11th International Conference on Intelligent Control and Information Processing
Dali, China, December 3-7, 2021

2021 11th International Conference on Intelligent Control and Information Processing (ICICIP) | 978-1-6654-2515-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICICIP53388.2021.9642212

A Collaborative Neurodynamic Optimization Algorithm Based on Boltzmann Machines for Solving the Traveling Salesman Problem

Hongzong Li
Department of Computer Science
City University of Hong Kong
Kowloon, Hong Kong
hongzli2-c@my.cityu.edu.hk

Jun Wang
Department of Computer Science
& School of Data Science
City University of Hong Kong
Kowloon, Hong Kong
jwang.cs@cityu.edu.hk

***Abstract*—The traveling salesman problem is known to be NP-hard and has numerous areas of applications. This paper proposes a collaborative neurodynamic optimization algorithm based on Boltzmann machines for solving the traveling salesman problem. A population of Boltzmann machines is employed for local search, and their initial states are repeatedly reinitialized by using the particle swarm optimization update rule for global repositioning. The efficacy of the proposed collaborative neurodynamic optimization algorithm is substantiated on four traveling salesman problem benchmark instances.**

***Index Terms*—Traveling salesman problem, collaborative neurodynamic optimization, Boltzmann machine.**

I. Introduction

The traveling salesman problem (TSP) is one of the most widely studied combinatorial optimization problems. It is to find a round trip visiting each city once with the minimal total distance. TSP arises in numerous applications, such as job sequencing [1], vehicle routing [2], wallpaper cutting [3], VLSI circuit wire routing [3], and warehouse order-picking [4].

Numerous methods are developed for solving TSP, including exact methods, approximate methods, and heuristic and meta-heuristic methods. Exact methods include branch and bound algorithm [5], branch and cut algorithm [6], etc. Approximate methods include restricted Lagrangean approach [7], etc. Heuristic and meta-heuristic methods include nearest neighbor algorithm [8], insertion method [8], generalized insertion procedure with unstringing and stringing [9], genetic algorithm [10], ant colony optimization [11], hybrid heuristic [12], particle swarm optimization [13], etc.

Since John Hopfield pointed out that the networks of simple and similar neurons collectively can serve as powerful computation models [14], [15], neurodynamic optimization approaches have attracted much attention. Specifically, the discrete and continuous Hopfield networks in [14], [15] are developed for linear programming and combinatorial optimization [16]–[18] including TSP [19], [20]. Numerous recurrent neural networks are developed for solving various optimization problems; e.g., [21]–[27]. Different from these deterministic neurodynamic optimization models, the Boltzmann machine is a stochastic neural network [28] with a local hill-climbing capability for combinatorial optimization [29], [30]. It is used for solving TSP [31], maximum independent problem [29], set partitioning problem [29], max cut problem [30], independent set problem [30], graph coloring problem [30], clique partitioning problem [30], etc. Although a Boltzmann machine is proved to be almost surely convergent to global optima, it entails a sufficient long cooling schedule.

It is easy to get stuck in a local optimum when a single neurodynamic model is applied to solve combinatorial optimization problems with binary variables [32]. To overcome the aforementioned shortcoming, collaborative neurodynamic optimization (CNO) is proposed for solving combinatorial optimization problems [33]. In a framework of CNO, a population of neurodynamic models work individually in parallel for scattered local search until convergence and their neuronal states are repeatedly initialized by using a meta-heuristic rule for global repositioning local search toward global optima. As a paradigm of hybrid intelligence to integrate neurodynamic optimization with swarm intelligence, it is proven to be almost surely convergent to global optima of optimization problems [34]. CNO approaches are developed for global optimization [33]–[36], biconvex optimization [37], multi-objective optimization [38], [39], and combinatorial and mixed-integer optimization [33], [40]. CNO is applied for model predictive control [41], [42], nonnegative matrix factorization [43], multi-vehicle task assignment [44], [45], feature selection [46], portfolio selection [47], sparse Bayesian learning [48], spiking neural network regularization [49], and hash bit selection [50].

In this paper, a CNO algorithm with on Boltzmann machines is proposed for solving the TSP based on a quadratic unconstrained binary optimization (QUBO) problem formulation in [51]. By leveraging the hill-climbing capability of Boltzmann machines and the global search capability of CNO, a CNO

This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region of China under Grant 11202318, and Grant 11202019; and in part by the Laboratory for AI-Powered Financial Technologies.

algorithm is developed based on Boltzmann machines for solving the formulated TSP. In the proposed CNO algorithm, a population of Boltzmann machines is employed for scatter search, and their initial states are re-initialized repeatedly by using a particle swarm optimization update rule for repositioning the global search.

The remaining paper is arranged as follows. Section II provides necessary preliminaries on discrete Hopfield network, Boltzmann machine, particle swarm optimization, and mutation operation. Section III states formulation and reformulation of TSP. Section IV details the proposed CNO-TSP algorithm. Section V discusses experimental results on four benchmark instances. Section VI concludes this paper.

II. Preliminaries

A. Neurodynamic Model

1) Discrete Hopfield Network (DHN): It consists of binary or bipolar valued nodes. its neuronal states are activated in discrete time [14]:

$$x_i(t) = \sigma(u_i(t)) = \begin{cases} 0 & \text{if } u_i(t) \leq 0, \\ 1 & \text{if } u_i(t) > 0, \end{cases} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state vector, and $u \in \mathbb{R}^n$ is the net-input vector. The net-input vector is updated as follows:

$$u(t+1) = Wx(t) - \theta, \tag{2}$$

where $W \in \mathbb{R}^{n \times n}$ is the connection weight matrix, and $\theta \in \mathbb{R}^n$ is the threshold vector.

It is shown that the DHN can stable at an equilibrium $\bar{x}$ (i.e., $\lim_{t\to\infty} x(t) = \bar{x}$) from a initial state, if $W = W^T$, $w_{ii} = 0$, $\forall i$, and the activation is carried out asynchronously [14].

The DHN [14] is globally convergent to a local minimum of the following combinatorial optimization problem:

$$\begin{aligned} \min \quad & -\frac{1}{2}x^T W x + \theta^T x \\ \text{s.t.} \quad & x \in \{0,1\}^n. \end{aligned} \tag{3}$$

The asynchronous activation of the DHN entails a long convergence time. To expedite the convergence of DHNs, several methods are available to activate neuronal states synchronously in batches [52]–[55]. A method for activating neurons in batches (Algorithm 2) is developed for solving the TSP in [51]. It is shown in [55] that the DHN is globally stable if the neurons are activated synchronously in batches, where the neurons in each batch are not directly connected.

2) Boltzmann Machine (BM): It is a well-known stochastic neural network, and it can be viewed as a generalized Hopfield network. It is a fully connected network comprising two states. The neurons are activated simultaneously based on the current state of their neighbors and the corresponding edges. The probability of accepting neural transition is defined as follows [28]:

$$\begin{cases} P(x_i(t) = 1) = \frac{1}{1+e^{-\frac{u(t)}{T}}}, \\ P(x_i(t) = 0) = 1 - P(x_i(t) = 1), \end{cases} \tag{4}$$

where $u(t)$ is defined as same as in equation (2), T is a temperature parameter. The temperature is updated according to exponential multiplicative cooling schedule:

$$T = T_0 \alpha^t,$$

where T_0 is a initial temperature and $\alpha \in [0, 1]$ is a cooling rate parameter.

Similar to DHN, BM neurons can also be activated in batches to expedite the convergence of BM, and its asymptotical convergence has been proved in [56].

B. Particle Swarm Optimization (PSO)

It is a stochastic optimization technique. It was motivated by the intelligent collective behavior of flocks of birds. It is based on a swarm, and each particle in the swarm keeps changing the position according to the experience of its own and the group-best particle.

For each particle, the velocity v_i and the position $x_i(t)$ are updated according to its best position x_i^* and the group-best position x^*:

$$\begin{cases} v_i(t+1) = & c_0 v_i(t) + c_1 r_1 (x_i^* - x_i(t)) \\ & + c_2 r_2 (x^* - x_i(t)), \\ x_i(t+1) = & x_i(t) + v_i(t+1), \end{cases} \tag{5}$$

where $x_i \in \mathbb{R}^n$ denotes the position vector of the i^{th} particle, x_i^* denote the best position found by the i^{th} particle individually, x^* denote the best position known to the swarm (solution set), $c_0 \in [0, 1]$ is an inertia weight, $c_1 \in [0, 1]$ is a cognitive learning factor, $c_2 \in [0, 1]$ is a social learning factor, and $r_1, r_2 \in [0, 1]$ are two random numbers.

C. Mutation Operation

Mutation operation is an effective method to avoid low diversity and trapping in local optima. When the diversity of a swarm is low than a threshold, carrying out a mutation operation can help the swarm intelligence algorithm escape local optima. A diversity measurement can be measured by the following function:

$$\delta(x) = \frac{1}{Nn} \sum_{i=1}^{N} \|x^{(i)} - x^*\|_2, \tag{6}$$

where N and n are respectively the number and dimensions of solutions, $x^{(i)}$ is the i^{th} solution, and x^* are the best solution among the N solutions.

Bit-flip mutation is a common mutation operator for evolutionary algorithms applied to optimization with binary variables and it is defined in [57]:

$$x_j = \begin{cases} \neg x_j & \text{if } \xi_j \leq P_m, \\ x_j & \text{otherwise }, \end{cases} \tag{7}$$

where $\bar{x}_j$ is the negation of x_j, ξ_j is a random number under the uniform distribution between [0, 1], P_m is the preset probability for mutation.

III. Problem Formulations

A. Problem Formulation

The TSP can be formulated as a quadratic assignment problem form [58]:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1,l\neq j}^{n} d_{jl}x_{ij}x_{i+1,l}, \tag{8a}$$

$$\text{s.t.} \sum_{i=1}^{n} x_{ij} = 1,\ j = 1,...,n, \tag{8b}$$

$$\sum_{j=1}^{n} x_{ij} = 1,\ i = 1,...,n, \tag{8c}$$

$$x_{ij} \in \{0,1\},\ i,j = 1,...,n. \tag{8d}$$

where d_{jl} denotes the distance between city j and city l, the distance matrix $D = [d_{jl}]$ is symmetric, The objective function (8a) to be minimized is the total distance of a tour. Constraints in (8b) ensure that exactly one city is visited once and only once. Constraints in (8c) ensure that each stop one and only one city being visited. The constraints in (8d) enforces the solution to be binary.

B. Problem Reformulation

Let $x = [x_{11}, x_{12}, ..., x_{1n}, x_{21}, x_{22}, ..., x_{2n}, ..., x_{nn}]^T \in \{0,1\}^{n^2}$ and the problem (8) can be written in a vector form:

$$\min\ x\hat{D}x, \tag{9a}$$

$$\text{s.t. } Ax = e, \tag{9b}$$

where $\hat{d}_{(i-1)n+j,(k-1)n+l} = d_{jl}$ when $k = i+1,\ i = 1,2,...,n-1$, and $k = 1,\ i = n$, A and e are defined as follows:

$$A = \begin{bmatrix} I_1 & I_2 & \cdots & I_n \\ I & I & \cdots & I \end{bmatrix} \in \{0,1\}^{2n\times n^2},$$
$$e = \{1\}^{2n\times 1}.$$

To handle the constraints (9b), a quadratic penalty function is defined:

$$p(x) = \frac{1}{2}\|Ax - e\|_2^2.$$

Then, a penalized objective function is written:

$$f_\rho(x) = f(x) + \rho p(x),$$

where ρ is a penalty parameter.

The original optimization problem (8) is cast into a QUBO framework via the penalty function:

$$\min\ f_\rho(x),$$
$$\text{s.t. } x \in \{0,1\}^{N^2}. \tag{10}$$

It is known that problems (8) and (10) are equivalent in terms of their optimal solutions if the penalty parameter is sufficiently large [59].

The penalized objective function can be simplified to:

$$\begin{aligned} f_\rho(x) &= x^T\hat{D}x + \frac{\rho}{2}\|Ax - e\|_2^2, \\ &= x^T\hat{D}x + \frac{\rho}{2}((Ax-e)^T(Ax-e)), \\ &= x^T\hat{D}x + \frac{\rho}{2}((x^TA^T - e^T)(Ax-e)), \\ &= x^T\hat{D}x + \frac{\rho}{2}(x^TA^TAx - x^TA^Te - e^TAx + e^Te), \\ &= x^T(\hat{D} + \frac{\rho}{2}A^TA)x - \rho e^TAx + \frac{\rho}{2}e^Te, \\ &= -\frac{1}{2}x^T(-2\hat{D} - \rho A^TA)x - \rho e^TAx + \frac{\rho}{2}e^Te. \end{aligned}$$

Therefore, the parameters of a DHN or BM are obtained:

$$W = -2\hat{D} - \rho A^TA,$$
$$\theta = -\rho e^TA.$$

To guarantee the stability of BM, we have to set $w_{ii} = 0$ according to the aforementioned stability conditions. Since for a binary variable x_i, $x_i^2 = x_i$. Therefore, the diagonal elements of W in the equation (2) can always be set to zeros and add an equivalently linear term $\text{diag}(w_{11}, \ldots, w_{nn})x$. Connection weight matrix and threshold are rewritten as:

$$\hat{W} = W - \text{diag}(W), \tag{11}$$

$$\hat{\theta} = \frac{1}{2}\text{diag}(W) - \rho e^TA. \tag{12}$$

The TSP is cast to the problem (3) with $\hat{W}$ in equation (11) and $\hat{\theta}$ in equation (12).

IV. CNO-TSP/BM Algorithm

In the proposed algorithm, a population of BMs is employed in parallel for scatter search, and the PSO update rule is used for re-initialization of neuronal states. Algorithm 1 details the proposed CNO procedure to TSP (CNO-TSP/BM). The equilibrium states are obtained, and the best solution of the ith BM is updated in steps 2 - 5, where N in step 2 denotes the number of BMs. Similar to the method for activating neurons of a DHN in batches in [51], the neurons of a BM in a batch are activated synchronously in step 3. The group-best solution is updated in steps 6 - 11. The states of BMs are reinitialized with PSO update rule in step 14. The diversity of the swarm is calculated in step 15. A bit-flip mutation is carried out in steps 16 - 18, where ϵ in step 22 is a diversity threshold. The CNO-TSP algorithm with a population of DHNs [51] can be implemented by replacing BM (equation (4)) in step 3 with DHN (equations (2) and (1)).

V. Experimental results

A. Setups

In this section, we evaluate the CNO-TSP algorithms on public available instances BURMA14, ULYSSES16, GR21, and ULYSSES22 in TSPLIB [60].

In the PSO update rule (5), $c_0 = 1$ and $c_1 = c_2 = 0.2$. In the BM, the cooling rate $\alpha = 0.5$. $\epsilon = 0.025$. For

Algorithm 1: CNO-TSP/BM algorithm

Input: Population size N, termination criteria M, initial temperature T_0, cooling rate parameter α, initial states of BMs $[x^{(1)}(0), ..., x^{(N)}(0)] \in \{0,1\}^{n^2 \times N}$, initial velocity matrix $[v^{(1)}, ..., v^{(N)}] \in [-1,1]^{n^2 \times N}$, PSO-based state initialization parameters c_0, c_1 and c_2, diversity threshold ϵ.

Output: x^*.

```
while m ≤ M do
    foreach i to N BMs do
        Each BM is updated in parallel accroding to Eq. (4)) with initial state x^(i)(0), initial temperature T_0, and cooling rate parameter α;
        Update the best position x^(i), and f(x^(i)) found by the i^th BM;
    end
    i* = arg min_i {f(x^(1)), ..., f(x^(i)), ..., f(x^(N))};
    if f(x^(i*)) < f(x*) then
        m ← 0;
        f(x*) ← f(x^(i*));
        x* ← x^(i*);
    else
        m ← m + 1;
    end
    Reposition the initial states of a population of BMs and update velocity by PSO update rule (5);
    Compute δ(x) according to Eq. (6);
    if δ(q) < ε then
        Update x_i(0) for all i according to Eq. (7);
    end
end
return x*.
```

performance comparison, the experimental results of the CNO-TSP algorithm with DHNs [51] (CNO-TSP/DHN) with the same parameters above are tabulated in the next subsection. All codes are implemented under Windows 10 (64-bit), Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz and 64.0GB RAM in MATLAB R2021b 64-bit.

B. Experimental Results

Fig. 1 snapshots the convergent behaviors of a single BM, including neuronal states x, objective function $f(x)$, penalty function $p(x)$ with $\rho = 10^6$ on instances BURMA14, ULYSSES16, GR21, and ULYSSES22. The snapshots show that BMs are stable at an equilibrium, their penalty function decreases to zero, and converges to a feasible solution. Fig. 2 shows the convergent behavior of the CNO-TSP/BM algorithm on instances BURMA14, ULYSSES16, GR21, and ULYSSES22. Fig. 3 illustrates the monte Carlo test results obtained by CNO-TSP algorithms on instances BURMA14, ULYSSES16, GR21, and ULYSSES22. Figs. 4-6 depict the optimal tours obtained by CNO-TSP/BM on instances BURMA14, ULYSSES16, and ULYSSES22, respectively. As coordinates are not provided for GR21, the optimal tour for this benchmark dataset is not plotted. Table I records the hyper-parameters, best/worst values, mean values, and standard deviations achieved by the CNO-TSP/BM algorithm and the CNO-TSP/DHN algorithm [51] on the four benchmark instances. It shows that the proposed CNO-TSP/BM algorithm can find the optimum tour of the four benchmark instances, and outperforms the CNO-TSP/DHN algorithm [51] on the four benchmark instances in terms of best, worst, and mean values of the objective function.

VI. Concluding Remarks

In this paper, the CNO-TSP/BM algorithm is proposed for solving the TSP. In the proposed algorithm, a population of Boltzmann machines is employed. They are activated in batches to expedite their convergence, and their states are re-initialized upon their convergence by using the PSO update rule. The experimental results demonstrate that the proposed CNO-TSP/BM algorithm outperforms the CNO-TSP/DHN algorithm in terms of solution quality and convergence time. Further investigations may aim to integrate unsupervised, supervised, or semi-supervised learning with CNO algorithms to improve their efficiency and scalability.

References

[1] J. Presby and M. Wolfson, "An algorithm for solving job sequencing problems," *Management Science*, vol. 13, no. 8, pp. B–454, 1967.

[2] J. K. Lenstra and A. R. Kan, "Some simple applications of the travelling salesman problem," *Journal of the Operational Research Society*, vol. 26, no. 4, pp. 717–733, 1975.

[3] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[4] R. Matai, S. P. Singh, and M. L. Mittal, "Traveling salesman problem: An overview of applications, formulations, and solution approaches," *Traveling salesman problem, theory and applications*, vol. 1, 2010.

[5] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem," *Operations Research*, vol. 11, no. 6, pp. 972–989, 1963.

[6] M. Fischetti, J. J. Salazar González, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, no. 3, pp. 378–394, 1997.

[7] E. Balas and N. Christofides, "A restricted Lagrangean approach to the traveling salesman problem," *Mathematical Programming*, vol. 21, no. 1, pp. 19–46, 1981.

[8] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II, "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 563–581, 1977.

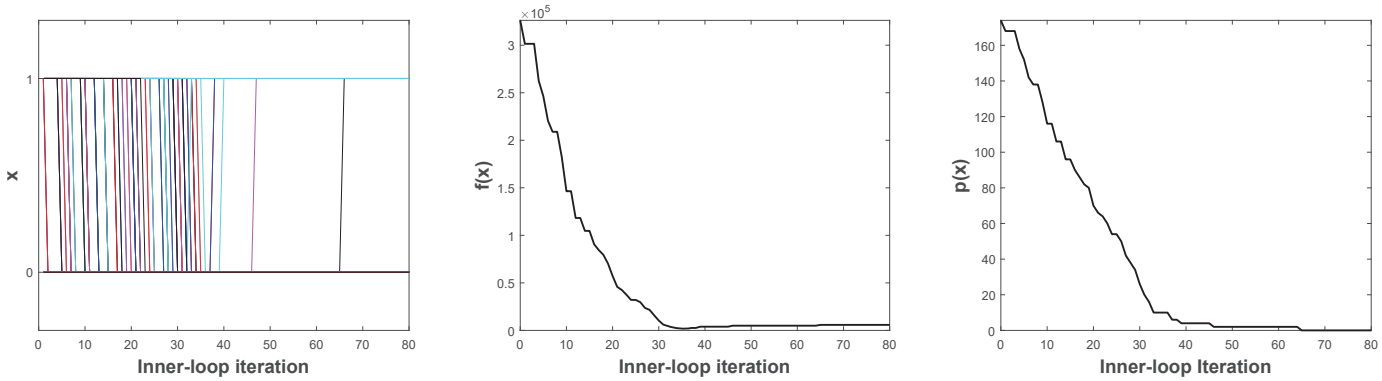
[9] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem," *Operations Research*, vol. 40, no. 6, pp. 1086–1094, 1992.

[10] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht, "Genetic algorithms for the traveling salesman problem," in *Proceedings of the first International Conference on Genetic Algorithms and their Applications*, vol. 160, no. 168. Lawrence Erlbaum, 1985, pp. 160–168.

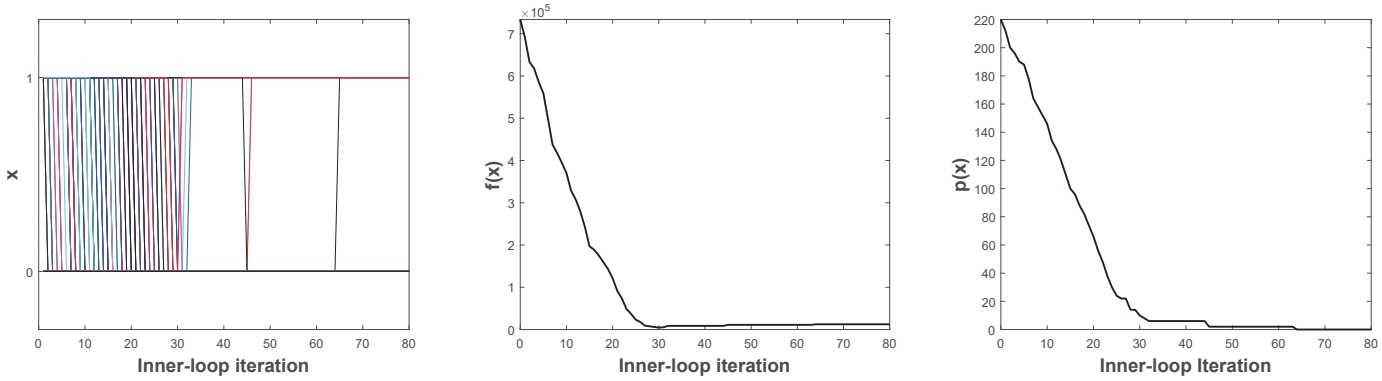
[11] T. Stützle and M. Dorigo, "ACO algorithms for the traveling salesman problem," *Evolutionary Algorithms in Engineering and Computer Science*, vol. 4, pp. 163–183, 1999.

[12] R. Baraglia, J. I. Hidalgo, and R. Perego, "A hybrid heuristic for the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 613–622, 2001.

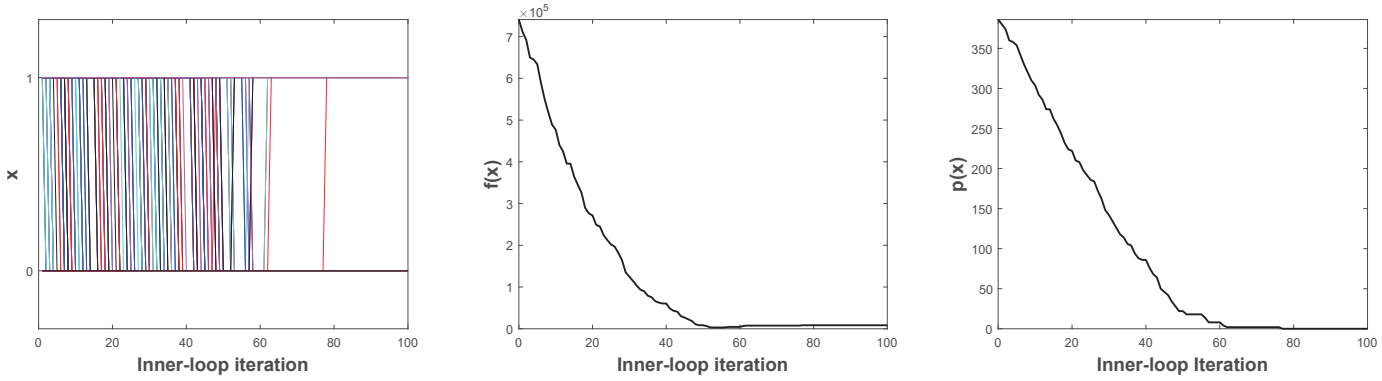
[13] K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, vol. 3. IEEE, 2003, pp. 1583–1585.



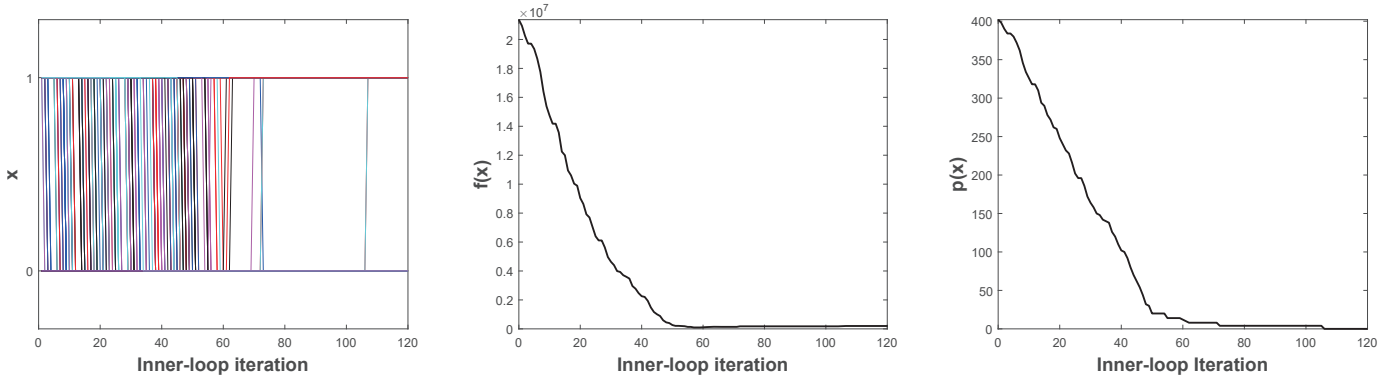
(a) BURMA14



(b) ULYSSES16

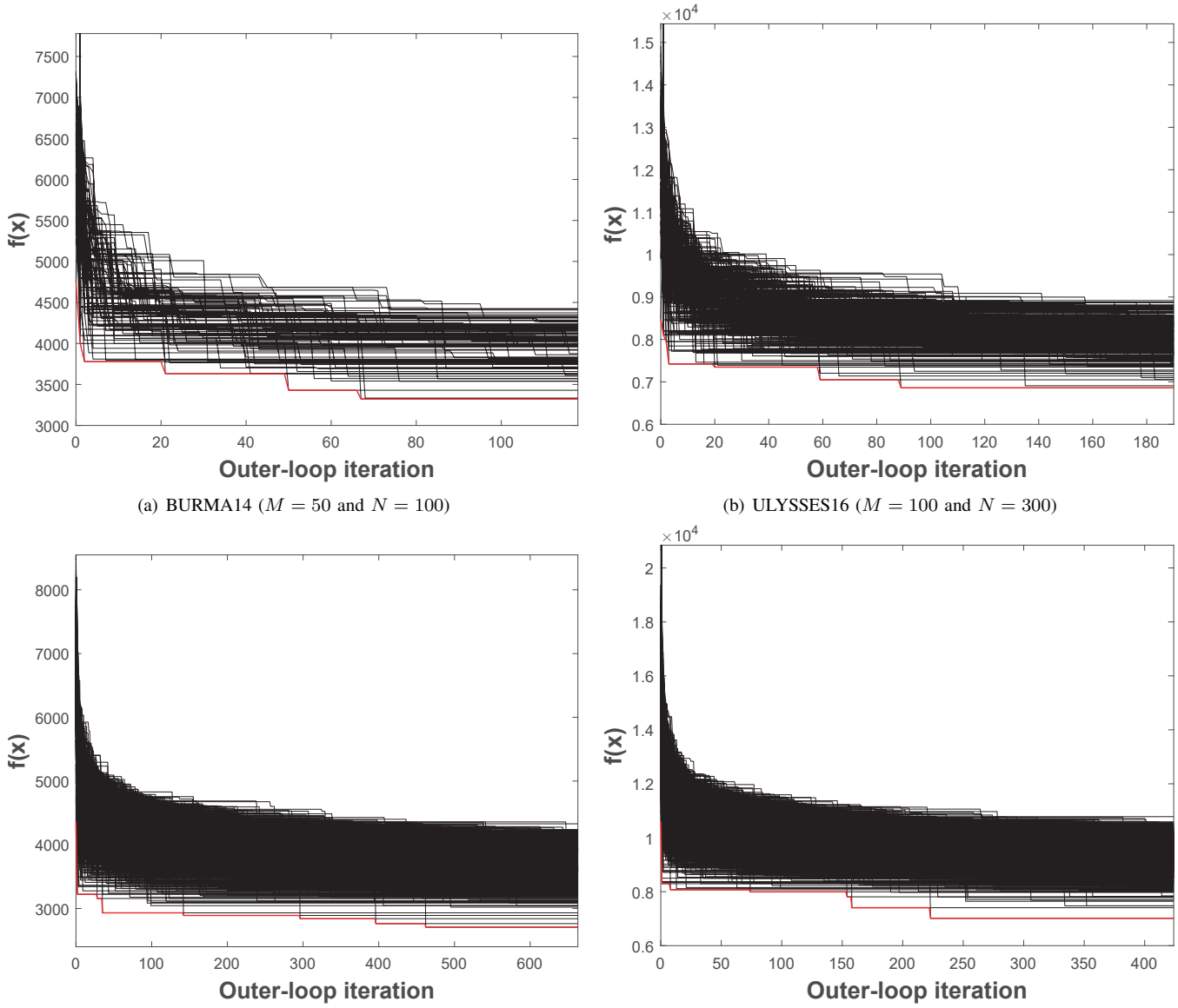


(c) GR21



(d) ULYSSES22

Fig. 1. Snapshots of neuronal states, objective function value and penalty function value of in the CNO-TSP/BM algorithm.



(a) BURMA14 ($M = 50$ and $N = 100$)

(b) ULYSSES16 ($M = 100$ and $N = 300$)

(c) GR21 ($M = 200$ and $N = 2600$)

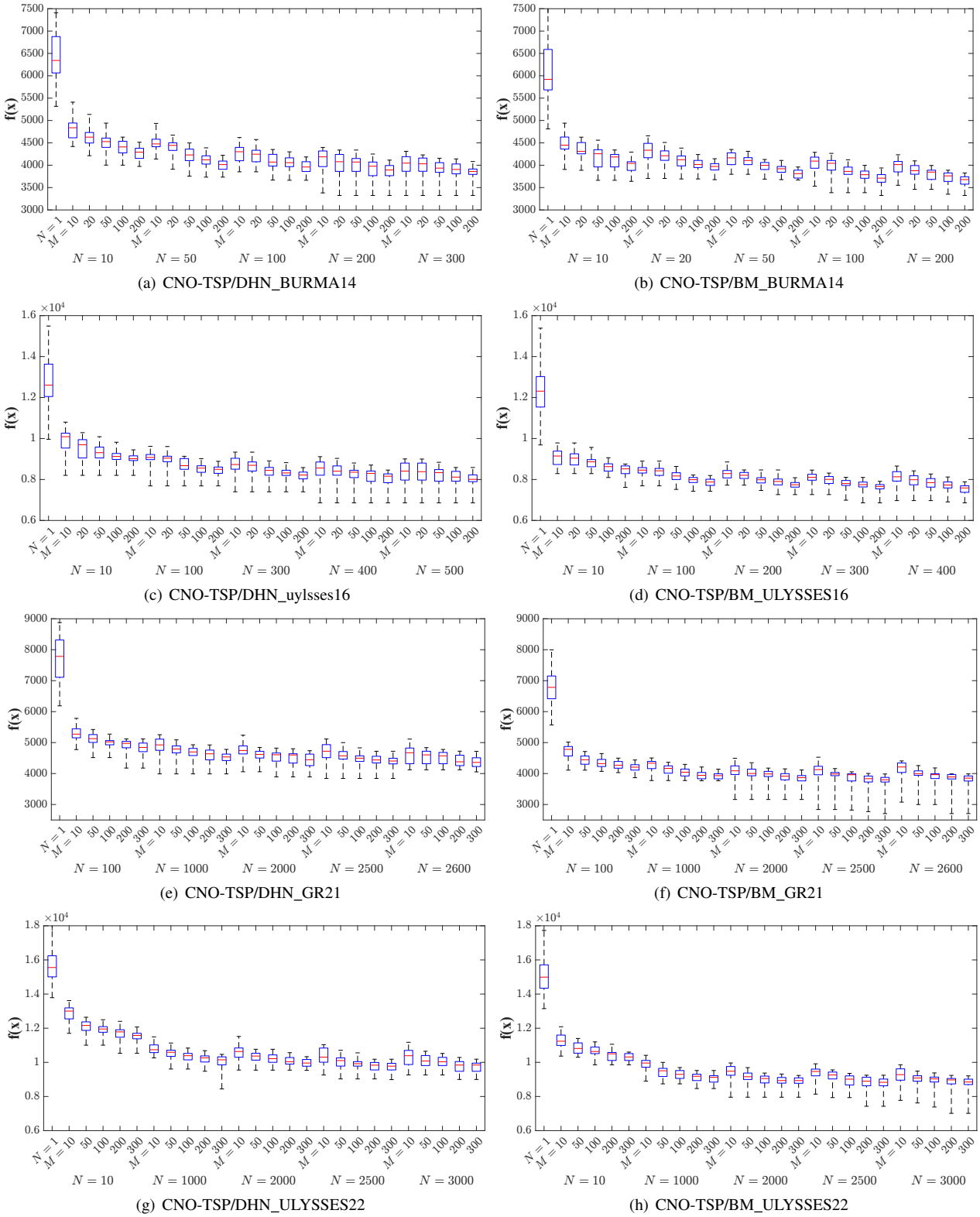
(d) ULYSSES22 ($M = 200$ and $N = 3000$)

Fig. 2. The convergent behavior of the CNO-TSP/BM algorithm on the four instances.

TABLE I
HYPER-PARAMETERS AND RESULTS OF CNO-TSP/DHN AND CNO-TSP/BM IN TERMS OF BEST/WORST VALUES, MEAN VALUES, AND STANDARD DEVIATIONS ON THE FOUR BENCHMARK INSTANCES

data set	# of cities	# of solutions	optimum	# of neurons	# of batches	algorithm	N	M	best/worst	mean $\pm$ std
BURMA14	14	8.7×10^{10}	3323	196	29	CNO-TSP/DHN	100	50	3668/4352	4085.72 $\pm$ 193.21
						CNO-TSP/BM	100	50	**3323**/4097	3860.56 $\pm$ 154.95
ULYSSES16	16	2.1×10^{13}	6859	256	33	CNO-TSP/DHN	300	100	7400/8822	8296.92 $\pm$ 299.55
						CNO-TSP/BM	300	100	**6859**/8100	7739.64 $\pm$ 295.78
GE21	21	5.1×10^{19}	2707	441	43	CNO-TSP/DHN	2600	200	4121/4723	4418.23 $\pm$ 196.48
						CNO-TSP/BM	2600	200	**2707**/3989	3833.24 $\pm$ 257.12
ULYSSES22	22	1.1×10^{21}	7013	484	45	CNO-TSP/DHN	3000	200	9000/10287	9774.12 $\pm$ 352.90
						CNO-TSP/BM	3000	200	**7013**/9237	8817.64 $\pm$ 457.69

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy*



(a) CNO-TSP/DHN_BURMA14

(b) CNO-TSP/BM_BURMA14

(c) CNO-TSP/DHN_uylsses16

(d) CNO-TSP/BM_ULYSSES16

(e) CNO-TSP/DHN_GR21

(f) CNO-TSP/BM_GR21

(g) CNO-TSP/DHN_ULYSSES22

(h) CNO-TSP/BM_ULYSSES22

Fig. 3. Monte Carlo test results of the CNO-TSP algorithms with several M and N on the four instances.

of Sciences, vol. 79, no. 8, pp. 2554–2558, 1982.

[15] ——, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[16] D. W. Tank and J. J. Hopfield, "Simple 'neural'optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit," *IEEE Trans. Circuits and Systems*, vol. 33, no. 5, pp. 533–541, 1986.

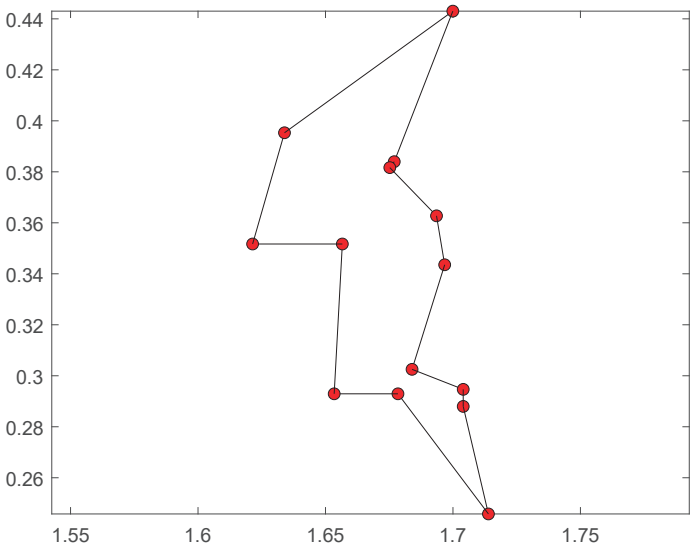


Fig. 4. The optimal tour of the dataset BURMA14 achieved by using the CNO-TSP/BM algorithm with $M = 50$ and $N = 100$.

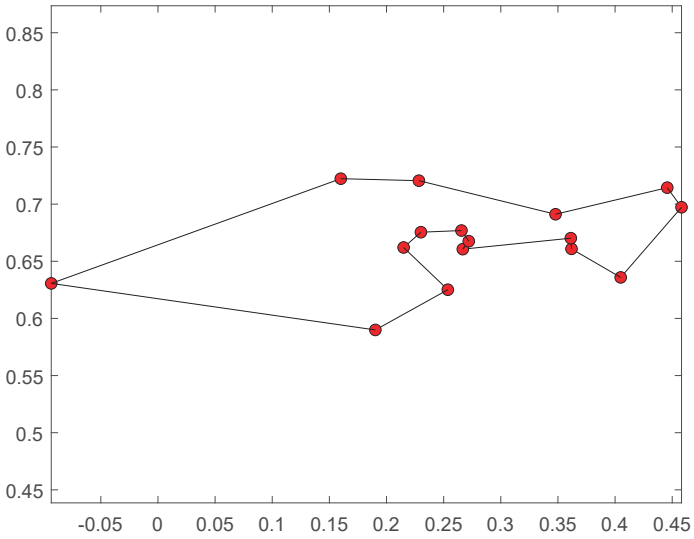


Fig. 5. The optimal tour of the dataset ULYSSES16 achieved by using the CNO-TSP/BM algorithm with $M = 100$ and $N = 300$.

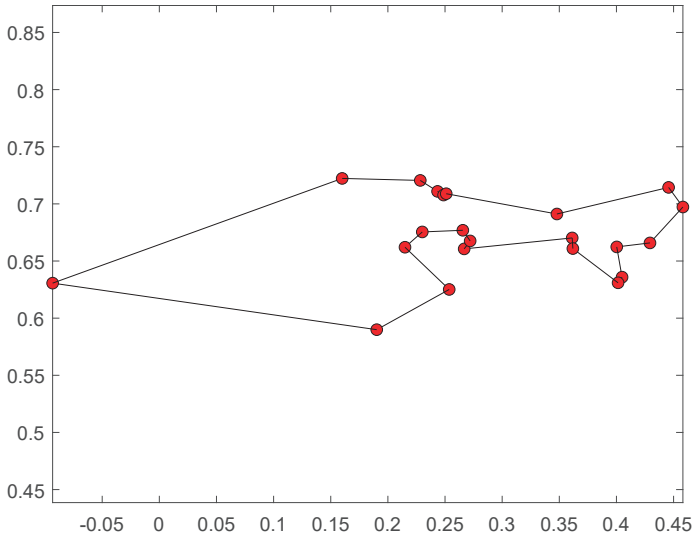


Fig. 6. The optimal tour of the dataset ULYSSES22 achieved by using the CNO-TSP/BM algorithm with $M = 200$ and $N = 3000$.

[17] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[18] ——, "Computing with neural circuits: a model," *Science*, vol. 233, no. 4764, pp. 625–633, 1986.

[19] P. M. Talaván and J. Yáñez, "Parameter setting of the Hopfield network applied to TSP," *Neural Networks*, vol. 15, no. 3, pp. 363–373, 2002.

[20] K. C. Tan, H. Tang, and S. S. Ge, "On parameter settings of Hopfield networks applied to traveling salesman problems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 5, pp. 994–1002, 2005.

[21] J. Wang, "Analysis and design of a recurrent neural network for linear programming," *IEEE Trans. Circuits and Systems: Part I*, vol. 40, no. 9, pp. 613–618, 1993.

[22] ——, "A deterministic annealing neural network for convex programming," *Neural Networks*, vol. 7, no. 4, pp. 629–641, 1994.

[23] Q. Liu and J. Wang, "A one-layer recurrent neural network with a discontinuous activation function for linear programming," *Neural Computation*, vol. 20, no. 5, pp. 1366–1383, 2008.

[24] Z. Guo, Q. Liu, and J. Wang, "A one-layer recurrent neural network for pseudoconvex optimization subject to linear equality constraints," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1892–1900, 2011.

[25] A. Hosseini, J. Wang, and S. M. Hosseini, "A recurrent neural network for solving a class of generalized convex optimization problems," *Neural Networks*, vol. 44, pp. 78–86, 2013.

[26] G. Li, Z. Yan, and J. Wang, "A one-layer recurrent neural network for constrained nonconvex optimization," *Neural Networks*, vol. 61, pp. 10–21, 2015.

[27] Y. Xia and J. Wang, "A bi-projection neural network for solving constrained quadratic optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 2, pp. 214–224, 2016.

[28] G. E. Hinton and T. J. Sejnowski, "Optimal perceptual inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1983, pp. 448–453.

[29] V. Zissimopoulos, V. T. Paschos, and F. Pekergin, "On the approximation of NP-complete problems by using the Boltzmann machine method: the cases of some covering and packing problems," *IEEE Transactions on Computers*, vol. 40, no. 12, pp. 1413–1418, 1991.

[30] J. H. Korst and E. H. Aarts, "Combinatorial optimization on a Boltzmann machine," *Journal of Parallel and Distributed Computing*, vol. 6, no. 2, pp. 331–357, 1989.

[31] E. H. Aarts and J. H. Korst, "Boltzmann machines for travelling salesman problems," *European Journal of Operational Research*, vol. 39, no. 1, pp. 79–95, 1989.

[32] M. Peng, N. K. Gupta, and A. F. Armitage, "An investigation into the improvement of local minima of the Hopfield network," *Neural Networks*, vol. 9, no. 7, pp. 1241–1253, 1996.

[33] H. Che and J. Wang, "A collaborative neurodynamic approach to global and combinatorial optimization," *Neural Networks*, vol. 114, pp. 15 – 27, 2019.

[34] Z. Yan, J. Fan, and J. Wang, "A collective neurodynamic approach to constrained global optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 5, pp. 1206–1215, 2017.

[35] Z. Yan, J. Wang, and G. Li, "A collective neurodynamic optimization approach to bound-constrained nonconvex optimization," *Neural Networks*, vol. 55, pp. 20–29, 2014.

[36] J. Wang and J. Wang, "Two-timescale multilayer recurrent neural networks for nonlinear programming," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[37] H. Che and J. Wang, "A two-timescale duplex neurodynamic approach to biconvex optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2503–2514, 2019.

[38] M.-F. Leung and J. Wang, "A collaborative neurodynamic approach to

multiobjective optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 11, pp. 5738 – 5748, 2018.
[39] S. Yang, Q. Liu, and J. Wang, "A collaborative neurodynamic approach to multiple-objective distributed optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 4, pp. 981–992, April 2018.
[40] H. Che and J. Wang, "A two-timescale duplex neurodynamic approach to mixed-integer optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 36–48, 2021.
[41] Z. Yan and J. Wang, "Nonlinear model predictive control based on collective neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 840–850, 2015.
[42] J. Wang, J. Wang, and Q. Han, "Neurodynamics-based model predictive control of continuous-time under-actuated mechatronic systems," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 1, pp. 311–321, Jan. 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9167474
[43] J. Fan and J. Wang, "A collective neurodynamic optimization approach to nonnegative matrix factorization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2344–2356, 10 2017.
[44] J. Wang, J. Wang, and H. Che, "Task assignment for multivehicle systems based on collaborative neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 4, pp. 1145–1154, 2020.
[45] J. Wang, J. Wang, and Q. Han, "Multi-vehicle task assignment based on collaborative neurodynamic optimization with discrete Hopfield networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021, doi=10.1109/TNNLS.2021.3082528, in press.
[46] Y. Wang, J. Wang, , and H. Che, "Two-timescale neurodynamic approaches to supervised feature selection based on alternative problem formulations," *Neural Networks*, vol. 142, pp. 180–191, Oct. 2021.
[47] M. Leung and J. Wang, "Minimax and biobjective portfolio selection based on collaborative neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2825–2836, Jul. 2021.
[48] W. Zhou, H.-T. Zhang, and J. Wang, "Sparse Bayesian learning based on collaborative neurodynamic optimization," *IEEE Transactions on Cybernetics*, pp. 1–15, 2021.
[49] J. Zhao, J. Yang, J. Wang, and W. Wu, "Spiking neural network regularization with fixed and adaptive drop-keep probabilities," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021.
[50] X. Li, J. Wang, and S. Kwong, "Hash bit selection via collaborative neurodynamic optimization with discrete Hopfield networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2021, doi=10.1109/TNNLS.2021.3068500, in press.
[51] H. Li, J. Wang, and J. Wang, "Solving the travelling salesman problem based on collaborative neurodynamic optimization with discrete Hopfield networks," in *2021 11th International Conference on Information Science and Technology (ICIST)*. IEEE, 2021, pp. 456–465.
[52] E. Goles-Chacc, F. Fogelman-Soulié, and D. Pellegrin, "Decreasing energy functions as a tool for studying threshold networks," *Discrete Applied Mathematics*, vol. 12, no. 3, pp. 261–277, 1985.
[53] B. Cernuschi-Frías, "Partial simultaneous updating in Hopfield memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 4, pp. 887–888, 1989.
[54] D.-L. Lee, "New stability conditions for Hopfield networks in partial simultaneous update mode," *IEEE Transactions on Neural Networks*, vol. 10, no. 4, pp. 975–978, 1999.
[55] J. Muñoz-Pérez, A. Ruiz-Sepúlveda, and R. Benítez-Rochel, "Parallelism in binary hopfield networks," in *Advances in Computational Intelligence*, J. Cabestany, I. Rojas, and G. Joya, Eds. Springer Berlin Heidelberg, 2011, pp. 105–112.
[56] E. H. Aarts and J. H. Korst, "Boltzmann machines as a model for parallel annealing," *Algorithmica*, vol. 6, no. 1, pp. 437–465, 1991.
[57] Y. Zhang, S. Wang, P. Phillips, and G. Ji, "Binary PSO with mutation operator for feature selection using decision tree applied to spam detection," *Knowledge-Based Systems*, vol. 64, pp. 22–31, 2014.
[58] J. Wang, "A deterministic connectionist machine for the traveling salesman problem," in *Proc. IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 1990, pp. 374–375.
[59] G. A. Kochenberger, F. Glover, B. Alidaee, and C. Rego, "A unified modeling and solution framework for combinatorial optimization problems," *OR Spectrum*, vol. 26, no. 2, pp. 237–250, 2004.
[60] G. Reinelt, "TSPLIB—a traveling salesman problem library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.