz 1.19 GHz and MATLAB 2018a.

Where two parameters N (i.e., pop) and M (i.e., termination criteria) in Table 2 are obtained based on experience.

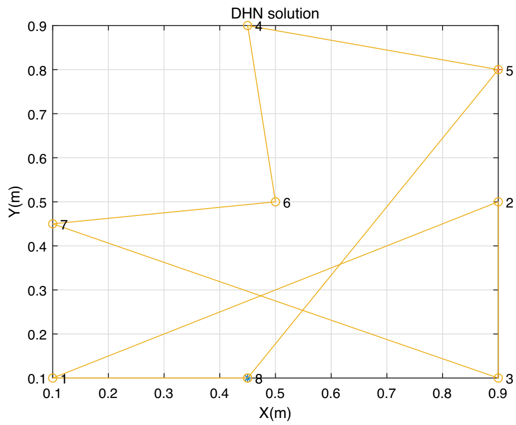
Note: DHN denotes Discrete Hopfield Network, CHN denotes Continuous Hopfield Network, and CHN_PSO denotes our proposed algorithm.

Numerical experiment

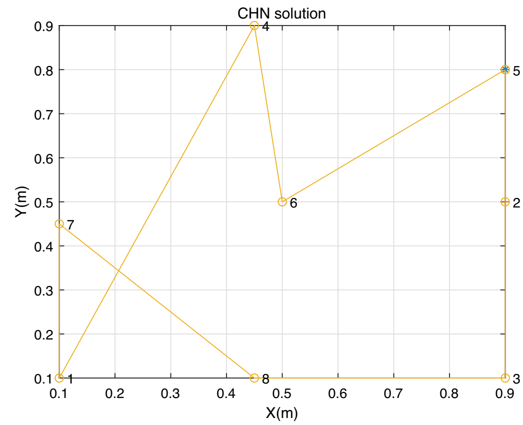
Let: $M = 500$, $N = 32$, $A = 2$, $D = 1$, $u_0 = 0.025$, $\triangle t = 0.002$ and the number of cities is 8. Figure 1 shows the

Table 2 The permutation matrix

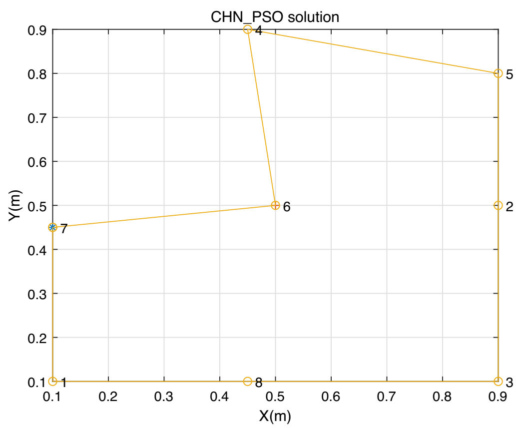
Datasets	A	D	u_0	$\triangle t$	Iter	N	M	c_1	c_2
ulysses16	15	0.2	0.02	0.0007	5000	96	500	2	2
ulysses22	500	0.01	0.02	0.00003	5000	96	500	2	2
burma14	10	0.01	0.02	0.0002	5000	96	500	2	2
att48	180	0.001	0.0025	0.00002	5000	32	500	2	2



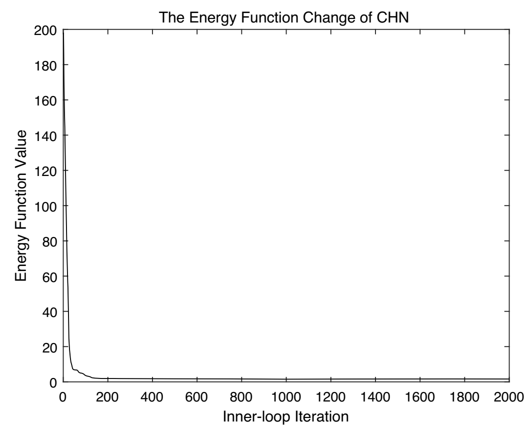
(a) The path of DHN



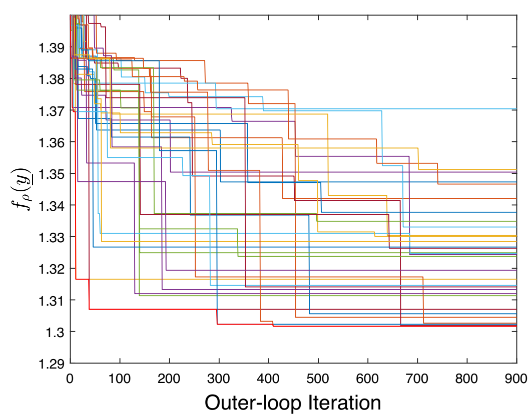
(b) The path of CHN



(c) The path of CHN_PSO



(d) The inner-convergence of CHN_PSO



(e) The outer-convergence of CHN_PSO

Fig. 1 The numerical experiment of the DHN, CHN, and CHN_PSO algorithm

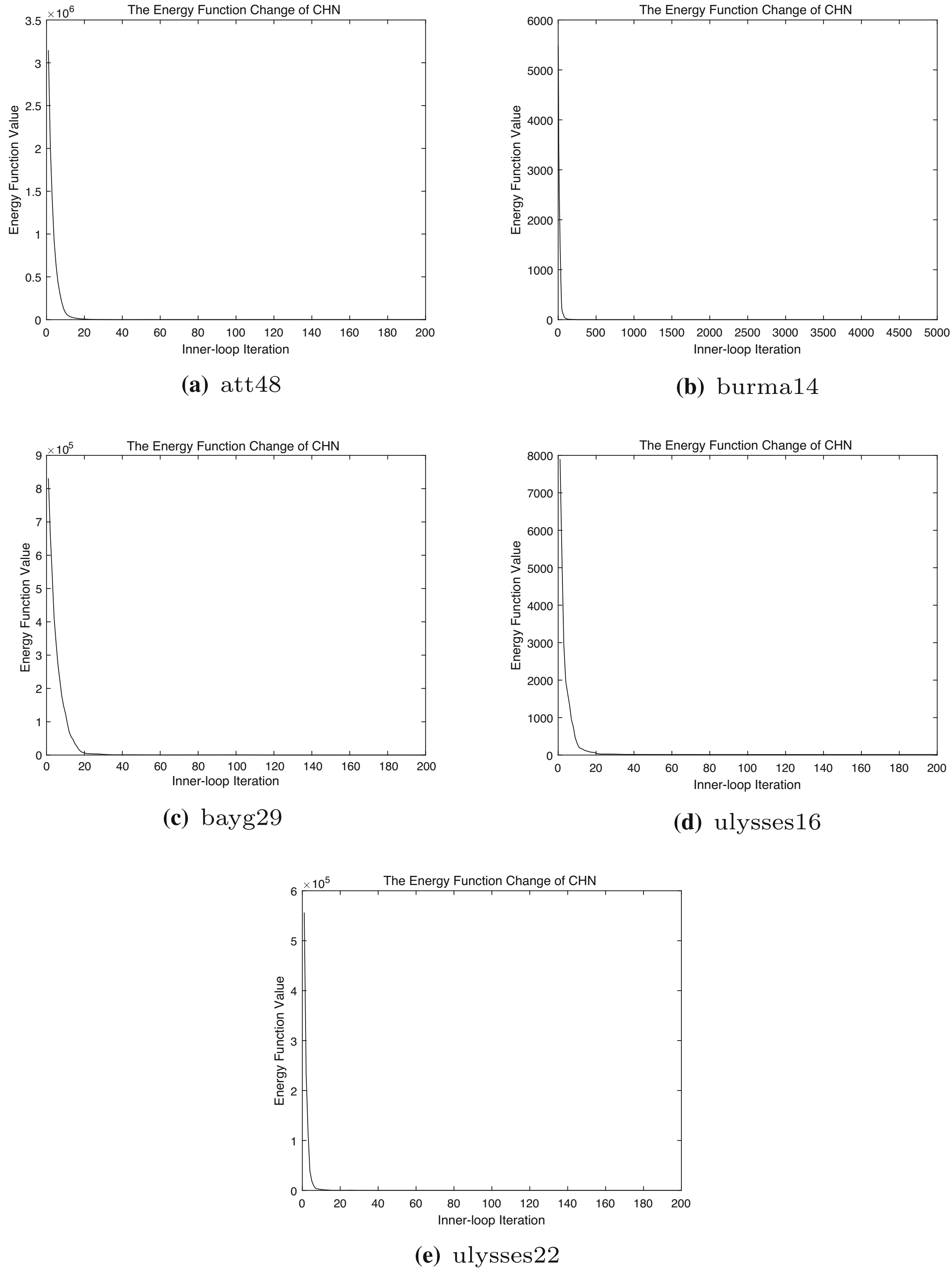
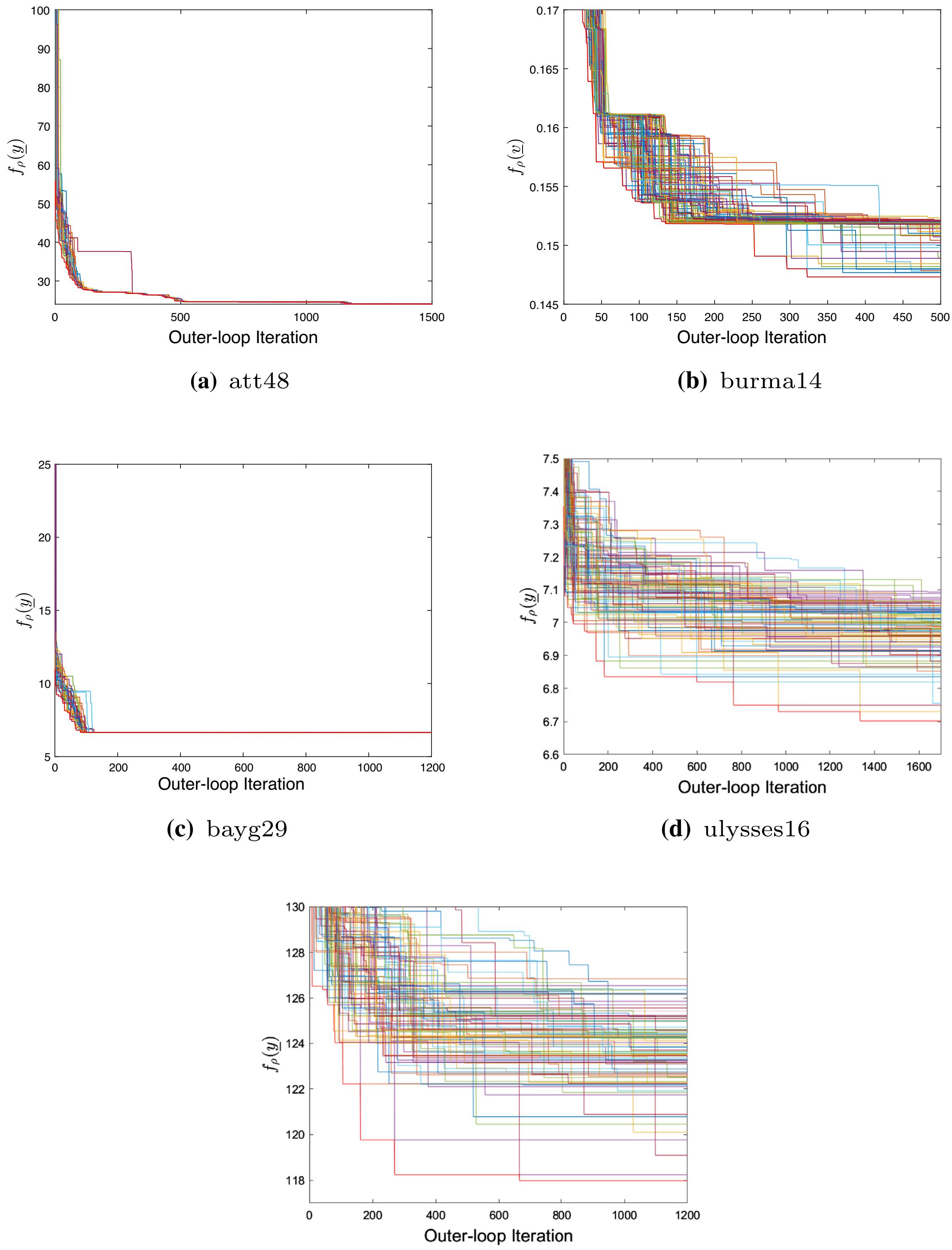


Fig. 2 The convergent behaviors (inner loop) of the CHN_PSO algorithm on the four datasets



(a) att48

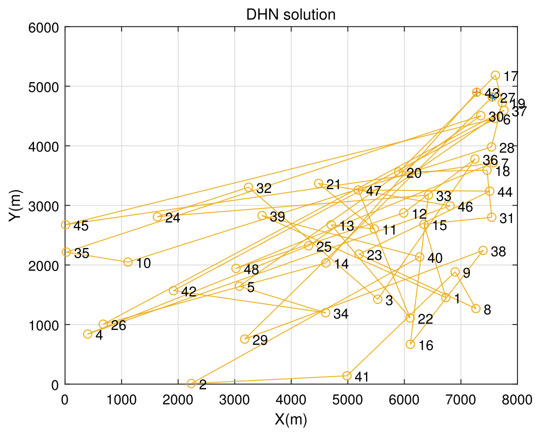
(b) burma14

(c) bayg29

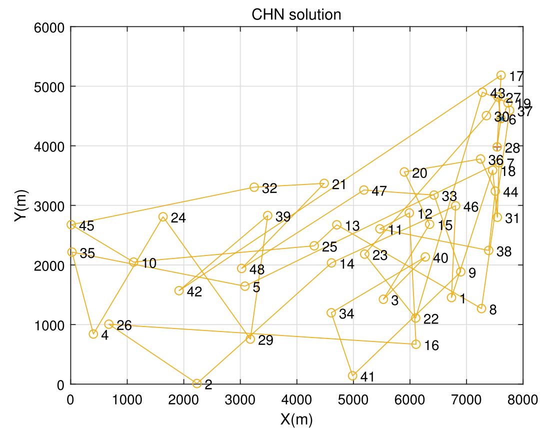
(d) ulysses16

(e) ulysses22

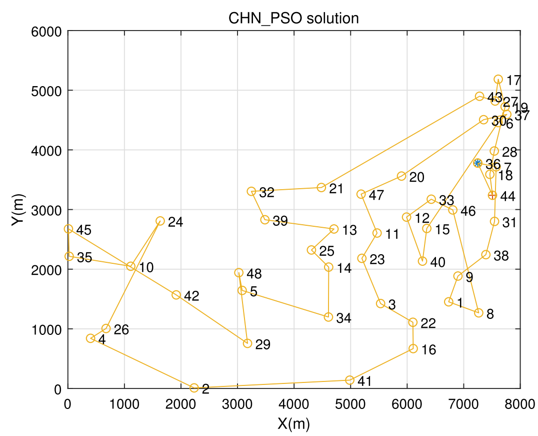
Fig. 3 The convergent behaviors (outer loop) of the CHN_PSO algorithm on the four datasets



(a) The path of DHN



(b) The path of CHN



(c) The path of CHN_PSO

Fig. 4 The optimization path of the PSO_CHN, CHN, and DHN algorithms on att48

numerical experiments conducted on eight randomly generated cities. To vividly show experimental results, we plot the experimental results in Fig. 1a–c. Figure 1d, e depicts the convergent value of the objective function CHN_PSO in the inner loop and the outer loop, respectively. Figure 1a–c shows the paths of DHN, CHN, and CHN_PSO algorithm in eight cities, respectively.

Real datasets' experiment

Datasets

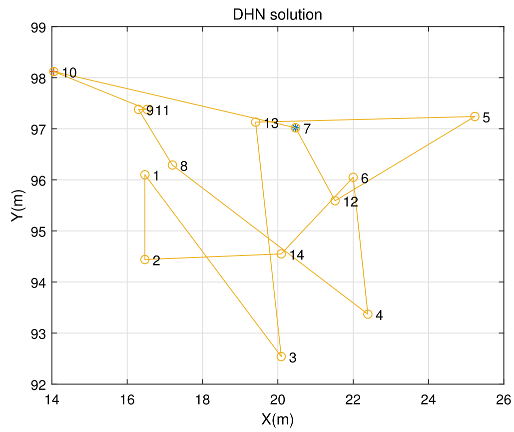
The att48, burma14, and bayg29 datasets contain 48, 14, and 29 instances, respectively. Each data set contains three columns of data, namely, serial number, abscissa, and ordinate. The ulysses16 and ulysses22 datasets contain 16 and 22 instances, respectively. Each dataset contains two columns of data, namely, abscissa and ordinate.

Convergence study

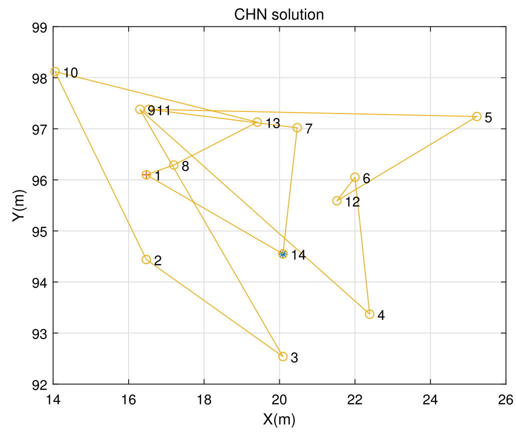
Figure 2 depicts the convergent behaviors of the objective function computed with CHN in the inner loop of our algorithm on datasets att48, burma14, bayg29, ulysses16, and ulysses22. Figure 3 depicts the convergent behaviors of the outer loop of our algorithm on datasets att48, burma14, bayg29, ulysses16, and ulysses22. These experiments show that outer loop iterations are less than inner loop iterations to reach function convergence.

Experiment results

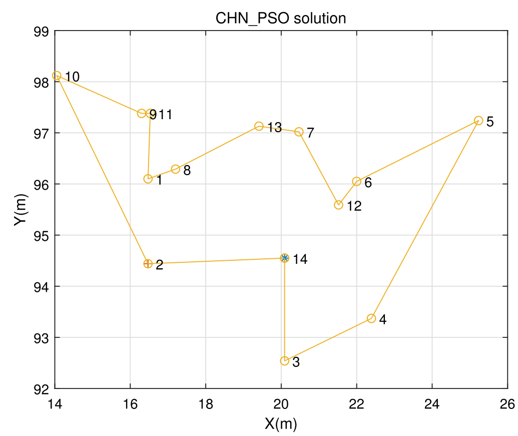
Figures 4, 5, 6, 7 and 8 show the algorithm performance of our method compared to CHN on att48, burma14, ulysses16, and ulysses22. The experiment results in the figure demonstrate that our proposed algorithm statistically outperforms the CHN and DHN algorithms in light of the given exper-



(a) The path of DHN



(b) The path of CHN



(c) The path of CHN_PSO

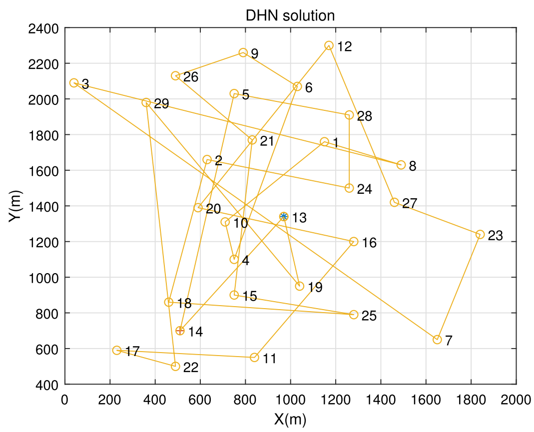
Fig. 5 The optimization path of the PSO_CHN, CHN, and DHN algorithms on burma14

iment result. According to the Figs. 4, 5, 6, 7 and 8, we summarize as follows:

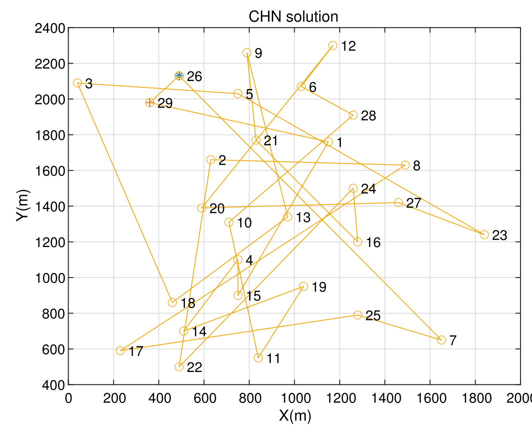
- The final optimization path of our method outperforms CHN and DHN algorithms on the att48, burma14, and ulysses16.
- For ulysses22, the final optimization path of the CHN and DHN algorithms is close to our algorithm, but they still perform unsatisfactorily.

Conclusion

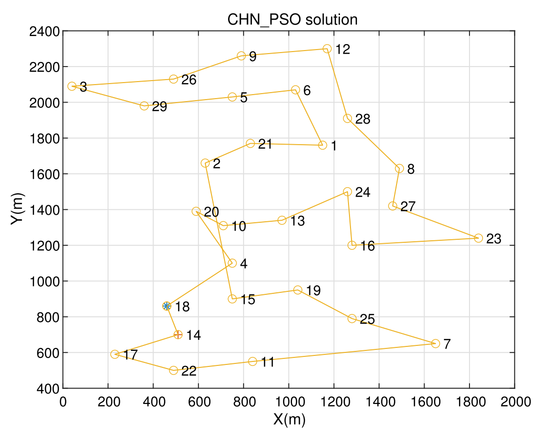
In this paper, a collaborative neurodynamic optimization (CNO) is proposed to solve the traveling salesman problem (TSP). The PSO and HNN are employed in the proposed algorithm. They are used to reach satisfactory results. Experimental results show the effectiveness of the CNO approach for solving four TSP benchmarks.



(a) The path of DHN



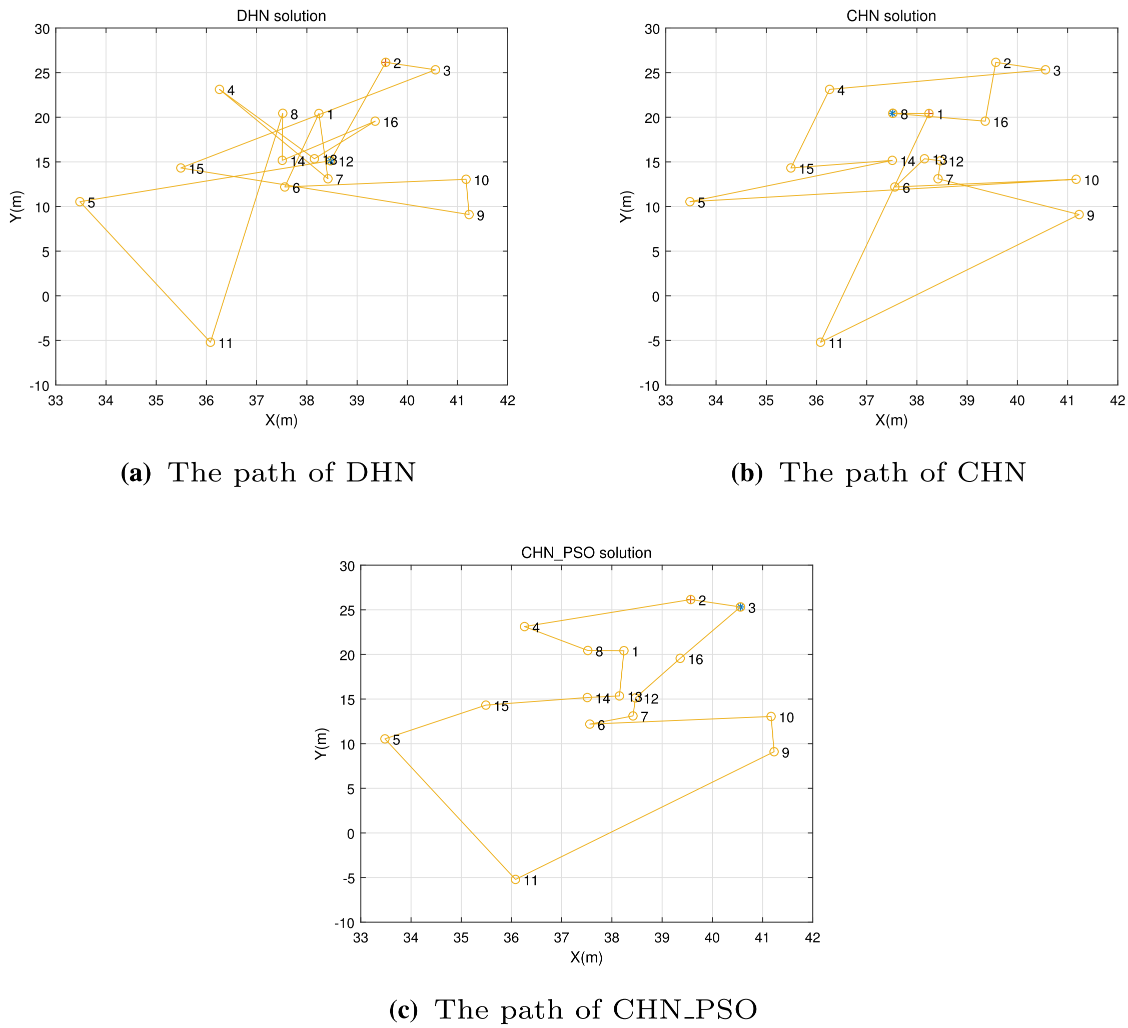
(b) The path of CHN



(c) The path of CHN_PSO

Fig. 6 The optimization path of the PSO_CHN, CHN, and DHN algorithms on bayg29

This paper use CHN and PSO to solve the TSP problem. In the future work, the discrete Hopfield networks can be use to solve this problem and combine with others Swarm intelligence algorithm. We are studying how to effectively and efficiently combine them at present.

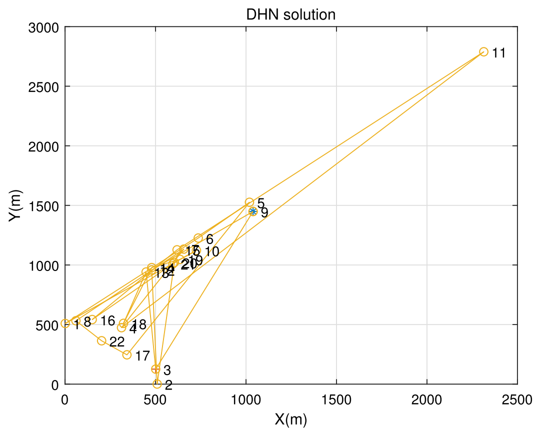


(a) The path of DHN

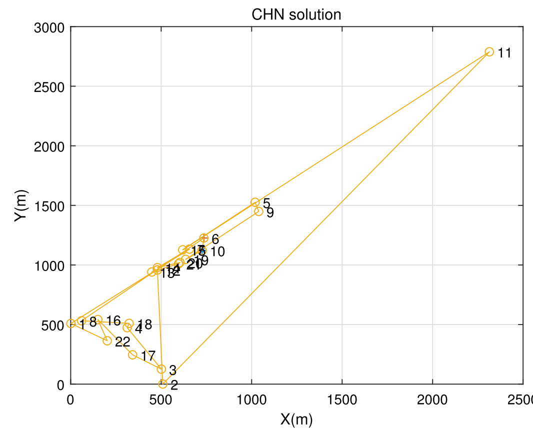
(b) The path of CHN

(c) The path of CHN_PSO

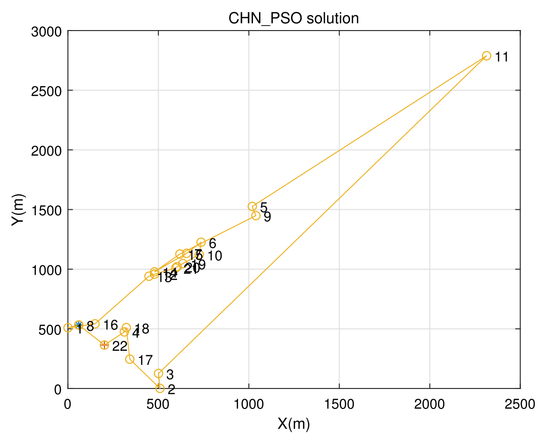
Fig. 7 The optimization path of the PSO_CHN, CHN, and DHN algorithms on ulysses16



(a) The path of DHN



(b) The path of CHN



(c) The path of CHN_PSO

Fig. 8 The optimization path of the PSO_CHN, CHN, and DHN algorithms on ulysses22

Acknowledgements This work is supported by the Chongqing Smart Ecotourism Science Group Open Fund , Natural Science Foundation of Chongqing (Grant No. cstc2018jcyjAX0502), Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant No. KJQN202001222) , National Natural Science Foundation of China (Grant No. 61602072), National Science Foundation (NSF) (2011927), and DoD (W911NF1810475).

References

1. Wang J (1990) A deterministic connectionist machine for the traveling salesman problem. IEEE International conference on systems, man, and cybernetics conference proceedings. IEEE, pp 374–375
2. Lawler EL, Lenstra JK, Rinnooy Kan AH, Shmoys DB (1986) Erratum: The traveling salesman problem: a guided tour of combinatorial optimization. J Oper Res Soc 37(6):655
3. Mulder SA, Wunsch DC II (2003) Million city traveling salesman problem solution by divide and conquer clustering with adaptive resonance neural networks. Neural Netw 16(5–6):827–832
4. Zhao K, Liu S, Rong Y, Yu JX (2021) Towards feature-free tsp solver selection: a deep learning approach In: 2021 InternationalJoint Conference on Neural Networks (IJCNN), IEEE, p 1–8
5. Berger A, Kozma L, Mnich et al (2020) Time- and space-optimal algorithm for the many-visits tsp. ACM Trans Algorithms 16(3):1–22
6. Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci 79(8):2554–2558
7. Tank D, Hopfield J (1986) Simple 'neural' optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit. IEEE Trans Circuits Syst 33(5):533–541

8. Hopfield JJ, Tank DW (1985) “Neural” computation of decisions in optimization problems. Biol Cybern 52(3):141–152
9. Talaván PM, Yáñez J (2002) Parameter setting of the Hopfield network applied to TSP. Neural Netw 15(3):363–373
10. Xia Y, Wang J (1998) A general methodology for designing globally convergent optimization neural networks. IEEE Trans Neural Netw 9(6):1331–1343
11. Liu S, Wang J (2006) A simplified dual neural network for quadratic programming with its KWTA application. IEEE Trans Neural Netw 17(6):1500–1510
12. Xia Y, Feng G, Wang J (2008) A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints. IEEE Trans Neural Netw 19(8):1340–1353
13. Hu X, Wang J (2008) An improved dual neural network for solving a class of quadratic programming problems and its k-winners-take-all application. IEEE Trans Neural Netw 19(12):2022–2031
14. Hu X, Wang J (2006) Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network. IEEE Trans Neural Netw 17(6):1487–1499
15. Guo Z, Liu Q, Wang J (2011) A one-layer recurrent neural network for pseudoconvex optimization subject to linear equality constraints. IEEE Trans Neural Netw 22(12):1892–1900
16. Liu Q, Wang J (2015) A projection neural network for constrained quadratic minimax optimization. IEEE Trans Neural Netw Learn Syst 26(11):2891–2900
17. Liu Q, Yang S, Wang J (2016) A collective neurodynamic approach to distributed constrained optimization. IEEE Trans Neural Netw Learn Syst 28(8):1747–1758
18. Peterson C (1990) Parallel distributed approaches to combinatorial optimization: benchmark studies on traveling salesman problem. Neural Comput 2(3):261–269
19. Yan Z, Fan J, Wang J (2016) A collective neurodynamic approach to constrained global optimization. IEEE Trans Neural Netw Learn Syst 28(5):1206–1215
20. Li X, Wang J, Kwong S (2020) Alternative mutation operators in collaborative neurodynamic optimization. In: 2020 10th International conference on information science and technology (ICIST), IEEE, pp 126–133
21. Leung MF, Wang J (2020) Minimax and biobjective portfolio selection based on collaborative neurodynamic optimization. IEEE Trans Neural Netw Learn Syst 32(7):2825–2836
22. Yan Z, Wang J, Li G (2014) A collective neurodynamic optimization approach to bound-constrained nonconvex optimization. Neural Netw 55:20–29
23. Glover F, Kochenberger G, Du Y (2019) Quantum bridge analytics I: a tutorial on formulating and using QUBO models. 4OR 17(4):335–371
24. Talavan PM, Yanez J (2005) A continuous Hopfield network equilibrium points algorithm. Comput Oper Res 32(8):2179–2196
25. Bruck J, Goodman JW (1988) A generalized convergence theorem for neural networks. IEEE Trans Inf Theory 34(5):1089–1092
26. Cottrell M (1988) Stability and attractivity in associative memory networks. Biol Cybern 58(2):129–139
27. Dasgupta S, Ghosh A, Cuykendall R (1989) Convergence in neural memories. IEEE Trans Inf Theory 35(5):1069–1072
28. Van Laarhoven PJ, Aarts EH (1987) Simulated annealing Simulated annealing: theory and applications. Springer, Dordrecht, pp 7–15
29. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
30. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
31. Ruiz R, Stützle T (2007) A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur J Oper Res 177(3):2033–2049
32. Price KV (2013) Differential evolution. Handbook of optimization. Springer, Berlin, pp 187–214
33. Wu X, Han J, Cui Q, Chen L, Liang Y, Huang H, Lee HP, Zhou Y, Wu C (2021) Surprisingly popular algorithm-based adaptive Euclidean distance topology learning PSO
34. Zhu SP, Keshtegar B, Seghier MEAB et al (2022) Hybrid and enhanced PSO: novel first order reliability method-based hybrid intelligent approaches. Comput Methods Appl. Proceedings of ICNN’95-international conference on neural networks. IEEE Mech Eng 393:114730
35. Kennedy J, Eberhart R (1995) Particle swarm optimization 4:1942–1948. IEEE
36. Dai X, Wang J, Zhang W (2022) Balanced clustering based on collaborative neurodynamic optimization. Knowl Based Syst 250:109026
37. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. International conference on systems, man, and cybernetics. Computational cybernetics and simulation. IEEE 5:4104–4108
38. Hopfield JJ, Tank DW (1985) “Neural” computation of decisions in optimization problems. Biol Cybern 52(3):141–152
39. Sun S, Zheng J (1995) An improved algorithm and theoretical proof of Hopfield network for solving TSP. J Electron 23(1):73–78