A Collaborative Neurodynamic Algorithm for Quadratic Unconstrained Binary Optimization

Hongzong Li and Jun Wang, *Life Fellow, IEEE*

***Abstract*—Quadratic unconstrained binary optimization (QUBO) is a typical combinatorial optimization problem with widespread applications in science, engineering, and business. As QUBO problems are usually NP-hard, conventional QUBO algorithms are very time-consuming for solving large-scale QUBO problems. In this paper, we present a collaborative neurodynamic optimization algorithm for QUBO. In the proposed algorithm, multiple discrete Hopfield networks, Boltzmann machines, or their variants are employed for scattered searches, and a particle swarm optimization rule is used to re-initialize neuronal states repeatedly toward global optima. With extensive experimental results on four classic combinatorial optimization problems, we demonstrate the efficacy and potency of the algorithm against several prevailing exact and meta-heuristic algorithms.**

***Index Terms*—Quadratic unconstrained binary optimization, combinatorial optimization, discrete Hopfield network, collaborative neurodynamic optimization, Boltzmann machine.**

I. Introduction

Quadratic unconstrained binary optimization (QUBO) problems, also known as unconstrained binary quadratic programs, are a major class of combinatorial optimization problems, most of which are NP-hard [1]. QUBO problems appear in a variety of application areas, such as quantum computing [2], [1], graph-cut optimization [3], [4], [5], video segmentation [6], visual recognition [7], data hashing [8], human tracking [9], vehicle routing [10], network flow optimization [11], material composition optimization [12], magnet array optimization [13], model predictive control [14], unit commitment [15], power network reconfiguration [16], scientific computing [17].

Existing QUBO solution methods can be classified as exact methods, approximate methods, heuristic and meta-heuristic methods, and hardware-based methods. Specifically, exact methods include branch and cut methods [18], branch and bound methods [19], mixed-integer quadratic programming solvers [20], etc. Approximation methods include max-flow approaches [21], quadratic convex reformulation methods [22], Lagrangian relaxation approaches [23], semidefinite programming methods [24], physics-inspired graph neural networks [25], mean-field approximation [26], stochastic neighborhood search algorithms [27], etc. In the view that exact algorithms are time-consuming or ineffective for large-scale optimization, they may be unable to obtain satisfactory solutions in given short periods of time. In addition, because of their approximation nature, the approximate methods cannot guarantee solution optimality or even feasibility. Heuristic and meta-heuristic methods include conditional simulated annealing [28], tabu search methods [29], local search heuristics [30], Markov chain search [31], evolutionary algorithms [32], recurrent neural networks [33], [34], quantum-inspired heuristic solver [35], deep reinforcement learning [36], etc. Hardware-based methods are based mainly on the D-Wave Quantum Annealer and Fujitsu's Digital Annealer; e.g., [5], [37], [14], [38], etc.

In his seminal papers, John Hopfield points out that recurrent neural networks can collectively serve as powerful computational models [39]. Specifically, the Hopfield networks are developed for linear programming and combinatorial optimization [40]. Since then, numerous neurodynamic optimization models are developed [41] for linear and nonlinear programming [42], [43], non-smooth optimization [44], [45], generalized convex optimization [46], minimax optimization [47], distributed optimization [48], [49], bi-level optimization [50], combinatorial optimization [51], [33], [52], and sparse optimization [53].

As an individual neurodynamic model is prone to be trapped in local minima, a hybrid intelligence framework called collaborative neurodynamic optimization (CNO) is developed for solving global optimization problems [54]. In the CNO approach, multiple neurodynamic models are leveraged for scattered searches, and a metaheuristic rule is used for the re-initialization of the neurodynamic models upon their local convergence. It is proven to be almost surely convergent to global optima of global optimization problems [54], [34]. In addition, CNO approaches are extended for multi-objective optimization [49], combinatorial optimization [34], and mixed-integer optimization [55].

In this paper, we propose a QUBO algorithm in the CNO framework (called CNO-QUBO). The CNO-QUBO algorithm employs multiple discrete Hopfield networks, or Boltzmann machines for scattered searches and re-initializes the neuronal states using a particle swarm optimization rule. We demonstrate the almost-sure global convergence of the proposed algorithm and its superior performance against several prevailing exact and meta-heuristic algorithms in terms of accuracy and precision with respect to optimal solutions. The salient features of the CNO-QUBO algorithm are summarized as follows.

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.

H.-Z. Li and J. Wang are with the Department of Computer Science, City University of Hong Kong, Hong Kong. J. Wang is also with the School of Data Science, City University of Hong Kong, Hong Kong (emails: hongzli2-c@my.cityu.edu.hk, jwang358@cityu.edu.hk).

i. CNO-QUBO is able to handle a variety of constrained quadratic binary problems by reformulating them as QUBO problems via the penalization of constraint violation.
ii. CNO-QUBO leverages the stochastic optimization capability of Boltzmann machines for scattered searches and the gradient-free updating feature of a particle swarm optimization rule to reposition the neuronal searches away from local minima.
iii. CNO-QUBO is experimentally demonstrated to outperform several prevailing exact and metaheuristic algorithms in terms of solution quality and consistency.

The remaining parts of the paper are structured as follows. In Section II, the preliminaries about neurodynamic optimization are introduced. In Section III, the problem formulation and reformulation are described. In Section IV, the proposed CNO approach to QUBO is described. In Section V, experimental results for solving four classic combinatorial problems are elaborated. In Section VI, concluding remarks are given.

II. Preliminaries

In this section, we provide background knowledge about neurodynamic optimization to facilitate the understanding of the results.

A. Neurodynamic Optimization

1) Discrete Hopfield Network: The discrete Hopfield network (DHN) exemplifies a recurrent neural network characterized by its binary or bipolar states and hard-limiter activation function [56]. Let W denote the neuron connection weight matrix, and θ denote the neuron bias vector. DHN operates in discrete time as follows:

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

where $u(t)$ is the net-input vector at the t-th iteration, $x(t)$ is the neuronal state vector at the t-th iteration, and $\sigma(\cdot)$ is a vector-valued hard-limiter activation function to determine the state of the $(t+1)$-th iteration based on the net input. The activation function is defined element-wise as:

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

meaning that the state is reset to zero if the net input is nonpositive or set to one otherwise.

As proved in [56], the global stability of DHN (1) is ensured at an equilibrium state $\bar{x}$ (i.e., $\lim_{t\to\infty} x(t) = \bar{x}$) under the following conditions: the symmetry of the weight matrix (i.e., $W = W^T$), the zero diagonal elements of the weight matrix (i.e., $w_{ii} = 0$, $\forall i$), and asynchronous activation (i.e., only one neuron is activated at a time, rather than all at the same time).

It is demonstrated in [56] that the DHN is globally convergent to a local minimum of the following combinatorial optimization problem:

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

It is worth noting that the states of the DHN are determined solely by the sign of the negative gradient of the objective function in (2) without being influenced by any historical effect.

If W in (2) is not symmetric, an equivalent approach is to replace it with $(W + W^T)/2$. In the view that the binary variables have $x_i^2 = x_i, \quad i = 1, 2, \ldots, n$, a linear term $\mathrm{diag}(w_{11}, \ldots, w_{nn})x$ is added to realize the zero diagonal elements of W.

Activating the DHN asynchronously entails a prolonged time for its convergence. For W with some special properties, synchronous activation in batches may expedite convergence. Specifically, several methods are developed to activate neuronal states synchronously in batches; e.g., [57], [58], [59], [60]. For example, the DHN is still convergent to a local minimum if the neurons without any direct connections are activated synchronously in batches [60].

A DHN with a momentum term (DHNm) is introduced [61] with the following neurodynamic equation:

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

DHNm (3) takes historical effects into account and enriches its dynamic behaviors by including the momentum term $u(t-1)$. It has been demonstrated that the synchronously activated neuronal states of DHNm (3) are convergent to a local optimum of (2) [62], [63].

2) Boltzmann Machine: The Boltzmann machine (BM) [64] is a stochastic version of DHN based on simulated annealing [65] for minimizing (2). Different from the DHN, the activation in the BM is carried out according to probability:

$$\begin{cases} p(x_i(t+1) = 1) = \dfrac{1}{1 + \exp(-\frac{u_i(t)}{T})}, \\ p(x_i(t+1) = 0) = 1 - p(x_i(t+1) = 1), \end{cases} \tag{4}$$

where T is the temperature parameter. T decrease gradually over time following a cooling schedule [66] defined as follows:

$$T(t) = T_0 \alpha^t, \tag{5}$$

where T_0 denotes the initial temperature parameter and α denotes a cooling rate parameter in the range of $(0, 1)$. It is known that if T is sufficiently large, and the cooling schedule is sufficiently long, BM is demonstrated almost surely convergent to global optima [67], [68].

In analogy to DHNm (3), a BM with a momentum term (BMm) [69] is expressed as:

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

B. Collaborative Neurodynamic Optimization

A CNO approach consists of two levels: In the lower level, multiple neurodynamic optimization models are employed for scattered searches. Various recurrent neural may be used. In the existing CNO paradigms, projection neural networks [70], [71], [72] and discrete Hopfield networks (1) [73], [74], [75] are often used. In the higher level, a gradient-free rule is used for state initialization. Various rules in existing metaheuristic algorithms may be used, e.g., the particle swarm optimization algorithm [70], [71], [72], [73], [74], [69], shuffled frog leaping algorithm [76], [77], the compressed coding scheme [78], or others [79], [80]. The particle swarm optimization rule is used in almost all of the CNO algorithms to reposition the initial states of the neurodynamic models. Among various PSO rules, the von Neumann topology stands out as an effective and well-studied variant [81]. In this topology, particles are organized in a grid-like structure, forming a lattice of interconnected neighborhoods. Let p_i^* denote the best position found by the i-th particle individually, p_i denote the position vector of the i-th particle, l_i^* denote the best neighbor of the i-th particle on all four sides of the two-dimensional lattice, and N denote the number of particles. The velocity v_i and the position p_i, for $i = 1, 2, \dots, N$, are updated as follows:

$$\begin{cases} v_i(t+1) = c_0 v_i(t) + c_1 r_1 (p_i^*(t) - p_i(t)) + \\ \qquad\qquad c_2 r_2 (l_i^*(t) - p_i(t)), \\ \text{if } (r_3 < S(v_{id}(t))), \text{then } p_{id}(t) = 1, \text{else } p_{id}(t) = 0, \end{cases} \tag{7}$$

where c_0 is an inertia parameter, c_1, c_2 are two acceleration constants, $r_1, r_2, r_3 \in [0, 1]$ are three random numbers, and $S(\cdot)$ is a sigmoid limiting transformation.

The diversity of global search is non-negligible in global and combinatorial optimization in the presence of convexity in objective functions or solution spaces. A simple diversity measure is defined as:

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

where n is the dimension of solutions, and p^* is the best solution among the N solutions.

In the literature, many mutation operators are used to ensure solution diversity. In particular, the following bit-flip mutation operation is defined in [82]: if $\delta(x) < \epsilon$, then

$$x_j = \begin{cases} \neg x_j & \text{if } \ \xi_j \le P_{mut}, \\ x_j & \text{otherwise }, \end{cases} \tag{9}$$

where ϵ is a threshold, $\bar{x}_j$ is the negation of x_j, ξ_j is a randomly generated number in the range of $[0, 1]$, and P_{mut} is a mutation probability.

CNO works as a computationally intelligent optimizer in a variety of applications, including vehicle-task assignment [71], hash bit selection [74], model predictive control [83], Boolean matrix factorization [84], binary matrix factorization [75], portfolio selection [85], sparse signal reconstruction [86], etc.

III. Problem Formulation

Numerous combinatorial optimization problems may be reformulated in a QUBO form. Consider a constrained quadratic binary optimization problem with a quadratic pseudo-Boolean objective function and linear constraints in the following form:

$$\min \ x^T Q x + q^T x, \tag{10a}$$
$$\text{s.t. } Ax = b, \tag{10b}$$
$$Cx \le d, \tag{10c}$$
$$x \in \{0, 1\}^n. \tag{10d}$$

where $Q \in^{n \times n}$, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{p \times n}$, and $d \in \mathbb{R}^p$.

The reformulation starts with defining a nonnegative penalty function to satisfy the constraints, such that the penalty function value is equal to zero for feasible solutions or it is positive for infeasible solutions. Problem (10) may be reformulated in a QUBO form by superposing a quadratic penalty function into the objective function. If the penalty function value decreases to zero, the augmented objective function degenerates to the original objective function.

To handle equality constraints (10b), a quadratic penalty term is defined below:

$$p_c(x) = \frac{1}{2} \sum_{i=1}^{m} \big(\sum_{j=1}^{n} a_{ij} x_j - b_i \big)^2 = \frac{1}{2} \|Ax - b\|_2^2. \tag{11}$$

There are two common ways to convert inequality constraints in (10c) to corresponding penalty terms: Some common inequality constraints (e.g., $x \le y$) can be directly formulated to corresponding penalty terms by using some existing transformation techniques without adding slack variables [87]. For example, the constraint $x \le y$ can be converted to the penalty term $P(x - xy)$ [1]. General inequality constraints can be converted to equalities by adding binary slack variables weighted by a series of integral powers of 2. As the left-hand side of each inequality constraint in (10c) is always an integer, by introducing a binary slack variable $y_{ik} \in \{0, 1\}$ for constraint i with $k = 0, ..., \lfloor \log_2(d_i) \rfloor$ to be expanded as an integer to fill the gap. In addition to the above ways, a penalty term for the inequality constraint in (10c) is defined by using a rectified activation function [69]:

$$p_b(x) = \frac{1}{2} \sum_{l=1}^{p} \Big(\max\{0, \sum_{i=1}^{n} c_{li} x_i - d_l\} \Big)^2. \tag{12}$$

A penalty function $p(x)$ and a penalized objective function $f_\rho(x)$ are defined by including the two quadratic penalty terms in (11) and (12), and a positive penalty parameter ρ:

$$p(x) = p_b(x) + p_c(x), \tag{13}$$
$$f_\rho(x) = f(x) + \rho p(x). \tag{14}$$

With (14), the original problem in (10) is reformulated as a QUBO problem:

$$\min_x \ f_\rho(x), \tag{15}$$
$$\text{s.t. } x \in \{0, 1\}^n.$$

It is well known that the optimal solution to (10) is the same as that to (15), provided that the penalty parameter ρ is sufficiently large [87]. It is worth pointing out that the QUBO problem is quite similar to the Ising model that is an archetypical Markov random field. The only difference lies in their different discrete variables (i.e., $\{0,1\}^n$ vs. $\{-1,1\}^n$), and straightforward linear translations can convert between them [88]. A Python library is available for formulating QUBO and Ising models [89].

In view of the different parametric ranges of different QUBO problems, to avoid numerical imbalance in optimization, it is better to normalize the parameter matrices and vectors in a given QUBO problem. In this paper, we normalize the problem parameters as follows. Let $Q_{\max}$ denote the maximal absolute value of the elements in Q and $2q^T$, $\hat{A}_{\max}$ denotes the maximal absolute value of the elements in A^TA and $2A^Tb$, and $\hat{C}_{\max}$ denotes the maximal absolute value of the elements in C^TC and $2C^Td$. $\bar{Q} = (Q+Q^T)/Q_{\max}, \bar{q} = q/Q_{\max}, \bar{A} = A^TA/\hat{A}_{\max}, \bar{b} = A^Tb/\hat{A}_{\max}$, $\bar{C} = C^TC/\hat{C}_{\max}$, and $\bar{d} = C^Td/\hat{C}_{\max}$. As a result of the parameter normalization, the penalty parameter ρ may be set in the order of 10's.

IV. CNO-QUBO Algorithms

A. Algorithm Description

In this section, we describe the CNO algorithm termed CNO-QUBO for solving the normalized QUBO in (15). Fig. 1 shows a schematic diagram of the CNO-QUBO algorithm. As shown in Fig. 1, CNO-QUBO is structured in two levels: a lower level and an upper level. In the lower level, a population of DHNs (1), DHNm's (3), BMs (4), or BMm's (6) with different initial states are leveraged to carry out scattered searches for optimal solutions. In the upper level, PSO rule (7) is used for state reinitialization upon their local convergence to reposition the scattered searches away from local optima at more promising points.

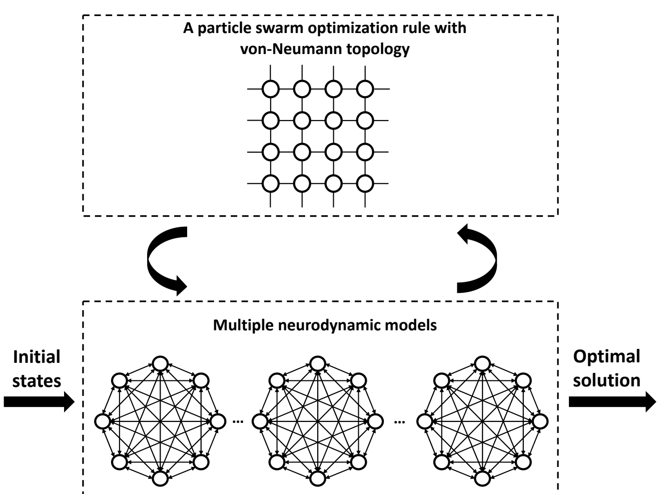


Fig. 1. A schematic diagram of the CNO-QUBO algorithm.

Let CNO-QUBO/DHN, CNO-QUBO/DHNm, CNO-QUBO/BM, and CNO-QUBO/BMm denote CNO-QUBO with DHN, DHNm, BM, and BMm, respectively. Algorithm 1 details the CNO-QUBO/BMm algorithm. Specifically, steps 1 - 29 span the outer loop for global search, steps 2 - 14 are the inner loop to perform scatter search by using the BMm's, steps 16 - 19 are to identify the best BMm in the population, steps 22 - 24 are to update the states of BMm's using the PSO rule in (7), and steps 26 - 28 are to perform mutation operation.

Algorithm 1: CNO-QUBO/BMm

Input: N, M, Initial states $[x^{(1)}(0), ..., x^{(N)}(0)]$, initial incremental vector $[v^{(1)}(0), ..., v^{(N)}(0)]$, individual/group best solutions $x^{(i)}$/ x^*, $\bar{f}_\rho(x^{(i)}) = \bar{f}_\rho(x^*) = \infty$, initial temperature T_0, mutation threshold ϵ, PSO parameters c_0, c_1 and c_2.

Output: x^*.

1 **while** $m \le M$ **do**
2   **for** $i = 1$ **to** N **do**
3     $T \leftarrow T_0$;
4     $t \leftarrow 0$;
5     $u^{(i)}(0) \leftarrow 2x^{(i)}(0) - 1$;
6   **while** $\tau \le p(x^{(i)}(t+1) = 1) \le 1 - \tau \wedge \mathrm{sign}(u^{(i)}(t)) \ne \mathrm{sign}(Wx^{(i)}(t) - \theta)$ **do**
7       $T \leftarrow T_0\alpha^t$;
8   Update $x^{(i)}(t+1)$ according to (6) with $u^{(i)}(t+1)$ and T;
9       $t \leftarrow t+1$;
10   **end**
11   **if** $f_\rho(\bar{x}^{(i)}) < f_\rho(x^{(i)})$ **then**
12       $x^{(i)} \leftarrow \bar{x}^{(i)}$;
13   **end**
14   **end**
15   $\hat{x} = \arg\min_{\hat{x}}\{f_\rho(x^{(1)}), ..., f_\rho(x^{(i)}), ..., f_\rho(x^{(N)}\}$;
16   **if** $f_\rho(\hat{x}) < f_\rho(x^*)$ **then**
17     $x^* \leftarrow \hat{x}$;
18     $m \leftarrow 0$;
19   **else**
20     $m \leftarrow m+1$;
21   **end**
22   **for** $i = 1$ **to** N **do**
23   Update velocity and initial neuronal state according to (7);
24   **end**
25   Compute $\delta(q)$ according to (8);
26   **if** $\delta(q) < \epsilon$ **then**
27   Perform the bit-flip mutation according to (9);
28   **end**
29 **end**
30 **return** x^*.

Besides the BMm, the DHN in (1), the DHNm in (3), and the BM in (4) may also be used in the CNO-QUBO algorithm. Algorithm 2 provides the pseudocodes for clustering indirectly connected neurons for partially synchronous activation in batches based on the idea in [60]. Algorithm 3 details a procedure of the batch-mode neuronal activation for the DHN

Algorithm 2: Clustering DHN or BM neurons into batches

```
Input: The DHN or BM connection weight matrix W,
       the number of neurons n.
Output: Batches of neurons B.
for i = 1 : n do
    if i = 1 then
        Add i to batch 1;
        The number of batches ← 1
    else
        for ℓ = 1 : the number of batches do
            FeasibleFlag ← 1;
            for the element j in batch ℓ do
                if weight w_ij ≠ 0 then
                    FeasibleFlag ← 0;
                end
            end
            if FeasibleFlag= 1 then
                Add i to batch ℓ;
                AddFlag ← 1;
                break
            end
        end
    end
    if AddFlag = 0 then
        Add i to a new batch;
        The number of batches ← The number of
         batches + 1;
    else
        AddFlag ← 0;
    end
end
return B.
```

Algorithm 3: DHN/BM activation in batches

```
   Input: W, θ, batch index B.
   Output: Equilibrium x̄ of DHN or BM.
1  while StableFlag = 1 do
2      StableFlag ← 0
3      Shuffle the batch index
4      for i = 1 : the number of batches do
5          for the elements j in batch i do
6              for DHN do
7                  u_j ← Σ_{k=1}^{n} w_jk x_k − θ_j;
8                  if x_j ≠ σ(u_j) then
9                      x_j ← σ(u_j);
10                     StableFlag ← 1;
11                 end
12             end
13             for BM do
14                 p_j ← 1 / (1+exp(−ΔE_i/T));
15                 Generate a random number γ
16                 if γ < p_j then
17                     if x_j = 0 then
18                         StableFlag ← 1
19                     end
20                     x_j ← 1;
21                 else if γ > p_j then
22                     if x_j = 1 then
23                         StableFlag ← 1
24                     end
25                     x_j ← 0;
26                 end
27             end
28         end
29     end
30 end
31 return x̄.
```

and BM, where the batch index is randomly shuffled at Step 3 in every iteration to enhance the DHN or BM search diversity, as in [74]. The CNO-QUBO algorithm with a population of the DHNs and BMs (termed as CNO-QUBO/DHN and CNO-QUBO/BM, respectively) can be implemented by replacing Steps 3 - 8 in CNO-QUBO/BMm with Algorithms 2 and 3. The DHNm's (termed as CNO-QUBO/DHNm) can be implemented by deleting steps 3 and 7 and replacing BMm in (6) in step 8 in CNO-QUBO/BMm with DHNm in (3).

In addition, the CNO-QUBO algorithm may be easily extended for the Ising model by using the DHN and DHNm with the bipolar hard-limiter activation function [56].

B. Inner-loop Termination Criteria

In the inner loop of the CNO-QUBO/DHN and CNO-QUBO/BM algorithms (i.e., the execution of DHN and BM), a termination criterion is to check whether the neuronal states between two consecutive iterations become unchanged, as implemented in Algorithm 3.

In the inner loop of the CNO-QUBO/DHNm and CNO-QUBO/BMm algorithms (i.e., the execution of DHNm and BMm), the following termination criteria are proposed to check the convergence of DHNm's and the stochastic convergence of BMm's as follows: Let $\mathrm{sign}(\cdot) \in \{-1, 1\}$ be a sign operator. If $\mathrm{sign}(u(t)) = \mathrm{sign}(Wx(t) - \theta)$, then the sum of $u(t)$ and $Wx(t) - \theta$ have the same sign as $u(t)$; i.e.,

$$\mathrm{sign}(u(t)) = \mathrm{sign}(u(t) + Wx(t) - \theta).$$

In view of $\sigma(u) = (\mathrm{sign}(u) + 1)/2$,

$$\frac{\mathrm{sign}(u(t)) + 1}{2} = \frac{\mathrm{sign}(u(t) + Wx(t) - \theta) + 1}{2},$$
$$\sigma(u(t)) = \sigma(u(t) + Wx(t) - \theta). \tag{16}$$

In DHNm's, as in (3),

$$x(t+1) = \sigma(u(t+1)) = \sigma(u(t) + Wx(t) - \theta). \tag{17}$$

Substituting (16) into (17), we have $x(t+1) = \sigma(u(t)) = x(t)$. Therefore, if $\mathrm{sign}(u(t)) = \mathrm{sign}(Wx(t) - \theta)$, then DHNm converges.

In BMm's, for a given small constant τ, if $p(x(t+1) = 1) < \tau$ or $p(x(t+1) = 1) > 1 - \tau$, then as in (6),

$$x(t+1) \approx \sigma(u(t+1)) = \sigma(u(t) + Wx(t) - \theta). \tag{18}$$

Substituting (16) into (18), we have $x(t+1) \approx \sigma(u(t)) = x(t)$. Therefore, if $(p(x(t+1)=1) < \tau \ \vee \ p(x(t+1)=1) > 1-\tau)$ and $\mathrm{sign}(u(t)) = \mathrm{sign}(Wx(t)-\theta)$, then the convergence takes place with probability $1-\tau$.

C. Hyper-Parameter Selection

In CNO-QUBO, there are two hyper-parameters: N (the DHN, DHNm, BM, or BMm population size) and M (the CNO-QUBO termination criterion in the outer loop). Determining these two hyper-parameters is usually carried out in an ad hoc manner, as their values depend on the complexity of the problem. Generally, the sufficiently large values of N and M result in fast and almost-sure convergence of CNO-QUBO to global optima.

V. BENCHMARK EXPERIMENTS

In this section, we elaborate on the results of experiments on four test instances of four classic set-theoretic and combinatorial optimization problems to substantiate the efficacy and superiority of the CNO-QUBO algorithm.

In the PSO rule (7), $c_0 = 1$ and $c_1 = c_2 = 2$. In the BMm, $\tau = 0.001$. For performance comparison, the experimental results of the PSO algorithm with the same parameters above and other baseline algorithms are tabulated in the next subsection. The experimental environment is Windows 10 (64-bit), Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz and 64.0GB RAM.

A. Set Partitioning Problem

The set partitioning problem (SPP) is to partition a set of items into a number of subsets so that the total cost of the partition is minimized. It is an NP-hard problem with probably the most widespread applications [90]. SPP is usually formulated as the following 0-1 linear program:

$$\begin{aligned} &\min_x \sum_{j=1}^{n} c_j x_j, \\ &\text{s.t.} \sum_{j=1}^{n} a_{ij} x_j = 1, \quad i = 1, ..., m, \\ &\quad x_j \in \{0,1\}, \quad j = 1, ..., n, \end{aligned}$$

where m is the number of items, n is the number of possible subsets, x_j indicates whether or not subset j is chosen, c_j is the cost coefficient associated with subset j, and a_{ij} is a binary indicator with its value being one if subset j contains item i or zero otherwise.

Consider an SPP instance with $m = 30, n = 100$, and cost coefficients c_js and binary indicators a_{ij}s are randomly generated in [0, 600] and $\{0, 1\}$, respectively. To ensure the existence of feasible solutions, the sparsity of a_{ij}s is kept at 90%.

Fig. 2(a) depicts two snapshots of $f(x)$ in (10a) and $p(x)$ in (13) using an individual BMm in the inner loop of CNO-QUBO/BMm (Steps 6-10). It shows that the objective function reaches equilibrium within 25 iterations, and the penalty function value decreases to zero, demonstrating BMm converges to a feasible solution. Fig. 3(a) illustrates the convergent behavior of CNO-QUBO/BMm with $M = 8$ and $N = 2$. Figs. 4(a)-4(b) illustrate the boxplots of Monte Carlo results on $f(x)$ obtained by using CNO-QUBO/DHNm and CNO-QUBO/BMm with various values of M and N for solving the SPP. The top of the box denotes the upper quartile $q_n(0.75)$, which is the median of the upper half of the result. The bottom of the box denotes the lower quartile $q_n(0.25)$, which is the median of the lower half of the result. The whiskers denote the highest result value and lowest result value. Fig. 4(a) and Fig. 4(b) show that the objective function values always reach their minima in all 100 runs if $M \geq 6$ and $N \geq 8$ by using CNO/DHNm, and $M \geq 8$ and $N \geq 2$

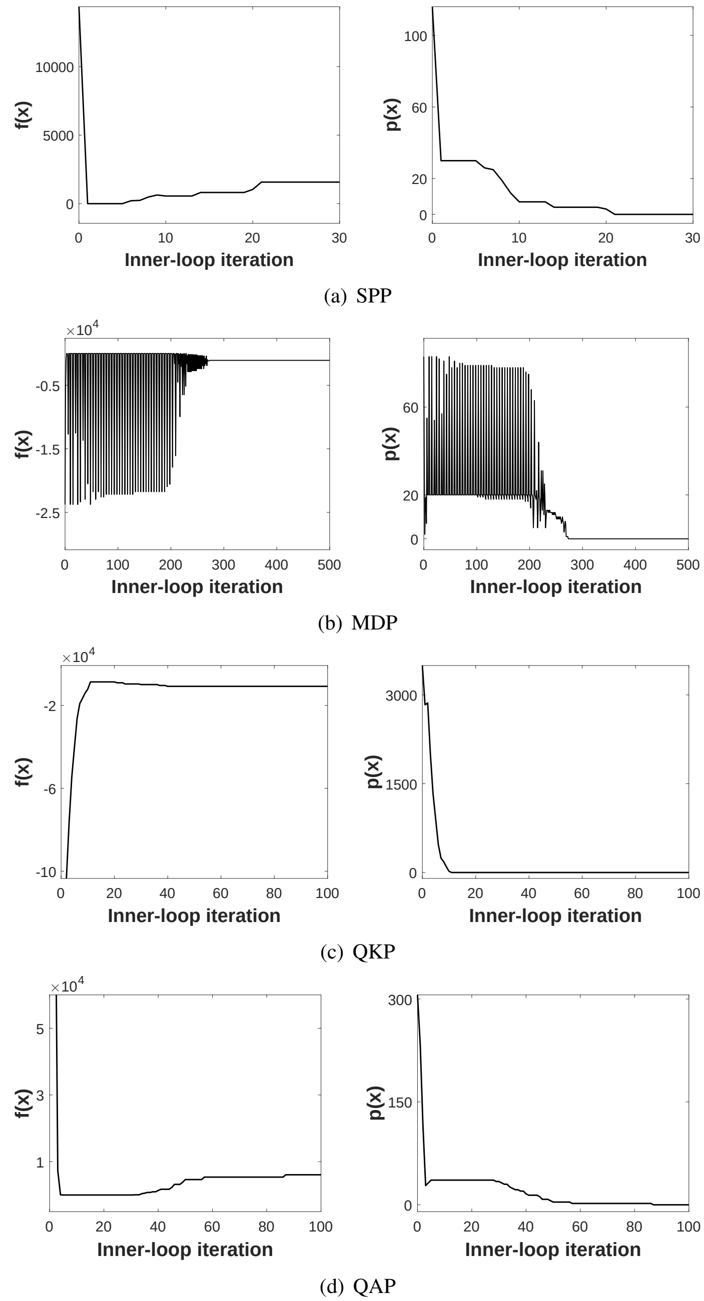


Fig. 2. Snapshots of objective function value of (10a) and penalty function value of (13) in CNO-QUBO/BMm.

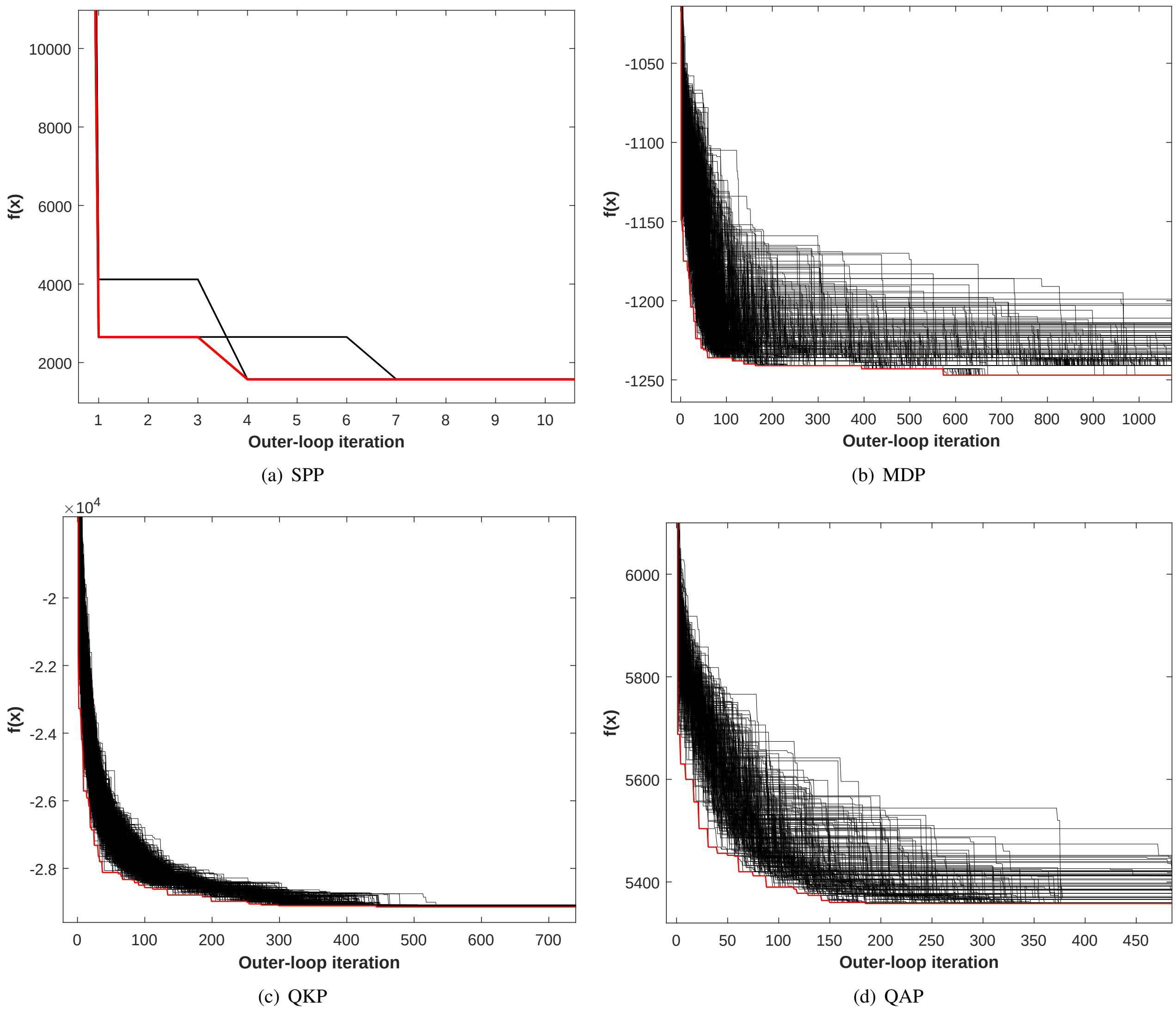


Fig. 3. The convergent behavior of CNO-QUBO/BMm.

by using CNO/BMm.

B. Maximum Diversity Problem

The maximum diversity problem is to select a subset of m elements from n elements that yield the sum of the distances between the chosen elements maximized. MDP is formulated as follows [91]:

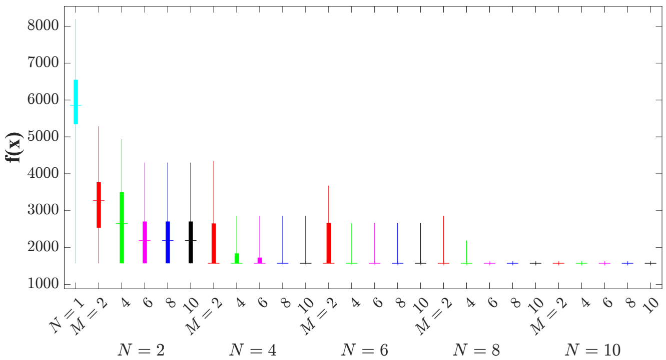
$$\max_x \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j,$$
$$\text{s.t. } \sum_{i=1}^{n} x_i = m,$$
$$x_i \in \{0,1\}, \quad i = 1, ..., n,$$

where d_{ij} is the distance between i and j, and $x_i = 1$ if element i is selected: else, $x_i = 0$. For dataset SOM-b_5_n200_m20 [91], the optimal objective function value is 1247.

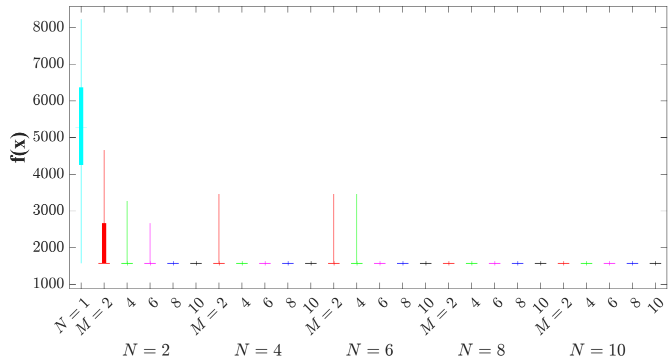
Fig. 2(b) depicts two snapshots of $f(x)$ in (10a) and $p(x)$ in (13) using an individual BMm in the inner loop of CNO-QUBO/BMm (Steps 6-10). It shows that the objective function reaches equilibrium within 300 iterations, and the penalty function value decreases to zero, demonstrating BMm converges to a feasible solution. Fig. 3(b) illustrates the convergent behavior of CNO-QUBO/BMm with $M = 500$ and $N = 1000$. Figs. 4(c)-4(d) illustrate the boxplots of Monte Carlo results on $f(x)$ obtained by using CNO-QUBO/DHNm and CNO-QUBO/BMm with various values of M and N for solving the MDP. Fig. 4(c) and Fig. 4(d) show that the objective function values always reach their minima in all 100 runs if $M \geq 500$ and $N \geq 1500$ by using CNO/DHNm, and $M \geq 500$ and $N \geq 1000$ by using CNO/BMm.

C. Quadratic Knapsack Problem

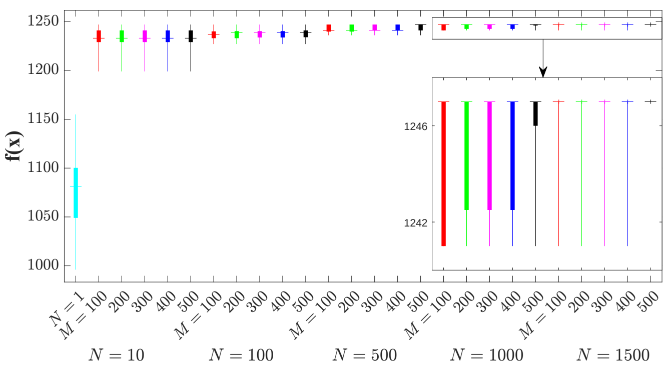
The quadratic knapsack problem (QKP) is to find a subset of items that yields the maximum total value of the items without exceeding given resource capacities.



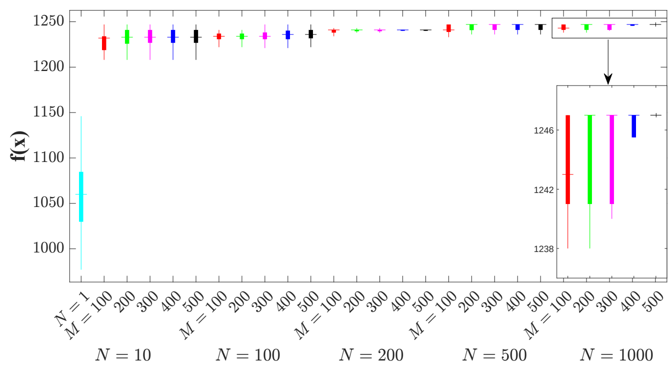
(a) CNO-QUBO/DHNm for solving SPP



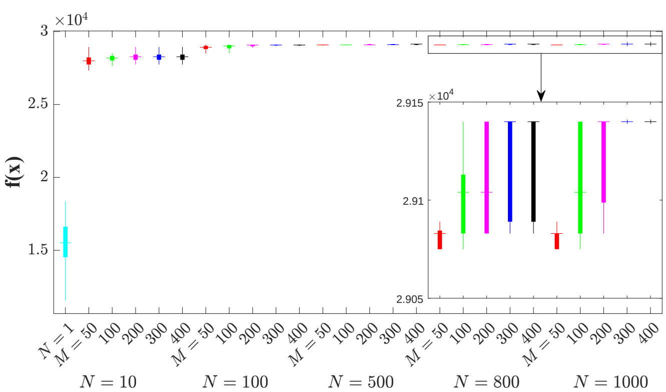
(b) CNO-QUBO/BMm for solving SPP



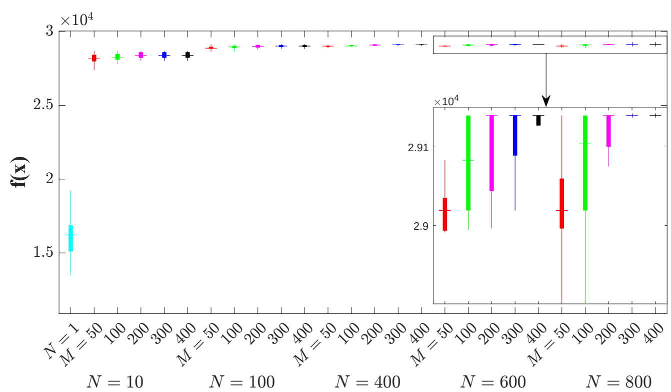
(c) CNO-QUBO/DHNm for solving MDP



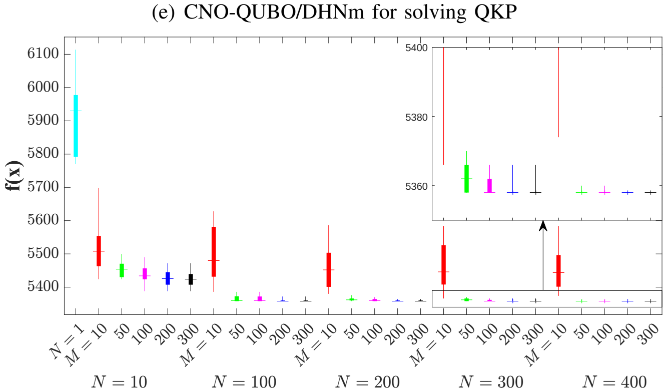
(d) CNO-QUBO/BMm for solving MDP



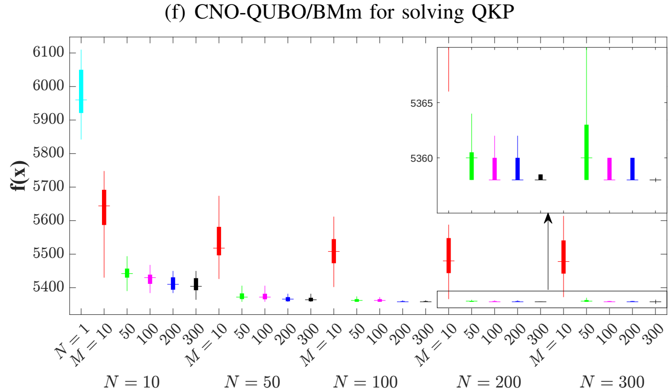
(e) CNO-QUBO/DHNm for solving QKP



(f) CNO-QUBO/BMm for solving QKP



(g) CNO-QUBO/DHNm for solving QAP



(h) CNO-QUBO/BMm for solving QAP

Fig. 4. Monte Carlo results obtained by using the CNO-QUBO/DHNm and CNO-QUBO/BMm.

QKP is formulated as follows [92]:

$$\begin{aligned} \max_{x} & \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_i x_j + \sum_{i=1}^{n} q_i x_i, \\ \text{s.t.} & \sum_{j=1}^{n} w_j x_j \le c, \\ & x_j \in \{0,1\}, \quad j = 1, ..., n, \end{aligned}$$

where n is the number of items, d_{ij} is the gain achieved if both item i and j are selected, q_i is the gain achieved if item i is selected, w_j is the resource requirement of item j, c is knapsack capacity and $x_j = 1$ if item j is chosen: else, $x_j = 0$. In particular, for a QKP instance, r_300_25_1, with $n = 300$ [92], the optimal objective function value is 29140[1].

Fig. 2(c) depicts two snapshots of $f(x)$ in (10a) and $p(x)$ in (13) using an individual BMm in the inner loop of CNO-QUBO/BMm (Steps 6-10). It shows that the objective function reaches equilibrium within 50 iterations, and the penalty function value decreases to zero, demonstrating BMm converges to a feasible solution. Fig. 3(c) illustrates the convergent behavior of CNO-QUBO/BMm with $M = 300$ and $N = 800$. Figs. 4(e)-4(f) illustrate the boxplots of Monte Carlo results on $f(x)$ obtained by using CNO-QUBO/DHNm and CNO-QUBO/BMm with various values of M and N for solving the QKP. Fig. 4(e) and Fig. 4(f) show that the objective function values always reach their minima in all runs if $M \ge 300$ and $N \ge 1000$ by using CNO/DHNm, and $M \ge 300$ and $N \ge 800$ by using CNO/BMm.

D. Quadratic Assignment Problem

The quadratic assignment problem (QAP) is a prototypical combinatorial optimization problem including many problems as its special cases, such as the traveling salesman problem and graph matching [93], [94]. It seeks to find the optimal assignments of pairs such that the total cost associated with the assignments is minimized [95]:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}, \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, ..., n, \\ & \sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, ..., n, \\ & x_{ij} \in \{0,1\}, \quad i, j = 1, ..., n, \end{aligned}$$

where f_{ik} is the flow of material or information between facilities i and k, d_{jl} is the distance between facilities j and l in the context of logistics.

In this paper, the experiment is conducted on dataset had18 [95]. The optimal objective function value is 5358[2].

Fig. 2(d) depicts two snapshots of $f(x)$ in (10a) and $p(x)$ in (13) using an individual BMm in the inner loop of CNO-QUBO/BMm (Steps 6-10). It shows that the objective function reaches equilibrium within 100 iterations, and the penalty function value decreases to zero, demonstrating BMm converges to a feasible solution. Fig. 3(d) illustrates the convergent behavior of CNO-QUBO/BMm with $M = 300$ and $N = 300$. Figs. 4(g)-4(h) illustrate the boxplots of Monte Carlo results on $f(x)$ obtained by using CNO-QUBO/DHNm and CNO-QUBO/BMm with various values of M and N for solving the QAP. Fig. 4(g) and Fig. 4(h) show that the objective function values always reach their minima in all runs if $M \ge 200$ and $N \ge 400$ by using CNO/DHNm, and $M \ge 300$ and $N \ge 300$ by using CNO/BMm.

E. Performance Comparisons

In this subsection, for comparison, we summarize the experimental results of the CNO-QUBO algorithm with the DHNs, DHNm's, BMs, and BMm's for solving the four classic problems presented in the preceding subsections, along with several state-of-the-art exact and meta-heuristic algorithms such as MINLP, CPLEX, greedy randomized adaptive search procedure (GRASP) [98], tabu search (TS) [29], simulated annealing (SA) [28], genetic algorithm (GA) [97], PSO algorithm [102], iterative greedy algorithm (IG) [99], ant colony optimization [105] and grey wolf optimizer (GWO) [106]. The sixth and seventh column boxes in Table I record the averaged results over 100 runs in terms of the best, worst, mean, and standard deviation of objective function values, the total number of iterations in the population, and CPU time in seconds on a PC in the MATLAB environment. In the CNO-QUBO/DHN algorithm and CNO-QUBO/BM algorithm, the numbers of batches for partially synchronous activations in the four problem instances are 19, 200, 300, and 324.

As shown in Table I, CNO-QUBO/DHNm and CNO-QUBO/BMm outperform CNO-QUBO/DHN, CNOQUBO/BM, and other baseline algorithms in terms of time efficiency as well as solution quality and consistency. In particular, only CNO-QUBO/DHNm and CNO-QUBO/BMm can reach global optima with zero standard deviation across all four benchmark problems, indicating the highest quality and consistency. In addition, it is shown that CNO-QUBO/DHNm is faster than most of the baselines in terms of the average number of iterations and the average CPU time. Note that the average numbers of iterations of CNO-QUBO/DHNm are much smaller than those of CNO-QUBO/DHN, owing to the fully synchronous activation in CNO-QUBO/DHNm instead of partially synchronous activation in CNO-QUBO/DHN.

VI. Concluding Remarks

In this paper, the collaborative neurodynamic algorithm is proposed for solving QUBO problems. The almost-sure convergence to global optima property inherited in the CNO approach is demonstrated experimentally. Experimental results of four well-known benchmark problems are elaborated to substantiate the efficacy and superiority of CNO-QUBO against several prevailing exact and meta-heuristic algorithms. The superior performance of CNO-QUBO is owing to the use of multiple Boltzmann machines with momentums for scattered searches assisted by the particle swarm optimization

[1] http://cedric.cnam.fr/~soutif/QKP/N300D25.txt

[2] https://www.opt.math.tugraz.at/qaplib/inst.html

TABLE I

AVERAGE RESULTS OF THE CNO-QUBO ALGORITHM AND SEVERAL BASELINES IN TERMS OF THE BEST/WORST OBJECTIVE FUNCTION VALUES, MEAN VALUES, STANDARD DEVIATIONS, THE NUMBER OF ITERATIONS, AND CPU TIME (SECONDS) AT EACH RUN FOR THE FOUR PROBLEM INSTANCES, WHERE GUROBI STANDS FOR GUROBI OPTIMIZER, CPLEX-DS FOR CPLEX DYNAMIC SEARCH, TS FOR THE TABU SEARCH ALGORITHM, GA FOR THE GENETIC ALGORITHM, CNTS FOR THE CONSTRAINED NEIGHBORHOOD TABU SEARCH ALGORITHM, MA FOR THE HYBRID METAHEURISTIC ALGORITHM, GRASP FOR THE GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE ALGORITHM, IG FOR THE ITERATED GREEDY ALGORITHM, ACO FOR THE ANT COLONY OPTIMIZATION ALGORITHM, HAS FOR THE HARMONY SEARCH ALGORITHM, SA FOR THE SIMULATED ANNEALING ALGORITHM, GWO FOR THE GREY WOLF OPTIMIZATION ALGORITHM, CNO/DHN FOR CNO-QUBO/DHN, CNO/DHNM FOR CNO-QUBO/DHNM, THE BEST RESULTS ARE BOLDFACED, / INDICATES "NOT APPLICABLE", — INDICATES "NOT AVAILABLE", †INDICATES THE PRESCRIBED MAXIMUM NUMBER OF ITERATIONS.

problem	dimension	# of solutions	optimal value	ρ	algorithm	N	M	best/worst	mean $\pm$ std	# of iterations	CPU time
SPP	100	1.27×10^{30}	1573	5	Gurobi	/	/	**1573/1573**	**1573.00 $\pm$ 0.00**	—	3.66
					CPLEX/DS	/	/	**1573/1573**	**1573.00 $\pm$ 0.00**	—	9.94
					TS [29]	/	/	3528/8053	5427.37 $\pm$ 961.85	10000.00†	10.66
					GA_SPP [96]	8	/	**1573**/3935	2078.78 $\pm$ 661.69	5498.96	4.72
					CNO/DHN	8	6	**1573**/4922	3159.16 $\pm$ 815.81	843.62	0.04
					CNO/BM	8	6	**1573**/4130	2782.68 $\pm$ 800.48	2327.12	0.07
					CNO/DHNm	8	6	**1573/1573**	**1573.00 $\pm$ 0.00**	1138.48	0.02
					CNO/BMm	8	6	**1573/1573**	**1573.00 $\pm$ 0.00**	586.86	0.02
MDP	200	1.61×10^{60}	1247	40	Gurobi	/	/	—	—	—	—
					TS [29]	/	/	1241/1179	1214.16 $\pm$ 15.35	2000000.00†	1490.98
					CNTS_MDP [91]	/	/	1161/1034	1102.24 $\pm$ 32.88	7000000.00†	297.62
					GA_MDP [97]	1500	/	1082/1040	1062.88 $\pm$ 8.99	1500000.00†	1965.04
					MA_MDP [91]	1500	500	1225/1183	1202.33 $\pm$ 10.67	225000000.00†	4107.12
					CNO/DHN	1500	500	**1247**/1231	1242.44 $\pm$ 4.98	149944.00	1582.42
					CNO/BM	1500	500	**1247**/1236	1242.48 $\pm$ 3.66	258112.00	5518.63
					CNO/DHNm	1500	500	**1247/1247**	**1247.00 $\pm$ 0.00**	86682.96	253.05
					CNO/BMm	1500	500	**1247/1247**	**1247.00 $\pm$ 0.00**	51616.41	273.45
QKP	300	2.04×10^{90}	29140	100	TS [29]	/	/	28608/25654	27346.84 $\pm$ 776.25	150000.00†	278.37
					GRASP_QKP [98]	/	/	15273/12879	13536.84 $\pm$ 473.34	1000000.00†	237.48
					SA_QKP^{a}	/	/	27615/21553	24170.08 $\pm$ 1746.38	300000.00†	297.32
					IG_QKP [99]	/	/	13019/11004	11941.88 $\pm$ 499.52	1500000.00†	343.36
					GA_QKP [100]	1000	/	**29140**/28757	29020.83 $\pm$ 96.98	717728.25	255.96
					ACO_QKP [101]	1000	/	2392/1959	2105.24 $\pm$ 87.98	500000000.00†	246.28
					PSO [102]	1000	300	28919/26404	27902.63 $\pm$ 534.35	792.17	0.47
					CNO/DHN	1000	300	**29140**/29083	29137.72 $\pm$ 11.40	86628.00	110.39
					CNO/BM	1000	300	**29140**/29075	29128.26 $\pm$ 22.46	148716.00	190.52
					CNO/DHNm	1000	300	**29140/29140**	**29140.00 $\pm$ 0.00**	28334.32	172.43
					CNO/BMm	1000	300	**29140/29140**	**29140.00 $\pm$ 0.00**	31423.21	230.43
QAP	324	3.42×10^{97}	5358	80	TS [29]	/	/	5690/6074	5893.62 $\pm$ 120.85	300000.00†	603.21
					TS_QAP [103]	/	/	**5358**/5400	5371.83 $\pm$ 11.15	1179873.92	328.49
					SA_QAP [104]	/	/	**5358**/5442	5387.54 $\pm$ 25.58	34948858.15	294.23
					GA_QAP^{b}	400	/	**5358**/5400	5375.07 $\pm$ 14.52	2081482.67	212.43
					GWO_QAP^{b}	400	/	5536/5694	5624.45 $\pm$ 44.62	16000000.00†	231.72
					PSO [102]	400	200	5604/5758	5678.84 $\pm$ 36.86	80400.00	0.37
					PSO_QAP^{c}	400	/	5388/5546	5464.88 $\pm$ 39.56	12000000.00†	260.34
					CNO/DHN	400	200	**5358/5358**	**5358.00 $\pm$ 0.00**	155312.64	171.48
					CNO/BM	400	200	**5358/5358**	**5358.00 $\pm$ 0.00**	166160.16	192.69
					CNO/DHNm	400	200	**5358/5358**	**5358.00 $\pm$ 0.00**	24124.82	21.95
					CNO/BMm	400	200	**5358/5358**	**5358.00 $\pm$ 0.00**	20180.92	34.58

[a] https://github.com/DaCasBe/Multiple-Quadratic-Knapsack-Problem-using-Population-based-Metaheuristics,
[b] https://github.com/zohrehraziei/QAP-Meta-heuristic-Algorithms,
[c] https://yarpiz.com/359/ypap104-quadratic-assignment-problem

rule for global repositioning. Further investigations may aim at implementing the CNO-QUBO algorithm in a parallel computing platform (e.g., CUDA), enhancing the robustness of the CNO-QUBO algorithm to handle noisy data, customizing the CNO-QUBO algorithm in specific application domains (such as constrained clustering, vehicle routing, and crowd tracking), and developing more efficient and versatile CNO-based QUBO algorithms assisted via deep learning or reinforcement learning.

REFERENCES

[1] F. Glover, G. Kochenberger, R. Hennig, and Y. Du, "Quantum bridge analytics I: a tutorial on formulating and using QUBO models," *Annals of Operations Research*, vol. 314, no. 1, pp. 141–183, 2022.

[2] F. Glover, G. Kochenberger, M. Ma, and Y. Du, "Quantum bridge analytics II: QUBO-Plus, network optimization and combinatorial chaining for asset exchange," *Annals of Operations Research*, vol. 314, no. 1, pp. 185–212, 2022.

[3] V. Kolmogorov and C. Rother, "Minimizing nonsubmodular functions with graph cuts - a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 7, pp. 1274–1279, 2007.

[4] I. Dunning, S. Gupta, and J. Silberholz, "What works best when? a systematic evaluation of heuristics for max-cut and QUBO," *INFORMS Journal on Computing*, vol. 30, no. 3, pp. 608–624, 2018.

[5] W. Cruz-Santos, S. E. Venegas-Andraca, and M. Lanzagorta, "A QUBO formulation of minimum multicut problem instances in trees for D-Wave quantum annealers," *Scientific Reports*, vol. 9, no. 1, pp. 1–12, 2019.

[6] C. Wang, Y. Guo, J. Zhu, L. Wang, and W. Wang, "Video object co-segmentation via subspace clustering and quadratic pseudo-Boolean optimization in an MRF framework," *IEEE Transactions on Multimedia*, vol. 16, no. 4, pp. 903–916, 2014.

[7] J. Ren, X. Jiang, J. Yuan, and G. Wang, "Optimizing LBP structure for visual recognition using binary quadratic programming," *IEEE Signal Processing Letters*, vol. 21, no. 11, pp. 1346–1350, 2014.

[8] X. Liu, J. He, and S. Chang, "Hash bit selection for nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5367–5380, Nov 2017.

[9] A. Dehghan and M. Shah, "Binary quadratic programing for online tracking of hundreds of people in extremely crowded scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 568–581, 2018.

[10] S. Harwood, C. Gambella, D. Trenev, A. Simonetto, D. E. B. Neira, and D. Greenberg, "Formulating and solving routing problems on quantum computers," *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–17, 2021.

[11] T. Krauss, J. McCollum, C. Pendery, S. Litwin, and A. J. Michaels, "Solving the max-flow problem on a quantum annealing computer," *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–10, 2020.

[12] Y. Imanaka, T. Anazawa, F. Kumasaka, and H. Jippo, "Optimization of the composition in a composite material for microelectronics application using the Ising model," *Scientific Reports*, vol. 11, no. 1, pp. 1–7, 2021.

[13] A. Maruo, H. Igarashi, H. Oshima, and S. Shimokawa, "Optimization of planar magnet array using digital annealer," *IEEE Transactions on Magnetics*, vol. 56, no. 3, pp. 1–4, 2020.

[14] D. Inoue and H. Yoshida, "Model predictive control for finite input systems using the D-Wave quantum annealer," *Scientific Reports*, vol. 10, no. 1, pp. 1–10, 2020.

[15] N. Nikmehr, P. Zhang, and M. A. Bragin, "Quantum distributed unit commitment: An application in microgrids," *IEEE Transactions on Power Systems*, vol. 37, no. 5, pp. 3592–3603, 2022.

[16] F. F. Silva, P. M. Carvalho, L. A. Ferreira, and Y. Omar, "A QUBO formulation for minimum loss network reconfiguration," *IEEE Transactions on Power Systems*, vol. 38, no. 5, pp. 4559–4571, 2023.

[17] B. Krakoff, S. M. Mniszewski, and C. F. Negre, "Controlled precision QUBO-based algorithm to compute eigenvectors of symmetric matrices," *Plos One*, vol. 17, no. 5, p. e0267954, 2022.

[18] P. M. Pardalos and G. P. Rodgers, "Computational aspects of a branch and bound algorithm for quadratic zero-one programming," *Computing*, vol. 45, no. 2, pp. 131–144, 1990.

[19] A. Billionnet and A. Sutter, "Minimization of a quadratic pseudo-Boolean function," *European Journal of Operational Research*, vol. 78, no. 1, pp. 106–115, 1994.

[20] A. Billionnet and S. Elloumi, "Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem," *Mathematical Programming*, vol. 109, no. 1, pp. 55–68, 2007.

[21] E. Boros, P. L. Hammer, R. Sun, and G. Tavares, "A max-flow approach to improved lower bounds for quadratic unconstrained binary optimization (QUBO)," *Discrete Optimization*, vol. 5, no. 2, pp. 501–529, 2008.

[22] A. Billionnet, S. Elloumi, and M.-C. Plateau, "Improving the performance of standard solvers for quadratic 0-1 programs by a tight convex reformulation: The QCR method," *Discrete Applied Mathematics*, vol. 157, no. 6, pp. 1185–1197, 2009.

[23] G. R. Mauri and L. A. N. Lorena, "A column generation approach for the unconstrained binary quadratic programming problem," *European Journal of Operational Research*, vol. 217, no. 1, pp. 69–74, 2012.

[24] P. Wang, C. Shen, A. van den Hengel, and P. H. Torr, "Large-scale binary quadratic optimization using semidefinite relaxation and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 470–485, 2016.

[25] M. J. Schuetz, J. K. Brubaker, and H. G. Katzgraber, "Combinatorial optimization with physics-inspired graph neural networks," *Nature Machine Intelligence*, vol. 4, no. 4, pp. 367–377, 2022.

[26] M. T. Veszeli and G. Vattay, "Mean field approximation for solving QUBO problems," *Plos One*, vol. 17, no. 8, p. e0273709, 2022.

[27] B. S. Lam and A. W. Liew, "A fast binary quadratic programming solver based on stochastic neighborhood search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 32–49, 2020.

[28] T. M. Alkhamis, M. Hasan, and M. A. Ahmed, "Simulated annealing for the unconstrained quadratic pseudo-Boolean function," *European Journal of Operational Research*, vol. 108, no. 3, pp. 641–652, 1998.

[29] F. Glover, "Tabu search—part I," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[30] E. Boros, P. L. Hammer, and G. Tavares, "Local search heuristics for quadratic unconstrained binary optimization (QUBO)," *Journal of Heuristics*, vol. 13, no. 2, pp. 99–132, 2007.

[31] D. Karapetyan, A. P. Punnen, and A. J. Parkes, "Markov chain methods for the bipartite Boolean quadratic programming problem," *European Journal of Operational Research*, vol. 260, no. 2, pp. 494–506, 2017.

[32] A. Lodi, K. Allemand, and T. M. Liebling, "An evolutionary heuristic for quadratic 0–1 programming," *European Journal of Operational Research*, vol. 119, no. 3, pp. 662–670, 1999.

[33] J. Wang, "Discrete Hopfield network combined with estimation of distribution for unconstrained binary quadratic programming problem," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5758–5774, 2010.

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

[35] H. Oshiyama and M. Ohzeki, "Benchmark of quantum-inspired heuristic solvers for quadratic unconstrained binary optimization," *Scientific Reports*, vol. 12, no. 1, pp. 1–10, 2022.

[36] S. Gu, T. Hao, and H. Yao, "A pointer network based deep learning algorithm for unconstrained binary quadratic programming problem," *Neurocomputing*, vol. 390, pp. 1–11, 2020.

[37] M. Ohzeki, "Breaking limitation of quantum annealer in solving optimization problems under constraints," *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.

[38] H. Ushijima-Mwesigwa, R. Shaydulin, C. F. Negre, S. M. Mniszewski, Y. Alexeev, and I. Safro, "Multilevel combinatorial optimization across quantum architectures," *ACM Transactions on Quantum Computing*, vol. 2, no. 1, pp. 1–29, 2021.

[39] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model," *Science*, vol. 233, no. 4764, pp. 625–633, 1986.

[40] 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.

[41] Y. Xia, Q. Liu, J. Wang, and A. Cichocki, "A survey of neurodynamic optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024, in press.

[42] Y. Xia and J. Wang, "A recurrent neural network for solving nonlinear convex programs subject to linear constraints," *IEEE Transactions on Neural Networks*, vol. 16, no. 2, pp. 379–386, Feb. 2005.

[43] ——, "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.

[44] W. Bian, L. Ma, S. Qin, and X. Xue, "Neural network for nonsmooth pseudoconvex optimization with general convex constraints," *Neural Networks*, vol. 101, pp. 1–14, 2018.

[45] C. Xu, Y. Chai, S. Qin, Z. Wang, and J. Feng, "A neurodynamic approach to nonsmooth constrained pseudoconvex optimization problem," *Neural Networks*, vol. 124, pp. 180–192, 2020.

[46] N. Liu and S. Qin, "A neurodynamic approach to nonlinear optimization problems with affine equality and convex inequality constraints," *Neural Networks*, vol. 109, pp. 147–158, 2019.

[47] Z. Xia, Y. Liu, J. Wang, and J. Wang, "Two-timescale recurrent neural networks for distributed minimax optimization," *Neural Networks*, vol. 165, pp. 527–539, 2023.

[48] Q. Liu, S. Yang, and J. Wang, "A collective neurodynamic approach to distributed constrained optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1747–1758, 2017.

[49] 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.

[50] S. Qin, X. Le, and J. Wang, "A neurodynamic optimization approach to bilevel quadratic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2580–2591, Nov 2017.

[51] Z.-B. Xu, G.-Q. Hu, and C.-P. Kwong, "Asymmetric Hopfield-type networks: theory and applications," *Neural Networks*, vol. 9, no. 3, pp. 483–501, 1996.

[52] A. Asheralieva, "Optimal computational offloading and content caching in wireless heterogeneous mobile edge computing systems with Hopfield neural networks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 3, pp. 407–425, 2021.

[53] Y. Zhao, X. Liao, and X. He, "Distributed continuous and discrete time projection neurodynamic approaches for sparse recovery," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 6, pp. 1411–1426, 2022.

[54] 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.

[55] 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, Jan. 2021.

[56] 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.

[57] B. Cernuschi-Frías, "Partial simultaneous updating in Hopfield memories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 4, pp. 887–888, 1989.

[58] A. Likas and A. Stafylopatis, "Group updates and multiscaling: An efficient neural network approach to combinatorial optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 2, pp. 222–232, 1996.

[59] 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.

[60] 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.

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

[62] 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.

[63] 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.

[64] 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.

[65] E. H. Aarts and J. H. Korst, "Boltzmann machines as a model for parallel annealing," *Algorithmica*, vol. 6, no. 1, pp. 437–465, 1991.

[66] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[67] 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.

[68] 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.

[69] H. Li and J. Wang, "Capacitated clustering via majorization-minimization and collaborative neurodynamic optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 6679–6692, May 2024.

[70] 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.

[71] 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.

[72] 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.

[73] J. Wang, J. Wang, and Q.-L. 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.

[74] 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. 33, no. 10, pp. 5116–5124, Oct. 2022.

[75] H. Li, J. Wang, N. Zhang, and W. Zhang, "Binary matrix factorization via collaborative neurodynamic optimization," *Neural Networks*, p. 106348, 2024.

[76] H. Che, C. Li, X. He, and T. Huang, "An intelligent method of swarm neural networks for equalities-constrained nonconvex optimization," *Neurocomputing*, vol. 167, pp. 569–577, 2015.

[77] Y. Liu, A. A. Heidari, Z. Cai, G. Liang, H. Chen, Z. Pan, A. Alsufyani, and S. Bourouis, "Simulated annealing-based dynamic step shuffled frog leaping algorithm: Optimal performance design and feature selection," *Neurocomputing*, vol. 503, pp. 325–362, 2022.

[78] Y. Chen, A. Zhou, and S. Das, "Utilizing dependence among variables in evolutionary algorithms for mixed-integer programming: A case study on multi-objective constrained portfolio optimization," *Swarm and Evolutionary Computation*, vol. 66, p. 100928, 2021.

[79] Y. Chen and A. Zhou, "Multiobjective portfolio optimization via pareto front evolution," *Complex & Intelligent Systems*, vol. 8, no. 5, pp. 4301–4317, 2022.

[80] E. H. Houssein, D. Oliva, N. A. Samee, N. F. Mahmoud, and M. M. Emam, "Liver cancer algorithm: A novel bio-inspired optimizer," *Computers in Biology and Medicine*, vol. 165, p. 107389, 2023.

[81] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the 2002 Congress on Evolutionary Computation.*, vol. 2, 2002, pp. 1671–1676.

[82] 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.

[83] X. Le, Z. Yan, and J. Xi, "A collective neurodynamic system for distributed optimization with applications in model predictive control," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 4, pp. 305–314, 2017.

[84] X. Li, J. Wang, and S. Kwong, "Boolean matrix factorization based on collaborative neurodynamic optimization with Boltzmann machines," *Neural Networks*, vol. 153, pp. 142–151, 2022.

[85] M.-F. Leung, J. Wang, and H. Che, "Cardinality-constrained portfolio selection via two-timescale duplex neurodynamic optimization," *Neural Networks*, vol. 153, pp. 399–410, 2022.

[86] H. Che, J. Wang, and A. Cichocki, "Sparse signal reconstruction via collaborative neurodynamic optimization," *Neural Networks*, vol. 154, pp. 255–269, 2022.

[87] 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.

[88] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, p. 5, 2014.

[89] M. Zaman, K. Tanahashi, and S. Tanaka, "PyQUBO: Python library for mapping combinatorial optimization problems to QUBO form," *IEEE Transactions on Computers*, vol. 71, no. 4, pp. 838–850, 2022.

[90] E. Balas and M. W. Padberg, "Set partitioning: A survey," *SIAM Review*, vol. 18, no. 4, pp. 710–760, 1976.

[91] Q. Wu and J.-K. Hao, "A hybrid metaheuristic method for the maximum diversity problem," *European Journal of Operational Research*, vol. 231, no. 2, pp. 452–464, 2013.

[92] A. Billionnet and É. Soutif, "An exact method based on Lagrangian decomposition for the 0–1 quadratic knapsack problem," *European Journal of Operational Research*, vol. 157, no. 3, pp. 565–575, 2004.

[93] R. E. Burkard, "Quadratic assignment problems," *European Journal of Operational Research*, vol. 15, no. 3, pp. 283–289, 1984.

[94] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, no. 2, pp. 657–690, 2007.

[95] N. Christofides and E. Benavent, "An exact algorithm for the quadratic assignment problem on a tree," *Operations Research*, vol. 37, no. 5, pp. 760–768, 1989.

[96] P. Chu and J. E. Beasley, "Constraint handling in genetic algorithms: the set partitioning problem," *Journal of Heuristics*, vol. 4, no. 4, pp. 323–357, 1998.

[97] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[98] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[99] C. García-Martínez, F. J. Rodriguez, and M. Lozano, "Tabu-enhanced iterated greedy algorithm: A case study in the quadratic multiple knapsack problem," *European Journal of Operational Research*, vol. 232, no. 3, pp. 454–463, 2014.

[100] T. Saraç and A. Sipahioglu, "A genetic algorithm for the quadratic multiple knapsack problem," in *International Symposium on Brain, Vision, and Artificial Intelligence*. Springer, 2007, pp. 490–498.

[101] M. Kong, P. Tian, and Y. Kao, "A new ant colony optimization algorithm for the multidimensional knapsack problem," *Computers & Operations Research*, vol. 35, no. 8, pp. 2672–2683, 2008.

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

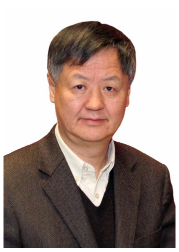
[103] J. Skorin-Kapov, "Tabu search applied to the quadratic assignment problem," *ORSA Journal on Computing*, vol. 2, no. 1, pp. 33–45, 1990.

[104] T. Peng, H. Wang, and D. Zhang, "Simulated annealing for the quadratic assignment problem: A further study," *Computers & Industrial Engineering*, vol. 31, no. 3-4, pp. 925–928, 1996.

[105] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[106] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

Hongzong LI received the B.E. degree in automation from Northeastern University, Shenyang, Liaoning, China, in 2020. He is currently a Ph.D. candidate with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include optimization, computational intelligence, and machine learning.



Jun Wang (Life Fellow) received his B.S. and M.S. degrees in 1982 and 1985 from Dalian University of Technology, Dalian, China, and his Ph.D. degree in 1991 from Case Western Reserve University, Cleveland, Ohio, USA. He held various academic positions at Dalian University of Technology, Case Western Reserve University, University of North Dakota, and the Chinese University of Hong Kong, Hong Kong. He also held various short-term or part-time visiting positions at the U.S. Air Force Armstrong Laboratory, Dayton, Ohio, USA; RIKEN Brain Science Institute, Tokyo, Japan; Huazhong University of Science and Technology, Wuhan, China; Shanghai Jiao Tong University, Shanghai, China; Dalian University of Technology, Dalian, China; and Swinburne University of Technology, Melbourne, Australia. He is currently a chair professor at City University of Hong Kong, Hong Kong. He was a recipient of several awards such as the Research Excellence Award from the Chinese University of Hong Kong (2008-2009), Outstanding Achievement Award from Asia-Pacific Neural Network Assembly (2011), *IEEE Transactions on Neural Networks* Outstanding Paper Award (2011), Neural Networks Pioneer Award from the IEEE Computational Intelligence Society (2014), and Norbert Wiener Award from the IEEE Systems, Man and Cybernetics Society (2019). He served as the General Chair of the 13th/25th International Conference on Neural Information Processing (2006/2018) and the IEEE World Congress on Computational Intelligence (2008). He is an IEEE Systems, Man, and Cybernetics Society Distinguished Lecturer (2017-2022), was a Distinguished Lecturer of IEEE Computational Intelligence Society (2010-2012, 2014-2016). He was the Editor-in-Chief of the *IEEE Transactions on Cybernetics* (2014-2019).