Digital CIM with Noisy SRAM Bit: A Compact Clustered Annealer for Large-Scale Combinatorial Optimization

Anni Lu[1], Junmo Lee[1], Yuan-Chun Luo[1], Hai Li[2], Ian Young[2], Shimeng Yu[1]
[1]Georgia Institute of Technology, Atlanta, GA [2]Exploratory Integrated Circuits, Components Research, Intel, Hillsboro, OR
Email: {alu75, junmolee, yluo369}@gatech.edu, {hai.li, ian.young}@intel.com, shimeng.yu@ece.gatech.edu

Abstract—**Combinatorial optimization problems (COP) are NP-hard and intractable to solve using conventional computing. The Ising model-based annealer has gained increasing attention recently due to its efficiency and speed in finding approximate solutions. However, Ising solvers for travelling salesman problems (TSP) usually suffer from a scalability issue due to quadratically increasing number of spins. In this paper, we propose a digital computing-in-memory (CIM) based clustered annealer to solve tens of thousands of city-scale TSP with only a few mega-byte (MB) of static random access memory (SRAM), using hierarchical clustering to solve input sparsity and digital CIM flexibility to solve weight sparsity. The intrinsic process variations between SRAM devices are utilized to generate the noisy bit errors during pseudo-read under reduced supply voltage, realizing the annealing process. The design space of cluster size and programmability is explored to understand the trade-offs of solution quality and hardware cost, for TSP scale ranging from 3080 to 85900 cities. The proposed design speeds up the convergence by >10^9× with <25% solution quality overhead compared with the CPU baseline. The comparison with state-of-the-art scalable annealers shows a >10^{13}× improvement on functionally normalized area and power.**

Keywords—travelling salesman problem, combinatorial optimization problem, simulated annealing, digital in-memory computing, Ising model

I. Introduction

Combinatorial optimization problems (COP) have a wide range of applications such as supply chain logistics, drug discovery, financial services, VLSI routing, big data analytics, etc. As an example, many routing problems can be mapped to the travelling salesman problems (TSP), which is to find the shortest route to visit all cities exactly once given a list of cities and the distances between each pair of cities. However, the searching space of COP explodes exponentially with its problem size, making it NP-hard [1] with no polynomial-time algorithms to solve. Ising model-based solvers obtain the near-optimal solution via the convergence of the system energy towards the ground state, and have gained prominence due to their abilities to quickly find approximate solutions for COP.

Under the Ising formulation, a N-city TSP requires N^2 spins and N^4 all-to-all interactions. The chip area, connectivity and performance of Ising annealer limit the manageable TSP size. A clustering approach [2] has been proposed to partially eliminate the spins and their coefficients so as to simplify the large-scale TSP and speed up the system convergence to a near-optimal solution. Applying the clustered approach on hardware annealer greatly relieves the scalability problem by removing the spins that are incompatible with the clusters [3]. In other words, clustering makes the spin states (input) sparse, so the hardware design benefits from the input sparsity to reduce the required memory capacity from $O(N^4)$ to $O(N^2)$ as shown in Fig. 1.

However, the sparsity of the coupling matrix (weight) was unexplored as it is limited by the crossbar nature of a memory array. To remap the valid matrix windows as a dense matrix, the original vector matrix multiplication (VMM) between spin states and coupling matrix becomes combinations of several smaller scale multiplications. Analog CIM is indivisible as it naturally accumulates the current along the entire column. While digital CIM has the flexibility to compute VMM partially, thanks to its logic accumulator. In this work, we propose a digital CIM-based Ising annealer to take advantage of the weight sparsity. The required memory capacity can be further reduced from $O(N^2)$ to $O(N)$, so a TSP with tens of thousands of cities can be solved using MB-level SRAM as shown in Fig. 1.

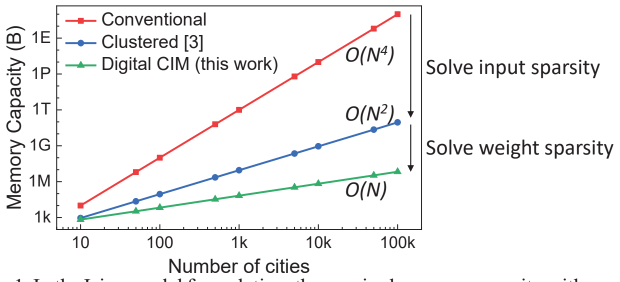


Fig. 1. In the Ising model formulation, the required memory capacity with respect to the scale of TSP. Clustered approach can reduce it from $O(N^4)$ to $O(N^2)$ by solving input sparsity [3], and digital CIM-based design can further reduce it to $O(N)$ by solving weight sparsity (proposed in this work).

Most of prior SRAM-based annealer designs require additional digital pseudo random number generators such as the linear feedback shift register (LFSR) for annealing, while prior work [4] tried to utilize the intrinsic spatial process variation of SRAM. However, the attempt in [4] does not perform well because annealing requires temporal noise instead of spatial noise. In this work, we propose to apply noise on the coupling matrix (weight) instead of spin states (input), so the spatial variation is converted to temporal changes when the devices at different locations are utilized in different clock cycles. The notable contributions of this work are summarized as follows:

- The proposed algorithm-hardware co-design for an Ising annealer greatly alleviates the scalability issue through clustering and digital CIM to leverage input and weight sparsity respectively, enabling MB-level SRAM to solve TSPs with tens of thousands of cities. Parallel updating of non-adjacent clusters also speeds up the convergence.
- Digital CIM is married with noisy SRAM bit cell, whose intrinsic process variation is used for annealing and validated by Monte Carlo simulations with the TSMC 16nm FinFET process-design-kit (PDK).

II. Background

A. Solving TSP Using Ising Model-based Annealer

Ising model is comprised of spins (σ), interaction weights (J) and external magnetic field (h). Each spin has either an upward (+1) or downward (-1) state. Depending on the interactions of adjacent spins and the external field for each spin, the system Hamiltonian energy (H) is expressed as Eq. (1)

$$H = -\sum_i \sum_j J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i \tag{1}$$

The local energy of spin i is shown as Eq. (2), which is determined by the weighted majority vote of adjacent spins and external field. Minimization of the total energy is achieved by updating each spin in the direction that lowers its local energy.

$$H(\sigma_i) = -(\sum_j J_{ij}\sigma_j + h_i)\sigma_i \tag{2}$$

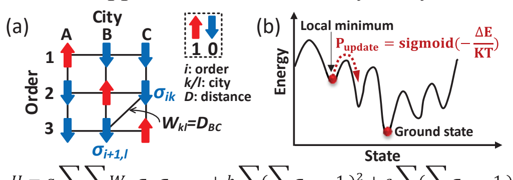
Generally, the system energy space has a global minimum and also multiple local minimum states. To prevent the system from being stuck into a local minimum, annealing process (Fig. 2(b)) such as simulated annealing has been widely used, where the temperature T adjusts the probability of spins flipped by "thermal fluctuation". During annealing process, the system is initialized with a high temperature and then converges towards the energy minimum state with gradually lowering temperature.

An example of mapping TSP to Ising model is shown in Fig. 2(a). A N-city TSP with N^2 spins is formulated as Eq. (3)

$$H_{TSP} = a\sum_{k\neq l}\sum_i W_{kl}\sigma_{ik}\sigma_{(i+1)l} + b\sum_i(\sum_k \sigma_{ik} - 1)^2 + c\sum_k(\sum_i \sigma_{ik} - 1)^2 \tag{3}$$

where σ_{ik} = +1 or 0 indicates whether the k-th city is visited at the i-th order; W_{kl} denotes the distance between the k-th and l-th cities. The first term represents the total distances between city pairs with adjacent visiting orders, which is the objective function. The last two terms are the one-hot constraints that represent the penalty for visiting multiple cities as the i-th visit and for visiting the city k more than once, respectively. *a, b* and *c* are hyperparameters to balance the relative strength of the objective function and the constraints. The energy penalty of infeasible routes (2nd and 3rd term in Eq. (3)) can be avoided through permutational Boltzmann machine (PBM) [5]. To ensure the two-way one-hot constraint being followed, four spins can be updated together (e.g., $\sigma_{ik}, \sigma_{il}, \sigma_{jk}, \sigma_{jl}$), which means two city visiting orders are swapped (e.g., k and l cities swap their orders i and j). Then the local energies of the flipped spins are compared with that before the swap to decide whether to accept or reject this swap.

The spin energy computation is actually a multiply-and accumulate (MAC) operation as expressed in Eq. (2). Further, the Ising model can be abstracted as a Hopfield neural network (HNN) to be mapped on a CIM memory array. The HNN is a fully connected recurrent network with binary neurons and a single layer. The neuron vector corresponds to the spin states and the weight matrix corresponds to the interactions in the Ising model. Therefore, the MAC output between neuron and weight is the spin energy and decides the spin state in the next cycle. When deployed on memory array, the Ising model is realized with full connections between all spins and parallel computation from the hardware perspective. Intrinsic noise of the memory devices can be used as randomness sources for annealing.



$$H = \underbrace{a\sum_{k\neq l}\sum_i W_{kl}\sigma_{ik}\sigma_{(i+1)l}}_{\text{Objective function}} + \underbrace{b\sum_i(\sum_k \sigma_{ik} - 1)^2 + c\sum_k(\sum_i \sigma_{ik} - 1)^2}_{\text{Penalty of one-hot constraint}}$$

Fig. 2. Illustration of solving TSP using the Ising model. (a) The Hamiltonian energy of TSP consisting of the objective function (sum of city distances following the order) and a high penalty (two-way one-hot constraint of order and city). (b) The optimal solution is obtained with the convergence of system Hamiltonian energy towards the ground state. The annealing process helps escape from local minimum.

B. Digital Compute-in-Memory

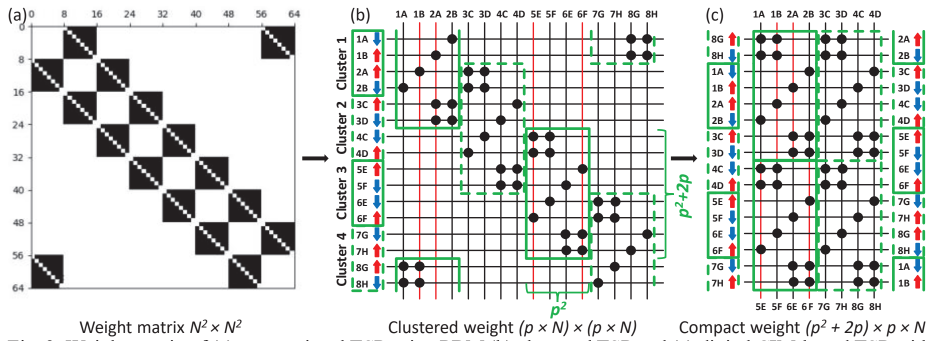
Traditional analog CIM crossbar arrays realize MAC operations through multiplication between input voltage and weight conductance and accumulation of resulting currents along a column in the analog domain. Instead, digital CIM [6-8] was proposed in recent years to utilize digital MACs while still enjoying the input parallelism. Most digital CIM silicon prototype chips reported today employ the SRAM bit cell that integrates local compute units followed by an adder tree. The adder tree-based accumulation enables more flexibility for digital CIM than its analog counterparts. Digital CIM is also compatible with the advanced foundry process such as 3 nm or beyond and eliminates the need for analog-to-digital converters, which is known as the bottleneck of power/performance/area (PPA) in analog CIM [9]. Furthermore, the software baseline accuracy can be maintained with digital CIM, thanks to the high precision computation and noise immunity of digital logic.

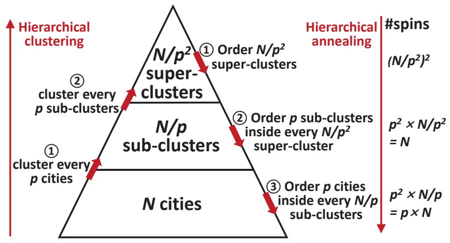
III. Digital CIM-based Ising Annealer Design

A. Hierarchical Clustered Approach for Input Sparsity

Clustering design has been applied to reduce the required number of spins and weights [3]. For example, in Fig. 3(a) and (b), the original spins are 64 permutations of cities (A to H) and orders (1 to 8) for an 8-city TSP, so the coupling matrix is 64×64. After clustering the cities with two cities in one group, there are only 16 permutations left (4 permutations each cluster × 4 clusters), thus the excluded spins (permutations) and their corresponding weights can be removed. With the problem scale further increased, hierarchical clustering is needed. As shown in Fig. 4, first, the hierarchical clustering is applied from bottom-up, clustering every p cities (or sub-clusters represented by the centroids) and repeated for all levels. Next, the hierarchical annealing is performed top-down. At the top level, TSP is solved for the order of the super-clusters. Then the orders of sub-clusters inside every super-cluster are solved as clustered TSPs level-by-level, until the order of all cities is obtained at the lowest level. In this way, the number of required neurons is reduced from N^2 to $p\times N$, and the dimension of weight matrix is reduced from $N^2\times N^2$ to $(p\times N)\times(p\times N)$.

Normally, the spins in the Ising model are updated one-by-one following the Gibbs sampling to make sure the ergodicity and the convergence. However, according to the chromatic Gibbs sampling [10], the spins that are independent (with no interactions) can be updated in parallel. In clustered TSP, the spins in non-adjacent clusters are always independent, because their represented cities cannot be ordered as neighbors. Therefore, as shown in Fig. 3(b), all the odd clusters (solid windows) can be computed together, and all the even clusters (dash windows) are computed in the other cycle.


(a)
(b)
(c)
Cluster 1
Cluster 2
Cluster 3
Cluster 4
p^2
p^2+2p
Weight matrix $N^2 \times N^2$
Clustered weight $(p \times N) \times (p \times N)$
Compact weight $(p^2 + 2p) \times p \times N$
Fig. 3. Weight matrix of (a) conventional TSP using PBM (b) clustered TSP and (c) digital CIM-based TSP with compact weight. N is number of cities and p is number of elements in each cluster. In this example, $N = 8$, p=2.


Hierarchical clustering
Hierarchical annealing
#spins
N/p^2 super-clusters
N/p sub-clusters
N cities
① Order N/p^2 super-clusters
② Order p sub-clusters inside every N/p^2 super-cluster
③ Order p cities inside every N/p sub-clusters
② cluster every p sub-clusters
① cluster every p cities
$(N/p^2)^2$
$p^2 \times N/p^2 = N$
$p^2 \times N/p = p \times N$
Fig. 4. The hierarchical clustering and hierarchical annealing of the clustered approach for TSP. $p{\times}N$ spins are required at most with enough hierarchies.

B. *Digital Compute-in-Memory for Weight Sparsity*

Although the clustered approach solves the input sparsity and reduces weight matrix from $O(N^4)$ to $O(N^2)$, a high weight sparsity still exists. As shown in Fig. 3(b), only the windows along the diagonals have valid weights. This is because the spins only have interactions with spins inside its own cluster and adjacent clusters. Each window consists of $(p^2+2p) \times p^2$ weights, where p is the number of cities or sub-clusters in each cluster. To exclude the useless weights, the windows are relocated as compact mapping as shown in Fig. 3(c). Therefore, the new weight matrix includes $(p^2+2p) \times p^2$ parameters in each window multiplied by N/p windows, so it becomes $O(N)$.

However, the relocation will cause wrong results using analog CIM because it naturally sums along the whole column in analog domain. While digital CIM has the flexibility to sum a section of column with a digital adder tree and makes the compact mapping feasible. As shown in Fig. 5(a), because of the parallel update of odd clusters (solid windows) together and even clusters (dash windows) together as introduced in section A, only one column of windows is enabled each time controlled by the window multiplexer (MUX). There are p^2 columns of parameters inside one window, so only one column is picked by cell MUX each time. One adder tree is responsible for the summation of such (p^2+2p) parameters with 8-bit precision.

As explained earlier, swapping city orders by updating four spins together can avoid the two-way one-hot constraint penalty. For example, if city k is visited at i-th order and city l is visited at j-th order, then $\sigma_{ik} = 1, \sigma_{il} = 0, \sigma_{jk} = 0, \sigma_{jl} = 1$. After swapping their orders, $\sigma'_{ik} = 0, \sigma'_{il} = 1, \sigma'_{jk} = 1, \sigma'_{jl} = 0$. So, the change of local spin energies is $H(\sigma'_{il}) + H(\sigma'_{jk}) - H(\sigma_{ik}) - H(\sigma_{jl})$. For its hardware implementation, as shown in Fig. 5 (a), the spin states before swapping are given as input in the first two cycles, and the local spin energies $H(\sigma_{ik})$ and $H(\sigma_{jl})$ are computed and stored in registers respectively. Then the spin states after swapping are inputted in the next two cycles to compute $H(\sigma'_{il})$ and $H(\sigma'_{jk})$. The energies before and after swapping are compared to decide which set of spin states is the new input.

Prior design of the digital CIM cell includes a 6-transistors (T) SRAM and a 4T NOR gate [6]. One bit of weight is stored as charges in the storage node of one SRAM cell. It is directly connected to one of the inputs of NOR gate, and input data is given to the other, so the NOR gate realizes the multiplication between one-bit input and weight without reading the SRAM using sense amplifiers. The one-bit multiplication output is sent to an adder tree to finish the shift-and-add and accumulation for multi-bit MAC operation. In our design, two 2T transmission gates (TG) are added as shown in Fig. 5(b) to support the required flexibility and programmability. The first TG is for the cell MUX to choose which column of parameters inside the window. Its control signal is shared along the entire row of windows. The second TG is for the window MUX to choose which column of windows. Its control signal is shared along the entire column of windows. The control signal connection is illustrated in Fig. 5(d). Hence, our design proposes a 14T cell.

For each 14T cell, the NOR and TGs are placed under the SRAM as shown in Fig. 5(b). According to previous design rule in 22nm [6], the 6T SRAM occupies around 500nm×500nm (no push rule), and the 4T NOR occupies around 500nm×250nm. Therefore, one 14T cell with our layout placement takes around 1μm×500nm if it is in 22nm. The horizontal wire routing congestion is a common challenge in digital CIM layout, but the cell height in our design should be sufficient to avoid it. In previous digital CIM design with 4-bit weight [6], the horizontal wires should include V_{DD}, V_{SS}, word-line (WL), input, and four outputs. In our design, as shown in Fig. 5(d), one cell MUX drain connection, one windows MUX drain connection and cell MUX gate control ($2p^2$ wires shared by p^2+2p rows) are added. 8-bit weight is adopted in the design to ensure solution quality so the output lines double. Overall, the horizontal wires roughly double and are feasible to be routed within the double cell height.

The array-level diagram is shown in Fig. 5(c). The peripherals include switch matrix to enable cells, decoder for MUXs, read and write control circuits for SRAM, and adder trees to realize shift-and-add and accumulation. For memory array, all the most significant bits (MSB) are placed together, and all the least significant bits (LSB) are placed together. Because the error caused by SRAM process variations will be utilized for randomness generation (details to be explained in section IV), the degree of stochasticity can be controlled by applying the noise on how many bits of LSBs. 8-bit weights are adopted to provide enough precision for weight representation and sufficient granularity for noise control. The placement of MSBs together and LSBs together is beneficial for easy control.

MB-level SRAM chip is usually comprised of several smaller memory arrays. The data transmissions inside and between arrays are very trivial in this proposed design. For intra-array dataflow, the input is the output of last update as HNN is a recurrent network. But the input register needs to be shifted up

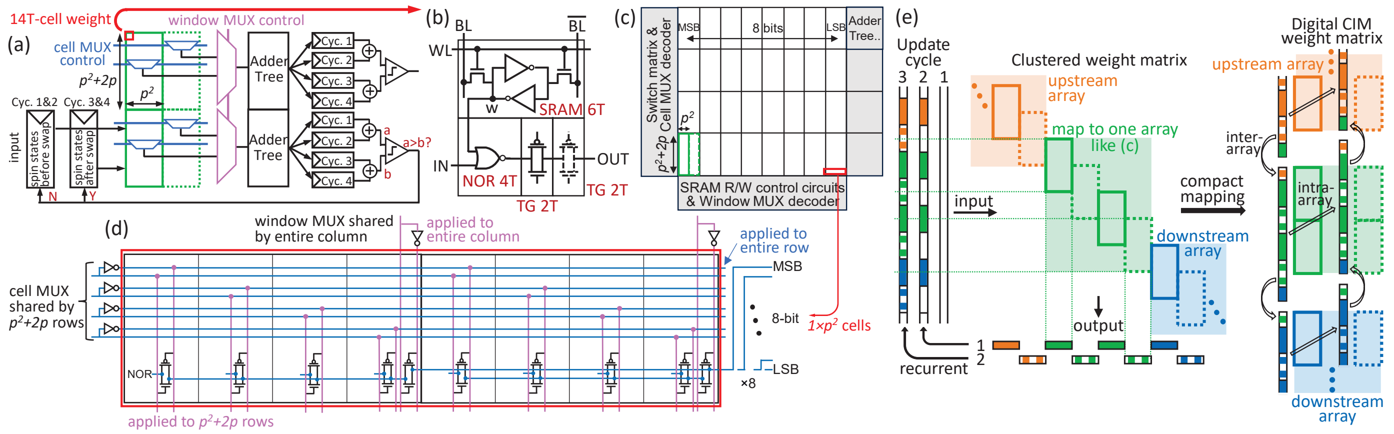


Fig. 5. Illustration of digital CIM-based Ising annealer design. (a) Design overview. Cell MUX is used to choose which column of cells inside window, and window MUX is used to choose which column of windows. Each adder tree computes multi-bit accumulation for p^2+2p parameters. Local spin energies are computed using spins states before and after city order swap respectively, and its change determines whether to accept or reject the swap and finishes one update cycle. (b) Proposed 14T digital CIM weight cell, with 6T SRAM, 4T NOR, and two 2T TG for MUXs. (c) Array-level diagram with peripherals for memory array. The MSBs are LSBs are placed together for easy control of noise generation. (d) Routings illustration. The cell MUX control is shared by entire row of windows, and the window MUX control is shared by entire column of windows. (e) Dataflow intra- and inter- memory arrays. Input register is shifted upward inside array to switch between solid and dash window updates. p-bit input is transmitted from upstream to downstream for solid window updates, and from downstream to upstream for dash window updates.

to align the relocated weight windows as shown in Fig. 5 (e). For inter-array dataflow, assuming the green windows are remapped to one array in digital CIM like Fig. 5(c), starting from initial spin states (update cycle 1), the solid windows are computed and their corresponding spin states are updated (solid colored segments in cycle 1 output and cycle 2 input), then the dash windows are computed and their corresponding spin states are updated (dash colored segments in cycle 2 output and cycle 3 input). In the next updates, the inputs for solid green window require a segment from upper orange dash window outputs; the inputs for dash green window require a segment from lower blue solid window outputs. Therefore, only the data at the edge need to be transmitted from neighbor arrays. When computing the solid windows, p-bit data needs to be transmitted from upstream to downstream; when computing the dash windows, p-bit data needs to be transmitted from downstream to upstream.

IV. Nosiy SRAM-based Annealing

Annealing requires gradually decreasing noise to converge to the global minimum. Compared with traditional digital implementation of LFSR, generating noise/error using the intrinsic process variation of SRAM is much more energy- and area-efficient, and provides an entropy for true randomness.

A. Noise Generation with SRAM Process Variation

SRAM bit-cell has a latch structure made of symmetric inverters to maintain the stored data robustly. However, the process variation causes mismatch between the two inverters and leads to smaller static noise margin (SNM). Voltage disturbance larger than the SNM will flip the data at storage nodes and cause error. With normal operation, the error rate is generally very low. To generate controllable errors, pseudo-read with low supply voltage is proposed in [4]. The WL is activated as normal read operation, while the supply voltage of the latch is decreased to shrink its butterfly curve and degrade the SNM, so a smaller voltage fluctuation could cause an error. There is no sense amplifier to sense the voltage changes on bit-line (BL) and truly read out the stored data, so the operation is named "pseudo-read". We performed Monte Carlo SPICE simulations of pseudo-read using TSMC 16nm PDK, sweeping the supply voltage from 800mV (nominal V_{DD} for 16nm) to 200mV. One thousand samples are taken and averaged to get the error rate at each supply voltage. The error rate increases from 0% to close to 50% with decreasing V_{DD} and behaves like a sigmoid curve. Higher BL capacitance leads to a sharper transition. The BL capacitance is mainly related to its length (i.e., memory array height).

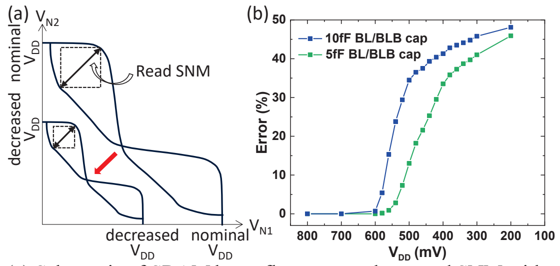


Fig. 6. (a) Schematic of SRAM butterfly curve to show read SNM with nominal and decreased V_{DD} respectively. (b) Error rate with respect to V_{DD} based on Monte Carlo SPICE simulations with 1000 samples using TSMC 16nm PDK.

B. Annealing with Spatial Noise

Since the errors are caused by transistor process variations, the mismatch of the latch for a specific SRAM cell is determined after fabrication and the errors will tend to happen in one direction. In other words, the error pattern is spatial, however, annealing requires temporal noise during samplings. The design in [4] does not perform well when the spatially noisy SRAM cells are used as spin representations. In the extreme case that the spin errors are purely spatial and temporally constant, the output in [4] will always follow a fixed trace with deterministic input errors and weight values, no matter how many attempts are made. Also, this design only applied 1V nominal V_{DD} (for 65nm) and a 0.7V lowered V_{DD}, without the gradually decreasing noise level for better convergence.

In our design, the noisy SRAM cells are used to represent weight values. Therefore, when spin states change, different rows are enabled and different columns are in the computation path, which means weights at different locations are utilized at different times. In this way, the spatial variation is successfully converted to temporal noise for annealing. The voltage flipping of storage nodes in pseudo-read is irreversible even if increasing the V_{DD} back to nominal value. To gradually "recover" the errors during annealing, the original correct weights need to be written back periodically. Besides using V_{DD} to control the error rate, the number of noisy bits provides finer granularity to tune the noise level. The procedure is summarized as follows:

1) Write the correct weights into arrays. Applying low V_{DD} (e.g., 0.3V) for some LSBs (e.g., 6 bits in 8-bit precision) and nominal V_{DD} for the remaining MSBs. Update for some iterations, during which the errors are fixed.
2) Write the correct weights back into arrays again. Apply a slightly increased V_{DD} (e.g., 0.4V) for less LSBs (e.g., 5 bits in 8-bit precision) and nominal V_{DD} for the remaining MSBs. Update for some iterations.
3) Repeat last steps until all bits work in nominal V_{DD} so that no noise is generated.

V. Algorithm-Hardware Co-Evaluation

We evaluate the performance of the clustered digital CIM-based annealer on several datasets from TSPLIB [11], where the problem size ranges from 3038 to 85900 cities. We apply 400 iterations of spins updating inside every cluster at each annealing level, and V_{DD} is increased from 300mV to 580mV with 40mV increments every 50 iterations. The evaluation includes both algorithm-level accuracy with simulated SRAM error rates, and system-level hardware performance, e.g., chip area, time-to-solution and energy-to-solution, with implementation using 16nm FinFET technology. The device and circuit macro models are modified from NeuroSim [12] to support the digital CIM.

A. Design Space Exploration of Cluster Size

The number of cities in each sub-cluster, or sub-clusters in each super-cluster (p) could impact the solution quality and the required hardware resources as shown in Table I. The optimal ratio reflects the solution quality, which is defined as the ratio of distance obtained from the proposed design with respect to the distance of the best-known solution. The exploration is first applied on two datasets with 3038 and 5915 cities respectively as examples. To begin with, the baseline is performed where the number of elements p is unlimited, and only the number of clusters is restricted at each level. In other words, each cluster has two elements on average, but the exact value is arbitrary. The absolute flexibility is beneficial to solution quality, but it causes great reconfigurability challenges in hardware design. To avoid this, p is strictly fixed in the following case, where the required memory size is $(p^2+2p)\times p^2\times N/p$. However, it is shown that the optimal ratio is degraded. Therefore, as a compromised strategy between the absolute flexibility and rigidity, we propose to restrict the maximum number of elements p_{max} in each cluster, so some flexibility is allowed to adjust each cluster size from 1 to p_{max}, if the average number of elements equals to $(1+p_{max})/2$. In this way, the hardware can be designed to support the $2N/(1+p_{max})$ clusters all with p_{max} elements, so the required memory size is $(p_{max}^2+2p_{max})\times p_{max}^2\times 2N/(1+p_{max})$ with some redundant columns. The results show that its solution qualities are much better than that for the strictly fixed strategy, and the optimal ratios are close to the unlimited strategy when p_{max} equals to 3 and 4. Based on the above insight, the semi-flexible strategy is applied on several datasets with city scales ranging from 3038 to 33810 cities as shown in Fig. 7(a). The solution quality generally improves with more p_{max} but starts to saturate when p_{max} reaches 3.

TABLE I. Exploration of Cluster Size and Strategy

dataset	pcb3038		rl5915	
#Elements each cluster	Capacity (kB)	Optimal Ratio	Capacity (kB)	Optimal Ratio
Arbitrary (baseline)	---	1.177	---	1.234
2	48.6	1.468	94.7	1.788
4	291.8	1.303	567.9	1.477
1/2	64.8	1.201	126.2	1.317
1/2/3	205.1	1.180	399.3	1.259
1/2/3/4	466.9	1.177	908.5	1.250

B. Performance, Power, and Area (PPA) Evaluation

The hardware PPA with different maximum number of elements p_{max} is evaluated for problems with 3038 up to 85900 cities. The settings are summarized in Table II. 16/14nm FinFET technology is assumed, and weight precision is 8-bit. One array is assumed to include five rows and two columns of windows. The results show that the chip area is almost proportional to the SRAM capacity (Fig. 7(b)). The latency and energy consumption with read and write breakdown are shown in Fig. 7(c) and (d). The write only happens every 50 iterations to recover the weights, so its portion is much less than the read/computation operation. When p_{max} equals to 2, it requires the least area, but longer latency because it requires the highest cluster hierarchies. When p_{max} equals to 4, the solution quality is the best, but it also requires more hardware cost. The best trade-off happens when p_{max} equals to 3 with moderate hardware cost and close-to-best solution quality.

TABLE II. PPA Evaluation Settings

Tech node	16/14nm FinFET		
Precision	8-bit weight, 1-bit input		
p_{max}	window size	Array size	Array area (μm×μm)
2	8×4	40×64	57×55
3	15×9	75×144	102×98
4	24×16	120×256	161×162

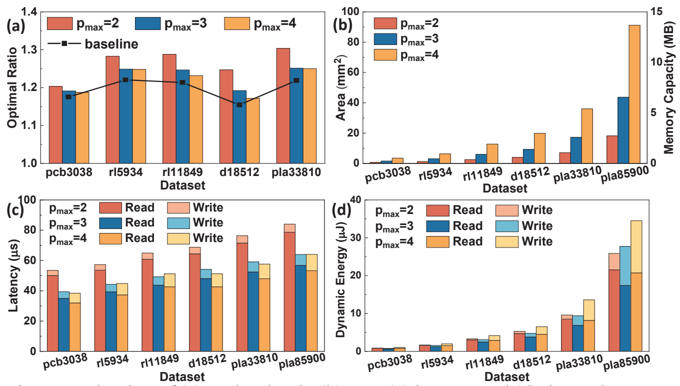


Fig. 7. Evaluation of (a) optimal ratio (b) area (c) latency and (d) dynamic energy of different datasets and p_{max}. The baseline in (a) corresponds to the case with unlimited number of elements p.

VI. Related Works

The CPU baseline reported from Concorde TSP solver [13] shows that the pcb3038, rl5934, and rl11849 require 22 hours, 7 days, and 155 days to be solved respectively, which means our proposed design speeds up 10^9 to 10^{11} with <25% additional travelling distance compared with the optimal tour. There are some algorithms proposed to speed up convergence by parallel updating or parallel replication, e.g., quantum-inspired simulated bifurcation [14-16], stochastic simulated annealing [17], stochastic cellular automata [18], improved parallel annealing [19], adaptive parallel tempering [20], etc. Although the classical simulated annealing is adopted in this design, the parallel updating of nonadjacent clusters enables comparable computing parallelism and convergence speed at the algorithm level. But it is hard to benchmark directly because the aforementioned algorithms were mostly tested on a simpler Max-Cut or small-scale TSP problems (up to 22 cities).

Besides the clustering approach [2], Neuro-Ising [21] was proposed to use graph neural networks to predict a global direction in the solution space and an Ising model to solve the clustered TSP. It was applied to rl5934 problem obtaining ~1.7 optimal ratio (i.e., 70% solution quality overhead) in 25 seconds, where Ising annealing takes ~8s. While our design solves rl5934 with 1.25 optimal ratio in 44μs for annealing step. Compared with [3] using 90Mb FinFET based charge-trap memory to solve 1060-city TSP, our proposed design uses only 46.4Mb SRAM to solve 85900-city problems. There are also several SRAM-based scalable Ising machines [18, 22-25] tested with Max-Cut problems as listed in Table III. The physical area and power efficiency of our design are slightly better than the best reported value there. However, the number of required spins for Max-Cut is equal to its number of nodes, instead of the quadratic relationship for TSP, and thus Max-Cut is a much simpler problem. Our design optimizes the input and weight sparsity for more challenging TSPs and utilizes 46.4Mb weights to achieve the same functionality that requires 4×10^{20}b weights to solve pla85900 before the optimization. If normalizing the metrics with the functionally equivalent weight bits (i.e., a reflection of problem scale), our proposed design improves $>10^{13}\times$ on area and power compared to the others reported in Table III.

TABLE III. COMPARISON WITH SOTA SCALABLE ANNEALERS

	STATICA [18]	CIM-Spin [22]	[23]	FlexSpin [24]	Amorphica [25]	This design
Technology	65nm CMOS	65nm CMOS	40nm CMOS	65nm CMOS	40nm CMOS	16/14nm CMOS
Problem	Max-Cut	Max-Cut	Max-Cut	Max-Cut	Max-Cut	TSP
# Spins	512	480	16k × 9 chips	1024	2k	0.39M* (7.4G)†
Weight Memory	1.31Mb	17.28kb	0.64Mb	57kb	8Mb	46.4Mb* (4×10^{20}b)†
Chip area	12mm^2	0.4mm^2	10.8mm^2	0.34mm^2	9mm^2	43.7mm^2
Chip Power	649mW	360uW	NA	1.17mW	313mW	433mW
Area / weight bit	9μm^2	23μm^2	16.5μm^2	6μm^2	1.1μm^2	0.94μm^2** (10^{-13}μm^2)††
Power / weight bit	495nW	21nW	NA	20nW	38nW	9.3nW** (10^{-12}nW)††

*Physical value, taking the case when p_{max}=3 for pla85900.
†Functional value, i.e., the required #spins(N^2) or #weights(N^4) before optimization (N=85900).
**Normalized with physical weight bit.
††Normalized with functionally equivalent weight bit.

VII. SUMMARY

We propose a digital CIM-based annealer for large-scale TSP with tens of thousands of cities. Clustering solves input sparsity and reduces the weight memory capacity from $O(N^4)$ to $O(N^2)$; flexibility of digital CIM further solves the weight sparsity and reduces the capacity to $O(N)$. Parallel updating is enabled between non-adjacent clusters. The intrinsic process variation of SRAM is controlled by supply voltage and utilized for annealing by converting spatial noise to temporal noise. The proposed design improves the convergence speed $>10^9\times$ with <25% solution quality overhead compared with CPU baseline, and improves the area and power efficiency $>10^{13}\times$ compared with other scalable annealers for large-scale COP.

ACKNOWLEDGMENT

This work was supported in part by Intel Emerging Technology Strategic Research Sector (SRS) funding and mentored by Exploratory Integrated Circuits, Components Research, Intel Corporation.

REFERENCES

[1] A. R. Karlin, et al., “A (slightly) improved approximation algorithm for metric TSP,” *ACM SIGACT STOC*, pp. 32-45, 2021.
[2] A. Dan, et al., “Clustering approach for solving traveling salesman problems via Ising model based solver”, *ACM/IEEE DAC*, 2020.
[3] A. Lu, et al., “Scalable in-memory clustered annealer with temporal noise of charge trap transistor for large scale travelling salesman problems,” *IEEE J. Emerg. Sel. Top. Circuits Syst.*, 13(1), pp. 422-435, 2023.
[4] M. Yamaoka, et al., “A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing,” *IEEE JSSC*, 51(1), pp. 303-309, 2015.
[5] M. Bagherbeik, et al., “A permutational boltzmann machine with parallel tempering for solving combinatorial optimization problems”, *PPSN*, pp. 317-331, 2020.
[6] Y.-D. Chih, *et al.*, "An 89TOPS/W and 16.3 TOPS/mm 2 all-digital SRAM-based full-precision compute-in memory macro in 22nm for machine-learning edge applications," *IEEE ISSCC*, 2021.
[7] C.-F. Lee, *et al.*, "A 12nm 121-TOPS/W 41.6-TOPS/mm2 all digital full precision SRAM-based compute-in-memory with configurable bit-width for AI edge applications," *IEEE VLSI*, 2022.
[8] H. Fujiwara, *et al.*, "A 5-nm 254-TOPS/W 221-TOPS/mm2 fully-digital computing-in-memory macro supporting wide-range dynamic-voltage-frequency scaling and simultaneous MAC and write operations," *IEEE ISSCC*, 2022.
[9] S. Yu, *et al.*, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits Syst. Mag.,* 21(3), pp. 31-56, 2021.
[10] J. Gonzalez, et al., “Parallel Gibbs sampling: From colored fields to thin junction trees,” *AISTATS, PMLR*, pp. 324-332, 2011.
[11] G. Reinelt, “TSPLIB—A traveling salesman problem library,” *ORSA J. Comput.*, 3(4), pp. 376-384, 1991.
[12] X. Peng, et al., “DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies,” *IEEE IEDM*, 2019.
[13] Concorde Benchmarks (from 1999), https://www.math.uwaterloo.ca/tsp/concorde/benchmarks/bench99.html.
[14] Y. Zou and M. Lin, “Massively simulating adiabatic bifurcations with FPGA to solve combinatorial optimization,” *ACM/SIGDA FPGA*, 2020.
[15] H. Goto, et al., “High-performance combinatorial optimization based on classical mechanics,” *Sci. Adv.*, 7(6), p. eabe7953, 2021.
[16] T. Zhang and J. Han, “ Efficient traveling salesman problem solvers using the Ising model with simulated bifurcation,” *DATE*, 2022.
[17] D. Shin, et al., “Memory-efficient fpga implementation of stochastic simulated annealing,” *IEEE J. Emerg. Sel. Top. Circuits Syst.*, 13(1), pp. 108-118, 2023.
[18] K. Yamamoto, et al., “STATICA: A 512-spin 0.25M-weight full-digital annealing processor with a near-memory all-spin-updates-at-once architecture for combinatorial optimization with complete spin-spin interactions,” *IEEE ISSCC*, 2020.
[19] Q. Tao, and J. Han, “Solving traveling salesman problems via a parallel fully connected Ising machine,” *ACM/IEEE DAC*, 2022.
[20] N.A. Aadit, et al., “Accelerating Adaptive Parallel Tempering with FPGA-based p-bits,” *IEEE VLSI*, 2023.
[21] S. Sanyal, K. Roy, “Neuro-Ising: Accelerating large scale travelling salesman problems via graph neural network guided localized Ising solvers”, *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, pp. 1-1, 2022.
[22] Y. Su, et al., “CIM-spin: A 0.5-to-1.2 V scalable annealing processor using digital compute-in-memory spin operators and register-based spins for combinatorial optimization problems,” *IEEE ISSCC,* pp. 480-482, 2020.
[23] T. Takemoto, et al., “A 144Kb annealing system composed of 9×16Kb annealing processor chips with scalable chip-to-chip connections for large-scale combinatorial optimization problems,” *IEEE ISSCC,* 2021.
[24] Y. Su, et al., “ Flexspin: A scalable CMOS Ising machine with 256 flexible spin processing elements for solving complex combinatorial optimization problems,” *IEEE ISSCC*, 2022.
[25] K. Kawamura, et al., “Amorphica: 4-replica 512 fully connected spin 336MHz metamorphic annealer with programmable optimization strategy and compressed-spin-transfer multi-chip extension,” *IEEE ISSCC*, 2023.