Slack-Free Spiking Neural Network Formulation for Hypergraph Minimum Vertex Cover

Tam Ngoc-Bang Nguyen[1], **Anh-Dzung Doan**[1], **Zhipeng Cai**[2], **Tat-Jun Chin**[1]
[1] Australian Institute for Machine Learning, The University of Adelaide,
[2] Intel Labs
{tam.nb.nguyen,dzung.doan,tat-jun.chin}@adelaide.edu.au
zhipeng.cai@intel.com

Abstract

Neuromorphic computers open up the potential of energy-efficient computation using spiking neural networks (SNN), which consist of neurons that exchange spike-based information asynchronously. In particular, SNNs have shown promise in solving combinatorial optimization. Underpinning the SNN methods is the concept of energy minimization of an Ising model, which is closely related to quadratic unconstrained binary optimization (QUBO). Thus, the starting point for many SNN methods is reformulating the target problem as QUBO, then executing an SNN-based QUBO solver. For many combinatorial problems, the reformulation entails introducing penalty terms, potentially with slack variables, that implement feasibility constraints in the QUBO objective. For more complex problems such as hypergraph minimum vertex cover (HMVC), numerous slack variables are introduced which drastically increase the search domain and reduce the effectiveness of the SNN solver. In this paper, we propose a novel SNN formulation for HMVC. Rather than using penalty terms with slack variables, our SNN architecture introduces additional spiking neurons with a constraint checking and correction mechanism that encourages convergence to feasible solutions. In effect, our method obviates the need for reformulating HMVC as QUBO. Experiments on neuromorphic hardware show that our method consistently yielded high quality solutions for HMVC on real and synthetic instances where the SNN-based QUBO solver often failed, while consuming measurably less energy than global solvers on CPU.

1 Introduction

Neuromorphic computing research aims to develop computational models inspired by neural architectures found in nature. Spiking neural networks (SNN), in which a network of processing units (neurons) asynchronously transmit spike-based messages to each other [34], is a notable representative of the neuromorphic paradigm. Through parallelism, stochastic behavior, event-driven computing and other biologically-inspired properties, SNNs promise higher energy efficiency than conventional computing, which includes the successful artificial neural networks (ANN) [16].

The advent of neuromorphic computing hardware that can implement SNNs has boosted research in the area. Notable examples include IBM TrueNorth [25] and Intel Loihi [12, 30, 35]. Rigorous benchmarking [13] indicate the promise of SNNs in delivering energy efficient computations, which not only benefit edge computing applications, but also reducing the energy consumption of data centers. The recent introduction of Intel Hala Point [1], the world's largest neuromorphic supercomputer with 1.15 billion neurons and 138.2 billion synapses, is a testament of confidence in the technology.

Two major classes of problems have been explored for SNNs: machine learning and combinatorial optimization [13]. Representative approaches in the former include training SNNs using asynchronous

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-2124

variants of backpropagation, and converting pre-trained ANNs into equivalent SNNs for deployment on neuromorphic hardware. Works in the latter develop SNNs to solve specific optimization problems, where the spike-based temporal information processing is exploited to achieve computational benefits over classical methods. Our work focuses on combinatorial optimization.

The concept of energy minimization of the Ising model underpins SNN techniques for combinatorial optimization. The task is closely related to quadratic unconstrained binary optimization (QUBO)

$$\min_{\mathbf{z}\in\{0,1\}^N} \mathbf{z}^T\mathbf{Q}\mathbf{z}, \tag{1}$$

where $\mathbf{Q} \in \mathbb{Z}^{N\times N}$ is a symmetric coefficient matrix (we restrict $\mathbf{Q}$ to integers to facilitate the SNN treatment; integers are sufficient nonetheless for the targeted combinatorial problem). Several SNNs have been developed to solve QUBO [6, 14, 3, 32]; Sec. 3.2 will describe such a method. To bring such SNNs to bear on other combinatorial problems, the problems must first be reformulated into QUBO [11, 22]. For problems with feasibility constraints, penalty terms and often slack variables will be added to the QUBO objective to implement the constraints. Examples are provided below.

Problem 1 (Minimum vertex cover (MVC))**.** Let $G = (V, E)$ be a graph with vertices $V = \{1, \ldots, N\}$ and edges $E = \{e_1, \ldots, e_K\}$, where each $e_k = \langle i^{(k)}, j^{(k)}\rangle \subset V$ connects two vertices. A vertex cover of G is a subset Z of V such that every edge is incident with at least a vertex in Z. The goal of MVC is to find the vertex cover with the smallest number of vertices. Let

$$\mathbf{z} = [z_1, \ldots, z_N]^T \in \{0, 1\}^N \tag{2}$$

be a binary vector whose role is to select a subset of V, where $z_i = 1$ indicates that the i-th vertex is selected, and $z_i = 0$ means otherwise. MVC can be expressed as

$$\begin{aligned} \min_{\mathbf{z}\in\{0,1\}^N} \quad & \|\mathbf{z}\|_1 \\ \text{s.t.} \quad & (1 - z_{i^{(k)}})(1 - z_{j^{(k)}}) = 0, \quad \forall k = 1, \ldots, K. \end{aligned} \tag{3}$$

Note that $\|\mathbf{z}\|_1 = \|\mathbf{z}\|_2$ for binary $\mathbf{z}$ and the equality constraints in (3) are quadratic in $\mathbf{z}$. Rewriting the equality constraints as penalty terms, we obtain the QUBO

$$\min_{\mathbf{z}\in\{0,1\}^N} \quad \|\mathbf{z}\|_2^2 + \lambda \sum_{k=1}^{K} (1 - z_{i^{(k)}})(1 - z_{j^{(k)}}). \tag{4}$$

Intuitively, solutions $\mathbf{z}$ that violate the constraints in (3) will raise the total cost in (4) and hence be penalized. The quantity $\lambda \in \mathbb{Z}$ is the penalty weight that controls the degree of penalization due to constraint violations. Note that while (4) is not exactly in the form (1) due to the existence of a constant, the remaining steps to rearrange (4) to (1) are minor; see supplementary material. □

Problem 2 (Hypergraph minimum vertex cover (HMVC))**.** Let $H = (V, F)$ be an $r-$uniform hypergraph with vertices $V = \{1, \ldots, N\}$ and hyperedges $F = \{f_1, \ldots, f_K\}$, where each $f_k \subset V$ is incident with exactly r vertices, $r \geq 2$. A vertex cover of H is a subset Z of V such that every hyperedge is incident with at least a vertex in Z. The goal of HMVC is to find the vertex cover with the smallest number of vertices. Note that HMVC reduces to MVC if $r = 2$. For each hyperedge f_k, define a binary vector

$$\mathbf{b}^{(k)} = [b_1^{(k)}, \ldots, b_N^{(k)}]^T \in \{0, 1\}^N, \tag{5}$$

where $\|\mathbf{b}^{(k)}\|_1 = r$, and $b_i^{(k)} = 1$ means that vertex i is incident to f_k and $b_i^{(k)} = 0$ means otherwise. HMVC can then be written as

$$\begin{aligned} \min_{\mathbf{z}\in\{0,1\}^N} \quad & \|\mathbf{z}\|_1 \\ \text{s.t.} \quad & {\mathbf{b}^{(k)}}^T\mathbf{z} \geq 1, \quad \forall k = 1, \ldots, K, \end{aligned} \tag{6}$$

which is a 0-1 integer linear program (ILP). Since r variables ($r \geq 2$) exist in each linear inequality constraint, in general it is not possible to express it as a quadratic equality constraint, *cf*. (3). Instead, the path to QUBO will involve converting each inequality into an equality constraint

$${\mathbf{b}^{(k)}}^T\mathbf{z} - \mathbf{1}_{r'}^T\mathbf{y}^{(k)} = 1, \tag{7}$$

which requires introducing r' binary slack variables for each f_k

$$\mathbf{y}^{(k)} = [y_1^{(k)}, \ldots, y_{r'}^{(k)}]^T \in \{0,1\}^{r'}, \tag{8}$$

where $r' = r - 1$ and $\mathbf{1}_{r'}$ is a column vector of 1 of size r'. Installing the equality constraints as quadratic penalty terms, we obtain the QUBO

$$\min_{\mathbf{z}\in\{0,1\}^N, \{\mathbf{y}^{(k)}\}_{k=1}^K \in \{0,1\}^{r'\times K}} \quad \|\mathbf{z}\|_2^2 + \lambda \sum_{k=1}^{K} \left(\mathbf{b}^{(k)^T}\mathbf{z} - \mathbf{1}_{r'}^T \mathbf{y}^{(k)} - 1\right)^2, \tag{9}$$

where $\lambda \in \mathbb{Z}$ is the penalty weight. See supp. material for rewriting the QUBO in form (1). □

A major difference between the QUBO reformulations for MVC and HMVC is that the latter requires slack variables, the quantity of which scales linearly with the number K and degree r of hyperedges. This significantly increases the search space and difficulty of the optimization. As we will show in Sec. 5, the existing SNN-based QUBO solver [3] is unable to satisfactorily solve HMVC based on (9). This presents an obstacle towards applying SNNs to a combinatorial problem with many practical applications, *e.g.*, computational biology [9], computer network security [19], resource allocation [7] and social network analysis [23]. Fundamentally, HMVC is a general optimization problem that encompasses several related formulations, such as set cover, hitting set, and traversal [5], giving it wide applicability. The issue calls for a more effective SNN for HMVC.

1.1 Contributions

To solve HMVC more effectively, we propose an SNN that is composed of a mix of non-equilibrium Boltzmann machine (NEBM) spiking neurons [32] and a custom *feedback* spiking neuron. One feedback neuron is introduced for each constraint in (6) to check for constraint violations and encourage the overall state to return to feasibility. A major benefit of our handcrafted SNN is obviating the need to reformulate HMVC as QUBO based on the penalty method, which not only avoids the usage of slack variables, but also removes the necessity to tune the penalty weight.

Results on Intel Loihi 2 [30] indicate that our SNN significantly outperformed the QUBO approach on HMVC, in that our method consistently yielded high-quality solutions on synthetic and real problem instances where the SNN-based QUBO solver [3] often failed to return feasible results. Moreover, our SNN consumed measurably much less energy than global solvers on CPU.

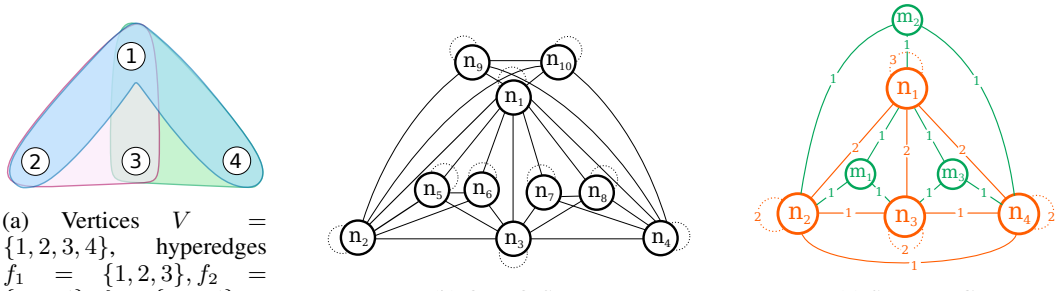
Our work follows the spirit of other handcrafted SNNs for combinatorial optimization [8, 31, 17, 21, 36]. However, such works have mainly been targeted at constraint satisfaction problems, whereas our SNN is aimed at a graph-based optimization problem, *i.e.*, HMVC (more details in Sec. 2).

2 Related work

Previous studies have shown that an SNN with a topology corresponding to the matrix $\mathbf{Q}$ can efficiently solve QUBO [3, 6, 14]. This enables neuromorphic computing to solve combinatorial problems that can be encoded as QUBO, such as graph partitioning [26]. In addition, via the penalty method [29, Chap. 17], other combinatorial problems with constraints can be reformulated into QUBO [15]. Problem 1 has discussed this for MVC, which has been evaluated on a neuromorphic processor [11]. Other works that employed QUBO reformulation include [32, 22] who solved maximum independent set and ILP. However, more complex problems will require the introduction of slack variables, which increases the search space and difficulty for an SNN solution. Problem 2 has illustrated this for HMVC.

The majority of handcrafted SNNs for combinatorial optimization aim to solve constraint satisfaction problems (CSPs). Jonke *et al*. [21] proposed a stochastic SNN for traveling salesman problem. Since then, several SNN solvers have been proposed for CSPs, such as, Sudoku [8, 31], graph coloring [17], and Boolean satisfiability problem [21, 36]. These existing approaches share a common strategy: constructing a specific SNN topology that is strongly tailored to the constraints of each CSP. This is to ensure that these SNN solvers can seek valid values for a set of variables that satisfy the specified constraints. In other words, the primary objective of these handcrafted SNNs is to find feasible solutions to the combinatorial optimization problem. Although our approach shares a similar spirit

 https://doi.org/10.52202/079017-2124



(a) Vertices $V = \{1, 2, 3, 4\}$, hyperedges $f_1 = \{1, 2, 3\}, f_2 = \{1, 3, 4\}, f_3 = \{1, 2, 4\}$

(b) QUBO-SNN

(c) SF-HMVC

Figure 1: (a) HMVC input hypergraph: a 3-uniform hypergraph with 4 vertices and 3 hyperedges. (b) SNN for QUBO (9) derived from (a). Both the variables z_i and introduced slack variables are NEBM neurons. The edge weights are omitted for brevity. (c) The SNN from our method for input (a). NEBM and FB neurons are in orange and green resp. Dashed circles indicate self-connections.

to these handcrafted SNNs in that we directly construct SNN topology using constraints, our SNN topological graph is specifically designed to seek not only a feasible solution but also the optimal one.

The QUBO formulation is amenable to Ising-specific analog and digital hardware solvers, including spintronics, memristors, and quantum annealers [27]. However, given our focus on SNNs, we leverage the state-of-the-art Loihi 2 neuromorphic accelerator to implement the SNN solutions for both QUBO and our method, due to its high flexibility in SNN design and programmable neuron model. For a comprehensive overview of Loihi 2's capabilities, we refer readers to [24, 32, 35].

There is a large body of work on SNNs for machine learning; we refer to [13] for a thorough survey. While SNNs are usually less accurate than ANNs [28], under certain temporal dynamics, SNNs can surpass ANNs. SNNs have also been used for robotics. For instance, [24] implemented their SNN-based quadratic programming solver for model predictive control on Loihi 2, achieving two orders-of-magnitude gains in energy-delay product compared to CPU solvers. Another study demonstrated the applications of SNN in optical flow estimation for event cameras [33], which showed significant potential for real-time operations. In addition, SNNs have been applied to depth estimation for event sensors [10], where their method computed the optical flow on the neuromorphic chip and integrated the optical flow with camera velocity to estimate depth.

3 Preliminaries

An SNN can be viewed as a "program" for a neuromorphic computer [34]. Formally, an SNN comprises a set of spiking neurons $\mathcal{N} = \{n_i\}_{i=1}^{N}$ and synapses, where each synapse connects a pair of neurons. The interconnections can be summarized by a matrix $\mathbf{W} = [w_{ij}] \in \mathbb{Z}^{N \times N}$, where the element w_{ij} indicates the strength of connection between n_i and n_j, and $w_{ij} = 0$ signifies an absence of connection between the pair. The architecture of an SNN and the model of the spiking neuron define the behavior of the program. Here, we describe the NEBM spiking neuron model (Sec. 3.1) and the NEBM-based SNN for QUBO (Sec. 3.2), adapted from Intel's original implementation [32].

3.1 NEBM

NEBM neurons produce outputs probabilistically based on the Boltzmann distribution [32]. An NEBM neuron n_i contains a binary state $s_i \in \{0, 1\}$ that indicates whether the neuron is firing $(s_i = 1)$ or not $(s_i = 0)$, and an internal state $u_i \in \mathbb{Z}$ that accumulates outputs from connected neurons.

While in theory spiking neurons operate asynchronously, practical neuromorphic computers such as Intel Loihi are fully digital devices [12]. The continuous time dynamics of a spiking neuron are approximated using fixed-size discrete timesteps t. It should be reminded that t relates to the algorithmic time of the computation rather than the time of a global synchronous clock.

Based on the algorithmic time formalism, at timestep t, an NEBM neuron n_i accumulates inputs from connected neurons into its internal state

$$u_i^{(t)} = u_i^{(t-1)} + \sum_{j \neq i} w_{ij} \Delta s_j^{(t-1)}, \tag{10}$$

where $\Delta s_j^{(t-1)}$ is the output spike of neuron n_j from the previous timestep. The internal state of n_i is then converted to a switching probability

$$P\Big(s_i^{(t)} = 1\Big) = \frac{1}{1 + e^{u_i^{(t)}/T}}, \tag{11}$$

where T is the *temperature*. If the switching probability (11) exceeds a randomly generated threshold $\theta_i \in [0, 1]$, s_i is set to 1. From here, an output or delta spike is calculated

$$\Delta s_i^{(t)} = s_i^{(t)} - s_i^{(t-1)}. \tag{12}$$

If $\Delta s_i^{(t)}$ takes a non-zero value, *i.e.*, the state s_i changes from the previous timestep, n_i propagates a delta spike to all its connected neurons and enters a *refractory period* for r_i timesteps. Within the refractory period, the binary state s_i remains unchanged. Note that temperature T and length of the refractory period r_i are hyperparameters of NEBM neurons. For more details of NEBM, its hyperparameters and how it is implemented on Loihi 2 given hardware constraints (i.e. no division operator), we refer readers to [32].

3.2 NEBM-based SNN for QUBO

The energy encoded by the neuronal states in an SNN is

$$\mathcal{E} = \sum_{i=1}^{N} s_i u_i. \tag{13}$$

Following [32], the NEBM-based SNN algorithm to minimize the energy is summarized in Alg. 2 (see Appendix). Intuitively, the method repeatedly samples and explores the neuronal states guided by the structure of the synapses and evolving state values [21]. The algorithm is executed on the neuromorphic hardware for M algorithmic timesteps, and the state configurations probed at the M timesteps are returned. The state configurations are evaluated off-chip, with the best one returned as the solution. Note that we employ an early version of NEBM where T is kept constant in Alg. 2, though T can be annealed to fine-tune the search. We refer interested readers to [32] for recent advancements of NEBM and the general-purpose NEBM-based QUBO solver.

To enable Alg. 2 to solve QUBO, following [32], we perform the following mapping:

- Assign each binary variable z_i to a neuron n_i, and equating the state s_i with the value of z_i.
- Associate the quadratic coefficients $\mathbf{Q} = [q_{ij}] \in \mathbb{Z}^{N \times N}$ with the synapse weights $\mathbf{W}$, *i.e.*, $\mathbf{Q} = \mathbf{W}$. This also implies the existence of self connections for the neurons, where $q_{ii} = w_{ii}$ is the strength of the self connection for neuron n_i.

Fig. 1b illustrates the mapping. Note that the "mapping" does not imply that $\mathcal{E}$ equates to the QUBO cost; rather, the minimization of $\mathcal{E}$ by Alg. 2 tends to lead to the minimization of the QUBO cost. In this sense, the SNN is a *heuristic* method to solve QUBO.

4 Slack-free SNN formulation for HMVC

Instead of converting HMVC to QUBO following the derivations in Problem 2 and applying the SNN described in Sec. 3.2, we develop a novel *slack-free* SNN that directly solves HMVC. Referring to the 0-1 ILP (6), we construct our SNN, named *SF-HMVC*, as follows:

- As before, each binary variable z_i is encoded as the state s_i of an NEBM neuron n_i.
- Each k-th constraint $\mathbf{b}^{(k)^T}\mathbf{z} \geq 1$ is represented by a neuron m_k, whose dynamics are governed by a custom neuron model called *feedback* or *FB* (more details below).

 https://doi.org/10.52202/079017-2124

Algorithm 1 Our proposed SF-HMVC (**note:** the algorithm is executed for each NEBM neuron n_i and FB neuron m_k on the neuromorphic hardware in a parallel way; see [13] for details).

Require: Co-occurrence matrix $\mathbf{F}$, adjacency matrix $\mathbf{A}$, temp. T and refract. period r_i.

Initialize $s_i^{(0)} \leftarrow 0,\ \Delta s_i^{(0)} \leftarrow 0,\ u_i^{(0)} \leftarrow -f_{ii},\ \text{refract\_counter}_i^{(0)} \leftarrow 0$
for each timestep t **do**
  $u_i^{(t)} \leftarrow u_i^{(t-1)} + \sum_{j \neq i}^{N} f_{ij} \Delta s_j^{(t-1)} + \sum_{k=1}^{K} a_{ik} \Delta c_k^{(t-1)}$
  $p_i^{(t)} \leftarrow \dfrac{1}{1 + \exp(u_i^{(t)}/T)}$
  $\theta_i \leftarrow rand(0,1)$
  if neuron n_i is not in refractory period **then**
    if $p_i^{(t)} \geq \theta_i$ **then**
      $s_i^{(t)} \leftarrow 1$
    else
      $s_i^{(t)} \leftarrow 0$
  else
    $s_i^{(t)} \leftarrow s_i^{(t-1)}$
    $\text{refract\_counter}_i^{(t)} \leftarrow \max(\text{refract\_counter}_i^{(t-1)} - 1, 0)$
  $\Delta s_i^{(t)} \leftarrow s_i^{(t)} - s_i^{(t-1)}$
  send $\Delta s_i^{(t)}$ to connected neurons
  if $\Delta s_i^{(t)} \neq 0$ **then**
    neuron n_i enters refractory period
    $\text{refract\_counter}_i^{(t)} \leftarrow r_i$

} NEBM neuron

Require: Adjacency matrix $\mathbf{A}$.

for each timestep t **do**
  $v_k^{(t)} \leftarrow \sum_{i=1}^{N} a_{ik} s_i^{(t)}$
  if $v_k^{(t)} = 0$ **then**
    $c_k^{(t)} \leftarrow 1, \Delta c_k^{(t)} \leftarrow -1$
  else
    if $c_k^{(t-1)} = 0$ **then**
      $c_k^{(t)} \leftarrow 0, \Delta c_k^{(t)} \leftarrow 0$
    else
      $c_k^{(t)} \leftarrow 0, \Delta c_k^{(t)} \leftarrow 1$
  send $\Delta c_k^{(t)}$ to connected neurons

} FB neuron

- The weight matrix of our SNN which consists of $N + K$ neurons $[s_1, \ldots, s_N, m_1, \ldots, m_K]$ is

$$\mathbf{W} = \begin{bmatrix} \mathbf{F} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0}_{K \times K} \end{bmatrix} \in \mathbb{Z}^{(N+K)\times(N+K)}, \tag{14}$$

  where $\mathbf{0}_{K \times K}$ is a $K \times K$ matrix of zeros,

  - $\mathbf{A} = [\mathbf{b}^{(1)} \cdots \mathbf{b}^{(K)}] \in \{0,1\}^{N \times K}$ is the adjacency matrix between NEBM and FB neurons (intuitively, the k-th FB neuron m_k is connected to the NEBM neurons corresponding to z_i's that appear in the k-th constraint); and
  - $\mathbf{F} = [f_{ij}] = \mathbf{A}\mathbf{A}^T \in \mathbb{Z}^{N \times N}$ is the co-occurrence matrix of the NEBM neurons (intuitively, f_{ij} is high if z_i and z_j appear frequently together in the constraints, while f_{ii} is high if z_i appear in many constraints).

Fig. 1c illustrates the proposed SNN construction, while Alg. 1 summarizes the associated algorithm.

Note that Alg. 1 involves two types of neurons that implement different dynamics, and the neurons are executed in parallel on the neuromorphic hardware [13]. The internal state for m_k is

$$v_k^{(t)} = \sum_{i=1}^{N} a_{ik} s_i^{(t)} \tag{15}$$

where a_{ik} is the element of i-th row and k-th column of matrix $\mathbf{A}$. An FB neuron m_k is deterministically activated (*i.e.*, its binary state c_k becomes 1) whenever the connected NEBM neurons are *all inactive* (*i.e.*, $v_k = 0$), and vice versa. Then, a negative spike Δc_k is generated if the value of c_k is 1, so as to send excitatory signals to the connected NEMB neurons to attempt to satisfy the constraint; otherwise no spikes are generated. The point here is not to over-excite NEBM neurons. Then, the internal state of an NEBM neuron n_i accumulates inhibitory signals from the connected NEBM neurons and excitatory signals from the connected FB neurons, *i.e.*,

$$u_i^{(t)} = u_i^{(t-1)} + \sum_{j \neq i}^{N} f_{ij} \Delta s_j^{(t-1)} + \sum_{k=1}^{K} a_{ik} \Delta c_k^{(t-1)}. \tag{16}$$

Similar to Alg. 2, Alg. 1 is executed on the neuromorphic hardware for M algorithmic timesteps, and the best state configuration probed and evaluated off-chip is taken as the solution. Note that the probing mechanism incurs significant overhead in the execution time of current implementation [10], and an on-chip evaluation strategy could be implemented to address this bottleneck [24, 32].

Scalability analysis Based on the construction of the QUBO-based SNN and the proposed SF-HMVC, we can model the number of spiking neurons required to encode each SNN:

- $n_{\text{neurons}}(\text{QUBO}) = N + (r-1)K$
- $n_{\text{neurons}}(\text{SF-HMVC}) = N + K$

As N, K and r collectively define the size of the input problem, we can conclude that SF-HMVC scales linearly whereas the QUBO-based SNN scales quadratically with respect to the problem size.

5 Experiments

Implementation Our method SF-HMVC was implemented on the Loihi assembly language through Lava framework [2] and deployed on the neuromorphic chip of Intel Loihi 2 [30] with $M = 1000$ algorithmic timesteps (denoted as `SF-HMVC-Loihi`). Using grid search, we selected temperature T from the range $[2, 4, 8, 16, 24, 32]$, refractory period r_i from the range $[8, 16, 32, 64, 128]$ (note that all NEBM neurons were given the same refractory period). We set $\lambda = 2$ for QUBO (4) (same as [11]) and grid search $\lambda = [1, 10, 100, 1000]$ for QUBO (9). Note that we were unable to change the seed of the random number generator on Intel Loihi 2, which prevented us from repeatedly executing SNNs to obtain error bars in the solution quality. Also, we noticed that energy consumption and runtime of our method and all competitors remained fairly consistent across different runs.

5.1 MVC

Dataset We employed the DIMACS benchmark dataset [20]. DIMACS includes graph instances that are both synthetically generated and sourced from real-world applications, including coding theory, fault diagnosis, Keller's conjecture, *etc*. The original DIMACS graphs were for maximum clique problems; we converted these to MVC instances by taking the complement graphs. Only 15 MVC instances where the corresponding SNN for SF-HMVC could fit on Loihi 2 were selected.

Competitors `SF-HMVC-Loihi` was compared to the following methods:

- `ILP-CPU`: MVC was solved using the ILP (3) with Gurobi [18] and evaluated on CPU.
- `QUBO-CPU`: MVC was solved using QUBO (4) with Gurobi [18] and evaluated on CPU. We set $\lambda = 2$ (same as [11]).
- `QUBO-Loihi`: MVC was solved using QUBO (4) with NEBM-based SNN (see Sec. 3.2) and evaluated on Intel Loihi 2. We set $T = 16, r_i = 64, \lambda = 2$.

https://doi.org/10.52202/079017-2124

Table 1: Solution quality $\|\mathbf{z}\|_1$ of the MVC methods for all DIMACS MVC instances.

Instance	$\|V\|$	$\|E\|$	ILP-CPU	QUBO-CPU	QUBO-Loihi	SF-HMVC-Loihi
C125.9	125	787	91	91	122	116
C250.9	250	3141	206	206	239	240
gen200_p0.9_44	200	1990	156	156	185	199
gen200_p0.9_55	200	1990	145	145	187	191
hamming6-2	64	192	32	32	51	32
hamming6-4	64	1312	60	60	64	61
hamming8-2	256	1024	128	128	236	128
johnson8-2-4	28	168	24	24	28	24
johnson8-4-4	70	560	56	56	59	69
johnson16-2-4	120	1680	112	112	120	119
keller4	171	5100	160	160	171	170
MANN_a9	45	72	29	29	38	37
MANN_a27	378	702	252	252	352	324
san200_0.9_1	200	1990	130	130	178	199
sanr200_0.9	200	2037	158	158	184	199

Table 2: Runtime (in seconds) and energy usage (in Joules) of MVC methods. Note that ∞ indicates that the energy was too high for pyJoules to measure.

Instance	ILP-CPU		QUBO-CPU		QUBO-Loihi		SF-HMVC-Loihi	
	Time	**Energy**	**Time**	**Energy**	**Time**	**Energy**	**Time**	**Energy**
C125.9	0.37	32.41	1.00	94.53	3.79	0.09	3.77	0.05
C250.9	>3600	144380.56	>3600	∞	7.36	0.26	7.37	0.18
gen200_p0.9_44	0.1	8.92	4.60	387.83	6.03	0.09	5.18	0.14
gen200_p0.9_55	0.03	2.12	2.59	203.84	5.98	0.04	5.17	0.01
hamming6-2	0	0.07	0.01	0.46	2.09	0.02	1.84	0.06
hamming6-4	0.03	2.25	0.36	25.83	2.91	0.07	1.83	0.06
hamming8-2	0	0.19	0.03	1.53	7.64	0.04	7.55	0.04
johnson8-2-4	0.01	0.41	0.05	3.16	1.06	0.08	0.94	0.02
johnson8-4-4	0	0.15	0.26	18.83	2.26	0.02	1.99	0.05
johnson16-2-4	0.5	3.74	4.46	316.75	3.71	0.03	3.72	0.05
keller4	1.14	114.13	1628.92	185173.36	5.13	0.11	5.15	0.03
MANN_a9	0	0.97	0.02	2.24	1.55	0.07	1.35	0.01
MANN_a27	0.18	12.06	0.82	72.92	10.95	0.31	9.58	0.09
san200_0.9_1	0.01	0.87	0.27	22.70	5.99	0.25	5.28	0.04
sanr200_0.9	77.92	10111.06	163.98	20591.27	5.95	0.18	5.17	0.3

All hyperparameters of the competitors were selected using grid search. The configuration that demonstrated consistent performance across all problem instances was selected for each method. For SF-HMVC-Loihi, we set $T = 4, r_i = 8$.

Metrics Solution quality, runtime (in seconds), and energy consumption (in Joules) were reported for all methods. Runtime and energy consumption on Intel Loihi 2's Oheo Gulch board were measured through the built-in profiler of Lava-Loihi v0.6.0 extension. Runtime on CPU was recorded using built-in functions of Gurobi and the energy usage was recorded using pyJoules [4] on a workstation with an Intel Core i7-11700K CPU @ 3.6GHz and 32GB RAM running Ubuntu 20.04.6 LTS.

Results Tabs. 1 and 2 display the results. Overall, SF-HMVC-Loihi was comparable to QUBO-Loihi in all three aspects: solution quality, runtime, and energy usage (the discrepancy between them will be clearer in the next experiment). ILP-CPU and QUBO-CPU outperformed SF-HMVC-Loihi in terms of solution quality (see Tab. 1), since both ILP-CPU and QUBO-CPU were globally optimal methods. The equal solution quality of the global methods also indicated the existence of a suitable penalty weight for QUBO (4) in the grid search range of λ. On runtime (see Tab. 2), though ILP-CPU and QUBO-CPU solved many instances in under 1 s, a few instances took them minutes to hours to solve. In contrast, SF-HMVC-Loihi consistently solved all instances within 10 s. Also, on energy usage, our method significantly outperformed ILP-CPU and QUBO-CPU.

Table 3: Solution quality $\|\mathbf{z}\|_1$ of the HMVC methods for all synthetic HMVC instances. Infeasible solutions are indicated in red, with number of constraint violations in brackets. `N/A` means that the SNN was not able to be embedded into the Loihi 2 due to capacity limitations.

Instance	$\|V\|$	$\|F\|$	`ILP-CPU`	`QUBO-CPU`	`QUBO-Loihi`	`SF-HMVC-Loihi`
3-uniform_HMVC01	30	981	3	3	3 (68)	3
3-uniform_HMVC02	30	2129	9	9	19 (1214)	9
3-uniform_HMVC03	30	2888	15	15	`N/A`	15
3-uniform_HMVC04	30	3327	22	30	`N/A`	29
3-uniform_HMVC05	50	3506	5	5	`N/A`	5
3-uniform_HMVC06	50	6538	8	8	`N/A`	8
3-uniform_HMVC07	50	7333	10	50	`N/A`	10
3-uniform_HMVC08	70	7326	4	4	`N/A`	4
3-uniform_HMVC09	100	5979	20	20	`N/A`	20
3-uniform_HMVC10	200	7430	10	10	`N/A`	19
4-uniform_HMVC11	30	4914	3	15	`N/A`	5

5.2 HMVC

Dataset We generated 11 synthetic HMVC instances as follows: first, a set of N vertices V were created, then a set of K degree-r hyperedges F were randomly generated. The values of N, K and r were selected to ensure that the SNN (for SF-HMVC) could fit on Loihi 2.

Competitors `SF-HMVC-Loihi` was compared to the following methods:

- `ILP-CPU`: HMVC was solved using the ILP Eq. (6) with Gurobi [18] and evaluated on CPU.
- `QUBO-CPU`: HMVC was solved using QUBO Eq. (9) with Gurobi [18] and evaluated on CPU. We set $\lambda = 10$.
- `QUBO-Loihi`: HMVC was solved using QUBO Eq. (9) with NEBM-based SNN (see Sec. 3.2) and evaluated on Intel Loihi 2 [30]. We set $T = 32, r_i = 8, \lambda = 10$.

All hyperparameters of these competitors were selected using grid search. The configuration that demonstrated consistent performance across all problem instances was selected for each method. For `SF-HMVC-Loihi`, we set $T = 8, r_i = 8$.

Results Tabs. 3 and 4 display the results. As expected, `ILP-CPU` outperformed `SF-HMVC-Loihi` in solution quality, since `ILP-CPU` was a global method. Interestingly, our method occasionally found better solutions than `QUBO-CPU`. This was probably because, as the problem difficulty (number of variables) increases, it became more challenging for `QUBO-CPU`. Note that `SF-HMVC-Loihi` handled all instances reasonably well. In contrast, `QUBO-Loihi` either could not find feasible solutions, or correspoding SNN could not be embedded onto Loihi 2. This suggests that the introduction of slack variables in `QUBO-Loihi` made the search space and/or problem size too large.

As presented in Table 4, while `ILP-CPU` performed better than `SF-HMVC-Loihi` in terms of runtime, our method was more efficient in terms of energy consumption. Furthermore, our method significantly surpassed `QUBO-CPU` and `QUBO-Loihi` in both runtime and energy efficiency.

6 Limitations and conclusions

Several limitations of our work can be identified:

L1: The capacity of the neuromorphic computer available to us (a single Intel Loihi 2 chip [30] which has 128 neuromorphic cores) was relatively low, which prevented testing of large problem instances.

L2: There is a lack of public benchmarks on HMVC problems. L1 and L2 together precluded an assessment of the generalizability of the methods more practical problem instances.

L3: Changing the seed of the random number generator on Intel Loihi 2 was inaccessible to us, which precluded error bars in solution quality.

https://doi.org/10.52202/079017-2124

Table 4: Runtime (in seconds) and energy usage (in Joules) of HMVC methods. Note that ∞ indicates that the energy was too high for pyJoules to measure, while `N/A` indicates the instance could not be embedded into Loihi 2 due to capacity limitations.

Instance	ILP-CPU		QUBO-CPU		QUBO-Loihi		SF-HMVC-Loihi	
	Time	**Energy**	**Time**	**Energy**	**Time**	**Energy**	**Time**	**Energy**
3-uniform_HMVC01	0.00	0.24	198.38	14748.67	57.82	18.89	1.02	0.04
3-uniform_HMVC02	0.01	0.27	1464.06	110916.35	174.32	269.43	1.00	0.02
3-uniform_HMVC03	0.09	4.64	21.89	1548.77	N/A	N/A	1.00	0.04
3-uniform_HMVC04	0.26	15.79	>3600	27582.99	N/A	N/A	1.02	0.01
3-uniform_HMVC05	0.01	0.58	>3600	∞	N/A	N/A	1.49	0.07
3-uniform_HMVC06	0.02	1.03	>3600	33883.58	N/A	N/A	1.78	0.08
3-uniform_HMVC07	0.03	1.52	>3600	54487.73	N/A	N/A	1.52	0.16
3-uniform_HMVC08	0.02	0.67	>3600	∞	N/A	N/A	2.32	0.05
3-uniform_HMVC09	0.02	0.66	>3600	∞	N/A	N/A	3.18	0.03
3-uniform_HMVC10	0.02	2.50	>3600	19186.52	N/A	N/A	6.03	1.38
4-uniform_HMVC11	0.02	0.45	>3600	106312.87	N/A	N/A	1.18	0.03

L4: SNNs are ultimately heuristic algorithms, which complicate theoretical analyses on solution quality and runtime complexity.

Despite the limitations above, the results showed clear trends of the greater scalability of the proposed method SF-HMVC, in that it was able to solve HMVC problem instances where the existing method could not. Moreover, SF-HMVC on Loihi 2 exhibited measurably lower energy consumption than global solvers on CPU, further supporting neuromorphic computing as an energy-efficient alternative.

Acknowledgments and Disclosure of Funding

We would like to acknowledge Intel Labs and the Intel Neuromorphic Research Community (INRC) for providing access to Loihi 2. We thank Intel's Neuromorphic Computing Lab (NCL) for developing NEBM algorithm and NEBM-based QUBO solvers, which was central to our work. We thank Philipp Stratmann from NCL for the documentation related to NEBM. We also thank all the members of NCL for their support on technical issues. Tat-Jun Chin is SmartSat CRC Professorial Chair of Sentient Satellites.

References

[1] Intel Builds World's Largest Neuromorphic System to Enable More Sustainable AI. https://www.intel.com/content/www/us/en/newsroom/news/intel-builds-worlds-largest-neuromorphic-system.html.

[2] Lava: A software framework for Neuromorphic Computing, Intel Corporation. https://github.com/lava-nc/lava.

[3] Quadratic Unconstrained Binary Optimization (QUBO) with Lava, Intel Corporation. https://github.com/lava-nc/lava-optimization/blob/main/tutorials/tutorial_02_solving_qubos.ipynb.

[4] Spirals, 2019. pyjoules. https://pyjoules.readthedocs.io/en/latest/.

[5] Vertex cover in hypergraphs, Wikipedia. https://en.wikipedia.org/wiki/Vertex_cover_in_hypergraphs.

[6] M. Z. Alom, B. V. Essen, A. T. Moody, D. P. Widemann, and T. M. Taha. Quadratic unconstrained binary optimization (qubo) on neuromorphic computing system. In *International Joint Conference on Neural Networks*, 2017.

[7] N. Bansal and S. Khot. Inapproximability of hypergraph vertex cover and applications to scheduling problems. In S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *Automata, Languages and Programming*, pages 250–261, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[8] J. Binas, G. Indiveri, and M. Pfeiffer. Spiking analog vlsi neuron assemblies as constraint satisfaction problem solvers. In *IEEE International Symposium on Circuits and Systems*, 2016.

[9] C. Chauve, M. Patterson, and A. Rajaraman. Hypergraph covering problems motivated by genome assembly questions. In T. Lecroq and L. Mouchard, editors, *Combinatorial Algorithms*, pages 428–432. Springer Berlin Heidelberg, 2013.

[10] S. Chiavazza, S. M. Meyer, and Y. Sandamirskaya. Low-latency monocular depth estimation using event timing on neuromorphic hardware. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4071–4080, 2023.

[11] K. Corder, J. V. Monaco, and M. M. Vindiola. Solving vertex cover via ising model on a neuromorphic processor. In *IEEE International Symposium on Circuits and Systems*, 2018.

[12] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[13] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud. Advancing neuromorphic computing with loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5):911–934, 2021.

[14] Y. Fang and A. S. Lele. Solving quadratic unconstrained binary optimization with collaborative spiking neural networks. In *IEEE International Conference on Rebooting Computing*, 2022.

[15] 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*, 2022.

[16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

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

[18] Gurobi. https://www.gurobi.com/. Accessed: 2024-05-22.

 https://doi.org/10.52202/079017-2124

[19] A. Guzzo, A. Pugliese, A. Rullo, and D. Saccà. Intrusion detection with hypergraph-based attack models. In M. Croitoru, S. Rudolph, S. Woltran, and C. Gonzales, editors, *Graph Structures for Knowledge Representation and Reasoning*, pages 58–73. Springer International Publishing, 2014.

[20] D. S. Johnson and M. A. Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc., 1996.

[21] Z. Jonke, S. Habenschuss, and W. Maass. Solving constraint satisfaction problems with networks of spiking neurons. *Frontiers in neuroscience*, 10:156676, 2016.

[22] A. Kumar and L. Vijaykumar. On neuromorphic computing: A case study on radio resource allocation with lava software framework. 2023.

[23] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In P. Berkhin, R. Caruana, and X. Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 420–429. ACM, 2007.

[24] A. R. Mangalore, G. A. Fonseca, S. R. Risbud, P. Stratmann, and A. Wild. Neuromorphic quadratic programming for efficient and scalable model predictive control: Towards advancing speed and energy efficiency in robotic control. *IEEE Robotics & Automation Magazine*, 2024.

[25] P. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, B. Brezzo, I. Vo, S. K. Esser, R. Appuswamy, B. Taba, A. Amir, M. Flickner, W. P. Risk, R. Manohar, and D. S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345:668 – 673, 2014.

[26] S. M. Mniszewski. Graph partitioning as quadratic unconstrained binary optimization (qubo) on spiking neuromorphic hardware. In *Proceedings of the International Conference on Neuromorphic Systems*.

[27] N. Mohseni, P. L. McMahon, and T. Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379, 2022.

[28] T. Moraitis, A. Sebastian, and E. Eleftheriou. Optimality of short-term synaptic plasticity in modelling certain dynamic environments. *arXiv preprint arXiv:2009.06808*, 2020.

[29] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer New York, New York, NY, 2006.

[30] G. Orchard, E. P. Frady, D. B. D. Rubin, S. Sanborn, S. B. Shrestha, F. T. Sommer, and M. Davies. Efficient neuromorphic signal processing with Loihi 2. In *IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259, 2021.

[31] C. Ostrau, C. Klarhorst, M. Thies, and U. Rückert. Comparing neuromorphic systems by solving sudoku problems. In *International Conference on High Performance Computing & Simulation*, 2019.

[32] A. Pierro, P. Stratmann, G. A. F. Guerra, S. Risbud, T. Shea, A. R. Mangalore, and A. Wild. Solving qubo on the loihi 2 neuromorphic processor. *arXiv preprint arXiv:2408.03076*, 2024.

[33] Y. Schnider, S. Woźniak, M. Gehrig, J. Lecomte, A. Von Arnim, L. Benini, D. Scaramuzza, and A. Pantazi. Neuromorphic optical flow and real-time implementation with event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4128–4137, 2023.

[34] C. D. Schuman, S. R. Kulkarni, M. Parsa, J. P. Mitchell, B. Kay, et al. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.

[35] S. B. Shrestha, J. Timcheck, P. Frady, L. Campos-Macias, and M. Davies. Efficient video and audio processing with loihi 2. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13481–13485. IEEE, 2024.

[36] C. Yakopcic, N. Rahman, T. Atahary, T. M. Taha, and S. Douglass. Leveraging the manycore architecture of the Loihi spiking processor to perform quasi-complete constraint satisfaction. In *International Joint Conference on Neural Networks*, 2020.

 https://doi.org/10.52202/079017-2124

Appendix

Algorithm 2 NEBM-based SNN for energy minimization (**note:** the algorithm is executed for each neuron n_i on the neuromorphic hardware in a parallel way; see [13] for details).

Require: Weight matrix $\mathbf{W}$, temperature T and length of refractory period r_i.

Initialize $s_i^{(0)} \leftarrow 0,\ \Delta s_i^{(0)} \leftarrow 0,\ u_i^{(0)} \leftarrow -w_{ii},\ \text{refract\_counter}_i^{(0)} \leftarrow 0$
for each timestep t **do**
    $u_i^{(t)} \leftarrow u_i^{(t-1)} + \sum_{j \neq i}^{N} w_{ij} \Delta s_j^{(t-1)}$
    $p_i^{(t)} \leftarrow \dfrac{1}{1 + \exp(u_i^{(t)}/T)}$
    $\theta_i \leftarrow rand(0, 1)$
    if neuron n_i is not in refractory period **then**
        if $p_i^{(t)} \geq \theta_i$ **then**
            $s_i^{(t)} \leftarrow 1$
        else
            $s_i^{(t)} \leftarrow 0$
    else
        $s_i^{(t)} \leftarrow s_i^{(t-1)}$
        $\text{refract\_counter}_i^{(t)} \leftarrow \max(\text{refract\_counter}_i^{(t-1)} - 1, 0)$
    $\Delta s_i^{(t)} \leftarrow s_i^{(t)} - s_i^{(t-1)}$
    send $\Delta s_i^{(t)}$ to connected neurons
    if $\Delta s_i^{(t)} \neq 0$ **then**
        neuron n_i enters refractory period
        $\text{refract\_counter}_i^{(t)} \leftarrow r_i$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: A novel SNN is introduced to solve HMVC. The derivation in Sec. 4 show the proposed SNN does not require new slack variables, as claimed in the abstract and introduction. The experimental results align with the claims that the proposed SNN consistently yields high quality solutions on synthetic and real HMVC instances where the established method mostly fails (to reach feasibility), and that the proposed SNN measurably consumed less energy on the neurmorphic hardware than the CPU solution.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 6 discusses the limitations of the work (see L1 to L4).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

https://doi.org/10.52202/079017-2124

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have provided a detailed methodology description and pseudo code in Sec. 4. The data generation process and the experiment setup are fully described in Sec. 5.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release all MVC and HMVC problem instances after the peer reviewing period. Release of source code is not possible at the moment due to internal organizational policy and contractual restrictions.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified all experiment details in Sec. 5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars are not available due to hardware limitations (see L3 in Sec. 6).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-2124

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Sec. 5 for details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and followed the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: The paper focuses on combinatorial optimization at the fundamental level, which does not have immediate societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: High risk data or models are not involved in this research.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have properly referenced all relevant data, code and papers.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

https://doi.org/10.52202/079017-2124

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have provided the pseudo code of the algorithm and all implementation details of the method.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No crowdsourcing or research with human subjects involved.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-2124