12th International Conference on Information Science and Technology (ICIST 2022)
Kaifeng, China, October 14-16, 2022

Collaborative Neurodynamic Algorithms for Solving Sudoku Puzzles

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

2022 12th International Conference on Information Science and Technology (ICIST) | 978-1-6654-8582-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIST55546.2022.9926961

***Abstract*—In this article, Sudoku is formulated as a quadratic unconstrained binary optimization, and a variables reduction algorithm is proposed based on given elements. Collaborative neurodynamic optimization algorithms based on discrete Hopfield networks or Boltzmann machines are developed for solving the formulated optimization problem. A population of discrete Hopfield networks or Boltzmann machines operating concurrently are employed for scatter search. A particle swarm optimization rule is used to re-initialize the initial states of discrete Hopfield networks or Boltzmann machines upon their local convergence. Experimental results on five Sudoku instances are elaborated to demonstrate the efficacy of the proposed collaborative neurodynamic optimization algorithms for solving Sudoku puzzles.**

***Index Terms*—Sudoku, discrete Hopfield network, Boltzmann machine, collaborative neurodynamic optimization.**

I. Introduction

Sudoku is a well-known logical puzzle game that was first published in 1979. It is presented with a 9 9 grid, where some of the cells contain integers between 1 and 9. The task is to fill the remaining cells to satisfy that each row, each column, and each 3 3 box contains the integers from 1 to 9 exactly once. It has various applications, such as secret digits embedding [1], photovoltaic array reconfiguration [2], and secret data hiding [3].

Sudoku is NP-complete [4], so numerous heuristic methods and meta-heuristic methods are developed for solving Sudoku. Heuristic methods include harmony search [5], Sinkhorn balancing algorithm [6], nested Monte Carlo search [7], entropy minimization [8], Monte Carlo search algorithm [9], backtracking algorithm with heuristic moves [10], set selection neural networks with partial memories [11], memetic coloring algorithm [12], etc. Meta-heuristic methods genetic algorithm [13], simulated annealing [14], tabu search with an arc-consistency 3 algorithm [15], integer-value particle swarm optimization [16], hybrid genetic algorithm [17], ant colony optimization [18], etc.

Various neurodynamic optimization models, such as Hopfield networks [19]–[22] and projection neural networks [23]–[29] have been developed to solve numerous optimization problems. Unlike these deterministic neurodynamic optimization models, the Boltzmann machine is a stochastic neural network [30] with a local hill-climbing capability for solving combinatorial optimization problems [31], [32]. The Boltzmann machines are developed for solving various combinatorial optimization [31]–[34]. Although the Boltzmann machine has been proved that it almost surely convergent to global optima, it entails a sufficient long cooling schedule.

It is well known that a single gradient-driven neurodynamic model easily gets stuck in a local optimum when facing combinatorial optimization problems with binary variables [35]. In such scenarios, multiple neurodynamic models are needed to work collaboratively in order to achieve the task. Collaborative neurodynamic optimization (CNO) is a hybrid intelligence framework that has been developed for solving combinatorial optimization problems in recent years [36] With multiple neurodynamic optimization models for scattered searches and a meta-heuristic rule to initialize the neuronal states upon neuronal search convergence, CNO is proven to be almost surely convergent to global optima of optimization problems [37]. In recent years, the CNO approach shows its admirable searching performance in numerous complex optimization problems, including global optimization [36]–[39], multi-objective optimization [40], [41], biconvex optimization [42], and combinatorial and mixed-integer optimization [36], [43]. CNO is customized in various applications, such as traveling salesman problem [22], [34], multi-vehicle task assignment [44], [45], model predictive control [46], [47], portfolio selection [48], hash bit selection [49], feature selection [50], nonnegative matrix factorization [51], [52], and spiking neural network regularization [53], etc.

In this article, Sudoku is formulated as a quadratic unconstrained binary optimization, and its variables are reduced based on given elements. The formulated problem is solved by CNO algorithms employing multiple discrete Hopfield networks or Boltzmann machines with momentum terms to search global optima collaboratively. In the proposed CNO algorithms, a population of discrete Hopfield networks or Boltzmann machines with momentum terms are employed for scatter search. Discrete Hopfield networks and Boltzmann machines with momentum terms are operated in fully parallel for

This work was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region of China under Grants 11202318, 11202019, and 11203721; and in part by the InnoHK initiative, the Government of the Hong Kong Special Administrative Region, and Laboratory for AI-Powered Financial Technologies.

expediting their convergence. A particle swarm optimization rule is used to re-initialize the initial states of discrete Hopfield networks or Boltzmann machines upon their local convergence to escape from local minima toward global minima.

The remainder of this article is organized as follows. In Section II, necessary preliminaries on discrete Hopfield network, Boltzmann machine, and collaborative neurodynamic optimization are introduced. In Section III, the formulation and reformulation of Sudoku are described. In Section IV, the proposed CNS/BMm and CNS/DHNm algorithms are delineated. In Section V, experimental results on five instances are discussed in detail. Finally, in Section VI, the concluding remarks are given.

II. Preliminaries

A. Neurodynamic Optimization

1) Discrete Hopfield Network: The discrete Hopfield network (DHN) is a classic recurrent neural network operating with binary or bipolar states and activation function in discrete time [54] as follows:

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

$$x(t+1) = \sigma(u(t)), \tag{2}$$

where $u \in \mathbb{R}^n$ is the net-input vector, $x \in \mathbb{R}^n$ is the state vector, $W \in \mathbb{R}^{n\times n}$ is the connection weight matrix, $\theta \in \mathbb{R}^n$ is the threshold vector, and $\sigma(\cdot)$ is a vector-valued discontinuous activation function defined element-wisely as follows:

$$x_i(t+1) = \sigma(u_i) = \begin{cases} 0 & \text{if} \;\; u_i(t) \le 0, \\ 1 & \text{if} \;\; u_i(t) > 0. \end{cases}$$

As a variant of the DHN, a DHN with a momentum term (DHNm) [55] is developed as follows:

$$\begin{cases} u(t+1) = u(t) + Wx(t) - \theta, \\ x(t) = \sigma(u(t)), \end{cases} \tag{3}$$

where $u(0) = 0$.

With the addition of the momentum term $u(t-1)$ in the DHN dynamic equation, the DHNm in (3) takes its historical effect into account and enriches its dynamic behaviors. It is shown that all neuronal states in the DHNm in (3) can be activated synchronously and are convergent to local or near optima [56], [57].

It is shown in [54] that the DHN is convergent to a local minimum of the following combinatorial optimization problem:

$$\min \;\; -\frac{1}{2}x^TWx + \theta^Tx \;\; \text{s.t.} \;\; x \in \{0,1\}^n. \tag{4}$$

2) Boltzmann Machine: The Boltzmann machine (BM) is a well-known stochastic neural network, and each state x_i is updated synchronously according to an acceptance probability [30]:

$$P(x_i(t) = 1) = 1/(1 + \exp(-u(t)/T(t))), \tag{5}$$

where $T(t)$ is a positive temperature parameter at iteration t that is updated as follows:

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

where T_0 is an initial temperature and $0 < \eta < 1$ is a cooling factor.

In analogy to DHNm, a BM with a momentum term (BMm) is developed as follows:

$$\begin{cases} u(t+1) = u(t) + Wx(t) - \theta, \\ p(x_i(t) = 1) = 1/(1 + \exp(-u_i(t)/T)), \\ p(x_i(t) = 0) = 1 - p(x_i(t) = 1). \end{cases} \tag{6}$$

3) Collaborative Neurodynamic Optimization: The neurodynamic models used in existing CNO approaches include projection neural networks [40], [44], [46]–[48], [51], [52], discrete Hopfield networks (2) [22], [45], [49], and Boltzmann machine (5) [34]. Almost all of the CNO algorithms use a particle swarm optimization rule as defined in [58]:

$$\begin{cases} v_i(t+1) = & c_0v_i(t) + c_1r_1(x_i^* - x_i(t)) \\ & +c_2r_2(x^* - x_i(t)), \\ x_i(t+1) = & x_i(t) + v_i(t+1), \end{cases} \tag{7}$$

where x_i is the current position of the i^{th} particle, $v_i \in \mathbb{R}$ is a velocity determining the searching direction of the i^{th} particle, x_i^* is the current best position of the i^{th} particle, x^* is the current best position of a group (solution set), $c_0 \in [0,1]$ is an inertia weight determining the weight of the previous velocity, $c_1 \in [0,1]$ 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.

The diversity of the particles is crucial for searching. Mutation operation is a frequently-used method to enhance diversity and avoid premature convergence. A diversity of a group is measured as follows:

$$\delta(x) = \frac{1}{Nn}\sum_{i=1}^{N} \|x^{(i)} - x^*\|_2, \tag{8}$$

where N is the population size of the swarm, n is the dimension of a solution, $x^{(i)}$ is the i^{th} particle, and x^* is the current best solution of the whole population.

The bit-flip mutation is a typical mutation operator for evolutionary algorithms applied to optimization with binary variables as defined in [59]:

$$x_j = \begin{cases} \neg x_j & \text{if} \;\; \kappa \le P_m, \\ x_j & \text{otherwise} \;, \end{cases} \tag{9}$$

where $\neg x_j$ is the negation of x_j, $\kappa \in [0,1]$ is a random number, P_m is the probability of mutation.

III. PROBLEM FORMULATION

Consider a binary formulation [60] of Sudoku puzzles in the following form:

$$\sum_{k=1}^{n} x_{ijk} = 1,\ i,j = 1,...,n, \tag{10a}$$

$$\sum_{i=1}^{n} x_{ijk} = 1,\ j,k = 1,...,n, \tag{10b}$$

$$\sum_{j=1}^{n} x_{ijk} = 1,\ i,k = 1,...,n, \tag{10c}$$

$$\sum_{i=\sqrt{n}(p-1)+1}^{\sqrt{n}p} \sum_{j=\sqrt{n}(q-1)+1}^{\sqrt{n}q} x_{ijk} = 1,$$

$$p,q = 1,...,\sqrt{n},\ k = 1,...,n, \tag{10d}$$

$$x_{ijk} = 1,\ \forall(i,j,k) \in \mathcal{G}, \tag{10e}$$

$$x_{ijk} \in \{0,1\}, \tag{10f}$$

where n is a size of a Sodoku puzzle (commonly, $n = 9$), $\sqrt{n}$ is the size of a block. x_{ijk} is a decision variable such that $x_{ijk} = 1$ means that the cell in ith row and jth column is set to be k and $x_{ijk} = 0$ otherwise, $\mathcal{G}$ denotes a set of binary-coded elements that are given. Constraints in (10a) ensure that every cell of a completed Sudoku grid is filled. Constraints in (10b) ensure that each column of a completed Sudoku grid contains each of 1 to n exactly once. Constraints (10c) ensure that each row of a completed Sudoku grid contains each of 1 to n exactly once. Constraints (10d) ensure that each $m \times m$ block of a completed Sudoku grid contains each of 1 to n exactly once. Constraints (10e) enforce the given elements $\mathcal{G}$. Constraints (10f) enforce the solution to be binary.

In the view that $\mathcal{X} = [x_{ijk}]$ is a 3-order tensor, the tensor $\mathcal{X}$ is vectorized to facilitate the following description. The vectorized decision variables are stated as follows:

$$\bar{x} = \mathrm{vec}(\mathcal{X}^{T_2}) = [x_{111}, x_{112}, x_{113}, \cdots, x_{997}, x_{998}, x_{999}] \in \{0,1\}^{n^3}$$

where $(\cdot)^{T_2}$ is the second transpose of tensor, which means $\mathcal{X}^{T_2}(i_3, i_2, i_1) = \mathcal{X}(i_1, i_2, i_3)$ for a 3-order tensor.

The constraint satisfaction problem can be re-formulated as an unconstrained problem by introducing a quadratic penalty function into the objective function as an alternative to imposing constraints. To handle constraints (10a)-(10e), quadratic penalty terms are defined as follows:

$$p_a(\bar{x}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\big(\sum_{k=1}^{n} x_{ijk} - 1\big)^2 = \frac{1}{2}\|A_a\bar{x} - e_{n^2}\|_2^2, \tag{11a}$$

$$p_b(\bar{x}) = \frac{1}{2}\sum_{j=1}^{n}\big(\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} - 1\big)^2 = \frac{1}{2}\|A_b\bar{x} - e_{n^2}\|_2^2, \tag{11b}$$

$$p_c(\bar{x}) = \frac{1}{2}\sum_{i=1}^{n}\big(\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} - 1\big)^2 = \frac{1}{2}\|A_c\bar{x} - e_{n^2}\|_2^2, \tag{11c}$$

$$\begin{aligned} p_d(\bar{x}) &= \frac{1}{2}\sum_{p=1}^{\sqrt{n}}\sum_{q=1}^{\sqrt{n}}\big(\sum_{i=\sqrt{n}(p-1)+1}^{\sqrt{n}p}\ \sum_{j=\sqrt{n}(q-1)+1}^{\sqrt{n}q}\ \sum_{k} x_{ijk} - 1\big)^2 \\ &= \frac{1}{2}\|A_d\bar{x} - e_{n^2}\|_2^2, \end{aligned} \tag{11d}$$

where

$$A_a = \begin{bmatrix} e_n^T & 0 & \cdots & 0 \\ 0 & e_n^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_n^T \end{bmatrix} \in \{0,1\}^{n^2 \times n^3},$$

$$A_b = \begin{bmatrix} I^n & 0 & \cdots & 0 \\ 0 & I^n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I^n \end{bmatrix} \in \{0,1\}^{n^2 \times n^3},$$

$$A_c = \begin{bmatrix} I_{n^2} & I_{n^2} & \cdots & I_{n^2} \end{bmatrix} \in \{0,1\}^{n^2 \times n^3},$$

$$A_d = \begin{bmatrix} \widetilde{I} & \widetilde{I} & \widetilde{I} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \widetilde{I} & \widetilde{I} & \widetilde{I} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \widetilde{I} & \widetilde{I} & \widetilde{I} \end{bmatrix} \in \{0,1\}^{n^2 \times n^3},$$

$$\widetilde{I} = \begin{bmatrix} I_n & I_n & I_n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_n & I_n & I_n & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_n & I_n & I_n \end{bmatrix} \in \{0,1\}^{3n \times n^2},$$

$$I^n = \underbrace{[I_n, I_n, ..., I_n]}_{n}, \quad e_n = \underbrace{[1, 1, ..., 1]}_{n}^T,$$

I_n is an $n \times n$ identity matrix.

For constraints in (10e), the given binary-coded elements can be substituted in (11) to reduce the number of variables. For a given binary coded element $x_{i_g j_g k_g}$, $x_{i_g j_g k_g} = 1$ and $x_{i_g j_g k} = 0,\ k \neq k_g$; e.g., in Fig. 1, $x_{114} = 1$, and $x_{11k} = 0$, where $k \neq 4$. Besides that, variables x_{ijk_g} with the same column, row, or sub-grid, and the same k_g with given binary coded elements should be 0; e.g., in Fig. 1, $x_{124} = 0$, and $x_{121} = 0$. The algorithm of variable reduction is detailed in the next chapter.

	1	2	3	4	5	6	7	8	9
1	4								
2	1			5		9	7	3	
3					4			1	9
4	2		4	6		8	1	9	7
5					5	1			2
6			3	4		2			8
7	3		5	8			9	2	6
8	7		6				8	5	
9	8		1			5	4	7	

Fig. 1. A Sudoku puzzle instance with given decimal-coded elements.

With the quadratic penalty terms to be minimized, a penalty function is defined as follows:

$$p(\bar{x}) = p_b(\bar{x}) + p_c(\bar{x}) + p_d(\bar{x}) + p_e(\bar{x}), \tag{12}$$

where $\bar{x} \in \{0,1\}^n$.

Based on the penalty function in (12), the original problem in (10) is expressed as follows:

$$\min_{\bar{x}} \; p(\bar{x}), \quad \text{s.t. } \bar{x} \in \{0,1\}^n. \tag{13}$$

IV. CNS Algorithms

Algorithm 1 details the variable reduction method. The every given variables $x_{i_g j_g k_g}$(i.e., $(i_g, j_g, k_g) \in \mathcal{G}$) is set to 1, and $x_{i_g j_g k}$ (i.e., $(i_g, j_g, k) \notin \mathcal{G}$) is set to 0 in steps 5-15, where δ_i is a flag such that $\delta_i = 1$ denotes the i^{th} variable is fixed, and $\delta_i = 0$ otherwise. The not-given variables that have the same row and k with given variables are set to 0 in steps 17-20. The not-given variables that have the same column and k with given variables are set to 0 in steps 21-24. The not-given variables that have the same sub-grid and k with given variables are set to 0 in steps 25-30. If an element that is not given has a unique k in a row, column, or sub-grid, the element can be fixed at k and it is described in steps 32-37. The dimensions of matrix A_a, A_b, A_c, and A_d are reduced according to δ and S in steps 39-49. If $\delta_i = 0$, then the variable $\bar{x}_i$ is fixed and can reduced. Therefore, the reduced variables $\hat{x}$ are attained.

The quadratic penalty terms in (11) are redefined with reduced variables $\hat{x}$ as follows:

$$p_b(\hat{x}) = \frac{1}{2}\|\hat{A}_b\hat{x} - b_b\|_2^2, \qquad p_c(\hat{x}) = \frac{1}{2}\|\hat{A}_c\hat{x} - b_c\|_2^2,$$
$$p_d(\hat{x}) = \frac{1}{2}\|\hat{A}_d\hat{x} - b_d\|_2^2, \qquad p_e(\hat{x}) = \frac{1}{2}\|\hat{A}_e\hat{x} - b_e\|_2^2.$$

Algorithm 1: Variable reduction

Input: A_a, A_b, A_c, A_d.
Output: $\hat{A}_a$, $\hat{A}_b$, $\hat{A}_c$, $\hat{A}_d$, b_a, b_b, b_c, b_d, δ.

1 $\delta \leftarrow \mathbf{0}^{n^3}$;
2 $s \leftarrow \mathbf{0}^{n^3}$;
3 $\Delta \leftarrow 1$;
4 **while** $\Delta = 1$ **do**
5   **foreach** $(i_g, j_g, k_g) \in \mathcal{G}$ **do**
6   **for** $k = 1$ **to** n **do**
7   **if** $k \neq k_g$ **then**
8         $s_{i_g \times n^2 + j_g \times n + k} \leftarrow 0$;
9         $\delta_{i_g \times n^2 + j_g \times n + k} \leftarrow 1$;
10   **else**
11         $s_{i_g \times n^2 + j_g \times n + k} \leftarrow 1$;
12         $\delta_{i_g \times n^2 + j_g \times n + k} \leftarrow 1$;
13   **end**
14   **end**
15   **end**
16   **foreach** $(i_e, j_e, k_e) \notin \mathcal{G}$ **do**
17   **foreach** $(i_e, c, k) \in \mathcal{G}$ **do**
18       $s_{i_e \times n^2 + c \times n + k} \leftarrow 0$;
19       $\delta_{i_e \times n^2 + c \times n + k} \leftarrow 1$;
20   **end**
21   **foreach** $(r, j_e, k) \in \mathcal{G}$ **do**
22       $s_{r \times n^2 + j_e \times n + k} \leftarrow 0$;
23       $\delta_{r \times n^2 + j_e \times n + k} \leftarrow 1$;
24   **end**
25   **foreach** *elements* (i, j) *in the sub-grid where* (i_e, j_e) *is in located* **do**
26   **if** $(i, k, k) \in \mathcal{G}$ **then**
27         $s_{i_e \times n^2 + j_e \times n + k} \leftarrow 0$;
28         $\delta_{i_e \times n^2 + j_e \times n + k} \leftarrow 1$;
29   **end**
30   **end**
31   **end**
32   **if** *A element* (i, j) *that is not given has a unique* k *in a row, column, or sub-grid* **then**
33   add (i, j, k) of the element to $\mathcal{G}$;
34     $\Delta \leftarrow 1$
35   **else**
36     $\Delta \leftarrow 0$
37   **end**
38 **end**
39 $b_a = e_{n^2} - A_a s$;
40 $b_b = e_{n^2} - A_b s$;
41 $b_c = e_{n^2} - A_c s$;
42 $b_d = e_{n^2} - A_d s$;
43 **foreach** $i \in \{i | \delta_i = 1\}$ **do**
44   delete the i^{th} column of A_a, A_b, A_c, and A_d;
45 **end**
46 $\hat{A}_a \leftarrow A_a$;
47 $\hat{A}_b \leftarrow A_b$;
48 $\hat{A}_c \leftarrow A_c$;
49 $\hat{A}_d \leftarrow A_d$;
50 **return** $\hat{A}_a$, $\hat{A}_b$, $\hat{A}_c$, $\hat{A}_d$, b_a, b_b, b_c, b_d.

A CNO procedure to Sudoku (CNS-BMm) is detailed in Algorithm 2. A population of BMms is employed in Steps 2 - 7 for scattered searches, where N in Step 2 is the population size of BMms. The best solution in the BMms is determined in Steps 9 - 12. The initial states of BMms are re-positioned with particle swarm optimization update rule in Steps 15 - 20, where $U(0,1)$ in Step 16 denotes a random number between zero and one, and $P_{[0,1]}(x)$ in Step 18 denotes a projection function with image $[0,1]$. The diversity is measured in Step 21. A bit-flip mutation is performed if the diversity measure is smaller than the threshold $\mathcal{T}$ in Steps 22 - 24. The CNS/DHNm algorithm with a population of DHNms can be implemented by replacing BMm (6) in Step 3 with DHNm (3).

Algorithm 2: CNS/BMm algorithm

Input: Number of neurodynamic models N, initial states $x^{(i)}(0) \in \{0,1\}^{n^2}$, $i = 1, ..., N$, velocity vector $v^{(i)} \in [-1,1]^{n^2}$, $i = 1, ..., N$, initial temperature T_0, cooling rate η, termination criterion M, parameters of particle swarm optimization rule c_0, c_1 and c_2, diversity threshold $\mathcal{T}$.

Output: x^*.

1 **while** $m \leq M$ **do**
2   **for** $i = 1$ **to** N **do**
3   Obtain the equilibrium state $\bar{x}^{(i)}$ of the ith BMm acorrding to eq. (6)) with initial state $x^{(i)}(0)$, initial temperature T_0, and cooling factor η;
4   **if** $f(\bar{x}^{(i)}) < f(x^{(i)})$ **then**
5       $x^{(i)} \leftarrow \bar{x}^{(i)}$;
6   **end**
7   **end**
8   $i^* = \arg\min_i \{f(x^{(1)}), ..., f(x^{(i)}), ..., f(x^{(N)}\}$;
9   **if** $f(x^{(i^*)}) < f(x^*)$ **then**
10     $x^* \leftarrow x^{(i^*)}$;
11     $m \leftarrow 0$;
12   **else**
13     $m \leftarrow m + 1$;
14   **end**
15   **for** $i = 1$ **to** N **do**
16   Update velocity $v^{(i)} = c_0 v^{(i)} + c_1 U(0,1)(x^{(i)} - \bar{x}^{(i)}) + c_2 U(0,1)(x^* - \bar{x}^{(i)})$;
17   Update initial state $x^{(i)}(0) = x^{(i)}(0) + v^{(i)}$;
18     $x^{(i)}(0) = P_{[0,1]}(x^{(i)}(0))$;
19     $x^{(i)}(0) = \text{round}(x^{(i)}(0))$;
20   **end**
21   Calculate the diversity of the swarm δ according to Eq. (8);
22   **if** $\delta < \mathcal{T}$ **then**
23   Perform the bit-flip mutation according to Eq. (9);
24   **end**
25 **end**
26 **return** x^*.

V. Experimental Results

Instances are selected based on previously used in the literature. Consider the ten instances used in [61] (labeled here Sabuncu1-Sabuncu10) that are all logically solvable. The variable reduction (Algorithm 1) is carried out before optimization. Table I records the number of variables after variable reduction. The number of variables before reduction is $n^3 = 729$ and can be reduced to $0\% - 28\%$ by using Algorithm 1. In particular, Sabuncu1, Sabuncu2, Sabuncu5, Sabuncu8, and Sabuncu10 can be solved directly by Algorithm 1.

TABLE I
THE NUMBER OF VARIABLES AFTER VARIABLE REDUCTION

Instance	# of remaining variables
Sabuncu1	0
Sabuncu2	0
Sabuncu3	171
Sabuncu4	95
Sabuncu5	0
Sabuncu6	209
Sabuncu7	168
Sabuncu8	0
Sabuncu9	163
Sabuncu10	0

Fig. 2 snapshots the transient states of a single BMm and corresponding penalty function values on Sabuncu3, Sabuncu4, Sabuncu6, Sabuncu7, and Sabuncu9. Fig. 3 depicts the convergent behavior of the CNS/BMm algorithm on Sabuncu3, Sabuncu4, Sabuncu6, Sabuncu7, and Sabuncu9. Fig. 9 illustrates monte Carlo test results on the instances using CNS algorithms. When the Sudoku is solved in all 100 runs by two CNS algorithms, N in CNS/BMm is much smaller than that in CNS/DHNm owning to the local hill-climbing capability of BMm. Table II records the solution dimensions of the datasets, the number of solutions, hyper-parameter values, and results of CNS/DHNm and CNS/BMm in terms of best/worst values, mean values, and standard deviations on the five instances. Figs. 4-8 show the feasible solutions obtained by variable reduction algorithm (Algorithm 1) and CNS/BMm (Algorithm 2) on instances Sabuncu3, Sabuncu4, Sabuncu6, Sabuncu7, and Sabuncu9, respectively. The left boards are before variable reduction, where the red cells indicate that it is a given cell, and the white cells indicate that they are not fixed (the numbers in it are allowable numbers). The middle boards are after variable reduction, where the blue cells are fixed after variable reduction. The number of allowable numbers in white cells is much smaller after variable reduction. The right boards are the results obtained by using the CNS/BMm algorithm, where the green cells are the cell solved by the CNS/BMm algorithm.

VI. Concluding remarks

In this article, Sudoku is formulated as a quadratic unconstrained binary optimization problem. A variable reduction algorithm is proposed to reduce the number of variables with the information in known cells. Two collaborative neurodynamic Sudoku algorithms are developed based on a population of discrete Hopfield networks and Boltzmann machines activated synchronously. The performance of the proposed algorithms is

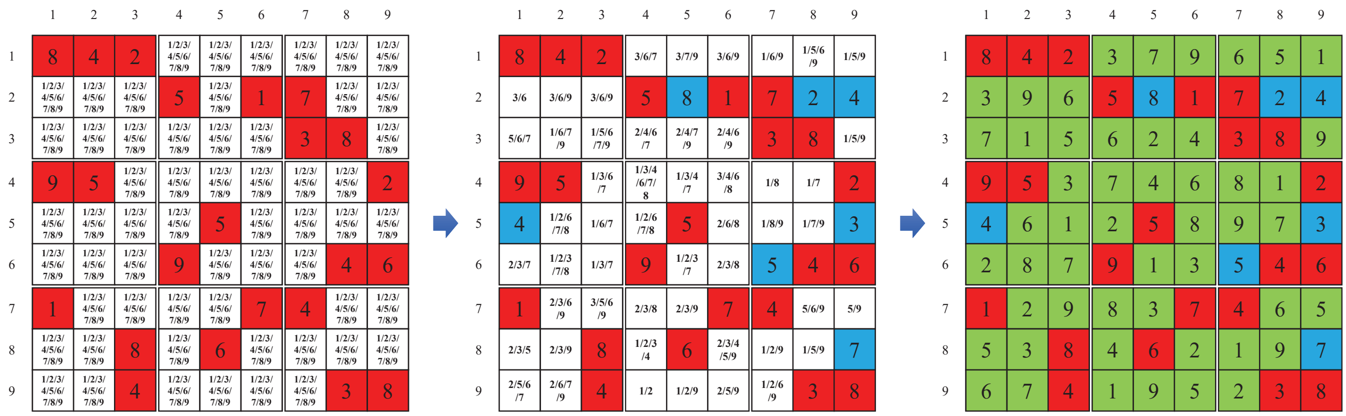


Fig. 4. Feasible results obtained by using the proposed variables reduction and CNS/BMm algorithm on Sabuncu3.

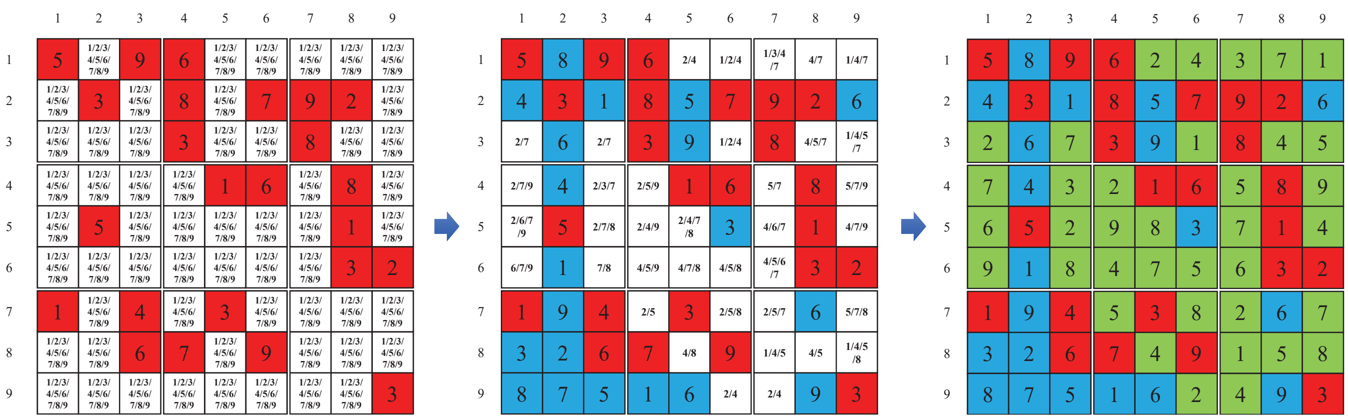


Fig. 5. Feasible results obtained by using the proposed variables reduction and CNS/BMm algorithm on Sabuncu4.

TABLE II
THE SOLUTION DIMENSIONS OF THE DATASETS, THE NUMBER OF SOLUTIONS, HYPER-PARAMETER VALUES, AND RESULTS OF CNS/DHNM AND CNS/BMM IN TERMS OF BEST/WORST VALUES, MEAN VALUES, AND STANDARD DEVIATIONS ON THE FIVE INSTANCES

Instance	n	# of dimensions	# of solutions	algorithm	N	M	best/worst	mean ± std
Sabuncu3	9	171	2.99×10^{51}	CNS/DHNm CNS/BMm	200 40	50 50	0/0 0/0	0.00 ± 0.00 0.00 ± 0.00
Sabuncu4	9	95	3.96×10^{28}	CNS/DHNm CNS/BMm	200 50	50 50	0/0 0/0	0.00 ± 0.00 0.00 ± 0.00
Sabuncu6	9	209	8.23×10^{62}	CNS/DHNm CNS/BMm	2000 500	200 200	0/0 0/0	0.00 ± 0.00 0.00 ± 0.00
Sabuncu7	9	168	3.74×10^{50}	CNS/DHNm CNS/BMm	1000 200	200 200	4/0 0/0	0.00 ± 0.00 0.00 ± 0.00
Sabuncu9	9	163	1.17×10^{49}	CNS/DHNm CNS/BMm	300 100	150 150	0/0 0/0	0.00 ± 0.00 0.00 ± 0.00

substantiated in five instances. The experimental results show that both algorithms are capable of solving the Sudoku puzzles effectively, and the algorithm based on Boltzmann machines entails a smaller population size. Further investigations may aim at the parallel implementation of the CNS algorithms to improve their efficiency.

REFERENCES

[1] W. Hong, “Adaptive image data hiding in edges using patched reference table and pair-wise embedding technique,” *Information Sciences*, vol. 221, pp. 473–489, 2013.

[2] S. G. Krishna and T. Moger, “Optimal Sudoku reconfiguration technique for total-cross-tied PV array to increase power output under non-uniform irradiance,” *IEEE Transactions on Energy Conversion*, vol. 34, no. 4, pp. 1973–1984, 2019.

[3] W. Chen, C.-C. Chang, S. Weng, and B. Ou, “Multi-layer mini-Sudoku based high-capacity data hiding method,” *IEEE Access*, vol. 8, pp. 69 256–69 267, 2020.

[4] T. Yato and T. Seta, “Complexity and completeness of finding another solution and its application to puzzles,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 86, no. 5, pp. 1052–1060, 2003.

[5] Z. W. Geem, “Harmony search algorithm for solving Sudoku,” in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2007, pp. 371–378.

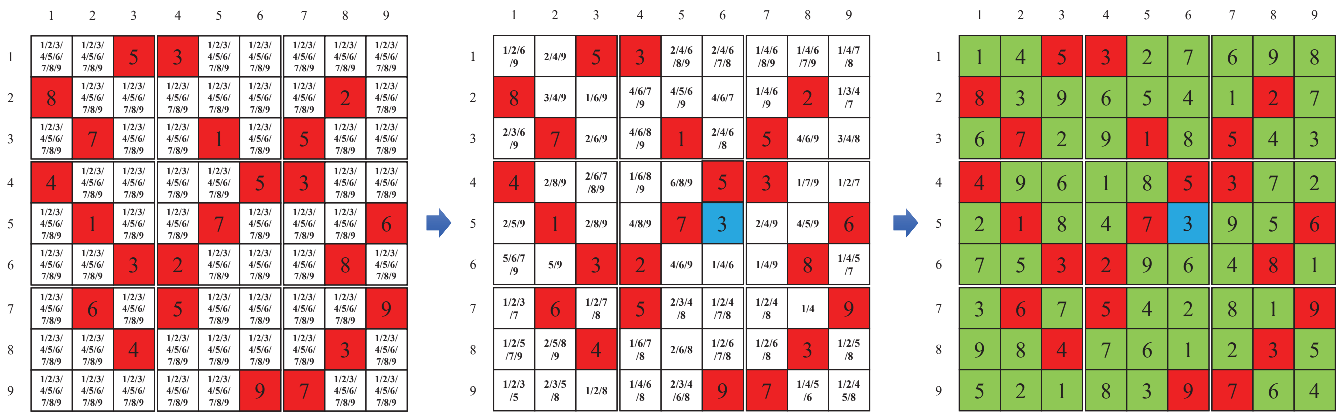


Fig. 6. Feasible results obtained by using the proposed variables reduction and CNS/BMm algorithm on Sabuncu6.

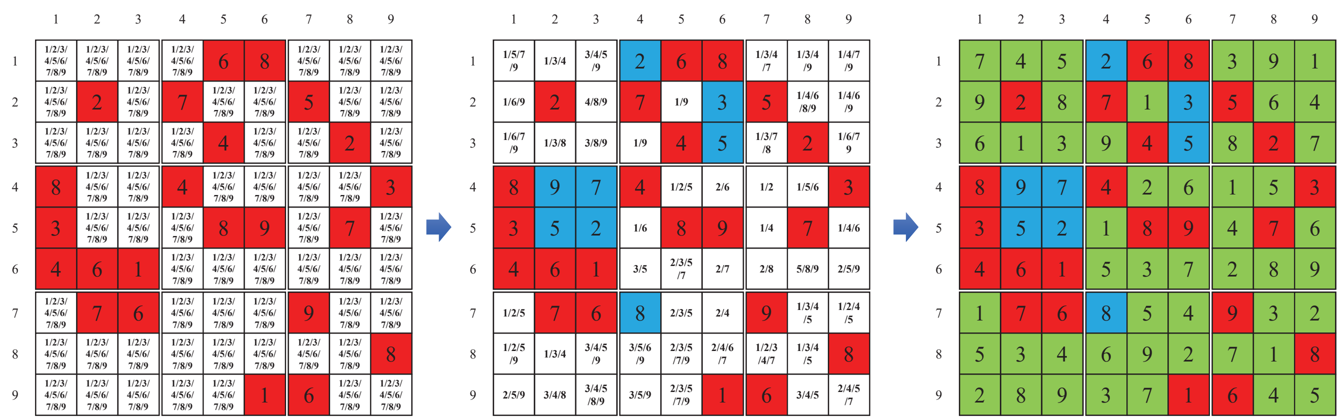


Fig. 7. Feasible results obtained by using the proposed variables reduction and CNS/BMm algorithm on Sabuncu7.

[6] T. K. Moon, J. H. Gunther, and J. J. Kupin, "Sinkhorn solves Sudoku," *IEEE Transactions on Information Theory*, vol. 55, no. 4, pp. 1741–1746, 2009.

[7] J. Méhat and T. Cazenave, "Combining UCT and nested Monte Carlo search for single-player general game playing," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, pp. 271–277, 2010.

[8] J. Gunther and T. Moon, "Entropy minimization for solving Sudoku," *IEEE Transactions on Signal Processing*, vol. 60, pp. 508–513, 2011.

[9] F. Maes, D. L. St-Pierre, and D. Ernst, "Monte carlo search algorithm discovery for single-player games," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 5, no. 3, pp. 201–213, 2013.

[10] L. C. Coelho and G. Laporte, "A comparison of several enumerative algorithms for Sudoku," *Journal of the Operational Research Society*, vol. 65, no. 10, pp. 1602–1610, 2014.

[11] B. Boreland, G. Clement, and H. Kunze, "Set selection dynamical system neural networks with partial memories, with applications to Sudoku and KenKen puzzles," *Neural Networks*, vol. 68, pp. 46–51, 2015.

[12] Y. Jin and J.-K. Hao, "Solving the Latin square completion problem by memetic graph coloring," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 1015–1028, 2019.

[13] T. Mantere and J. Koljonen, "Solving, rating and generating Sudoku puzzles with GA," in *2007 IEEE Congress on Evolutionary Computation*. IEEE, 2007, pp. 1382–1389.

[14] P. Malakonakis, M. Smerdis, E. Sotiriades, and A. Dollas, "An FPGA-based Sudoku solver based on simulated annealing methods," in *2009 International Conference on Field-Programmable Technology*. IEEE, 2009, pp. 522–525.

[15] R. Soto, B. Crawford, C. Galleguillos, E. Monfroy, and F. Paredes, "A hybrid ac3-tabu search algorithm for solving Sudoku puzzles," *Expert Systems with Applications*, vol. 40, no. 15, pp. 5817–5821, 2013.

[16] J. M. Hereford and H. Gerlach, "Integer-valued particle swarm optimization applied to Sudoku puzzles," in *2008 IEEE Swarm Intelligence Symposium*, 2008, pp. 1–7.

[17] X. Q. Deng and Y. D. Li, "A novel hybrid genetic algorithm for solving Sudoku puzzles," *Optimization Letters*, vol. 7, no. 2, pp. 241–257, 2013.

[18] H. Lloyd and M. Amos, "Solving Sudoku with ant colony optimization," *IEEE Transactions on Games*, vol. 12, no. 3, pp. 302–311, 2019.

[19] 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.

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

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

[27] A. Hosseini, J. Wang, and S. M. Hosseini, "A recurrent neural network

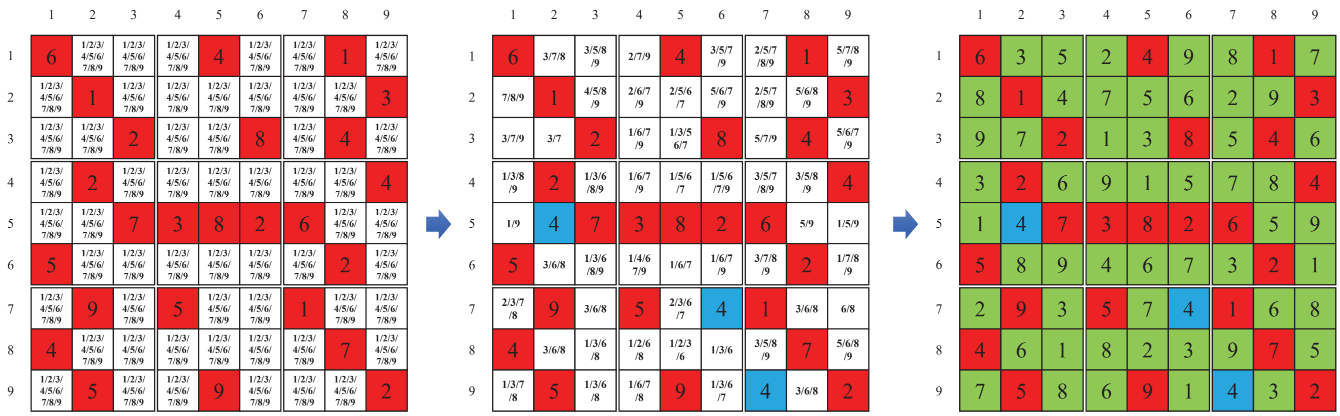


Fig. 8. Feasible results obtained by using the proposed variables reduction and CNS/BMm algorithm on Sabuncu9.

for solving a class of generalized convex optimization problems," *Neural Networks*, vol. 44, pp. 78–86, 2013.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] H. Li and J. Wang, "A collaborative neurodynamic optimization algorithm based on Boltzmann machines for solving the traveling salesman problem," in *2021 11th International Conference on Intelligent Control and Information Processing (ICICIP)*. IEEE, 2021, pp. 325–333.

[35] 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.

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

[37] 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.

[38] 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.

[39] J. Wang and J. Wang, "Two-timescale multilayer recurrent neural networks for nonlinear programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 1, pp. 37–47, Jan. 2022.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[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*, vol. 32, no. 12, pp. 5274–5286, Dec. 2021.

[46] 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.

[47] J. Wang, J. Wang, and Q. Han, "Neurodynamics-based model predictive control of continuous-time under-actuated mechatronic systems," *IEEE/ASME Transactions on Mechatronics*, no. 1, pp. 311–321, Jan. 2021.

[48] M.-F. 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.

[49] 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*, vol. 32, 2022, in press.

[50] Y. Wang, X. Li, and J. Wang, "A neurodynamic optimization approach to supervised feature selection via fractional programming," *Neural Networks*, vol. 136, pp. 194–206, Apr. 2021.

[51] 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, 2017.

[52] H. Che, J. Wang, and A. Cichocki, "Bicriteria sparse nonnegative matrix factorization via two-timescale duplex neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, 2022, in press.

[53] 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*, 2022, in press.

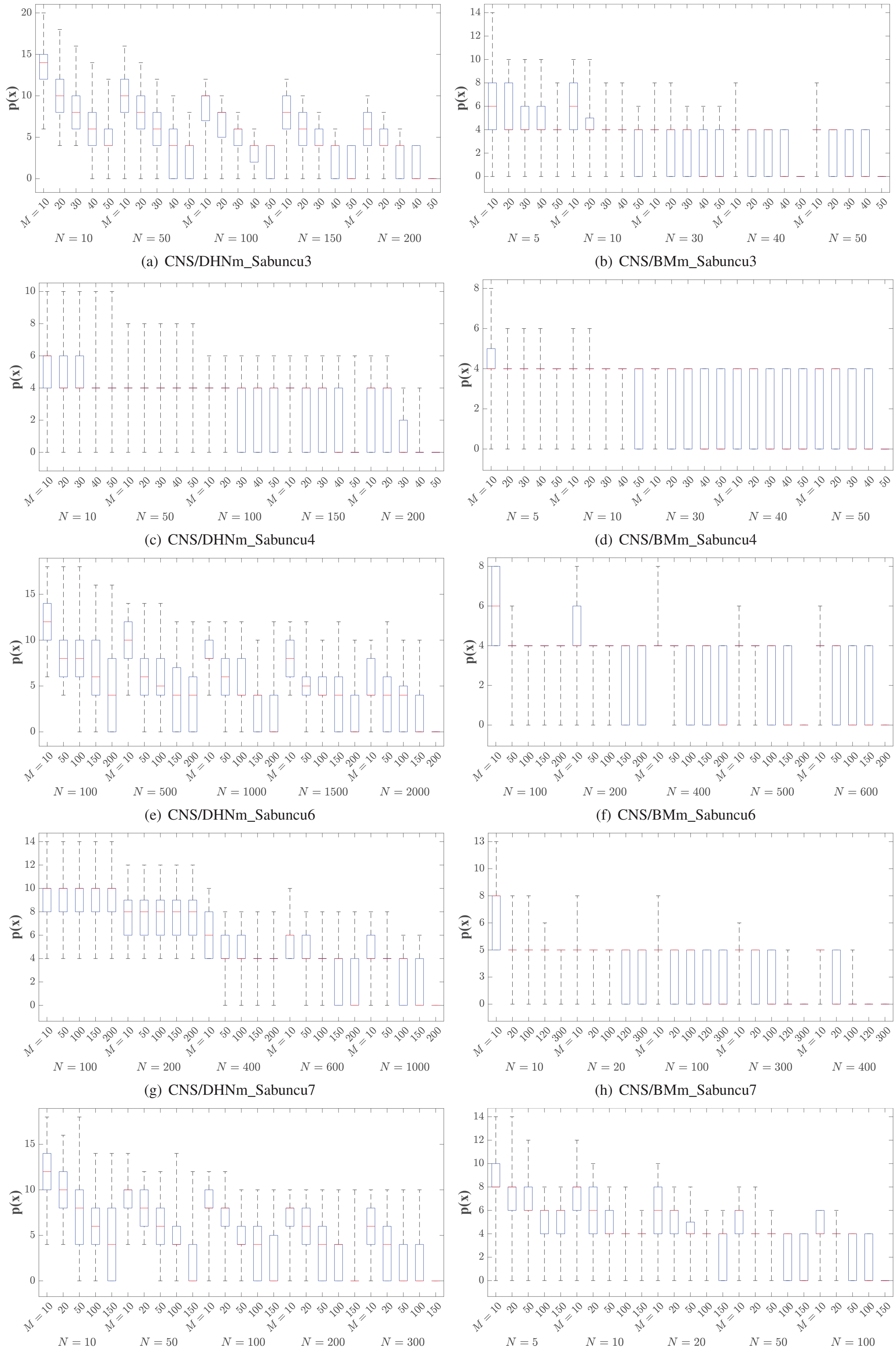
[54] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[55] Y. Takefuji and K.-C. Lee, "A near-optimum parallel planarization algorithm," *Science*, vol. 245, no. 4923, pp. 1221–1223, 1989.

[56] Y. Takefuji and K. C. Lee, "Artificial neural networks for four-coloring map problems and k-colorability problems," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 3, pp. 326–333, 1991.

[57] G. Galán-Marín and J. Muñoz-Pérez, "Design and analysis of maximum Hopfield networks," *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 329–339, 2001.

[58] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.



(a) CNS/DHNm_Sabuncu3 (b) CNS/BMm_Sabuncu3

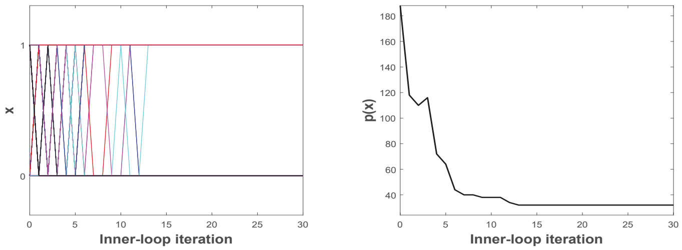
(c) CNS/DHNm_Sabuncu4 (d) CNS/BMm_Sabuncu4

(e) CNS/DHNm_Sabuncu6 (f) CNS/BMm_Sabuncu6

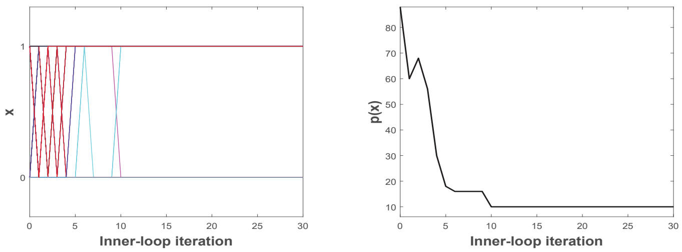
(g) CNS/DHNm_Sabuncu7 (h) CNS/BMm_Sabuncu7

(i) CNS/DHNm_Sabuncu9 (j) CNS/BMm_Sabuncu9

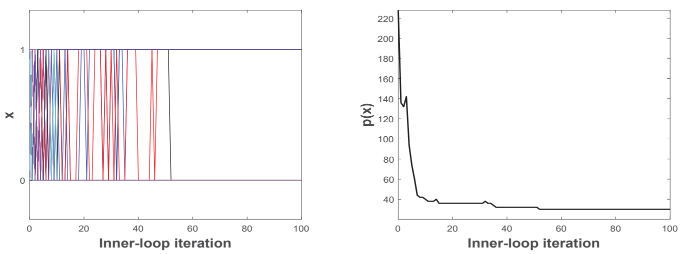
Fig. 9. Monte Carlo test results obtained by using the CNS algorithms with several M and N on the five instances.



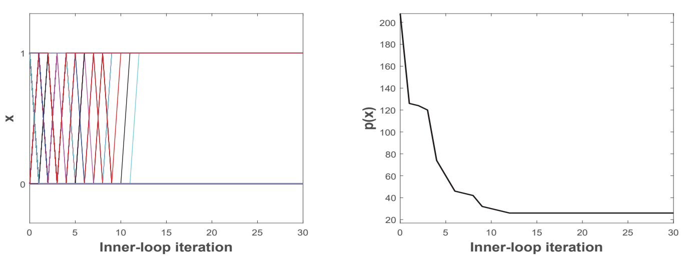
(a) Sabuncu3



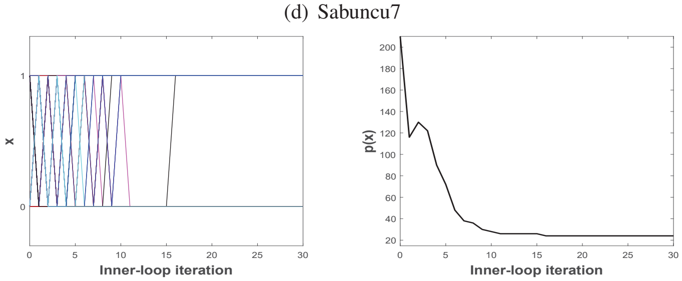
(b) Sabuncu4



(c) Sabuncu6

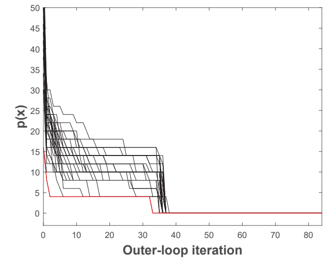


(d) Sabuncu7

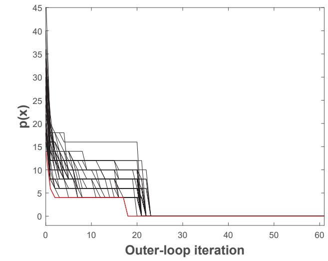


(e) Sabuncu9

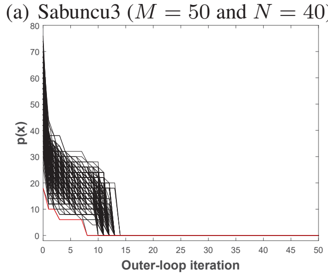
Fig. 2. Snapshots of neuronal states, and penalty function value of in the CNS/BMm algorithm.



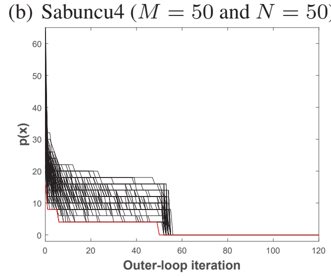
(a) Sabuncu3 ($M = 50$ and $N = 40$)



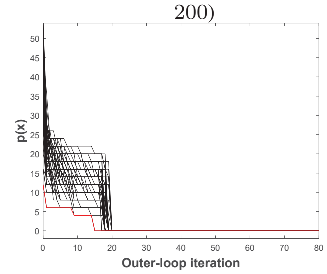
(b) Sabuncu4 ($M = 50$ and $N = 50$)



(c) Sabuncu6 ($M = 200$ and $N = 500$)



(d) Sabuncu7 ($M = 200$ and $N = 200$)



(e) Sabuncu9 ($M = 150$ and $N = 100$)

Fig. 3. The convergent behavior of the CNS/BMm algorithm in the five instances.

[59] 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.

[60] A. Bartlett, T. P. Chartier, A. N. Langville, and T. D. Rankin, "An integer programming model for the Sudoku problem," *Journal of Online Mathematics and its Applications*, vol. 8, no. 1, 2008.

[61] I. Sabuncu, "Work-in-progress: solving Sudoku puzzles using hybrid ant colony optimization algorithm," in *2015 1st International Conference on Industrial Networks and Intelligent Systems*. IEEE, 2015, pp. 181–184.