Solving Quadratic Unconstrained Binary Optimization with Collaborative Spiking Neural Networks

Yan Fang
Department of Electrical and Computer Engineering
Kennesaw State University
Marietta, GA
yfang9@kennesaw.edu

Ashwin Sanjay Lele
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA
alele9@gatech.edu

***Abstract*—Quadratic Unconstrained Binary Optimization (QUBO) problem becomes an attractive and valuable optimization problem formulation in that it can easily transform into a variety of other combinatorial optimization problems such as Graph/number Partition, Max-Cut, SAT, Vertex Coloring, TSP, etc. Some of these problems are NP-hard and widely applied in industry and scientific research. Meanwhile, QUBO has been discovered to be compatible with two emerging computing paradigms, neuromorphic computing, and quantum computing, with tremendous potential to speed up future optimization solvers. In this paper, we propose a novel neuromorphic computing paradigm that employs multiple collaborative spiking neural networks to solve QUBO problems. Each SNN conducts a local stochastic gradient descent search and shares the global best solutions periodically to perform a meta-heuristic search for optima. We simulate our model and compare it to a single SNN solver and a mult-SNN solver without collaboration. Through tests on benchmark problems, the proposed method is demonstrated to be more efficient and effective in searching for QUBO optima. Specifically, it exhibits x10 and x15-20 speedup respectively on the multi-SNN solver without collaboration and the single-SNN solver.**

***Index Terms*—neuromorphic computing, spiking neural networks, combinatorial optimization, QUBO**

I. Introduction and Background

A. Spiking Neural Network

Along with the renaissance of neural networks in recent decades, deep learning models and algorithms have successfully dominated most fields of AI research and still keep improving state-of-the-art performance on various benchmarks [1]. However, the success of training deep neural networks relies on tremendous data and powerful computing resources, which commonly lead to substantial consumption of energy. On the other hand, the human brain exhibits superior energy efficiency over artificial neural networks on digital computers. Spiking Neural Network (SNN) is the third generation of neural networks that are developed by computational neuroscientists to model the dynamics of the biological neural system [2]. The "spiking" comes from the action potential generated by the membrane of biological neurons. The value of SNN in AI computing was discovered and explored recently. SNN encodes information with the timing or rate of spikes and can process highly parallel spatial-temporal information with a small number of neurons and spikes [3], [4]. SNN computes in the timing of binary spikes, instead of large matrices of weights. Also, it is a naturally sparse model that brings less computing intensity. Another advantage of SNN comes from the recent progress of neuromorphic computing hardware [5], [6]. Novel nanotechnologies in semiconductor devices and materials like resistive RAMs (RRAM) [7], spintronic devices [8], and Ferroelectric devices [9], are thrusting the design of mixed-signal neuromorphic computing systems, showing promising future of energy-efficient computing. SNNs have been successfully applied to various AI tasks demanding energy efficiency, such as visual recognition [10], natural language processing [11], brain-computer interface [12], and robot control [13].

B. Neural Networks for Optimization

The idea of using neural networks to solve optimization is not new. Using traditional neural networks to solve combinatorial optimization problems was explored as early as 1986 by Hopfield and Tank [14]. Aarts and Korst (1989) proposed to use Boltzmann machines to solve constraint satisfaction problems [15]. Recently there are a few related works that employ SNNs to solve optimization problems. Jonke et al present a method for designing stochastic SNN based on energy functions [16]. These SNNs operate like Boltzmann machines and perform stochastic searches for lower energy status, which represents possible solutions to NP-hard optimization problems such as 3-SAT and the traveling salesman problem (TSP). A similar idea that using SNN to solve constraint satisfaction problem has been explored by Guerra and Furber [17].Coder et al demonstrate an Ising model based on SNN and implement it on IBM's Truenorth neuromorphic processor to solve vertex cover problems, which aim at finding a minimum vertex cover of a graph [18]. Most of these works design neural networks with similar dynamics as simulated annealing, which evolve towards energy minimum for optimization.

C. Quadratic Unconstrained Binary Optimization

In this work, we focus on Quadratic Unconstrained Binary Optimization (QUBO) problem because it can be formulated into numerous combinatorial problems with concise mathematic transformations. These problems including but not limited to Graph/number Partition, Max-Cut, SAT, Graph Coloring, TSP, etc [19], [20]. Some of these problems are NP hard and widely exists in engineering and scientific research. QUBO is also the fundamental algorithm carried out by the quantum computer developed by D-wave [21], due to its equivalent form as Ising model [22]. Similar to Ising model, QUBO can also be implemented on neuromorphic systems such as IBM Truenorth [23] and Intel Loihi [24]. Comparing to the {+1,-1} variables in the Ising model, the binary variables {0,1} in the QUBO model are more compatible with the output of SNN. Differ from the previous works, we explore a set of collaborative SNNs to solve QUBO problem.

A QUBO problem can be defined as:

$$min/max\ y = x^t Q x \tag{1}$$

where x is a binary vector and Q is a square matrix of constants. Q matrix is usually symmetric or upper-triangular form. Under different circumstances, the object can be minimization or maximization, depending on the recreation of application problems. For instance, a number partition problem demands the minimization of QUBO term (1) while the MAX-CUT problem equals to the maximization of (1) [20].

II. Methods

A. QUBO on SNN

The QUBO problem can be visualized as a graph partition problem. One example of such a problem is shown in Fig.1(a). A graph contain four vertices that are connected by four weighted edges. The QUBO problem in this case can be formatted as labeling vertices with binary variables 1 or 0 so that we can maximize the sum of edge weights in the graph of vertices label as 1. The Q in Equ.(1) would be the symmetric weight matrix of all the edges, given in Fig1(a). For this specific problem, the solution is {1,1,0,1}, which selected node 1,2 and 4 and achieved the optimum 9, or 18 in terms of Equ.(1).

The graph problem in Fig.1(a) can be solved by an SNN with the same topology and weight connection. Such techniques have been demonstrated previously in [23]. The SNN behaves like a recurrent network that repetitively generates solutions and results in a reduction of the system energy related the objective function. To avoid local minimum, certain stochastic perturbation is introduced into the network to enhance the ability of "exploration". Although the theoretical proof has not been provided due to the difficulty of NP-hard combinatorial problems. The explanation of the neural dynamics can be refer to Ising model based methods, which use the same gradient descent techniques [22].

Fig.1(b) depicts a fully-connected SNN that is used to solve the graph problem in Fig.1(a). We use a discretized digital Leaky Integrate-and-Fire (LIF) neuron model, modified based

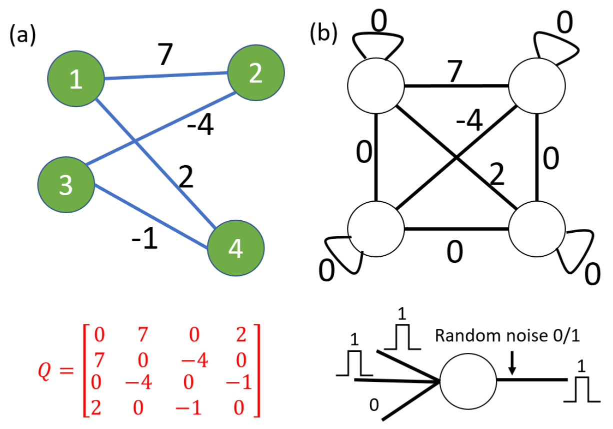

Fig. 1. (a) Example of QUBO graph problem and the edge weight matrix;(b) Corresponding SNN structure of (a) and discretized spiking neuron

on [25], [26]. Assume the SNN has N neurons, the membrane voltage of ith neuron at each time step t can be calculated as:

$$\begin{gathered} V_i(t) = \alpha V_i(t-1) + \sum_{j=1}^{N} q_{ij}(x_j(t-1)|r_j(t-1)) \\ x_i(t) = 1\ on\ V_i(t) > V_{th} \end{gathered} \tag{2}$$

Where the α is the leaky term and q_{ji} is the element in the synaptic weight matrix. r is the stochastic noise added to the presynaptic input via an OR operation with other neurons' spiking output x_j. The presynaptic inputs are weighted and added upon the membrane voltage V_i. The spike status x_i of a neuron is a binary variable determined by the membrane voltage and threshold voltage. If the membrane voltage is higher or equal to the threshold voltage, the neuron will fire a spike and set $x = 1$. In addition, $V_i(t+1) = 0$ as a result of the neuron falling into a refractory period. Otherwise, there is no spike fired, and $x = 0$. As we mentioned, the firing status of neurons is the binary vector that represents a candidate solution at the current time step. In the next step, the fired spikes are broadcast to all the other neurons for updating of membrane voltage. For this optimization problem, the fully-connected SNN uses the edge weight matrix Q as the synaptic weight matrix. Therefore, the spikes are not sent between the neurons with a synaptic weight equal to zero, representing a missing edge in the graph.

B. Collaborative SNNs

Creating multiple threads or agents to perform a meta-heuristic search is a commonly used technique in evolutionary and population-based (swarm intelligence) optimization algorithms [27], [28]. [29] proposed a computing paradigm that implements the swarm intelligence (SI) on multiple coupled SNNs with LIF spiking neuron model. Such an SI-SNN fused model is designed to solve parameter optimization on continuous objective functions. This model is extended in [30] and [31] and is utilized to solve NP-hard combinatorial problems, such as TSP. In this work, we design a similar

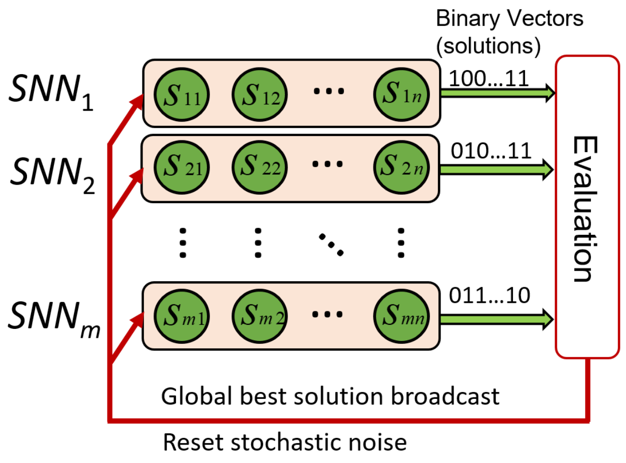
Binary Vectors (solutions)
SNN_1
SNN_2
SNN_m
S_{11} S_{12} S_{1n}
S_{21} S_{22} S_{2n}
S_{m1} S_{m2} S_{mn}
100...11
010...11
011...10
Evaluation
Global best solution broadcast
Reset stochastic noise

Fig. 2. Architecture of Collaborative SNNs to solve QUBO problem

architecture of multiple homogeneous SNNs that conduct a collaborative search to expedite the process of solving QUBO problems.

Fig.2 illustrates the architecture of our proposed model. The entire QUBO solver is composed of m SNNs, shown as rows of pink blocks in Fig.2. Note that the full synaptic connections in these SNNs are not shown here (refer to Fig.1(b)). Each SNN has the same structure, including the same number (n) of neurons (green) and the same weight matrix Q (determined by the QUBO problem), as we described in the previous section. Each SNN runs in parallel with the same dynamics but different initial status, determined by the stochastic noise term $r(t)$ at the first step $t = 1$. For the collaborative SNN model, SNNs run within an initial period from step 0 to step T_ini without noise, so that each SNN can reach the local optima and stabilize the output there. After T_ini, the interaction between SNNs is activated. Normally, every SNN still maintains its own dynamics (Equ.(2)) with a decayed noise term on each neuron:

$$r(t) = \eta\beta^{\gamma(t)} \tag{3}$$

where$\eta$is a normal distributed random variable range from 0 to 1, β is the constant decay factor and $\gamma(t)$ is a function that increments by 1 at each time step ($\gamma(t) = \gamma(t-1)+1$). The binary vectors of solutions, $x(t)$, generated by SNNs at each time step, are evaluated with Equ.(1). A current global best solution $gb(t)$ is kept in memory and compared to every batch of solutions at each step. The SNN interaction happened when a new best solution in history is found. All the presynaptic inputs of SNNs in the next time step are overwritten by this new best solution and the decay of noise term will be reset with $\gamma(t) = 0$. Namely, the membrane voltage at time step $t+1$ would be:

$$V_i(t+1) = \alpha V_i(t) + \sum_{j=1}^{N} q_{ij} gb_j(t)) \tag{4}$$

The motivation of such a design is to speed up the convergence among solutions of SNN and simultaneously maintain the

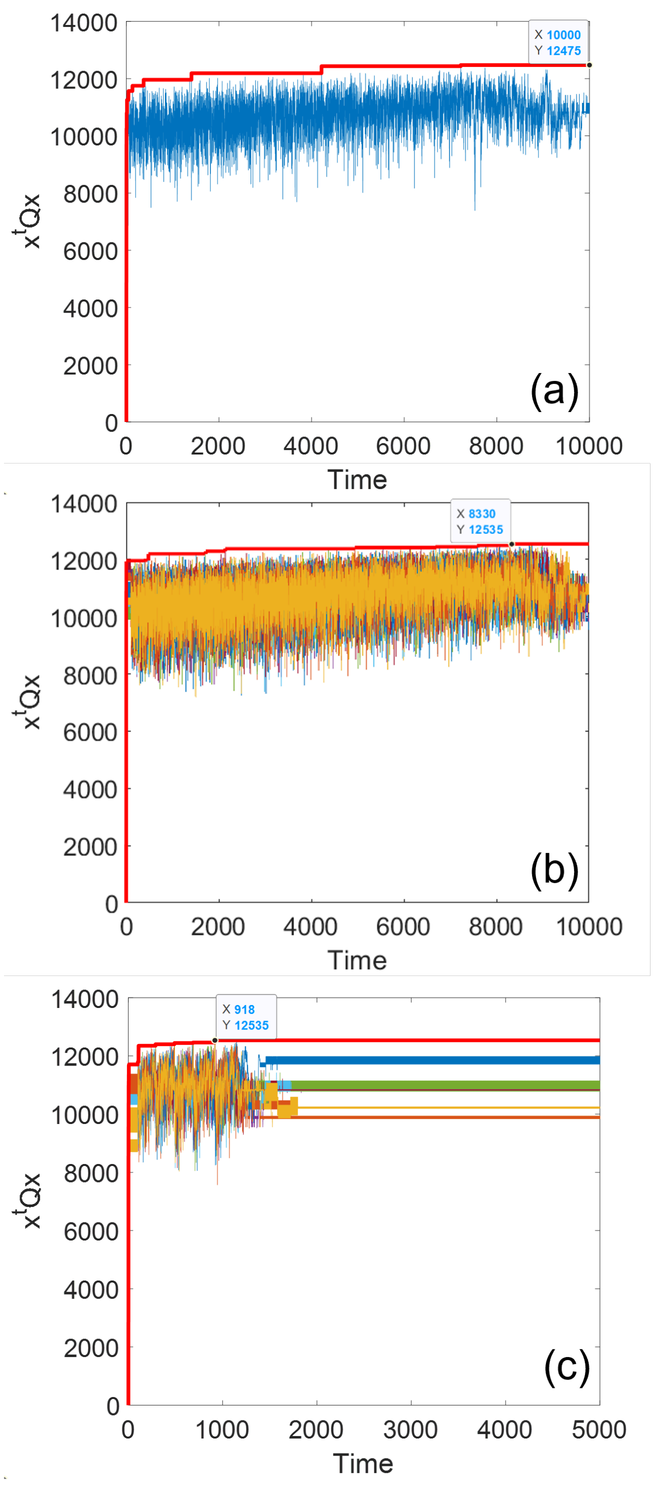

Fig. 3. Diagrams of the solution evolution by time step. x-axis is the time steps of simulation, and y-axis is the evaluation of solutions by object function Equ.(1). The wide red line represents the evaluation history of global best solution. Other lines plot the evaluation of candidate solutions from each SNN. The data label in each plot indicate the best performance in this simulation and when it was achieved (convergence time) (a) A single SNN. (b) 10 SNNs run in parallel, without any collaboration. (c) 10 collaborative SNNs.

exploration capability of the entire system.

III. Results

We simulate the proposed model in Matlab. We primarily use two metrics to evaluate the performance of solving QUBO problems. One is the quality of the solution the solver can find, evaluated via the QUBO object function Equ.(1).

Another is the time steps that cost the solver to reach the optimum. No real-time unit is adopted for evaluation because the proposed model could be implemented on various computing platforms such as GPU, neuromorphic chips, or systems built on emerging nanodevices, which may scale the frequency and time and lead to divergence of computing speed. Thus, the basic time unit here is time steps in simulation, similar to the "ticks" in [23]. The benchmark used in our simulation is a QUBO dataset "Glover, Kochenberger and Alidaee instances", which is available in [32], [33]. We picked two subsets of the maximum QUBO problem: 1). gkaid, 10 instances with dimension 100, 10 edge densities from 0.1 to 1. Diagonal coefficients from [-75,75], off-diagonal coefficients from [-50,50]; 2). gkaie, Five instances with dimension 200, five densities from 0.1 to 0.5. Diagonal coefficients from [-100,100], off-diagonal coefficients from [-50,50].

For comparison, we also test another two solvers, a single SNN QUBO solver (section II.A) and Multiple SNNs with no collaboration. Because these two solvers do not reset noise term, the exponential decay of noise term results in early convergence into local optima in most simulations. We replace the exponential decay with a linear reduction in these two solver. We keep the exponential decay with $\beta = 0.99$ for the proposed model. To improve the search speed, we use $\alpha = 1$, $V_{th} = 1$. Combining with the refractory mechanism of spiking neuron, the SNN's dynamics becomes close to a recurrent network with spiking activation function. It generates different solution at every time step.

Fig.3 gives an example and plots the diagram of evaluating solutions over time for three solvers. In this case, the problem is a QUBO with 100-dimension and 0.5 edge density. We use ten SNNs in our proposed model run the simulation for 10k steps. The maximum value obtained from best solution is 12535. The proposed model find this optimum at $t = 918$, while the SNNs without collaboration reach it at $t = 8330$. The single SNN solver could not find the optimum in this test. Instead, it provided a near-optimum with 12475. We run multiple simulations with these three solvers on the 15 QUBO problems in the datasets mentioned above. We calculate the average times steps of the simulations that solvers successfully find the global optima. Fig.4 provides these data in two chart on two QUBO problem sets respectively. We notice that the other two solvers are respectively 15-20 and 10 times slower than the collaborative SNNs.

The simulation results above demonstrate that Multiple SNNs are more effective and efficient than a single SNN solver in searching for optima of QUBO problems. Within limited time steps, a QUBO solver with multiple SNNs can find better solutions than a single-SNN solver. On the other hand, it takes

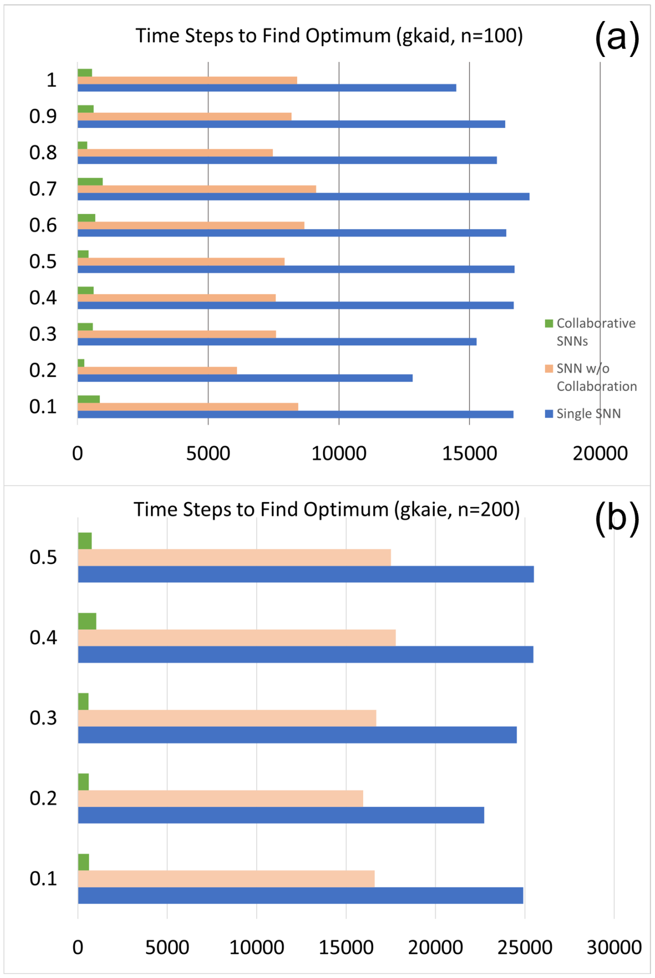

Fig. 4. Time spent by three QUBO solvers to find optima. (a) gkaid dataset (n=100); (b) gkaie dataset(n=200)

a single SNN solver more time to find the global optimum. Further, the collaborative SNN improves the efficiency of convergence compared to SNNs without collaboration. This is attributed to the periodic synchronization on the global best solution. After the synchronization, the reset of noise perturbation help maintain the exploration in the solution space and reduce the chances of being trapped by local optima. Obviously, the methods of coupling multiple SNNs are not limited to the proposed algorithm in this paper. Meanwhile, there is a vast design space remaining for the proposed model. For example, the trade-offs between the hardware cost and number of SNNs, search speed and performance, etc.

Table 1 shows the average performance of global best solutions founded on gkaid(n=100) when we use different numbers of SNNs (swarm size m). The result indicates more SNNs can improve the quality of solutions slightly. However, it is difficult to evaluate such an improvement due to the diversity of problems. Improving the value of the objective function by 1 may be easy for one problem but hard for the other.

TABLE I
AVERAGE PERFORMANCE OF GLOBAL BEST SOLUTION ON GKAID

Edge Density	m=5	m=10	m=20	m=40
0.1	4620.5	4672.1	4679.2	4728.3
0.2	7817.6	7847	7848.8	7872.1
0.3	10087	10087.6	10103.1	10134.5
0.4	12164.5	12162.8	12192.3	12194
0.5	12476.5	12523.8	12514.1	12528.2
0.6	12482	12503.1	12517.9	12522.4
0.7	11973.4	12143.4	12216.4	12272.9
0.8	16421	16421	16421	16421
0.9	17576	17583.2	17587	17587.2

IV. CONCLUSION

In summary, we propose a new neuromorphic computing paradigm that employs multiple collaborative spiking neural networks to solve QUBO problems. Each SNN conducts a local stochastic gradient descent search and shares the global best solutions periodically to perform a meta-heuristic search for optima. We simulate our model and compare it to the single-SNN QUBO solver and multiple SNN QUBO solvers without collaboration. We test three solvers on a few benchmark QUBO problems. The proposed method is demonstrated to be more efficient and effective in searching for QUBO optima. Specifically, it exhibits x10 and x15-20 speedup respectively on the multi-SNN solver without collaboration and the single-SNN QUBO solver.

V. ACKNOWLEDGEMENT

This work is supported by NSF grant CCF-2153440 and CBRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[3] F. Ponulak and A. Kasinski, "Introduction to spiking neural networks: Information processing, learning and applications.," *Acta neurobiologiae experimentalis*, vol. 71, no. 4, pp. 409–433, 2011.

[4] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[5] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[6] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *Ieee Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[7] G. Indiveri, B. Linares-Barranco, R. Legenstein, G. Deligeorgis, and T. Prodromakis, "Integration of nanoscale memristor synapses in neuromorphic computing architectures," *Nanotechnology*, vol. 24, no. 38, p. 384010, 2013.

[8] M. Romera, P. Talatchian, S. Tsunegi, F. Abreu Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota, *et al.*, "Vowel recognition with four coupled spin-torque nano-oscillators," *Nature*, vol. 563, no. 7730, pp. 230–234, 2018.

[9] E. Covi, H. Mulaosmanovic, B. Max, S. Slesazeck, and T. Mikolajick, "Ferroelectric-based synapses and neurons for neuromorphic computing," *Neuromorphic Computing and Engineering*, 2022.

[10] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.

[11] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *2016 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–8, IEEE, 2016.

[12] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014.

[13] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2010.

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

[15] 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.

[16] Z. Jonke, S. Habenschuss, and W. Maass, "Solving constraint satisfaction problems with networks of spiking neurons," *Frontiers in neuroscience*, vol. 10, p. 118, 2016.

[17] G. A. Fonseca Guerra and S. B. Furber, "Using stochastic spiking neural networks on spinnaker to solve constraint satisfaction problems," *Frontiers in neuroscience*, vol. 11, p. 714, 2017.

[18] K. Corder, J. V. Monaco, and M. M. Vindiola, "Solving vertex cover via ising model on a neuromorphic processor," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2018.

[19] G. A. Kochenberger and F. Glover, "A unified framework for modeling and solving combinatorial optimization problems: A tutorial," *Multiscale optimization methods and applications*, pp. 101–124, 2006.

[20] 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*, pp. 1–43, 2022.

[21] H. Ushijima-Mwesigwa, C. F. Negre, and S. M. Mniszewski, "Graph partitioning using quantum annealing on the d-wave system," in *Proceedings of the Second International Workshop on Post Moores Era Supercomputing*, pp. 22–29, 2017.

[22] A. Lucas, "Ising formulations of many np problems," *Frontiers in physics*, p. 5, 2014.

[23] M. Z. Alom, B. Van Essen, A. T. Moody, D. P. Widemann, and T. M. Taha, "Quadratic unconstrained binary optimization (qubo) on neuromorphic computing system," in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3922–3929, IEEE, 2017.

[24] Intel, "Lava documentation - neuromorphic constraint optimization library." https://lava-nc.org/optimization.html, 2022.

[25] A. S. Cassidy, P. Merolla, J. V. Arthur, S. K. Esser, B. Jackson, R. Alvarez-Icaza, P. Datta, J. Sawada, T. M. Wong, V. Feldman, *et al.*, "Cognitive computing building block: A versatile and efficient digital neuron model for neurosynaptic cores," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10, IEEE, 2013.

[26] H. Soula, G. Beslon, and O. Mazet, "Spontaneous dynamics of asymmetric random recurrent spiking neural networks," *Neural Computation*, vol. 18, no. 1, pp. 60–79, 2006.

[27] J. A. Foster, "Evolutionary computation," *Nature Reviews Genetics*, vol. 2, no. 6, pp. 428–436, 2001.

[28] C. Blum and X. Li, "Swarm intelligence in optimization," in *Swarm intelligence*, pp. 43–85, Springer, 2008.

[29] Y. Fang and S. J. Dickerson, "Achieving swarm intelligence with spiking neural oscillators," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, pp. 1–4, IEEE, 2017.

[30] Y. Fang, Z. Wang, J. Gomez, S. Datta, A. I. Khan, and A. Raychowdhury, "A swarm optimization solver based on ferroelectric spiking neural networks," *Frontiers in neuroscience*, p. 855, 2019.

[31] T. Sasaki and H. Nakano, "An optimizer based on spiking neural-oscillator networks," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2924–2929, IEEE, 2021.

[32] J. E. Beasley, "Or-library." http://people.brunel.ac.uk/ mastjjb/jeb/orlib/bqpinfo.html, 1990.

[33] "Biq mac library - binary quadratic and max cut library." https://biqmac.aau.at/biqmaclib.html.