PAPER • **OPEN ACCESS**

Performance evaluation of coherent Ising machines against classical neural networks

To cite this article: Yoshitaka Haribara *et al* 2017 *Quantum Sci. Technol.* **2** 044002

View the article online for updates and enhancements.

You may also like

- Roadmap for unconventional computing with nanotechnology
Giovanni Finocchio, Jean Anne C Incorvia, Joseph S Friedman et al.
- Computational phase transitions: benchmarking Ising machines and quantum optimisers
Hariphan Philathong, Vishwa Akshay, Ksenia Samburskaya et al.
- Analog errors in Ising machines
Tameem Albash, Victor Martin-Mayor and Itay Hen

This content was downloaded from IP address 18.118.30.253 on 01/05/2024 at 01:32

Quantum Science and Technology

OPEN ACCESS

RECEIVED
9 April 2017

REVISED
17 July 2017

ACCEPTED FOR PUBLICATION
24 July 2017

PUBLISHED
14 August 2017

Original content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence.

Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.

PAPER

Performance evaluation of coherent Ising machines against classical neural networks

Yoshitaka Haribara[1,2], **Hitoshi Ishikawa**[3], **Shoko Utsunomiya**[4], **Kazuyuki Aihara**[1,2] and **Yoshihisa Yamamoto**[5]

[1] Department of Mathematical Informatics, University of Tokyo, Tokyo 113-8656, Japan
[2] Institute of Industrial Science, University of Tokyo, Tokyo 153-8505, Japan
[3] PEZY Computing, Tokyo 101-0052, Japan
[4] National Institute of Informatics, Tokyo 101-8430, Japan
[5] ImPACT Program, Japan Science and Technology Agency, Tokyo 102-0076, Japan

E-mail: haribara@sat.t.u-tokyo.ac.jp

Keywords: degenerate optical parametric oscillator, measurement feedback system, combinatorial optimization problem, maximum cut problem

Abstract

The coherent Ising machine is expected to find a near-optimal solution in various combinatorial optimization problems, which has been experimentally confirmed with optical parametric oscillators and a field programmable gate array circuit. The similar mathematical models were proposed three decades ago by Hopfield *et al* in the context of classical neural networks. In this article, we compare the computational performance of both models.

1. Introduction

In recent trends in semiconductor technologies, the Moore's law is slowing down mainly due to the limitation of micro-fabrication heat dissipation and communication bottleneck problems on a chip [1, 2]. Many efforts to boost the processor performance have been made for parallelized architectures including GPU, other multi/many-core processors, and neuromorphic hardwares [3]. An optics-based special purpose computer, which is named the coherent Ising machine (CIM), has been proposed to exploit a rapid physical convergence time for accelerating the solution search in hard optimization problems [4].

One of the well-known examples of combinatorial optimization problems is a maximum cut problem (MAX-CUT) on a graph, which is essentially equivalent to the Ising model in statistical mechanics [5, 6]. It is a problem to find the largest cut in a given graph $G = (V, E)$, where the number of edges at the boundary of a partition of vertices into two subsets is maximized. The size of the cut is defined as the total weight of edges separated by the cut, i.e., edges which have each endpoints in the different sides of the cut. This objective function can be written as

$$\mathrm{CUT}(x) = \sum_{1 \leqslant i < j \leqslant n} w_{ij} \frac{1 - x_i x_j}{2}, \tag{1}$$

where the graph order $n = |V|$ is the number of vertices, w_{ij} is the weight of the edge $(i, j) \in E$ and $x_i = \pm 1$ is a binary value indicating which side of the cut the vertex $i \in V$ belongs to. We focus on the MAX-CUT in this study since it is suitable for fundamental benchmarks. It is a classically known example of NP-hard problems as in [5] and also a well-studied problem with several approximation algorithms including the convex relaxation algorithm using semi-definite programming (SDP) [7]. In addition, any NP problems can be mapped onto the MAX-CUT since it belongs to NP-hard class.

To implement the above problem on a physical system, the injection-locked lasers [4] and the degenerate optical parametric oscillators (DOPOs) [8] were proposed to use. With a series of experimental challenges, the implementations of the optical delay line based [9, 10] and measurement feedback based [11, 12] CIMs have

© 2017 IOP Publishing Ltd

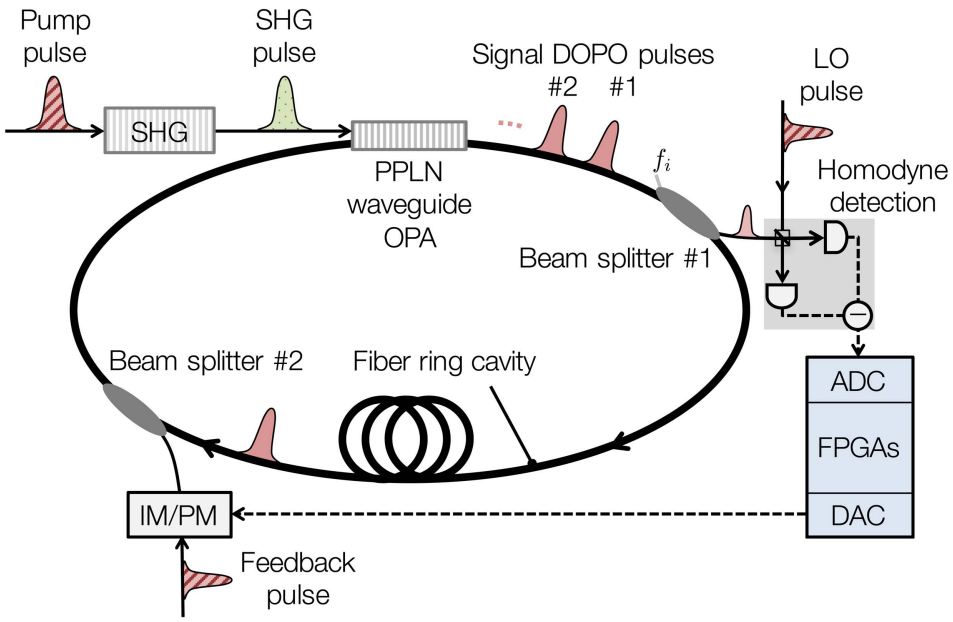

Figure 1. Experimental schematic of a coherent Ising machine implemented on a fiber DOPO with an FPGA measurement feedback circuit.

been realized. Stimulated by the experimental success, its applications including drug discovery [13], image processing wireless communication are investigated.

As a metaheuristic algorithm, the CIM can be interpreted as a mathematical model to solve combinatorial optimization problems using recurrently updated neurons with nonlinear activation function (with linear growth and nonlinear saturation of amplitudes [14]). From this point of view, there have been related and interesting approaches by mathematical models of the neurons (e.g., [15, 16]) and their networks (e.g., [17]). Hopfield developed the optimization algorithm by using such neural networks [18]. Then Hopfield and Tank extended it to the continuous valued model to improve the performance and applied it to the combinatorial optimization problems [19, 20]. Simulated annealing (SA) is proposed in the same period [21].

Here, we try to clarify the relative performance of our CIM against a family of classical neural network approaches and SA for combinatorial optimization problems especially for MAX-CUT. This paper is organized as follows. In section 2.1, we describe the basic concept and experimental configuration of CIM. In section 2.2, we describe models of neural network algorithm for the combinatorial optimization problems followed by section 2.3 to describe suitable hardware implementation. Then the numerical experiments are performed in section 3. We discuss the results and other possibilities of implementations in section 4. Finally, we conclude the paper in section 5.

2. Method

2.1. Coherent Ising machine (CIM)

We intend to solve combinatorial optimization problems by mapping the cost function (1) to the energy of an Ising spin system. CIM is initially proposed as an injection-locked laser system [4], followed by the proposal using a DOPO system [8]. So far, several experimental machines are demonstrated with $n = 4, 16, 100, 2048$-pulse systems [9–12]. Since the original MAX-CUT has binary variables, we use a bistable optical device, DOPO at the output stage of computation, while an analog optical device, degenerate optical parametric amplifier, at the solution search stage of computation.

Figure 1 depicts the schematic of the measurement feedback based CIM [11, 12]. Here we describe the typical experimental configurations in [12]. The DOPO part consists of a 1 km optical fiber (round trip time of 5 μs) with an externally pumped periodically poled lithium niobate waveguide. The pulsed pump laser, at the 1 GHz repetition rate of 5000 times as the cavity circulation frequency, generates 5000 individual DOPO pulses in a single fiber ring cavity. A segment of them (2000 pulses) is used as the signal pulses for computation and the remaining portion (3000 pulses) is used for the cavity stabilization.

Table 1. Classical neural-network approaches for combinatoral optimization problems.

	Deterministic	Stochastic
Binary	Derandomized Hopfield network (HN)	Simulated annealing (SA)
Analog	Hopfield–Tank neural network (HTNN)	

The feedback circuit stores the interaction strength for each pair of DOPO pulses. A portion of optical pulse is picked-off by a beamsplitter (numbered as 1 in the figure 1) and measured by balanced homodyne detectors. The measured values of DOPO pulse amplitudes are fed into an analog–digital converter, followed by field programmable gate arrays (FPGAs). Here, 1 GHz repetition rate of signal pulses is downclocked to 125 MHz (8 parallel) and the measured amplitudes $\tilde{c}_i$ are sliced into the digital signals of 5 bits. Then 2 FPGAs sum up the coupling effect from the other vertices (in the given topology) $\sum_j J_{ij}\tilde{c}_j$ for the ith pulse. The feedback pulse train is modulated in intensity and phase by this output electrical signal after a digital–analog converter. The feedback pulse is injected to the signal DOPO pulse running through the main fiber ring cavity via a beamsplitter #2.

The DOPO is operated near the oscillation threshold by crossing the pump rate from below to above the threshold in the case of [12]. In the beginning, the DOPO is biased at below the threshold in which all phase configuration is established so as a superposition state and the quantum parallel search is implemented [22]. Then, the external pump rate (or the feedback) strength is gradually increased, and once the whole system reaches the oscillation threshold, it selects a particular phase configuration which corresponds to the near-optimal solution of the original optimization problem.

The dynamics of the CIM can be simulated by the quantum master equation. Instead of numerically integrating the master equation for the DOPO density operator, we can expand the density operator by the quasi-probability function in the phase space. One quasi-probability function need for this purpose is the positive $P(\alpha,\ \beta)$ representation in terms of the off-diagonal coherent state expansion, $|\alpha\rangle\langle\beta|$. The Fokker–Planck equation for $P(\alpha,\ \beta)$ is derived from the master equation and then the c-number stochastic differential equations for α and β are obtained using the Ito calculus (see [23] for detail). Another quasi-probability function used for this purpose is the truncated Wigner representation $W(\alpha)$. The corresponding c-number stochastic differential equations are derived in [22]. We will use the latter approach in this paper to evaluate the performance of the CIM.

2.2. Classical neural networks

We describe in this section the classical neural network models to solve the same combinatorial optimization problems, which are summarized in the table 1.

2.2.1. Derandomized Hopfield network (HN)

Hopfield implemented a classical neural network model solving combinatorial optimization problems in his 1982 paper [18], which is referred to the HN. The neuron in this model has the discrete output values $x_i = \pm 1$ with a simple majority voting update rule:

$$x_i \leftarrow \mathrm{sgn}\left(\sum_{j=1}^{n} J_{ij}x_j\right), \tag{2}$$

which will execute asynchronously. The spin index i is selected randomly in the original paper but we derandomized it to enhance the speed, i.e., the spin indices from $i = 1$ to $i = n$ are updated sequentially. Simultaneous updates introduce the instability or periodic solution into the system. Since the update is local and deterministic, the system will converge into the nearest local minimum, which is determined by the initial state. Note that the model is originally proposed with $\{0,\ 1\}$-binary neurons, but for comparison, we use equivalent $\{+1,\ -1\}$-valued neurons.

2.2.2. Simulated annealing (SA)

While the HN will often get stacked at poor local minima, Kirkpatrick *et al* introduced a stochastic spin update strategy in SA algorithm to mimic thermal annealing [21]. The probability of stochastic spin flip is governed by the Boltzmann factor in the Metropolis–Hastings procedure as follows:

$$P = \exp(-\Delta E_i/T), \tag{3}$$

even if the energy difference to flip the ith spin

$$\Delta E_i = 2x_i \sum_{j=1}^{n} J_{ij} x_j \tag{4}$$

makes the total energy increased, namely $\Delta E_i > 0$. The spin index i is selected randomly while temperature T is gradually decreased.

2.2.3. Hopfield–Tank neural network (HTNN)

Hopfield and Tank proposed another neural network approach using an analog valued neuron $x_i \in [-1, 1]$, which is referred to the HTNN [20]. The time evolution of the HTNN is described by ordinary differential equations (ODEs):

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = -\alpha x_i + \beta \sum_{j=1}^{n} J_{ij} f(x_j), \tag{5}$$

where $f(x)$ is a nonlinear sigmoid function. In this study, $\tanh(x)$ is used as $f(x)$. In the extremely high linear gain limit, i.e., when the slope of the sigmoid function around 0 is steep, this HTNN model becomes close to the HN model described above. The parameters in later section are optimized as the neuron decay rate $\alpha = 6$ and the synaptic connection strength $\beta = 0.1$ to achieve the best performance for the given MAX-CUT problems. The numerical integration of (5) is performed by the Euler method with the discrete time step $\Delta t = 0.01$.

2.3. Hardware used for Implementation of classical neural networks

Here we describe the hardware configuration needed to implement the classical neural networks, which will be used in the benchmark section. Note that all codes are implemented with C++[6].

2.3.1. CPU (for SA and HN)

SA and HN are iterative updating algorithms for discrete spins. We can achieve CPU implementation efficiently by SIMD bitwise operations in parallel[7]. In this paper, we mainly used Intel Xeon E3-1225 v3 @ 3.2 GHz (Haswell architecture shipped in 2013). Note that the performance of SA is slightly improved from the previous paper, in which SA is run on an older processor (Intel Xeon X5650 @ 2.67 GHz Westmere architecture shipped in 2010) [12]. We did not use any accelerators for HN and SA in this study since it is already parallelized by SIMD operations in CPU and the cache hit rate is high enough as 98.8%.

2.3.2. MIMD many core processor PEZY-SC (for HTNN)

Since HTNN is based on ODEs (a continuous-valued continuous-time system) and requires floating-point arithmetic, it is better to parallelize by accelerators. We used a MIMD many core processor PEZY-SC @ 733 MHz with 1024 cores and 8192 threads on a chip (the architecture is shown in figure 2), which is set in Shoubu or Satsuki supercomputers at Riken (Japan). We parallelized matrix-vector multiplication and neuron updates in 8192-thread parallel. The coupling matrix is efficiently stored as a 1 bit matrix (since $J_{ij} = \pm 1$ has no empty entry) and neuronal states as floating points (32 bit float). Note that it was 1.4 times faster than storing matrix values in 32 bits. The benchmark of the hardware itself is shown in appendix A.1.

3. Results

We compared the performance of HN, SA, HTNN, and CIM by solving the MAX-CUT problems on a dense graph. The particular problem instance is a complete graph, in which all pair of vertices are connected and edges are weighted by $\{+1, -1\}$ in uniform distribution. We used the identical instance for $n = 2000$ as in [12] and generated a larger instance of $n = 20\,000$ in the same manner. Figure 3 shows the performance on the complete graph, while the detailed computation time to target and the hardware configurations are summarized in table 2.

We ran 100 different trials for the same problem instance (except for CIM experiment, which consists of 26 trials). Each solid line in the figure 3 indicates the ensemble average of all trials, while the lower and upper shaded lines indicate the best and worst case envelopes, respectively. Here, parameters for SA and HTNN are optimized to achieve the shortest computation time to the target which is obtained by the SDP relaxation algorithm [7]. The computation time to the SDP-produced target is shorter in the order of CIM, HN, SA, HTNN on the instance. The data from CIM in figure 3(a) are noisy due to experimental noise, but it can find better solutions than the

[6] We used Ubuntu 16.04.4 with GCC 5.4.0 (CPU) and CentOS 7.1.1503 with GCC 4.8.3 (PEZY-SC).

[7] The code is available here https://github.com/haribara/SA-complete-graph

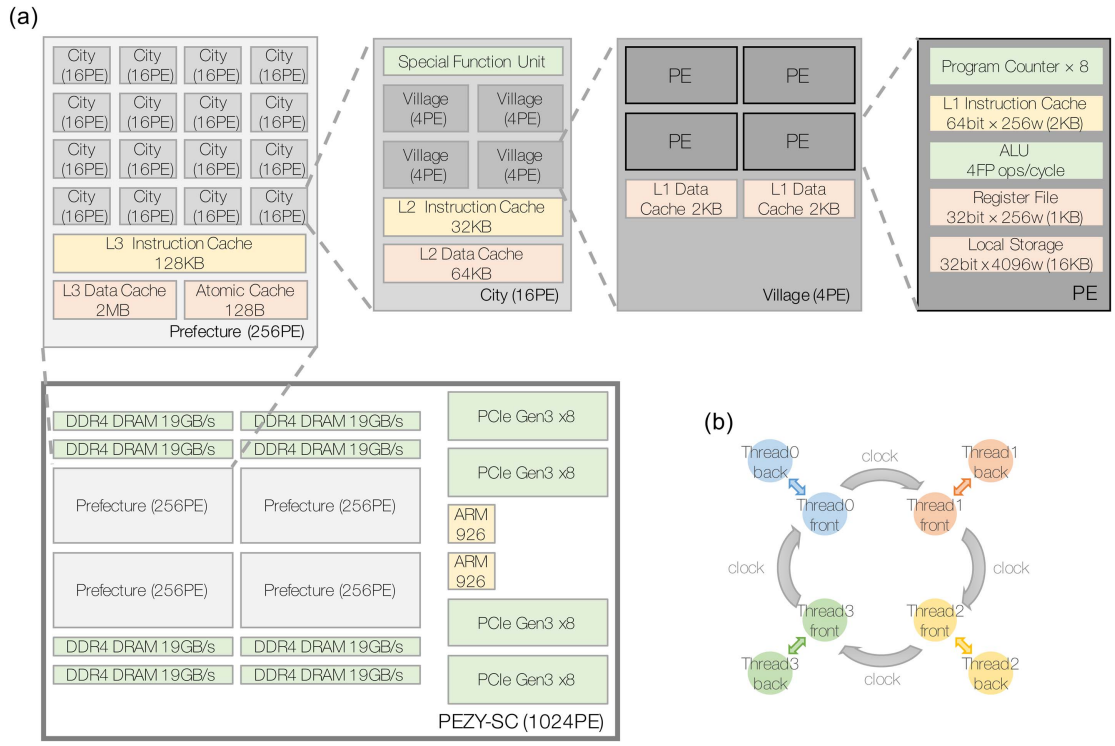

Figure 2. (a) Hierarchical architecture of a PEZY-SC many core processor. There are 1024 processing elements (PE) packed in a single PEZY-SC chip. (b) Each PE core handles 8 threads independently.

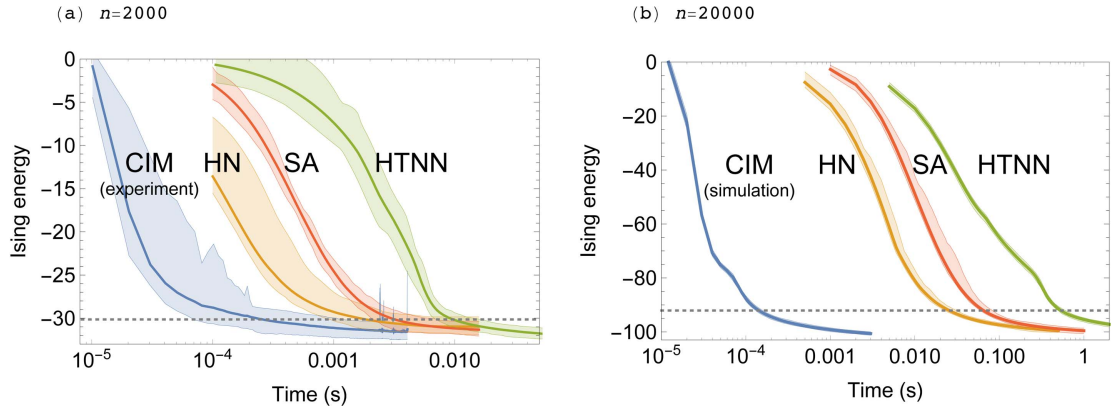

Figure 3. Energy descent when solving $\{+1, -1\}$-weighted (a) $n = 2000$ and (b) $n = 20\,000$ complete graphs. Each thick line is the ensemble average of 100 trials (except for CIM experiment, which consists of 26 trials), while the lower and upper shaded error bars show the best and worst envelopes for each computational model. The gray dotted line is the target values $-60\,278/n$ and $-1841\,216/n$ for $n = 2000$ and 20 000, respectively, which are obtained by the SDP relaxation algorithm [7]. In the CIM simulation of $n = 20\,000$, the cavity round trip time is assumed to be 10 μs.

target in all 26 trials. HN is faster than SA since HN can be regarded as a derandomized version of SA. Note that in the worst case, HN cannot reach the target (it fails 3 times in 100 trials as it can be seen partly in the worst case in figure 3(a)). It can be understand that HTNN performs much slower than HN/SA since it solves ODEs which deals with the analog variables. Note that HTNN achieves lower energy than SA in figure 3 but the performance of SA heavily depends on temperature scheduling. We optimized to reach the target shorter but slower scheduling ends up lower energy generally.

The computation time to the SDP target is in the same sequence when the number of vertices increases to $N = 20\,000$. Here the cavity round trip time of CIM is assumed to be 10 μs. Then the relative speed-up of CIM is raised to 2–3 orders of magnitude compared to other implementations.

Table 2. Time to target and hardware configurations. The best (shortest) time to reach the target value and the time to cross the ensemble averaged line are listed in the upper table. Note that the target value is obtained by the SDP relaxation algorithm which has performance guaranty to the 87% of the optimal value.

	$n = 2000$		$n = 20\,000$		
	Best (ms)	Avg. (ms)	Best (ms)	Avg. (ms)	Hardware
CIM	0.071	0.264	0.14	0.15	Fiber DOPO+2FPGAs
HN	0.924	1.84	23	26	CPU
SA	2.10	3.20	60	65	CPU
HTNN	7.04	9.67	500	540	PEZY-SC

Hardware	Model	Clock and architecture	Release date
FPGA	Xilinx Virtex-7 VX690T	125 MHz 693k logic cells	2010
CPU	Intel Xeon E3-1225 v3	3.2 GHz, Haswell	2013
MIMD many core	PEZY-SC	733 MHz, 1024 core	2014

4. Discussion

In this section, we will add the two discussions to justify the above conclusions:

- Validity for the hardware selection, and
- Optimization for PEZY-SC implementation for HTNN.

4.1. Validity for the hardware selection

HTNN is apparently efficient on PEZY-SC than on CPU, which is shown in appendix A.1. On the other hand, we did not use any accelerator for HN and SA in this study. This is because we do not expect significant speed-up by naive implementation since they are already parallelized by SIMD operations in CPU and the cache hit rate is so high as 98.8% (measured by perf command in Linux). Generally, asynchronous update in HN/SA seems to be not suitable for parallel implementation.

4.2. Optimization for PEZY-SC implementation for HTNN

We tried to optimize the implementation by storing matrix data efficiently. Since the given adjacency matrix has only the 1 bit entry ($J_{ij} = \pm 1$), we packed each value in 1 bit. This contributes the 1.4 times speed-up than having a 32 bit float matrix for $n = 2000$. But putting the data in local memory does not contribute to significant speed-up since its bottleneck in computation is not in memory transfer. There is a possibility of speed-up if we replace the multiplication by the selector. Rather, it is possible to scale out for parallel distributed processing by using multiple PEZY-SC chips in Soubu supercomputer, especially when the problem size is larger.

5. Conclusion

In this paper we compared the performance of the CIM implemented on DOPOs and FPGAs against the family of classical neural-network-based algorithms: HN, SA and HTNN. To accelerate the performance of the classical neural networks, HN and SA are implemented on CPU with bit operations and HTNN is implemented on a many core processor PEZY-SC. It is shown on the $n = 2000$ complete graph that the experimental CIM can achieve faster computational time than HN (13.0 times for the best case and 6.97 times for the average), SA (29.6 times for the best case and 12.1 times for the average) and HTNN (99.2 times for the best case and 36.7 times for the average). To estimate the performance on a larger instance, we performed numerical simulations on $n = 20\,000$ graph, in which the relative CIM performance is 173, 433 and 3600 times faster than HN, SA and HTNN in average, respectively, where the 2 km fiber ring cavity with 10 μs round trip time is assumed for CIM.

Acknowledgments

The authors thank H Takesue and T Inagaki for providing the experimental data, K Kawarabayashi, S Tamate and T Sonobe for accelerating SA implementation, Y Saito for comments on the FPGA configuration, T Leleu and M Oku for general discussion, K Hiraki for the comment on the CPU architecture. We used super

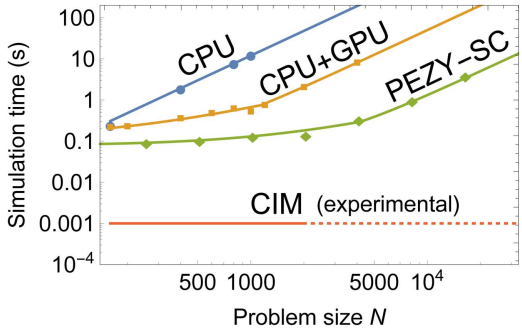

Figure A1. Simulation time for the *c*-number stochastic differential equations (200 round trips on complete graphs).

Table A1. Accelerator configurations for parallel implementation of the *c*-number stochastic differential equations in figure A1.

Hardware	Model	Clock and architecture	Release date
CPU	Intel Xeon W3530	2.80 GHz, Nehalem-WS	2010
GPU	NVIDIA Tesla C2075	1.15 GHz, Fermi	2011
MIMD many core	PEZY-SC	733 MHz, 1024 core	2014

computers Shoubu and Satsuki in Riken (Saitama, Japan) to benchmark on PEZY-SC, thanks to T Ebisuzaki, M Kurokawa and H Kenzaki. This research is supported by the Impulsing Paradigm Change Through Disruptive Technologies (ImPACT) Program of the Council of Science, Technology and Innovation (Cabinet Office, Government of Japan).

Appendix

A.1. Processor performance of PEZY-SC

We show the elapsed time for the CIM simulation by the following *c*-number stochastic differential equations [22, 24]

$$\mathrm{d}c_i = [(p - c_i^2 - s_i^2)c_i]\,\mathrm{d}t + \frac{1}{A_s}\sqrt{c_i^2 + s_i^2 + \frac{1}{2}}\,\mathrm{d}W_{ci}, \tag{A.1}$$

$$c_i(t + \Delta t) \mapsto \sqrt{1 - T_{\mathrm{mes}}}\,c_i(t) + \sqrt{T_{\mathrm{mes}}}\,\frac{f_i}{A_s}, \tag{A.2}$$

$$c_i(t + \Delta t) \mapsto \sqrt{1 - T_{\mathrm{inj}}}\,c_i(t) + \sqrt{T_{\mathrm{inj}}}\,\xi \sum_{j=1}^{n} J_{ij}\tilde{c}_j, \tag{A.3}$$

implemented on different processor configurations. Note that the simulation by the Langevin equation is simplified version and we employed this model to contrast the processor performance (detailed simulations are reported in [25, 26]). Here, CPU indicates serialized calculation on a single thread, CPU+GPU indicates matrix-vector multiplication is off-roaded to GPU while other part of differential equation is calculated on the same processor as CPU, PEZY-SC indicates that all processes are paralellized. We conclude that is it preferable to implement HTNN on PEZY-SC than CPU or GPU which we listed in the table A1 since the Langevin equations are similar to ODEs of HTNN except for random number generation.

References

[1] Hennessy J L and Patterson D A 2011 *Computer Architecture: A Quantitative Approach* (Amsterdam: Elsevier)
[2] Waldrop M M 2016 *Nat. News* **530** 144
[3] Khan M, Lester D, Plana L A, Rast A, Jin X, Painkras E and Furber S B 2008 Spinnaker: mapping neural networks onto a massively-parallel chip multiprocessor *IEEE Int. Joint Conf. Neural Networks, 2008, IJCNN 2008, (IEEE World Congress on Computational Intelligence)* (Piscataway, NJ: IEEE) pp 2849–56
[4] Utsunomiya S, Takata K and Yamamoto Y 2011 *Opt. Express* **19** 18091–108
[5] Karp R M 1972 *Reducibility Among Combinatorial Problems Complexity of Computer Computations* (Berlin: Springer) pp 85–103
[6] Garey M R and Johnson D S 2002 *Computers and Intractability* vol 29 (New York: Freeman)

[7] Goemans M X and Williamson D P 1995 *J. ACM (JACM)* **42** 1115–45
[8] Wang Z, Marandi A, Wen K, Byer R L and Yamamoto Y 2013 *Phys. Rev.* A **88** 063853
[9] Marandi A, Wang Z, Takata K, Byer R L and Yamamoto Y 2014 *Nat. Photon.* **8** 937–42
[10] Takata K, Marandi A, Hamerly R, Haribara Y, Maruo D, Tamate S, Sakaguchi H, Utsunomiya S and Yamamoto Y 2016 *Sci. Rep.* **6** 34089
[11] McMahon P L *et al* 2016 *Science* **354** 614–7
[12] Inagaki T *et al* 2016 *Science* **354** 603–6
[13] Sakaguchi H, Ogata K, Isomura T, Utsunomiya S, Yamamoto Y and Aihara K 2016 *Entropy* **18** 365
[14] Hamerly R, Inaba K, Inagaki T, Takesue H, Yamamoto Y and Mabuchi H 2016 *Int. J. Mod. Phys.* B **30** 1630014
[15] McCulloch W S and Pitts W 1943 *Bull. Math. Biophys.* **5** 115–33
[16] Hodgkin A L and Huxley A F 1952 *J. Physiol.* **117** 500
[17] Fukushima K 1980 *Biol. Cybern.* **36** 193–202
[18] Hopfield J J 1982 *Proc. Natl Acad. Sci.* **79** 2554–8
[19] Hopfield J J 1984 *Proc. Natl Acad. Sci.* **81** 3088–92
[20] Hopfield J J and Tank D W 1986 *Science* **233** 625–33
[21] Kirkpatrick S, Gelatt C D and Vecchi M P 1983 *Science* **220** 671–80
[22] Maruo D, Utsunomiya S and Yamamoto Y 2016 *Phys. Scr.* **91** 083010
[23] Takata K, Marandi A and Yamamoto Y 2015 *Phys. Rev.* A **92** 043821
[24] Haribara Y, Utsunomiya S and Yamamoto Y 2016 *Entropy* **18** 151
[25] Shoji T, Aihara K and Yamamoto Y 2017 arXiv:1706.02034 [quant-ph]
[26] Yamamura A, Aihara K and Yamamoto Y 2017 arXiv:1706.02454 [quant-ph]