# Memristor-based hardware and algorithms for higher-order Hopfield optimization solver outperforming quadratic Ising machines

Mohammad Hizzani*†, Arne Heittmann*, George Hutchinson‡, Dmitrii Dobrynin*†,
Thomas Van Vaerenbergh§, Tinish Bhattacharya‡, Adrien Renaudineau¶, Dmitri Strukov‡, John Paul Strachan*†
*Forschungszentrum Jülich GmbH, Jülich, Germany
†RWTH Aachen University, Aachen, Germany
‡University of California, Santa Barbara, Santa Barbara, CA, USA
§Hewlett Packard Enterprise, Brussels, Belgium
¶Université Paris-Saclay, Paris, France
Corresponding Email: m.hizznai@fz-juelich.de

***Abstract*—Ising solvers offer a promising physics-based approach to tackle the challenging class of combinatorial optimization problems. However, typical solvers operate in a quadratic energy space, having only pair-wise coupling elements which already dominate area and energy. We show that such quadratization can cause severe problems: increased dimensionality, a rugged search landscape, and misalignment with the original objective function. Here, we design and quantify a higher-order Hopfield optimization solver, with 28nm CMOS technology and memristive couplings for lower area and energy computations. We combine algorithmic and circuit analysis to show quantitative advantages over quadratic Ising Machines (IM)s, yielding 48x and 72x reduction in time-to-solution (TTS) and energy-to-solution (ETS) respectively for Boolean satisfiability problems of 150 variables, with favorable scaling.**

***Index Terms*—Optimization, Hopfield neural network, Ising machine, Boolean satisfiability**

## I. Introduction

With diminishing performance gains from CMOS in traditional hardware, alternative mechanisms to deliver high-performance, particularly for challenging (NP-hard) optimization problems has growing importance. A variety of physical systems designed to accelerate intractable combinatorial optimization has attracted recent attention, in many cases implementing a quadratic Ising solver in a novel computing substrate such as superconductor-based quantum annealing, optics, CMOS-based coupled oscillators, etc [1]–[8].

An Ising-based solver typically maps the desired optimization problem into a quadratic energy function to be minimized. Quadratic terms represent pair-wise interactions, which are typically easy to implement in a physical system, such as with resistive or capacitive couplings, while higher-order (3-body or higher) are more challenging. Numerous approaches exist for conversion of an arbitrary target objective function into quadratic form (so called QUBO: Quadratic Unconstrained Binary Optimization). However, it is shown here that a quadratic-only solver is severely disadvantaged compared to a solver supporting higher-order interactions (so called PUBO: Polynomial Unconstrained Binary Optimization). For example, there are naturally cubic terms in the case of 3-SAT problems. Quadratization requires the introduction of additional auxiliary variables, typically much more than the original problem variables, leading to an exponentially increased search space. The new search space may also have additional ruggedness that hinders a solver, as will be illustrated later.

In the present work, we utilized 3-SAT optimization problems as representative for our proposed PUBO implementation. We chose 3-SAT because it is NP-Complete and has widespread applications [9]–[11].

This paper is structured as follows: Section II gives a brief introduction to Hopfield Neural Networks and 3-SAT. Section III compares QUBO and PUBO algorithms and highlights the advantages of the latter through novel energy landscape analysis. Finally, in Section IV, we present hardware circuit designs supporting both algorithms, enabling quantitative comparisons of key metrics TTS and ETS.

## II. Background and Motivation

Hopfield Neural Networks (HNNs), originally proposed by J. J. Hopfield in 1982 [12], are a type of recurrent neural network, the dynamics of which is governed by an overall energy function, as in Eq. 1. Just as in physical systems, the network evolves to minimize this energy, leading to a neuron update rule proportional to the negative energy gradient with respect to each dynamical neuron.

$$E(\{s\}) = -\frac{1}{2}\sum_{\langle ij \rangle} w_{ij} s_i s_j + \sum_i b_i s_i, \;\; s_i \in \{-1, 1\} \quad (1)$$

Ising Machines (IM) operate based on a Hamiltonian, a mathematical description of the energy of a physical system. In the context of IMs, the Hamiltonian represents the problem to be solved. It consists of two key components: the "spins" representing the variables in the problem, and their interactions which are quantified as coupling strengths (manifesting similar to HNN eq. 1). By manipulating these spins, IMs seek the lowest energy state of the system, which corresponds to the optimal solution for the given problem.

Satisfiability problems are typically represented in conjunctive normal form (CNF) as shown in eq. 2. The goal is to find input values that satisfy the Boolean function, making it `true`. In $k$-SAT, 'k' represents the maximum number of literals in a SAT clause (eq. 2). Each clause, such as $(x_i \vee \neg x_j \vee x_k)$, consists of positive $x_i$ and negative $\neg x_j$ literals.

$$F(x_1, \ldots, x_N) = (x_i \vee \neg x_j \vee x_k) \wedge (x_a \vee x_b \vee \neg x_c) \wedge \ldots \quad (2)$$

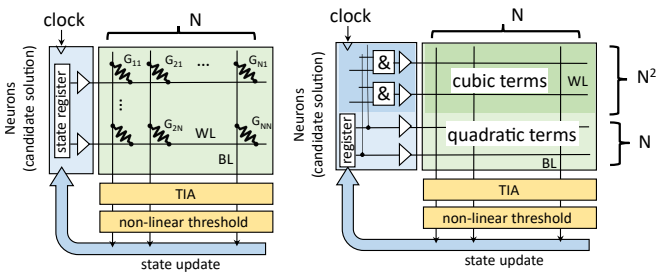


Figure 1: Left: QUBO HNN that utilizes memristors to store synapse weights in conductance to perform VMM. Right: PUBO HNN with support for cubic interactions. N is number of variables ($n_{var}$)

SAT solvers can be exact or non-exact. Exact solvers, like DPLL and CDCL [13]–[17], verify SAT problem satisfiability or unsatisfiability. Non-exact solvers, such as SLS algorithms, use heuristics and random assignments for local search, including variable flips. Examples include QUBO IMs/HNNs, PUBO HNNs, and others relying on the `make` and `break` evaluation such as WalkSAT, GSAT, and probSAT [18], [19].

Recent advancements in non-volatile analog memory technologies [20], [21], particularly 2-terminal memristive devices [22], [23], have made mixed-signal implementations of certain computing primitives increasingly attractive due to their low area footprint, monolithic 3D potential, and multi-bit capacity. Furthermore, a cross-bar array of memristive devices can store coupling matrices and efficiently perform vector-matrix multiplications (VMM), as needed for both QUBO (2D matrix) and PUBO (3D tensor), allowing for computing-in-memory (CIM), as illustrated in Fig. 1.

## III. Comparing Quadratic and Higher-Order Solvers and Energy Landscapes

This section compares quadratic (QUBO) and higher-order (PUBO) solvers in terms of optimized algorithms, the solution space that must be navigated, and introduces some novel techniques for visualization. Performance is compared quantitatively, and 3-SAT problems are used for concrete analysis . The development and testing of algorithms necessitates adaptability while considering hardware compatibility. To ensure a fair comparison, both algorithms underwent in-depth optimization.

### A. QUBO

Quadratization of higher-order problems, such as 3-SAT, typically involve substituting products of variables $x_i, x_j$ with a new auxiliary variable $y$ and introducing a constraint to maintain this substitution, converted into a penalty term added to the objective function (Alg. 1). A penalty term, initially introduced by Rosenberg [24], emerged as highly effective, especially when combined with stochastic group parallel updates. In essence, a QUBO HNN is governed by an energy function:

$$E(\{s\}) = \sum_{\langle ij \rangle} w_{ij} s_i s_j + \sum_i b_i s_i + c; \begin{cases} \{s\} = \{x\} \cup \{y\} \\ s \in \{0,1\} \end{cases} \quad (3)$$

This function comprises quadratic terms, linear terms (biases), and a constant. While the resulting QUBO mapping is indeed quadratic, it exhibits high sparsity since the auxiliary variables ($y$) only couple to the original variables ($x$). This sparsity provides an opportunity for parallel updates within the algorithm, employing simulated annealing to overcome barriers.

**Algorithm 1** Quadratizing a 3-SAT with a Rosenberg penalty

**Require:** 3-SAT problem in CNF
//example $(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_3 \vee x_4)$
**Convert** dissatisfiability $\leftarrow$ Satisfiability (negating)
//example $(x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge \neg x_3 \wedge \neg x_4)$
**Unroll** each clause into penalty
//example $(x_1(1-x_2)(1-x_3)) + ((1-x_1)(1-x_3)(1-x_4))$
**Substitute** the product of two variables in cubic term with aux variable
//example $(y_1(1-x_3)) + (y_2(1-x_4)) : x_1(1-x_2) = y_1, (1-x_1)(1-x_3) = y_2$
**Add** penalty for each auxiliary equality constraint
//example $(y_1(1-x_3)) + (y_2(1-x_4)) + P(x_1(1-x_2) - 2x_1y_1 - 2(1-x_2)y_1 + 3y_1) + P((1-x_1)(1-x_3) - 2(1-x_1)y_2 - 2(1-x_3)y_2 + 3y_2) : P$ is a parameter

To implement stochastic parallel updates, we randomly distribute neurons into a desired number of groups at each step, then update neurons of each group sequentially. This follows an annealing schedule starting at high temperature to zero. We found stochastic parallel updates significantly improve convergence time of QUBO HNN for 3-SAT compared with all single-neuron update algorithms tested.

A detailed analysis of the QUBO HNN's behavior revealed cases where the solver traversed through a satisfiability condition (i.e. 3-SAT problem solved) despite the QUBO energy not reaching zero, leading to a departure from the solution. To address this challenge, we introduced an additional component, the SAT checker, to monitor the solver's evolution cycle and halt the search upon finding satisfiability. This enhancement significantly accelerated the QUBO HNN (Fig. 5a), which we implemented using a cross-bar array with variables at word-line and clauses at bit-line, the clause is satisfied when the current is above a threshold representing zero.

### B. PUBO

To natively accommodate the energy function of 3-SAT within the Hopfield Neural Network (HNN), cubic interactions must be supported. The conversion of 3-SAT into this function closely resembles Alg. 1, albeit without variable substitution, resulting in the following equation:

$$E(\{s\}) = \sum_{\langle ijk \rangle} ijk w_{ijk} s_i s_j s_k + \sum_{\langle ij \rangle} w_{ij} s_i s_j + \sum_i b_i s_i + c; \; s \in \{0,1\} \quad (4)$$

Adhering to the classical HNN update rule of one neuron per step was observed to give quickest convergence and superior scaling compared to our best QUBO (see Fig. 5a). However, top performance was found by implementing a modified update rule inspired by Aramon, et al. [8] that increases the probability of flips. This starts by computing state changes for all spins, selects a flip if it exists (focus), if not it adds $\Delta E += E_{\text{offset}}$ to all gradients to induce a flip on the next step (offset), and if a flip is found set $\Delta E = 0$.

### C. Landscape Advantages of PUBO over QUBO

The degradation of the QUBO version of a higher-order problem is attributed to the substitution of original variables $x$ and augmenting the landscape with auxiliary variables $y$.

This process not only increases exponentially the search space (Fig. 2a), but also the new landscape is not faithful to the native space (see Fig. 2b), whereby reducing QUBO energy does not perfectly correlate to solving more SAT clauses. An

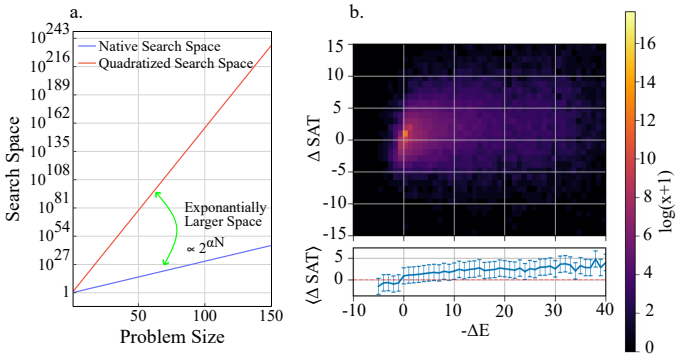


Figure 2: a. Search space vs problem size for quadratized and native space for hard 3-SAT random instances with number of clauses 4.23× number of variables, resulting in $2^N$ and $2^{5.23}$ size of search space for native and quadratized representation respectively. b. Histogram of running 3-SAT QUBO solver, showing energy reduction ($-\Delta E$) does not correlate well with satisfying more clauses $\Delta$SAT .

intrinsic property of the 3-SAT problem is the degeneracy of the configuration space, that it is possible to group configurations into large valleys based on their connectivity with each other by zero-energy-change bit-flips (e.g. Fig. 3 left), an HNN solver will diffuse through the optimization *landscape* until a downward path is found, or a barrier at the valley border is overcome. When mapped to a QUBO problem $g(\mathbf{x}, \mathbf{y})$ with the Rosenberg penalty,

$$f(\mathbf{x}) = \min_{\mathbf{y}} g(\mathbf{x}, \mathbf{y}) \,, \tag{5}$$

the QUBO landscape increases the ruggedness (e.g. Fig. 3 right) of the original PUBO manifold due to auxiliary variables $\mathbf{y}$ for each fixed configuration $\mathbf{x}$. To numerically estimate

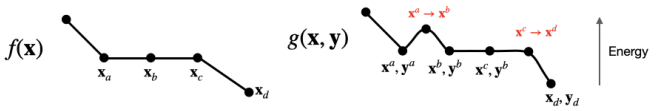


Figure 3: The deformation of the native (PUBO) energy landscape $f(\mathbf{x})$ when reformulated as a QUBO problem $g(\mathbf{x}, \mathbf{y})$ with the penalty quadratization of Eq. (5). Every vertex denotes a PUBO/QUBO configuration, edges represent bit-flip neighbours. $\mathbf{x}_a$ and $\mathbf{x}_b$ have different optimal $\mathbf{y}_a$ and $\mathbf{y}_b$, while $\mathbf{y}_b$ is optimal for both $\mathbf{x}_b$ and $\mathbf{x}_c$. An unstable point $\mathbf{x}_c$ in PUBO can become a saddle in QUBO.

degenerate properties in PUBO/QUBO landscapes, we employ the Generalized Wang-Landau algorithm [25] to sample the lowest 200 degenerate valleys for 100 instances from SATLIB *(uf50-[901,1000])*. Following sampling, we discard valleys with zero-barrier exits (saddle points). For each remaining local minimum in a sampled PUBO degenerate valley, denoted as $\{\mathbf{x}\}^i$, we assess the potential for single bit-flip moves $\mathbf{x}_a^i \rightarrow \mathbf{x}_b^i$ in the QUBO space (in PUBO there is a zero barrier $E_{\text{PUBO}}(\mathbf{x}_a^i) = E_{\text{PUBO}}(\mathbf{x}_b^i)$) through the number $N_{\mathbf{y}}$ of combinations of auxiliary variables $\mathbf{y}$ satisfying $E_{\text{QUBO}}(\mathbf{x}_a^i, \mathbf{y}) = E_{\text{PUBO}}(\mathbf{x}_a^i)$. If a barrier $\Delta E(\mathbf{y}) = E_{\text{QUBO}}(\mathbf{x}_b^i, \mathbf{y}) - E_{\text{QUBO}}(\mathbf{x}_a^i, \mathbf{y})$ is overcome with probability $p = \frac{1}{N_{\mathbf{y}}} \sum_{\mathbf{y}} \exp\left(-\Delta E(\mathbf{y})/T\right)$, then we consider $\mathbf{x}_a$ and $\mathbf{x}_b$ connected in QUBO. Based on the resulting QUBO connectivity, we count the QUBO valley entropies $s$, i.e. the number of configurations $\exp\{Ns\}$ within a valley, and complexities $\Sigma(s)$, i.e. the number $(\exp\{N\Sigma(s)\})$ of valleys of size $s$, at different temperatures (see Fig. 4).

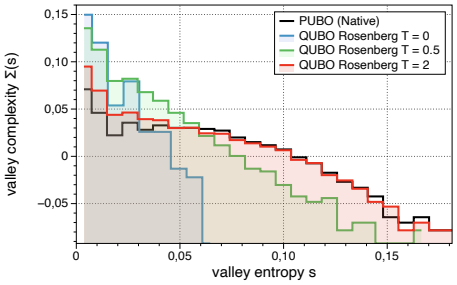


Figure 4: Sampled histogram of local minima valley complexity $\Sigma(s)$ vs valley entropy $s$ for the PUBO and Rosenberg QUBO landscapes, averaged over 100 SATLIB instances *uf50-[901,1000]*. Penalty hyperparameter for QUBO is $P = 0.5$. $T$ is temperature of QUBO connectivity method.

At small temperatures the QUBO landscape is very rugged, i.e. has more disconnected local minima valleys compared to PUBO, which results in higher rejection of local moves and thus drastic performance slow-down. At higher temperature the QUBO energy barriers can of course be overcome and the entropic barriers of the PUBO landscape are restored. However, low-temperature performance determines convergence to solutions. The above results reinforce both quantitatively and conceptually the challenges of a quadratic-limited solver.

## IV. 28nm QUBO and PUBO Hardware Designs

For quantitative analysis and comparison, we designed the needed hardware blocks for both QUBO and PUBO memristor-based solvers. The hardware must offer full flexibility for current and future algorithm development, yet provide realistic performance metrics. Circuit elements at the device level were designed, optimized and simulated with respect to energy consumption and time requirements based on a TSMC 28nm CMOS technology. The central block is the VMM, which is built as a memristor-based CIM architecture [26]. The needed periphery includes driver circuits to activate word lines, current-voltage converters (transimpedance amplifiers, TIAs [27]), DACs [28] for additive noise signals in annealing, comparators [29], digital random number generators (PRNGS) [30] and custom 1-out-of-n encoders (n/2-out-of-n encoders) for updating state vectors.

Circuit performance is expected to be dominated by an extended interconnect, thus it is first necessary to have an area planning of the individual components, after which an estimate of expected interconnect lengths of individual modules can be made. The interconnect lengths are then converted into effective capacitances and integrated into the Spectre circuit simulation model. From simulation, realistic values for delays and switching energies are obtained.

Fig. 6 shows a designed floor plan of true-scale PUBO/QUBO implementations for 20-variable 3-SAT problems. Transient simulations of realistic state vector sequences provided numbers for aggregated power and timing numbers

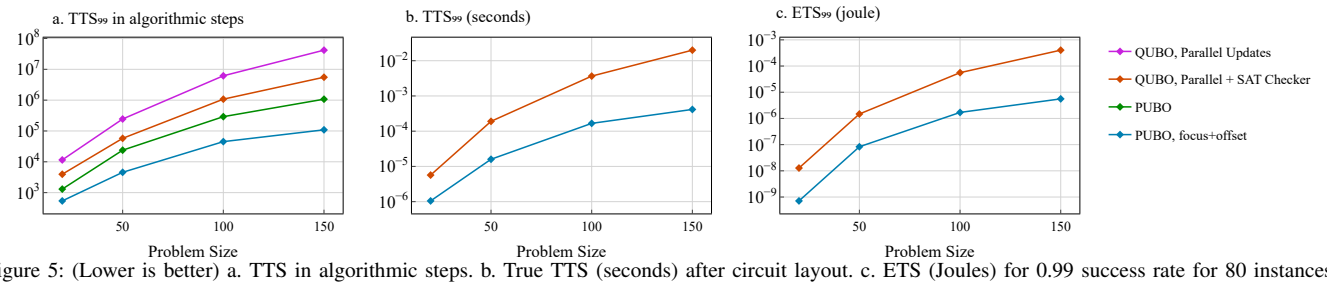


Figure 5: (Lower is better) a. TTS in algorithmic steps. b. True TTS (seconds) after circuit layout. c. ETS (Joules) for 0.99 success rate for 80 instances at each size. Curves are median values for each size.

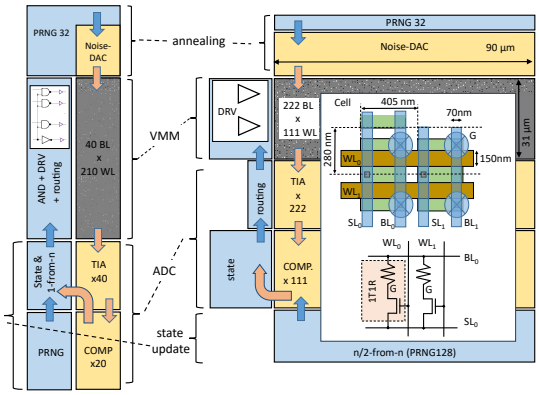


Figure 6: Floor plan of PUBO (left) and QUBO (right) for 20-variable 3-SAT problems with true-scaled circuit blocks based on 28 nm CMOS. Areas colored gray represent weight matrices, areas colored blue represent digital components, and areas colored ocher are analog/mixed-signal components.

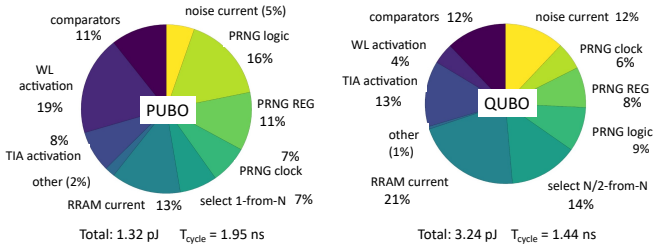


Figure 7: Energy total and breakdown for the various components in the PUBO (left) and QUBO (right) solvers.

which were re-scaled to average energy contributions per cycle. Fig. 7 shows the corresponding distribution of the energy contributions to the individual sub-operations.

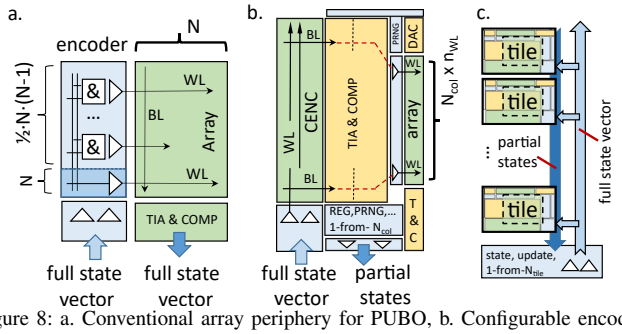


Figure 8: a. Conventional array periphery for PUBO, b. Configurable encoder (CENC) replacing the AND-encoder. True scale layout for $N = 150$ and $N_{col} = 19$, c. Tile architecture and interconnect.

Scaling PUBO/QUBO to problems with larger numbers of variables poses challenges for the hardware implementation. First, the number of word lines (PUBO) grows with $\sim \frac{1}{2}N(N-1)$ which leads to highly asymmetric, elongated layouts for $N \gg 20$. The gradient matrix (for well-randomized problems) contains many rows in which weights are zero. With these aspects in mind, we developed a tiling architecture to support scaling. Instead of encoding all possible pairwise products of variables (see Fig. 8a), the encoder array is replaced by a memristor-based Configurable Encoder (CENC) (see Fig. 8). The CENC receives the complete state vector for input. For each horizontal bit line of the CENC block, a word line is now provided in the gradient array. The CENC array is programmed so that only active lines of the gradient matrix (associated to specific state bit combinations) are represented by corresponding patterns in the CENC.

The circuit components designed for the smaller 20-var implementation are utilized in the fully scalable CENC circuit. For a rectangular, fully filled layout of a tile, $n_{WL} = 400$ with variable number of CENC inputs proves to be favorable. Each tile can thus accommodate sub-problems of larger problems. For given 3-SAT problems from the SATLIB, an upper bound of possible state variables is obtained whose gradient can be fully mapped with a restricted and fixed number of word lines. In particular, for $n_{WL} = 400$, $N_{col} = 19$ state variables are obtained. An N-variable problem is thus mapped to $N_{tile} = \lceil N/N_{col} \rceil$ tiles (see Fig. 8c). A separate unit holds the state vector and the update logic, distributes the state vector as well as control signals to the tiles and collects the partial state vectors and other control signals from the tiles. Again, average energy and timing values are obtained from circuit simulation of the tiled architecture and translated into true TTS and ETS values (see Fig. 5(b and c)).

The quantitative hardware designs of QUBO and PUBO implementations are now combined with the algorithmic explorations of Section III. We see (Fig. 7) that PUBO cycles consume $2.45\times$ less energy, but take $1.35\times$ longer than QUBO. However, Section III showed a significant algorithmic advantage for PUBO, reducing the needed cycles to converge to solutions. This PUBO advantage scales with problem size (Fig. 5a.), and our hardware analysis here quantifies the circuit costs associated with the problem scaling. We thus combine both the hardware and algorithmic analysis in Figs. 5b and c to yield time-to-solution and energy-to-solution numbers. Our analsyis shows a substantial and growing advantage of PUBO, with $48\times$ and $72\times$ improved speed and energy, respectively, over QUBO at the largest problem sizes. This performance gap is expected to continue for larger problems, which is future work.

## V. Conclusion

In this work we developed a memristor-based hardware implementation of a higher-order HNN (i.e. PUBO) showing

significantly reduced time and energy consumption compared to more standard quadratic IM (QUBO). We separately quantified the gains from both the hardware and algorithmic aspects. We showed operating in the higher-order search space reduced variables (exponentially smaller search space), provides a more faithful energy landscape to the original problem landscape, and can be smoother for the solver to explore.

## VI. Acknowledgment

We gratefully acknowledge computing time on the supercomputer JURECA [31] at Forschungszentrum Jülich under grant no. 'optimization'. We also are gratfully acknowledge the generous funding of this work with is under NEUROTEC II (Verbundkoordinator / Förderkennzeichen: Forschungszentrum Jülich: 16ME0398K) by the Bundesministerium für Bildung und Forschung. This project was also under the DARPA QuICC project under contract no. FA8650-23-3-7313.

## References

[1] N. Mohseni, P. L. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Reviews Physics*, vol. 4, no. 6, pp. 363–379, 2022.

[2] M. J. Schuetz, J. K. Brubaker, and H. G. Katzgraber, "Combinatorial optimization with physics-inspired graph neural networks," *Nature Machine Intelligence 2022 4:4*, vol. 4, pp. 367–377, 4 2022.

[3] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura, "High-performance combinatorial optimization based on classical mechanics," *Science Advances*, vol. 7, no. 6, p. eabe7953, 2021.

[4] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.

[5] W. R. Clements, J. J. Renema, Y. H. Wen, H. M. Chrzanowski, W. S. Kolthammer, and I. A. Walmsley, "Gaussian optical ising machines," *Physical Review A*, vol. 96, 2017.

[6] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "A 20k-spin ising chip to solve combinatorial optimization problems with cmos annealing," *IEEE Journal of Solid-State Circuits*, vol. 51, 2016.

[7] R. Hamerly, T. Inagaki, P. L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo, K. Enbutsu, T. Umeki, R. Kasahara, S. Utsunomiya, S. Kako, K. I. Kawarabayashi, R. L. Byer, M. M. Fejer, H. Mabuchi, D. Englund, E. Rieffel, H. Takesue, and Y. Yamamoto, "Experimental investigation of performance differences between coherent ising machines and a quantum annealer," *Science Advances*, vol. 5, 2019.

[8] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers in Physics*, vol. 7, p. 48, Apr 2019.

[9] Y. Vizel, G. Weissenbacher, and S. Malik, "Boolean satisfiability solvers and their applications in model checking," *Proceedings of the IEEE*, vol. 103, 2015.

[10] M. Stojadinović, "Air traffic controller shift scheduling by reduction to csp, sat and sat-related problems," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8656 LNCS, pp. 886–902, 2014.

[11] F. Massacci, L. M. J. of Automated Reasoning, and undefined 2000, "Logical cryptanalysis as a sat problem," *disi.unitn.it*, vol. 24, pp. 165–203, 2000.

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

[13] M. Davis and H. Putnam, "A computing procedure for quantification theory," *Journal of the ACM (JACM)*, vol. 7, 1960.

[14] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Communications of the ACM*, vol. 5, 1962.

[15] R. J. Bayardo and R. C. Schrag, "Using csp look-back techniques to solve real-world sat instances," 1997.

[16] J. P. M. Silva and K. A. Sakallah, "Grasp - a new search algorithm for satisfiability," 1996.

[17] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient sat solver," 2001.

[18] B. Selman, H. A. Kautz, and B. Cohen, "Noise strategies for improving local search," vol. 1, 1994.

[19] A. Balint and U. Schöning, "Choosing probability distributions for stochastic local search and the role of make versus break," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7317 LNCS, pp. 16–29, 2012.

[20] Z. Wang, H. Wu, G. W. Burr, C. S. Hwang, K. L. Wang, Q. Xia, and J. J. Yang, "Resistive switching materials for information processing," *Nature Reviews Materials*, vol. 5, no. 3, pp. 173–195, 2020.

[21] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature nanotechnology*, vol. 15, no. 7, pp. 529–544, 2020.

[22] J. J. Yang, D. B. Strukov, and D. R. Stewart, "Memristive devices for computing," *Nature nanotechnology*, vol. 8, no. 1, pp. 13–24, 2013.

[23] M. Rao, H. Tang, J. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang *et al.*, "Thousands of conductance levels in memristors integrated on cmos," *Nature*, vol. 615, no. 7954, pp. 823–829, 2023.

[24] I. G. Rosenberg, "Reduction of bivalent maximization to the quadratic case," *Cahiers du Centre d'Etudes de Recherche Opérationnelle*, vol. 17, pp. 71–79, 1975.

[25] A. Barbu and S. Zhu, *Monte Carlo Methods*. Springer Nature Singapore, 2020.

[26] C. Li, Y. Li, H. Jiang, W. Song, P. Lin, Z. Wang, J. J. Yang, Q. Xia, M. Hu, E. Montgomery, J. Zhang, N. Davila, C. E. Graves, Z. Li, J. P. Strachan, R. S. Williams, N. Ge, M. Barnell, and Q. Wu, "Large memristor crossbars for analog computing," vol. 2018-May, 2018.

[27] M. Atef and H. Zimmermann, "10gbit/s 2mw inductorless transimpedance amplifier," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2012, pp. 1728–1731.

[28] D. Karadimas, M. Papamichail, and K. Efstathiou, "A most-only r-2r ladder-based architecture for high linearity dacs," 2008.

[29] M. Al-Qadasi, A. Alshehri, A. S. Almansouri, T. Al-Attar, and H. Fariborzi, "A high speed dynamic strongarm latch comparator," vol. 2018-August, 2019.

[30] G. Marsaglia, "Xorshift rngs," *Journal of Statistical Software*, vol. 8, pp. 1–6, 2003.

[31] F. Jülich and J. S. Centre, "Jureca: Data centric and booster modules implementing the modular supercomputing architecture at jülich supercomputing centre," *Journal of large-scale research facilities JLSRF*, vol. 7, pp. A182–A182, 10 2021. [Online]. Available: https://www.journal-of-large-scale-research-facilities.org/index.php/lsf/article/view/182